

https://dspace.org/

A transfer learning approach to wildlife identification and classification

Naran, Jagruthi

2021
https://hdl.handle.net/10539/33366

Downloaded from a DSpace repository. https://dspace.org/



A transfer learning approach to wildlife identification and classification

Jagruthi Naran (450671)

School of Mechanical, Industrial and Aeronautical Engineering
University of the Witwatersrand
Johannesburg, South Africa

Supervisor: Dr. J. Bührmann

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

October 2021

Plagiarism Declaration

I hereby declare the following:

- This dissertation is my own unaided work. It is being submitted for the degree of Master of Science to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other University;
- I am aware that plagiarism (the use of someone else's work without their permission and/or without acknowledging the original source) is wrong;
- I confirm that the work submitted herewith for assessment in the above course is my own unaided work except where I have explicitly indicated otherwise;
- I have followed the required conventions in referencing the thoughts and ideas of others;
- I understand that the University of the Witwatersrand may take disciplinary action against me if it can be shown that this task is not my own unaided work or that I have failed to acknowledge the sources of the ideas or words in my writing.

Signature:  Date: 3 October 2021

ANIMAL RESEARCH ETHICS COMMITTEE

Registration number: AREC-101210-002

August 2, 2019

Ethical Clearance

To whom it may concern,
Reference: Waiver J Naran Eng 20190802-O
Re: Waiver from the Animal Research Ethics Committee of the University of the Witwatersrand

Applicant/Student: Jagruthi Naran
Staff number: 450671
Degree: MSc (Engineering)

Study Title: "*A transfer learning approach to wildlife identification and classification*".

Department: School of Mechanical, Industrial and Aeronautical Engineering

Supervisor: Dr J Bührmann

Tel: + 011-717-7360; Email: joke.buhrmann@wits.ac.za

Full clearance certificate from the AREC of the University of the Witwatersrand is NOT required.

Reason: Study uses mathematical modelling and machine learning to recognize animals by image recognition. It does not involve animals.

Comment/Notes: - Please contact me should you require further information.

Yours sincerely

GP Candy

Geoffrey Candy
Chair : Animal Research Ethics Committee, University of the Witwatersrand

Geoffrey P Candy PhD
Professor, Department of Surgery, Faculty of Health Sciences, University of the Witwatersrand, *7 York Road Parktown 2193, Johannesburg; Tel: 27-11-717-2574; Email: geoffrey.candy@wits.ac.za*

Abstract

Global climate change is having a significant effect on ecosystems, causing species to vary their behaviour and migrate to more suitable regions. This will require more sustainable management of wildlife populations. Despite research conducted on the identification of wildlife through traditional and deep learning techniques, the challenge of large, labelled datasets required to perform these tasks still exists. In this study, a transfer learning approach is presented for wildlife classification applications. Four pretrained models were trained to identify wildlife in 48 species from the Snapshot Serengeti, Greenpeace and African Wildlife Foundation image datasets. This was done by using two transfer learning techniques, namely freeze-layer and fine-tuning, varying the learning rate and using different pretrained models, namely AlexNet, GoogLeNet, ResNet-101 and VGG-19. Results show that deeper, fine-tuned networks with lower learning rates result in higher classification accuracies. In addition, a comparison of the training times confirmed that the run times for the freeze-layer models were shorter than those of the fine-tuned models. The best result achieved a top-1 and top-5 accuracy of 100% for the ResNet-101 model.

Acknowledgements

I would like to express my sincere gratitude to the following individuals for their contribution to this research.

My supervisor, Dr Joke Bührmann, for introducing me to the world of machine learning and for her constant support, patience and guidance throughout this project and for her faith in my capabilities.

Professor Barend Erasmus from the Global Change Institute for affording me the time to discuss my research and for introducing me to various individuals who were able to provide me with the image datasets.

Dr Mark Keith from the University of Pretoria and Dr Nakedi Maputla from the African Wildlife Foundation for providing me with camera-trap images that were used in this research.

The Southern African Wildlife Management Association (SAWMA) for assisting me and introducing me to various individuals who were able to assist with this research.

The Zooniverse Team for allowing me to use the Snapshot Serengeti image datasets for my research.

Greenpeace for providing me with access to use the media library and allowing me to use the images in my research.

Portia Molepo and Tinyeko Mdhlovu for assisting me with the MATLAB software.

Danillo van Eck, Ashref Makra, Lando Alexander and Keogile Mofokeng for sponsoring the use of the GPU and hardware required to run the model simulations.

Lastly, I would like to thank my dad for all his help with this research and for doing everything in his power to ensure that things run smoothly and going above and beyond to assist wherever he could. Also, to my mum and sister for their continued support, guidance, and motivation in everything I do. I could not have done this without you.

Table of Contents

Plagiarism Declaration i

Ethical Clearance ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Figures x

List of Tables xx

Nomenclature xxii

1 PROJECT BACKGROUND 1

1.1 Research Background 1

1.2 The Problem Context 2

1.3 Purpose of Study 3

1.4 Research Motivation 3

1.5 Research Questions 4

1.6 Research Objectives 4

1.7 Delimitations 4

1.8 Assumptions 5

1.9 Report Structure Overview 5

2 LITERATURE REVIEW 6

2.1 Artificial Intelligence: A Brief History 6

2.2 Machine Learning 7

2.2.1 Big data and machine learning 8

2.2.2 Machine learning algorithms 8

2.2.3 Transfer learning 10

2.3 Deep Learning 13
2.3.1 History of neural networks 13
2.3.2 Overview of deep learning 14
2.3.3 Deep network architectures 18
2.3.4 Optimisation 29
2.3.5 Training & learning 31
2.3.6 Understanding CNN's 38
2.4 Related Work on Wildlife Imaging, Monitoring, and Identification 40
2.4.1 Traditional approaches 40
2.4.2 Challenges with camera trapping 41
2.4.3 Deep learning approaches 45
3 RESEARCH METHODOLOGY 47
3.1 Introduction 47
3.2 Theoretical Foundation 47
3.3 Research Design 48
3.3.1 Research instruments 48
3.3.2 Datasets 49
3.3.3 Model architecture 55
3.3.4 Activation visualisations 56
3.3.5 Feature visualisations 72
3.3.6 Model development 82
3.3.7 Experimental framework 90
4 RESULTS 97
4.1 Model Experiments 97
4.2 Training & Validation Results 97

4.2.1 Experiment 1 results 97
4.2.2 Experiment 2 results 104
4.2.3 Experiment 3 results 111
4.3 Test Results 115
4.3.1 Experiment 1 115
4.3.2 Experiment 2 124
4.3.3 Experiment 3 132
4.4 Summary 136
5 ANALYSIS 139
5.1 Training & Validation Results Analysis 139
5.1.1 Experiment 1 139
5.1.2 Experiment 2 142
5.1.3 Experiment 3 144
5.2 Test Results Analysis 146
5.2.1 Experiment 1 146
5.2.2 Experiment 2 150
5.2.3 Experiment 3 154
5.3 Summary 156
6 DISCUSSION 159
6.1 Introduction 159
6.2 Effect of transfer learning techniques 159
6.3 Effect of class imbalance 162
6.4 Effects of learning rates 164
6.5 Effect of architecture 166
7 CONCLUSION & RECOMMENDATIONS 168

REFERENCE LIST 170
Appendix A 183
Experiment 1A 183
AlexNet 183
GoogLeNet 184
ResNet-101 185
Experiment 1B 187
AlexNet 187
GoogLeNet 188
ResNet-101 189
VGG-19 190
Experiment 2A 192
GoogLeNet 192
ResNet-101 193
VGG-19 194
Experiment 2B 195
AlexNet 195
GoogLeNet 197
ResNet-101 198
VGG-19 199
Experiment 3 200
AlexNet 200
GoogLeNet 202
ResNet-101 203
VGG-19 204

Appendix B..... 206

Appendix C..... 224

List of Figures

Figure 1: Transfer learning strategies (Pan & Yang 2010) 12

Figure 2: An artificial neuron (Lobova et al. 2007) 14

Figure 3: Training a CNN (adapted from Thenmozhi & Reddy 2019) 18

Figure 4: Top1 accuracy versus the network (Canziani et al. 2016) 19

Figure 5: Top1 accuracy versus the network operations and size proportional to the parameters (Canziani et al. 2016) 20

Figure 6: AlexNet architecture (Adapted from Krizhevsky et al. 2012) 24

Figure 7: GoogLeNet architecture (Adapted from Szegedy et al. 2015) 25

Figure 8: Inception-v3 architecture (Adapted from Szegedy et al. 2016) 26

Figure 9: ResNet architecture (Adapted from He et al. 2016) 27

Figure 10: VGG-19 architecture (Adapted from Garcia-Gasulla et al. 2017) 28

Figure 11: Images with different exposure levels and day and night-time images 42

Figure 12: Images with partial of the animal species in the frame 43

Figure 13: Empty images with no animal species in the frame 43

Figure 14: Intra-species images with animals from the same species in a frame 44

Figure 15: Multiple animal species appear in a single frame 44

Figure 16: Full-body images of wildlife species 45

Figure 17: Practical design process adapted from Goodfellow et al. (2016) 47

Figure 18: GPU device for computer 1 49

Figure 19: GPU device for computer 2 49

Figure 20: Examples of augmented images from the SS dataset 55

Figure 21: AlexNet input image example for visualising activations 57

Figure 22: Visualisation of activations extracted from convolution layer one (AlexNet) 57

Figure 23: Visualisation of activations extracted from channels 33, 34 and 45 from convolution layer one (AlexNet) 58

Figure 24: Visualisation of activations extracted from ReLU layer one (AlexNet)..... 58

Figure 25: Visualisation of activations extracted from normalisation layer one (AlexNet) 58

Figure 26: Visualisation of activations extracted from pooling layer one (AlexNet) 59

Figure 27: Visualisation of activations extracted from convolution layer five (AlexNet) 60

Figure 28: Visualisation of activations extracted from pooling layer five (AlexNet).... 60

Figure 29: Visualisation of activations extracted from fully connected layer eight (AlexNet) 60

Figure 30: Visualisation of activations extracted from the output layer (AlexNet) 61

Figure 31: GoogLeNet input image example for visualising activations..... 61

Figure 32: Visualisation of activations extracted from convolution layer 7x7 (GoogLeNet)..... 62

Figure 33: Visualisation of activations extracted from channels 33, 34 and 45 from convolution layer one (GoogLeNet)..... 62

Figure 34: Visualisation of activations extracted from ReLU layer 7x7 (GoogLeNet) . 63

Figure 35: Visualisation of activations extracted from pooling layer one (GoogLeNet) 63

Figure 36: Visualisation of activations extracted from normalisation layer one (GoogLeNet)..... 63

Figure 37: Visualisation of activations extracted from inception layer 3a 1x1 (GoogLeNet)..... 64

Figure 38: Visualisation of activations extracted from inception layer 5b 5x5 (GoogLeNet)..... 64

Figure 39: Visualisation of activations extracted from the output layer (GoogLeNet).. 64

Figure 40: ResNet-101 input image example for visualising activations..... 65

Figure 41: Visualisation of activations extracted from convolution layer one (ResNet-101)..... 65

Figure 42: Visualisation of activations extracted from channels 33, 34 and 45 from convolution layer one (ResNet-101)..... 66

Figure 43: Visualisation of activations extracted from ReLU layer one (ResNet-101) . 66

Figure 44: Visualisation of activations extracted from pooling layer one (ResNet-101)66

Figure 45: Visualisation of activations extracted from batch normalisation layer one (ResNet-101) 67

Figure 46: Visualisation of activations extracted from convolution layer 2b_branch2c (ResNet-101) 67

Figure 47: Visualisation of activations extracted from convolution layer 4b22_branch2c (ResNet-101) 67

Figure 48: Visualisation of activations extracted from convolution layer 5c_branch2c (ResNet-101) 68

Figure 49: Visualisation of activations extracted from the output layer (ResNet-101).. 68

Figure 50: VGG-19 input image example for visualising activations..... 69

Figure 51: Visualisation of activations extracted from convolution layer one (VGG-19 model)..... 69

Figure 52: Visualisation of activations extracted from channels 33, 34 and 45 from convolution layer one (VGG-19)..... 70

Figure 53: Visualisation of activations extracted from ReLU layer one (VGG-19) 70

Figure 54: Visualisation of activations extracted from pooling layer one (VGG-19).... 70

Figure 55: Visualisation of activations extracted from convolution layer 4_4 (VGG-19) 71

Figure 56: Visualisation of activations extracted from convolution layer 5_2 (VGG-19) 71

Figure 57: Visualisation of activations extracted from fully connected layer seven (VGG-19)..... 71

Figure 58: Visualisation of activations extracted from fully connected layer eight (VGG-19)..... 71

Figure 59: Visualisation of activations extracted from the output layer (VGG-19)..... 72

Figure 60: Input image example for visualising features 73

Figure 61: Visualisation of features learned by convolutional layer one (AlexNet)..... 73

Figure 62: Visualisation of features learned by convolutional layer two (AlexNet)..... 74

Figure 63: Visualisation of features learned by convolutional layer five (AlexNet) 74

Figure 64: Visualisation of features learned by fully connected six (AlexNet) 75

Figure 65: Visualisation of features learned by fully connected seven (AlexNet)..... 75

Figure 66: Visualisation of features learned by fully connected layer seven - channel two (AlexNet) 75

Figure 67: Visualisation of features learned by inception output layer 3b (GoogLeNet) 76

Figure 68: Visualisation of features learned by inception output layer 4a (GoogLeNet) 76

Figure 69: Visualisation of features learned by inception output layer 4e (GoogLeNet) 77

Figure 70: Visualisation of features learned by inception output layer 5b (GoogLeNet) 77

Figure 71: Visualisation of features learned by convolution layer one (ResNet-101) ... 78

Figure 72: Visualisation of features learned by convolution layer 3b_branch2c (ResNet-101)..... 78

Figure 73: Visualisation of features learned by convolution layer 4b10_branch2c (ResNet-101) 79

Figure 74: Visualisation of features learned by convolution layer 5c_branch2c (ResNet-101)..... 79

Figure 75: Visualisation of features learned by convolution layer 5c_branch2c - channel one (ResNet-101) 80

Figure 76: Visualisation of features learned by the output layer (ResNet-101) 80

Figure 77: Visualisation of features learned by convolution layer one (VGG-19) 81

Figure 78: Visualisation of features learned by convolution layer 4_1 (VGG-19) 81

Figure 79: Visualisation of features learned by fully connected layer eight (VGG-19) 82

Figure 80: Visualisation of features learned by the output layer (VGG-19) 82

Figure 81: Frozen layers of pretrained AlexNet model (network illustration adapted from Krizhevsky et al. 2012) 83

Figure 82: Frozen layers of pretrained GoogLeNet model (network illustration adapted from Szegedy et al. 2015) 84

Figure 83: Frozen layers of pretrained ResNet-101 model (network illustration adapted from He et al. 2016) 84

Figure 84: Frozen layers of pretrained VGG-19 model (network illustration adapted from Garcia-Gasulla et al. 2017) 85

Figure 85: Fine-tuned layers of pretrained AlexNet model (network illustration adapted from Krizhevsky et al. 2012) 86

Figure 86: Fine-tuned layers of pretrained GoogLeNet model (network illustration adapted from Szegedy et al. 2015) 87

Figure 87: Fine-tuned layers of pretrained ResNet-101 model (network illustration adapted from He et al. 2016) 87

Figure 88: Fine-tuned layers of pretrained VGG-19 model (network illustration adapted from Garcia-Gasulla et al. 2017) 88

Figure 89: Class imbalance of Snapshot Serengeti dataset 92

Figure 90: Experiment 1A - AlexNet accuracy curve 98

Figure 91: Experiment 1A - AlexNet loss curve 98

Figure 92: Experiment 1A - GoogLeNet accuracy curve 99

Figure 93: Experiment 1A - GoogLeNet loss curve 99
Figure 94: Experiment 1A - ResNet-101 accuracy curve 100
Figure 95: Experiment 1A - ResNet-101 loss curve 100
Figure 96: Experiment 1B - AlexNet accuracy curve 101
Figure 97: Experiment 1B - AlexNet loss curve 101
Figure 98: Experiment 1B - GoogLeNet accuracy curve 102
Figure 99: Experiment 1B - GoogLeNet loss curve 102
Figure 100: Experiment 1B - ResNet-101 accuracy curve 103
Figure 101: Experiment 1B - ResNet-101 loss curve 103
Figure 102: Experiment 1B - VGG-19 accuracy curve 104
Figure 103: Experiment 1B - VGG-19 loss curve 104
Figure 104: Experiment 2A - GoogLeNet accuracy curve 105
Figure 105: Experiment 2A - GoogLeNet loss curve 105
Figure 106: Experiment 2A - ResNet-101 accuracy curve 106
Figure 107: Experiment 2A - ResNet-101 loss curve 106
Figure 108: Experiment 2A - VGG-19 accuracy curve 107
Figure 109: Experiment 2A - VGG-19 loss curve 107
Figure 110: Experiment 2B - AlexNet accuracy curve 108
Figure 111: Experiment 2B - AlexNet loss curve 108
Figure 112: Experiment 2B - GoogLeNet accuracy curve 109
Figure 113: Experiment 2B - GoogLeNet loss curve 109
Figure 114: Experiment 2B - ResNet-101 accuracy curve 110
Figure 115: Experiment 2B - ResNet-101 loss curve 110
Figure 116: Experiment 2B - VGG-19 accuracy curve 111
Figure 117: Experiment 2B - VGG-19 loss curve 111

Figure 118: Experiment 3 - AlexNet accuracy curve 112

Figure 119: Experiment 3 - AlexNet loss curve 112

Figure 120: Experiment 3 - GoogLeNet accuracy curve 113

Figure 121: Experiment 3 - GoogLeNet loss curve 113

Figure 122: Experiment 3 - ResNet-101 accuracy curve 114

Figure 123: Experiment 3 - ResNet-101 loss curve 114

Figure 124: Experiment 3 - VGG-19 accuracy curve 115

Figure 125: Experiment 3 - VGG-19 loss curve 115

Figure 126: Confidence levels for classification examples (Experiment 1A - AlexNet) 116

Figure 127: Confidence levels for classification examples (Experiment 1A - GoogLeNet) 117

Figure 128: Confidence levels for classification examples (Experiment 1A - ResNet-101) 119

Figure 129: Confidence levels for classification examples (Experiment 1B - AlexNet) 120

Figure 130: Confidence levels for classification examples (Experiment 1B - GoogLeNet) 121

Figure 131: Confidence levels for classification examples (Experiment 1B - ResNet-101) 122

Figure 132: Confidence levels for classification examples (Experiment 1B - VGG-19) 123

Figure 133: Confidence levels for classification examples (Experiment 2A - GoogLeNet) 125

Figure 134: Confidence levels for classification examples (Experiment 2A - ResNet-101) 126

Figure 135: Confidence levels for classification examples (Experiment 2A - VGG-19) 127

Figure 136: Confidence levels for classification examples (Experiment 2B - AlexNet) 128

Figure 137: Confidence levels for classification examples (Experiment 2B - GoogLeNet) 129

Figure 138: Confidence levels for classification examples (Experiment 2B - ResNet-101) 130

Figure 139: Confidence levels for classification examples (Experiment 2B - VGG-19) 131

Figure 140: Confidence levels for classification examples (Experiment 3 - AlexNet) 132

Figure 141: Confidence levels for classification examples (Experiment 3 - GoogLeNet) 133

Figure 142: Confidence levels for classification examples (Experiment 3 - ResNet-101) 134

Figure 143: Confidence levels for classification examples (Experiment 3 - VGG-19) 135

Figure 144: Summary of experiments by top-1 and top-5 accuracy 137

Figure 145: Summary of training time in minutes by experiment 138

Figure 146: Model predictions with confidence level (Experiment 1A - AlexNet)..... 184

Figure 147: Model predictions with confidence level (Experiment 1A - GoogLeNet) 185

Figure 148: Model predictions with confidence level (Experiment 1A - ResNet-101) 186

Figure 149: Model predictions with confidence level (Experiment 1B - AlexNet)..... 188

Figure 150: Model predictions with confidence level (Experiment 1B - GoogLeNet) 189

Figure 151: Model predictions with confidence level (Experiment 1B - ResNet-101) 190

Figure 152: Model predictions with confidence level (Experiment 1B - VGG-19)..... 191

Figure 153: Model predictions with confidence level (Experiment 2A - GoogLeNet) 193

Figure 154: Model predictions with confidence level (Experiment 2A - ResNet-101) 194

Figure 155: Model predictions with confidence level (Experiment 2A - VGG-19) 195
Figure 156: Model predictions with confidence level (Experiment 2B - AlexNet) 196
Figure 157: Model predictions with confidence level (Experiment 2B - GoogLeNet) 198
Figure 158: Model predictions with confidence level (Experiment 2B - ResNet-101) 199
Figure 159: Model predictions with confidence level (Experiment 2B - VGG-19)..... 200
Figure 160: Model predictions with confidence level (Experiment 3 - AlexNet)..... 201
Figure 161: Model predictions with confidence level (Experiment 3 - GoogLeNet) .. 203
Figure 162: Model predictions with confidence level (Experiment 3 - ResNet-101) .. 204
Figure 163: Model predictions with confidence level (Experiment 3 - VGG-19) 205
Figure 164: Confusion matrix for Experiment 1A - AlexNet 206
Figure 165: Confusion matrix for Experiment 1A - GoogLeNet 207
Figure 166: Confusion matrix for Experiment 1A - ResNet-101 208
Figure 167: Confusion matrix for Experiment 1B - AlexNet..... 209
Figure 168: Confusion matrix for Experiment 1B - GoogLeNet 210
Figure 169: Confusion matrix for Experiment 1B - ResNet-101 211
Figure 170: Confusion matrix for Experiment 1B - VGG-19 212
Figure 171: Confusion matrix for Experiment 2A - GoogLeNet 213
Figure 172: Confusion matrix for Experiment 2A - ResNet-101 214
Figure 173: Confusion matrix for Experiment 2A - VGG-19 215
Figure 174: Confusion matrix for Experiment 2B - AlexNet..... 216
Figure 175: Confusion matrix for Experiment 2B - GoogLeNet 217
Figure 176: Confusion matrix for Experiment 2B - ResNet-101 218
Figure 177: Confusion matrix for Experiment 2B - VGG-19 219
Figure 178: Confusion matrix for Experiment 3 - AlexNet 220
Figure 179: Confusion matrix for Experiment 3 - GoogLeNet 221

Figure 180: Confusion matrix for Experiment 3 - ResNet-101 222

Figure 181: Confusion matrix for Experiment 3 - VGG-19 223

List of Tables

Table 1: Summary of deep network architectures 24

Table 2: Confusion matrix for binary classification (Hossin & Sulaiman 2015) 37

Table 3: Evaluation metrics 37

Table 4: Summary of image datasets 51

Table 5: Summary of image datasets download procedure 52

Table 6: Input image sizes 54

Table 7: Network training hyperparameters from Norouzzadeh et al. (2017) study 89

Table 8: Experiments for testing 89

Table 9: Training and test data split 91

Table 10: Experiment 1 with training parameters 93

Table 11: Experiment 2 with training parameters 93

Table 12: Experiment 3 with training parameters 94

Table 13: Summary of hyperparameters for Experiments 1, 2 and 3 97

Table 14: Results for Experiment 1A - AlexNet 117

Table 15: Results for Experiment 1A - GoogLeNet 118

Table 16: Results for Experiment 1A - ResNet-101 119

Table 17: Results for Experiment 1B - AlexNet 120

Table 18: Results for Experiment 1B - GoogLeNet 122

Table 19: Results for Experiment 1B - ResNet-101 123

Table 20: Results for Experiment 1B - VGG-19 124

Table 21: Results for Experiment 2A - GoogLeNet 125

Table 22: Results for Experiment 2A - ResNet-101 126

Table 23: Results for Experiment 2A - VGG-19 127

Table 24: Results for Experiment 2B - AlexNet 128

Table 25: Results for Experiment 2B - GoogLeNet 129

Table 26: Results for Experiment 2B - ResNet-101 130

Table 27: Results for Experiment 2B - VGG-19 131

Table 28: Results for Experiment 3 - AlexNet 132

Table 29: Results for Experiment 3 - GoogLeNet 133

Table 30: Results for Experiment 3 - ResNet-101 135

Table 31: Results for Experiment 3 - VGG-19 136

Table 32: Precision values per animal class for all experiments[1] 224

Table 33: Recall values per animal class for all experiments[2] 226

Nomenclature

Acronym	Definition
AI	Artificial intelligence
AWF	African Wildlife Foundation
ANN	Artificial neural network
API	Application Programming Interface
BGD	Batch Gradient Descent
CNN	Convolutional neural network
CUDA	Computer Unified Device Architecture
GAN	Generative Adversarial Networks
GPU	Graphical Processing Unit
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
ML	Machine Learning
NaN	Not a Number
ReLU	Rectified Linear Units
SGD	Stochastic Gradient Descent
SGDm	Stochastic Gradient Descent with Momentum
VGG	Visual Geometry Group

1 PROJECT BACKGROUND

This chapter introduces the research topic and provides context for the research.

1.1 Research Background

Global climate change is having a significant effect on species, ecosystems, and biodiversity (Levin 2007; Gitay et al. 2002). This is likely to alter species composition in protected areas as more species move in or out of parks and reserves as the climate changes (Hannah et al. 2005; Parmesan 2006). Effects of climate change include (Levin 2007; Gitay et al. 2002; Mawdsley et al. 2008):

- shifts in species distributions;
- changes to migratory patterns;
- invasive species outbreaks;
- effects of demographic survival and mortality rates;
- loss of habitats;
- reductions in population size; and
- extinction of populations.

As climate change affects ecosystems, species will vary their behaviour and migrate to more suitable regions (Levin 2007). This will require more effective and sustainable management of wildlife populations with regards to population size, movement patterns and species distribution (Gilchrist et al. 2005).

Although extensive research has been conducted on understanding and predicting climate change, researchers have developed an interest in identifying practical strategies to reduce or improve the negative effects caused by climate change (Mawdsley et al. 2008). The approach of implementing human activities to minimise the adverse effects of climate change on ecosystems and the natural environment has been termed as adaptation strategies. A review on climate change and adaptation strategies for wildlife management and biodiversity conservation outlined four broad categories for climate change adaptation strategies (Mawdsley et al. 2008):

- Land and water protection and management
- Direct species management
- Monitoring and planning

- Law and policy

The scope of this study will focus on adding to literature on strategies relating to monitoring and planning of wildlife through the use of artificial intelligence techniques.

Traditional methods of collecting information and extracting knowledge from wildlife imaging have been expensive and tedious. The use of artificial intelligence, and more specifically deep learning, a type of machine learning (ML) technique, have led to advances in the field of artificial intelligence, computer vision and image recognition. This has resulted in the automation of the information and knowledge extraction process, allowing for ecologists to perform their scientific, management and conservation missions (Norouzzadeh et al. 2017).

Despite research that has been conducted on the identification of wildlife through camera trap images using traditional methods and deep learning techniques, there are key issues with these methods. Some of these issues include vast amounts of labelled data and significant computational resources which are required to perform the required tasks (Norouzzadeh et al. 2017).

1.2 The Problem Context

Breakthrough performance improvements in deep learning for image classification, object detection, speech recognition and natural language processing have proven to be particularly effective in processing pixel data. In particular, convolutional neural networks (CNN) have become widely used in image classification and this has been demonstrated in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), an annual competition in object recognition based on a dataset of over a million images in hundreds of object categories (Krizhevsky et al. 2012).

However, deep learning works well with vast amounts of labelled data and requires significant computational resources. The use of transfer learning to overcome the problem of large-labelled datasets for wildlife identification is proposed in this study. Transfer learning is a ML technique in which a model is trained on one task and the learnings are repurposed on a second, related task with comparatively less data.

The varying size of datasets can be noted in wildlife identification studies conducted by Chen et al. (2014), Villa et al. (2017) and Norouzzadeh et al. (2017). Results from these

studies show a 92.0% accuracy on the best network developed by Norouzzadeh et al. (2017) who used 1.4 million images for training and 105 000 images for testing, versus 57% accuracy on 2 597 images used in the study conducted by Villa et al. (2017) and 38% accuracy from the study conducted by Chen et al. (2014) who used 20 000 images and 20 classes.

The size of the datasets as well as constraints in identifying and obtaining large-labelled image datasets re-emphasize the need for a method to conduct image classification analysis in the absence of large-labelled datasets.

1.3 Purpose of Study

The purpose of this study is to use transfer learning to overcome the problem of large-labelled datasets for wildlife identification, which range in the order of over 750 000 images. This is based on 25% of the Snapshot Serengeti (SS) image dataset, which contains 757 000 non-empty images or images that contain animals (Norouzzadeh et al. 2017) as well as datasets obtained from the African Wildlife Foundation (AWF) and Greenpeace.

In this study, a deep neural network will be trained using transfer learning through pre-trained networks. Training and validation data from a larger labelled dataset will be used and test data from a smaller dataset will be used to classify the images. The test datasets containing a substantially smaller number of images, in the region of 50 000 images, will be used together with deep convolutional neural networks (CNNs) to identify and classify wildlife based on camera trap imaging. The results will be compared to existing results of camera trap wildlife imaging obtained from deep neural networks, in which training and learning data were from the small, labelled datasets.

1.4 Research Motivation

Although research into the identification of wildlife through camera trap images has been conducted through both traditional hand-designed features and ML techniques, limited research exists on the use of transfer learning applied to wildlife identification.

A study conducted by Norouzzadeh et al. (2017) identified the use of transfer learning to automate animal identification as a field for further study. In addition, previous research on transfer learning applications to wildlife identification and classification used subsets

of existing labelled datasets, adopted the use of manual data augmentation techniques to modify the data and used training and test data from the same dataset.

This dissertation aims to add to existing literature on the use of transfer learning and wildlife identification and classification by presenting an approach through the use of various pre-trained models and varying training parameters.

1.5 Research Questions

Based on the need for further study in this field, a transfer learning approach to wildlife identification and classification is proposed in this dissertation. The following question will be addressed in this research:

How do different ML models compare in wildlife identification in the absence of large-labelled datasets?

With sub questions:

- What effect does learning rate have on the accuracy of prediction?
- What effect does transfer learning technique have on the accuracy of prediction?
- What effect does class imbalance have on the accuracy of the prediction?

1.6 Research Objectives

The research objectives are as follows:

1. Investigate and develop a ML model using transfer learning to overcome the problem of large labelled datasets, which range in the order of over 750 000 images, in the field of wildlife identification.
2. Train the deep CNN on existing datasets and use the knowledge obtained to classify images from datasets with fewer labelled images, which range in the order of 50 000 images.
3. Compare the results obtained to existing results of camera trap wildlife imaging obtained from deep neural networks, in which training and learning data are from the same labelled dataset.

1.7 Delimitations

The delimitations of this research include the following:

- This study is not an extensive overview of machine learning and deep learning theories and research. Only those methodologies relevant to the study will be discussed.
- This study does not aim to provide an in-depth understanding of activation maps and feature visualisation and all content regarding this are aimed at facilitating explanation of the models' training and learning process.
- The image datasets that will be used have been analysed and only non-empty images were identified for use in this study. Non-empty images are those which contain a partial or full view of an animal in the frame.
- This study will only consider instances where there is a single animal species in the frame. All images with multiple species in the frame were excluded.
- Only African wildlife image datasets with wildlife species that are indigenous to the African savanna have been used in this project.
- This study does not aim to analyse the difference between capture events, which consist of several photographs taken at a single burst per trigger, and single images. Single images were used in this research study.

1.8 Assumptions

The assumptions in this research include the following:

- Augmented image datastores provide sufficient data augmentation to prevent overfitting of data.

1.9 Report Structure Overview

The structure of the remaining sections of the report is as follows. The literature review in Chapter 2 will present topics relating to the history of artificial intelligence, machine learning, deep learning, transfer learning techniques and related work on wildlife imaging and classification. Chapter 3 highlights the research methodology employed in this study. The research findings and model results will be discussed in Chapter 4. An analysis of the results will be described in Chapter 5. An explanation and interpretation of the findings will be discussed in Chapter 6. Finally, Chapter 7, will describe the conclusions and recommendations based on the research findings.

2 LITERATURE REVIEW

This chapter presents the literature on a brief history on artificial intelligence, machine learning, deep learning, transfer learning techniques and related work on wildlife imaging and classification.

2.1 Artificial Intelligence: A Brief History

In its simplest terms, artificial intelligence (AI) is a field of computer science that aims to make computers more intelligent (Kononenko 2001). The beginnings of modern AI can be traced back to the classical era where philosophers attempted to describe human thinking as a system (Buchanan 2005). The field of AI, however, was formally founded in 1956 when John McCarthy first defined the term at a conference at Dartmouth College, New Hampshire (Benko & Lanyi 2009; Buchanan 2005). Although the emerging field received its name in 1956, computer intelligence, known as machine intelligence, was being actively pursued for approximately ten years by this time (Copeland 2000).

In 1950, Alan Turing suggested a definition for deciding whether software is intelligent or not. His seminal paper in the philosophy journal *Mind*, developed ideas about the possibility of programming a computer to behave intelligently and proposed the question 'can machines think?' (Turing 1950: 433; Buchanan 2005). The imitation game, also known as the Turing test, suggested that a software is intelligent if a human does not know if he or she is talking to a software or another human (Benko & Lanyi 2009).

Although AI flourished from 1957 to 1974, limitations in computing power resulted in research and funding constraints. Computers were only able to execute commands rather than store them. Hence, they could not remember the tasks that they executed. In addition, due to the size and speed of processors, computing was extremely expensive and only prestigious universities or large technological companies could afford to trial and test artificial intelligence theories and methods (Buchanan 2005). During this time, knowledge-based systems as described in Goldstein and Papert (1977) and Lindsay et al. (1980) saw a 'paradigm shift' in AI towards these types of systems. The Mycin expert system (Buchanan & Shortliffe 1984) and those that followed demonstrated the power of small amounts of knowledge to enable intelligent decision-making.

As the industrial revolution led to a more mechanised world, machinery became more sophisticated and robots became a part of the public's perception of intelligent computers (Buchanan 2005). As a result, robotics and artificial intelligence became synonymous with mechanical engineering rather than intelligent control. Artificial intelligence, however, is not just about robots, but rather using computers as an experimental means to understand the link between intelligent thought and action (Buchanan 2005).

After decades of setbacks in AI research and development due to computational constraints, AI was reignited during the 1980s due to advancements in algorithms and funding. In addition, the development of expert systems aimed to simulate the decision-making process in human beings, which enabled computers to solve complex reasoning problems. Over the last half a century, developments in computational devices and programming languages have enabled researchers to test ideas of what intelligence is (Buchanan 2005).

In 1947, Turing gave, what is known as the earliest public lecture, on computer intelligence stating that 'what we want is a machine that can learn from experience' and having 'the possibility of letting the machine alter its own instructions provides the mechanism for this' (Turing 1947: 13). Most modern researchers today agree that learning is a requirement for intelligence. Having the ability to learn from memory and improve its ability to perform is a notion that lies at the core of artificial intelligence (Minsky & Papert 1969; Buchanan 2005). As a result, machine learning, one of the major branches of artificial intelligence and one of the most rapidly developing subsets of AI research, came about (Shavlik & Dietterich 1990; Kononenko 2001).

2.2 Machine Learning

Machine learning (ML) uses experience, or past information, to improve performance or make accurate predictions thus relying on the use of learning algorithms, data analysis and statistics (Mohri et al. 2012). ML algorithms have been used in a variety of applications including text classification, natural language processing, speech recognition, image recognition, fraud detection, unassisted vehicle controls and medical diagnosis, to name a few. At the core of machine learning is data.

2.2.1 Big data and machine learning

The term ‘big data’ refers to large volumes of data that are both structured and unstructured. Characteristics of big data sets include the volume, velocity, and variety of the data. The volume of available data has increased exponentially with companies having developed the ability to collect petabytes of data transactions on an hourly basis (McAfee & Brynjolfsson 2012).

Velocity, or the speed at which data is being created, has enabled organisations to use real-time data and information to their advantage. Digitisation has allowed data to come in a variety of forms ranging from images to sensors to signals. In addition, the declining costs of computing and improvements in computational storage, memory, processing, and bandwidth have allowed data analytics to become more economical (McAfee & Brynjolfsson 2012). Defined as datasets that are too large for traditional data-processing systems (Provost & Fawcett 2013), big data often requires new technologies to be processed. Through machine learning algorithms, which are based on statistics, trends can be extracted from the data and can be analysed accordingly.

According to the McKinsey Global Institute (James et al. 2011) ML will be one of the main drivers of the big data revolution. It is at the core of big data analytics as it provides a means to extract value from data and provide data driven insights, decisions and predictions (L’Heureux et al. 2017).

According to Shavlik and Dieterrich (1990), there are two ways in which a system can learn: the system can gain new knowledge from external sources or the system can modify itself by exploiting its current knowledge more effectively. These are known as empirical or inductive learning and speedup or skill acquisition learning, respectively.

2.2.2 Machine learning algorithms

Empirical or inductive learning is further divided into three broad categories of machine learning algorithms, namely supervised, unsupervised and reinforcement learning. Supervised learning refers to mapping an input variable to an output variable, where new input data can be used to predict corresponding output values based on an approximate mapping function. Given sufficient input data, the model will learn the behaviour for any data input entry (Maruster 2003).

Supervised learning problems can be further grouped into classification or regression problems. Classification occurs when an object needs to be assigned into a predefined group based on several attributes related to that specific object (Sathya & Abraham 2013) whereas regression problems use labelled data to make predictions.

Unsupervised learning, on the other hand, refers to input data which does not have corresponding output data, or predefined classes. Through the use of unsupervised learning algorithms, unlabelled data can be organised into clusters based on similar metrics (Maruster 2003).

Reinforcement learning is a more general learning algorithm which encompasses algorithms that do not fall within the supervised or unsupervised learning space. This type of algorithm can best be described as learning about learning (Sutton & Barto 1998). This learning task relies on monitoring the response of actions taken by trying to maximise the rewards, without direct instructions. The most distinguishing feature of this type of learning is trial and error. Unlike supervised learning, in practical applications, it is not always possible to obtain behavioural outputs that are correct and that fairly represent all situations which the agent may be exposed to (Sutton & Barto 1998). Hence, such a learning agent must be able to exploit previously learned behaviour while exploring actions that it has not selected before.

Most traditional supervised learning algorithms assume that training, test and future data fall within the same feature space and distribution and that there is sufficient training data to make predictions about future data (Pan & Yang 2010). In real-world applications, however, labelled training data is both expensive and difficult to obtain. More recently, the use of semi-supervised learning tasks has aimed to address the problem of limited labelled training data. Transfer learning algorithms use borrowed labelled data from a related domain and apply it to a domain of interest (Pan & Yang 2010).

Previous studies conducted with transfer learning reveal that transferring results show better performance than other approaches with different datasets (Huang et al. 2017). For example, ImageNet, which contains 22 000 classes and 15 million labelled images, is widely used as a dataset for transfer learning cases. A review of transfer learning conducted by Pan and Yang (2010) indicated that either the feature spaces between the domain data are different or the source and target tasks focus on different topics. Transfer

learning with deep CNNs can be applied to various fields. In the field of wildlife identification, research on the use of transfer learning remains limited.

2.2.3 Transfer learning

Although there are varying definitions of transfer learning, in the most basic sense, transfer learning can be defined as the ability to identify deep, subtle connections, which is the hallmark of human intelligence (Cook et al. 2013). All transfer learning problems depend on the assumption that there is a relationship between the source and target domains, for the successful transfer of knowledge from the source to the target (Cook et al. 2013). By using data gained from one machine learning problem to another related, yet different, problem, the need for large amounts of training data can be overcome (Becherer et al. 2017).

According to Torrey and Shavlik (2009), there are three measures by which transfer learning might improve learning:

1. *The initial performance* achieved in the target task using only transferred knowledge compared to the initial performance of an unlearned task;
2. *The amount of time* to fully learn the target task given the transferred knowledge compared to the amount of time to learn the task from scratch; and
3. *Final performance level* achievable in the target task compared to the final level without transfer of knowledge.

One of the major challenges in developing transfer methods is creating positive transfer between related tasks while avoiding negative transfer between less related tasks. There are three ways in which to reduce negative transfer namely by rejecting bad information to ensure that the performance is no worse than learning the target task without transfer or knowledge, choosing an appropriate source task and modelling task similarity (Torrey & Shavlik 2009).

A common assumption of many machine learning models is that the training and test data are from the same feature space and are of the same distribution (Pan & Yan 2010). In real-world applications, it can be expensive or impossible to collect the required training data and hence in these scenarios, transfer learning between task domains is desirable. The terms domain refers to knowledge about the environment in which the source and

target operated. In the study conducted by Pan et al. (2008), in examples when data can be easily outdated and the labelled data in one time period (the source domain) may not follow the same distribution in a later time period (the target domain), transfer learning can be used to address the problem when training and test data follow different distributions or are represented in different feature spaces.

According to Pan and Yang (2010), the following three questions must be answered during the process of transfer learning:

1. What to transfer – what part of knowledge can be transferred across domains or tasks.
2. When to transfer - in which situations transferring should or should not be done. If the source and target domain are unrelated, this can result in negative transfer.
3. How to transfer – identifying ways of transferring knowledge through existing algorithms and techniques.

Based on the different situations between source and target domains and tasks as shown in Figure 1, transfer learning can be categorised into three strategies (Pan & Yang 2010):

1. *Inductive transfer learning* takes place when the target task is different to the source task, irrespective if the source and target domains are different or not. Depending on whether the source domain contains labelled data or not, this can be further divided into multitask learning and self-taught learning, respectively. An example of inductive transfer learning can be found in competitive games involving robots in which transferring knowledge learned from one task to another is crucial to acquire the necessary skills to beat the opponent (Vilalta et al. 2011). Knowledge gained from the first activity must be transferred to the next activity.
2. In *transductive transfer learning*, the source and target tasks are the same, however, the source and target domains are different. In this category, the source domain contains labelled data, whereas the target domain contains no labelled data. An example of transductive learning can be found in text classification problems in which a fixed collection of documents needs to be organised (Pechyony 2008). In this scenario, each document is an example and since the collection is fixed, no other documents will appear. The full sample of documents

is known prior to the learning process and can be used to organise documents that were not originally in the full sample.

3. *Unsupervised transfer learning* is when the target task is different from but related to the source task. In this case, there is no labelled data available in both source and target domains during training. An example of unsupervised transfer learning can be found in biomedical applications. An example can be found in the study conducted by Chang et al. (2018), which demonstrated the effectiveness of convolutional sparse coding models for unsupervised learning on classification and learning tasks, specifically in biomedical applications where the scale of labelled data is very limited.

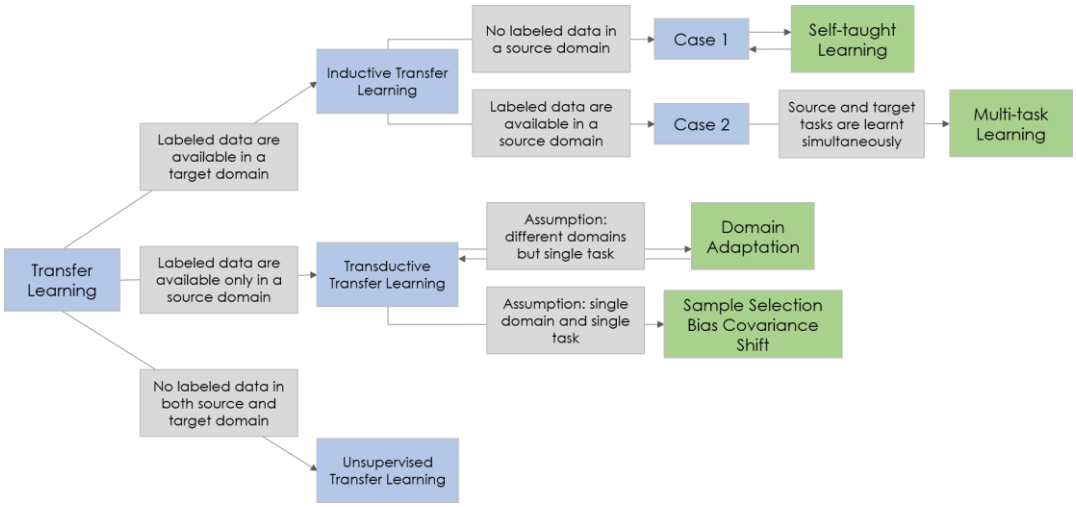


Figure 1: Transfer learning strategies (Pan & Yang 2010)

These transfer learning categories address the question of where transfer learning can be applied. To address the question of what to transfer, the following approaches can be used (Pan & Yang 2010):

- Instance-transfer re-weights some labelled data in the source domain for use in the target domain.
- Feature-representation transfer aims to reduce the difference between the source and the target domains by finding a "good" feature representation and by reducing the error.
- Parameter-transfer identifies shared parameters between the source and target domains.

- Relational-knowledge transfer maps relational knowledge between the source and target domains.

In image classification, transfer learning can be accomplished by using pre-trained models. Pre-trained models are models which are trained on large datasets and can be used as a benchmark to extract already learned features from images, using this knowledge as a starting point to learn a new task (Lundstrom 2017). This will be discussed in further detail in section 2.3.3.

2.3 Deep Learning

Deep learning is a branch of machine learning that teaches computers to learn from experience using neural networks (Moulder et al. 2017). Conventional machine learning methods were limited in their ability to process data in the raw form and for decades, required precise engineering and domain expertise to design feature extractors to transform the raw data into suitable representations to detect or classify the input data (LeCun et al. 2015).

2.3.1 History of neural networks

An artificial neural network (ANN) is a computational model that has been inspired by neurons in the human brain. The neurons in an ANN pass the information from one computational unit to further neurons. The processing of neurons in one-layer feed into the calculations of the next and the result of the final layer of the neural network can be interpreted for classification.

The basic concepts of the structure of a neural network date back to the McCulloch-Pitts neuron of the 1940's (McCulloch & Pitts 1943; Goodfellow et al. 2016). Through training and learning, the weights of the ANN can be adjusted through algorithms to obtain the desired output from the neural network.

As shown in Figure 2, the ANN consists of inputs which are multiplied by weights and computed by a mathematical function which determines the activation of the neuron itself (Lobova et al. 2007):

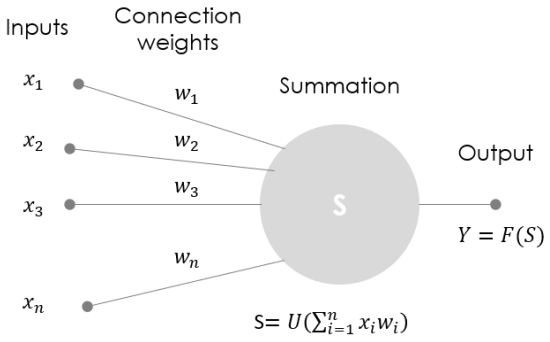


Figure 2: An artificial neuron (Lobova et al. 2007)

This model only allowed for binary outputs from each neuron, summing the input and comparing the result to zero, without defining an update rule to allow the neural network to adapt to new information (McCulloch & Pitts 1943).

The perceptron algorithm was introduced in the 1950s and it simplified the McCulloch-Pitts neuron to have weights on the connection as well as an update rule. The perceptron learning rule, however, was only capable of training networks with a single layer (Rosenblatt 1958). During the 1960's and 70's there was very little progress on the development of neural networks due to lack of funding as well as a combination of theoretical limitations and practical failures (Williams 2017).

Research conducted by Minsky and Papert (1969) showed that although single layer perceptrons could not classify and represent many common problems, it exhibited that a network with as little as two layers could model any function. Developments in the field of backpropagation, capable of training neural networks with multiple layers, saw renewed interest and advancements in the field of neural networks during the 1980s (Williams 2017). Despite these developments, lack of processing power and lack of techniques to resolve theoretical problems associated with the backpropagation error resulted in little progress in the field of neural networks once again.

Research conducted by Hinton et al. in 2006, however, proposed the Restricted Boltzmann Machines to train one layer at a time. This development paved the way for modern deep learning.

2.3.2 Overview of deep learning

The performance of machine learning algorithms depends on the representation of the data that it is presented with. Each piece of information from the dataset is known as a

feature. However, understanding which features to extract is often difficult. Representation learning aims to overcome this problem by discovering a suitable set of features for a specific task by separating factors of variation within the data (LeCun et al. 2015). In real-world applications, there are countless factors of variation that influence the observed dataset and extracting abstract features from raw data has thus proven to be difficult. Deep learning solves the problems of representation learning by enabling the computer to build complex concepts out of simpler concepts.

A convolutional neural network (CNN) is a specific type of ANN which is used in image recognition and classification, with the key concept being that CNN's automatically select the most important characteristics of the object (Cicero et al. 2016). This type of ANN is used to extract features from input images and convert them into lower dimensions without losing any of the feature characteristics. The main operations in the CNN are convolution, non-linearity, pooling and classification.

The primary purpose of the convolution layer is to extract features from an input image. Each convolution layer acts like a filter or feature detector that results in an activation (Goodfellow et al. 2016). Repeated application of the same filter, or kernel, to an input, results in a map of activations called a feature map, indicating the location and strength of a detected feature in an image input (Brownlee 2020).

Mathematically, the convolution step is a linear operation that involves the multiplication of a set of weights with the input through use of dot product element-wise multiplication (Dumoulin & Visin 2016). This is done by systematically applying the filter to a patch of input data from left to right and top to bottom. This results in a feature map which is then passed onto the next processing step. Different filters over the same image create different feature maps. A feature map is the output of a convolution operation and is created by sliding a filter, or kernel, over the input image (Goodfellow et al. 2016).

Feature maps are controlled by three parameters (Dumoulin & Visin 2016):

- Depth which corresponds to the number of filters used during convolution. The number of filters determines the number of feature maps.
- Stride which corresponds to the number of pixels by which the filter matrix slides over the input matrix. Having a larger stride produces smaller feature maps and vice versa.

- Zero-padding is used to ‘pad’ the input matrix with zeroes around the border so the filter can be applied to bordering elements of the input image matrix. This is used to control the size of the feature map.

The weights defining the feature map are referred to as shared weights and the bias defining the feature map are shared biases. Together, the shared weights and bias are defined as a kernel or filter (Nielsen 2015).

There are two aspects of computation that are used in convolution, namely location invariance and compositionality. Location invariance refers to the classification of an object in an image and is not concerned with the location of that object within the image (Britz 2015). Compositionality occurs when each filter composes a patch of lower-level features which are then used to represent higher-level features, thus building edges from pixels, shapes from edges and more complex objects from shapes (Britz 2015). Intuitively, averaging each pixel with its neighbouring values blurs the image and the difference between the averaged pixel and its neighbours detects edges (Britz 2015).

Lower-level convolution layers operate directly on the pixel values and will extract lower-level features such as lines and edges. The subsequent layers that operate on the output of the first layers combine the lower-level features, such as features that comprise to form shapes. In general, the more convolution steps that are present, the more complicated features the network will learn and be able to recognise.

Yosinski et al. (2014) conducted a study on the transferability of features in deep neural networks. Findings from the study indicated that transferability is negatively affected by specialisation of high-level neurons to their original task and co-adapted neurons, which refers to the dependability that some neurons have on each other. The point at which features are being transferred i.e. from the bottom, middle or top of the network, can influence which of these issues appear dominantly.

Non-linearity is introduced after each convolution step. Prior to the paper by Krizhevsky et al. (2012), the default activation functions were sigmoid and tanh functions (Goodfellow et al. 2016). However, rectified linear units or ReLUs are the default recommendation for activation functions (Glorot et al. 2011; He et al. 2015; Goodfellow et al. 2016). Specifically, ReLU is an element-wise operation that provides computational simplicity (Glorot et al. 2011), sparse representation and linearity (Goodfellow et al.

2016). ReLUs are capable of outputting true zero values, allowing the hidden layers in the network to contain one or more zero values, which is referred to as sparsity (Goodfellow et al. 2016). This property of non-linearity through sparsity can accelerate and simplify the model and expedite the convergence of the training procedure (Krizhevsky et al. 2012). According to Goodfellow et al. (2016) because rectifier functions behave like linear activation functions, there is no vanishing gradient effect, as gradients flow well on the active path of the neurons.

ReLU activation functions return a zero value for negative inputs, represented by the colour black on activation maps and return a value back for positive values, represented by the colour white. Adding more nodes in each layer provides the model with a greater ability to represent these interactions and non-linearity. Without rectification, what is propagated by the preceding layers is just noise in the input (Jarret et al. 2009).

Theoretically, all features extracted from the convolution layer can be input directly into the classifier, however, this will increase the computational requirements, especially for large images, and can lead to over-fitting (Al-Saffar et al. 2017). Pooling, which can be defined as aggregating the operations on characteristics of certain locations on the image (Al-Saffar et al. 2017), assists in reducing the dimensionality of features but making the representations invariant to small translations from the input (Goodfellow et al. 2016). This is especially useful if the presence and location of certain features are required. A pooling layer takes each feature map output from the convolution layer and prepares a condensed feature map (Nielsen 2015). Through translation invariance, even if the object in the original image produces a small translation, the classifier can still output the same classification result (Al-Saffar et al. 2017). Specifically, with max pooling, the largest element from the feature map is used, making the network more impervious to small transformations, distortions and translations.

The output of the convolution and pooling layers represent high-level features of the input image. Through several pairs of convolution and pooling layers, these features are then used to classify the input image into various classes based on the training dataset (Khan et al. 2016).

These training steps can be summarised in Figure 3.

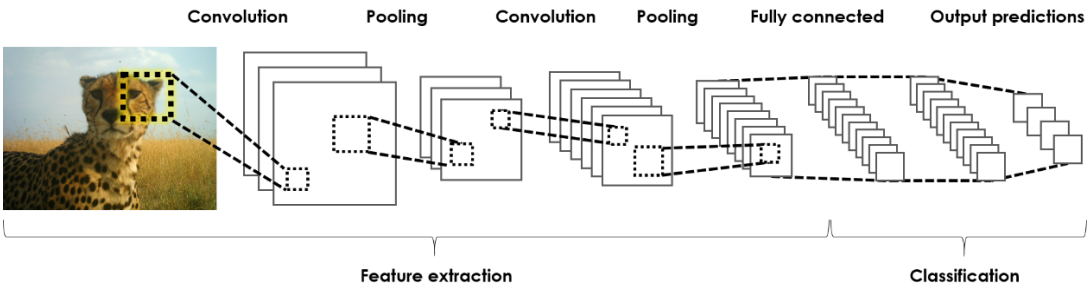


Figure 3: Training a CNN (adapted from Thenmozhi & Reddy 2019)

Research conducted by Urban et al. (2017) on shallow versus deep networks revealed that shallow networks that contain only one or two convolution layers are unable to achieve the accuracies comparable to deep networks of the same number of parameters. In addition, deep learning can execute feature engineering on its own and can identify features and correlate and combine them later to execute faster learning.

More recent CNN's have advanced the field of deep learning by making the basic CNN architecture deeper. However, all techniques require large amounts of training data to be effective. This is primarily due to CNN training being a highly complex optimisation problem that uses stochastic gradient descent (SGD) to find the minima of a loss function, which requires a large, labelled training dataset (Becherer et al. 2017). However, the use of transfer learning techniques can be used to overcome the lack of training data.

2.3.3 Deep network architectures

ImageNet is a dataset of over 15 million labelled high-resolutions images belonging to roughly 22 000 categories, which were collected from the internet and were labelled by volunteers using Amazon's Mechanical Turk crowd-sourcing tool (Krizhevsky et al. 2012). The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) is a benchmark for large-scale object recognition which aided in the development of various deep learning architectural advancements to address image classification and object recognition problems. Deep network architectures that participated in the ILSVRC with successful results include AlexNet, Visual Geometry Group (VGG), ResNet and GoogLeNet. These models have made significant contributions to the field of computer vision. Table 1 provides a summary of some of these deep network architectures.

Pretrained networks

Pre-trained models are models used to extract already learned features from images using this knowledge as a starting point to learn a new task (Lundstrom 2017). The majority of pretrained networks are trained on a subset of the ImageNet database. The purpose of pretrained networks include the following (MathWorks 2020a):

- Classification – applying a pretrained network to classify new images
- Feature extraction – using a pretrained network as a feature extractor and then using those features to train a new classifier
- Transfer learning – using layers from a network that has been trained on a large dataset and fine-turning this on a new dataset.

A comparison of the pretrained models as evaluated by Canziani et al. (2016) indicated several challenges when comparing the various architectures. The use of ensemble models increases the amount of computation required to achieve the published accuracy. The various models evaluate their validation images a different number of times and the reported accuracy is based on the specific sampling technique used (Canziani et al. 2016). Hence, a direct comparison of resource utilisation could not be done across the models due to these reasons (Canziani et al. 2016). The top-1 accuracy, or model accuracy with the highest probability, can be seen in Figure 4.

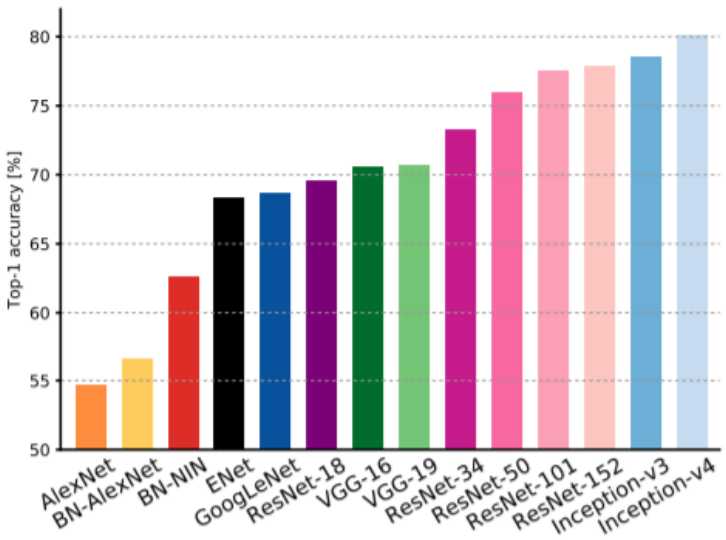


Figure 4: Top1 accuracy versus the network (Canziani et al. 2016)

The comparison of various architecture accuracies and their operations and size proportional to the number of parameters is shown in Figure 5.

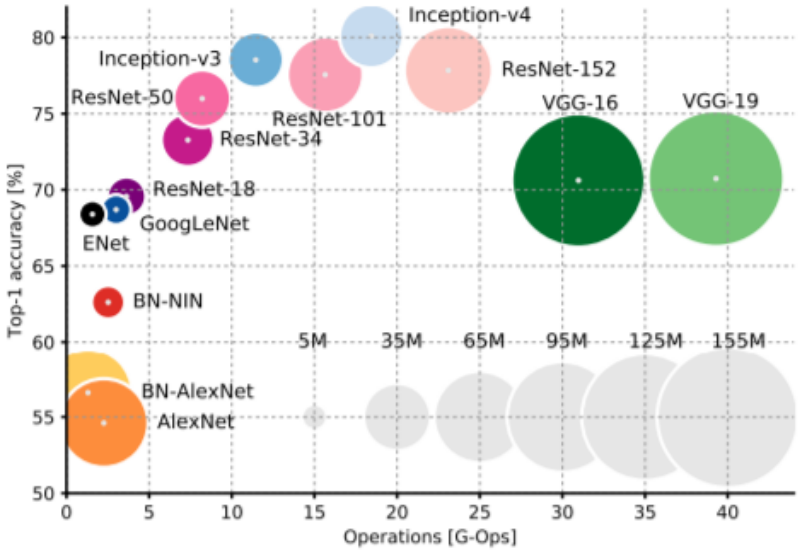


Figure 5: Top1 accuracy versus the network operations and size proportional to the parameters (Canziani et al. 2016)

AlexNet was the first convolution network that won the ILSVRC challenge in 2012 and achieved significant recognition accuracy against traditional machine learning approaches. This was a significant breakthrough in the field of machine learning and marked the point in history that paved the way for deep learning (Alom et al. 2018). As shown in Figure 6, AlexNet consists of eight learned layers – five convolutional and three fully connected layers. The input image layer is 227 x 227 x 3 and all images were resized to these dimensions. There are three max pooling layers and ReLU is applied after every convolution operation and fully connected layer (Krizhevsky et al. 2012). A dropout ratio of 0.50 is applied to the first and second fully connected layers, each of which contain 4 096 neurons (Thenmozhi & Reddy 2019). The final fully connected layer has 1 000 neurons which are configured to classify 1 000 classes of the ImageNet dataset. The loss is calculated via a softmax function.

GoogLeNet, the winner of ILSVRC in 2014, aimed to reduce computational complexity by increasing the width of the model, rather than the depth. This model proposed having filters with multiple sizes that can operate on the same level rather than stacking pooling layers in a sequential structure which can increase the chance of overfitting (Szegedy et

al. 2015). Through the use of an Inception module, rather than layering pooling and convolution operations sequentially, all these operations can be performed in parallel, each performing its own function in extracting information while still being computationally efficient (Szegedy et al. 2015).

Figure 7 shows that the GoogLeNet architecture consists of 22 layers that contain two convolution layers, four max pooling layers, nine inception modules which are stacked linearly and an average pooling layer which is applied at the end of the inception module (Thenmozhi & Reddy 2019). The inception modules perform convolution on three filter sizes, namely (1x1), (3x3) and (5x5). Between each inception module there are max pooling layers to send the convolution layer outputs to the next inception module. This results in a model that is wider. A 1x1 convolution operation is used in each inception module that performs dimension reduction before performing 3x3 and 5x5 convolutions. A dropout ratio of 0.40 is applied before the fully connected layer, which contains 1 000 neurons which are configured to classify 1 000 classes of the ImageNet dataset. The loss is calculated via a softmax function. The model uses an input image layer of 224 x 224 x 3.

The inception architecture was designed to perform well even under memory and computational budget constraints (Szegedy et al. 2016). Inception networks are feasible in big data scenarios such as Movshovitz-Attias et al. (2015) and Schroff et al. (2015) where large amounts of data need to be processed at a reasonable cost and where computational memory or capacity is limited. Although there are several versions of the Inception module for this architecture, the Inception-v3 model, consisting of 48 layers, achieved the highest top-1 and top-5 error rate (Szegedy et al. 2016). The inception architecture can be seen in Figure 8.

Recent evidence revealed that network depth is of crucial importance (He et al. 2016). When deeper networks start converging, a degradation problem occurs, known as the vanishing gradient problem: with the depth of the network increasing, the accuracy gets saturated and then degrades rapidly (He et al. 2016). Such degradation is not caused by overfitting and adding more layers to a deep model leads to higher training error, highlighting that not all neural network architectures are equally easy to optimise (He et al. 2016). In order to address the problem of degradation, a deep residual framework,

known as ResNet, was proposed by Microsoft Research that uses a technique called skip connections, which skips training from a few layers and connects directly to the output. If a layer hurts the performance of the model, it will be skipped by regularisation, resulting in very deep neural networks without the problem of degradation of vanishing gradients (He et al 2016). There are several ResNet networks namely ResNet-34, ResNet-50, ResNet-101 and ResNet-152 that consist of 34, 50, 101 and 152 parameter layers, respectively.

ResNet-101 consists of 101 parametrised layers with recurrent connections, which can be described as skip connections that are parallel to the normal convolution layers that allow the network to understand global features (Thenmozhi & Reddy 2019). As shown in Figure 9, the ResNet-101 model contains a 7x7 convolution layers with 64 kernels, a 3x3 max pooling layer, 33 residual building blocks, a 7x7 average pooling layer and a fully connected layer configured to classify 1 000 classes of the ImageNet dataset. The loss is calculated via a softmax function. This model uses an input image layer of 224 x 224 x 3.

The VGG model was the runner up for the 2014 ILSVRC. This model showed that the depth of a network is a critical component to achieving better recognition and classification accuracy (Alom et al. 2018). However, although VGG is widely used in many applications, it is by far the most expensive architecture in terms of computational requirements and number of parameters (Canziani et al. 2016). Its deployment on most modern-sized GPU's is challenging due to the large computational requirements in terms of memory and time and can become inefficient due to the large width of convolutional layers.

VGG-19 is a convolutional neural network that is 19 layers deep, consisting of 13 convolutional layers and three fully connected layers. The number of parameters is reduced using small 3x3 convolution filters (Thenmozhi & Reddy 2019). As shown in Figure 10, this model consists of convolution layers with consecutive 3x3 convolution operations and 2x2 max pooling layers, followed by fully connected layers and a softmax loss function. The VGG-19 model uses four 3x3 convolution layers in each convolution block. The model uses an input image layer of 224 x 224 x 3 and a dropout ratio of 0.50.

In this study, the following four models will be used: AlexNet, GoogLeNet, VGG-19 and ResNet-101. The number of parameters, operations and depth of architecture were considered when choosing these models to assist in answering the research question in this study.

Table 1: Summary of deep network architectures

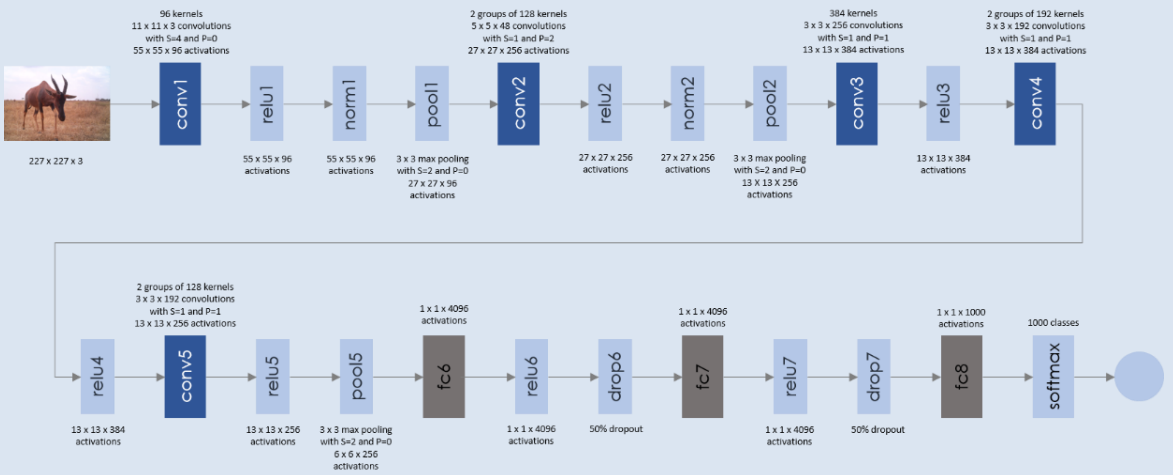
Network	Year	Number of Parameters	Top-5 Error Rate	Architecture illustration
AlexNet (Krizhevsky et al. 2012)	2012	60 million	15.3%	

Figure 6: AlexNet architecture (Adapted from Krizhevsky et al. 2012)

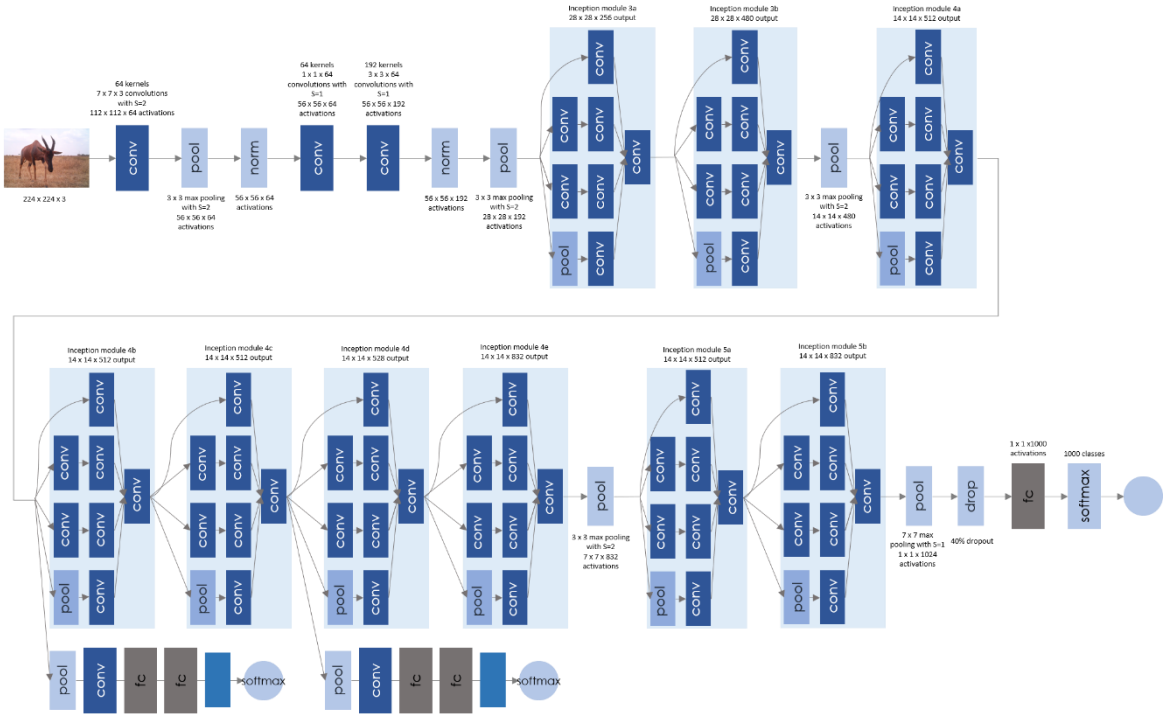
GoogLeNet (Szegedy et al. 2015)	2014	4 million	6.67%	

Figure 7: GoogLeNet architecture (Adapted from Szegedy et al. 2015)

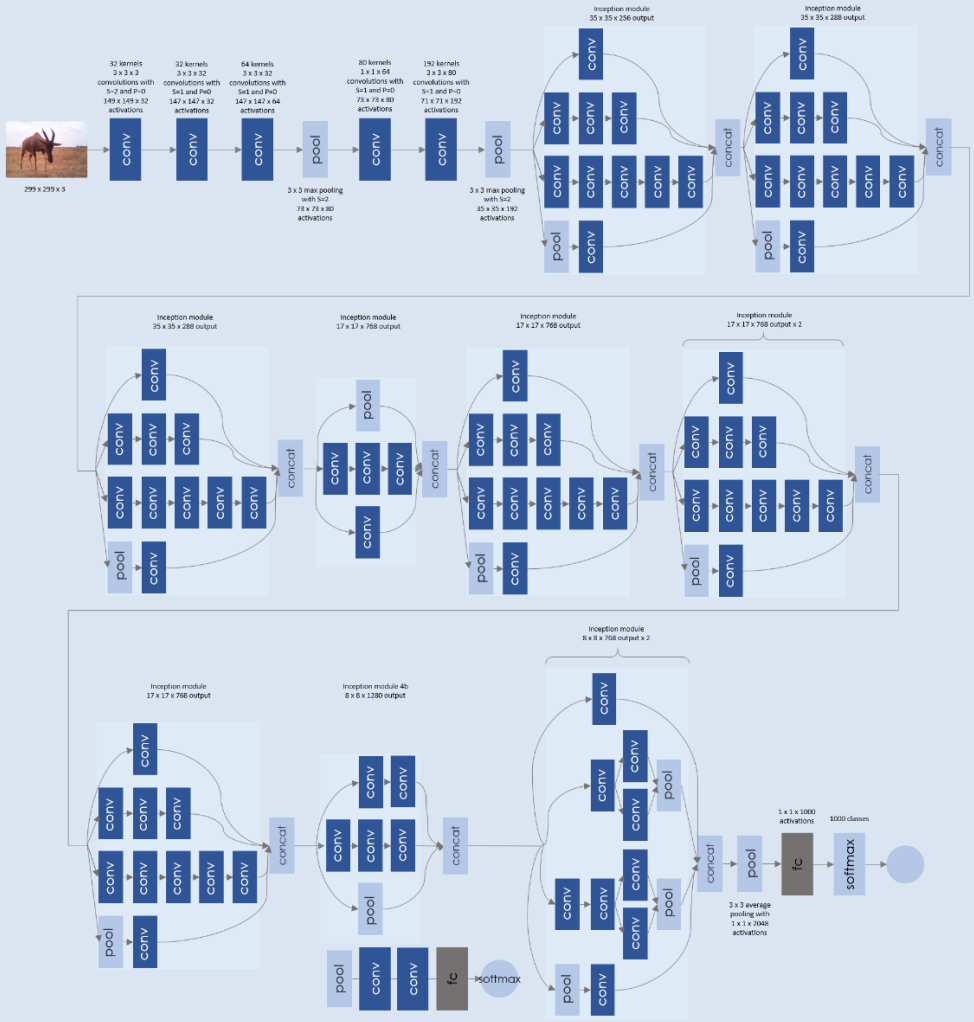
Inception-v3 (Szegedy et al. 2016)	2015	24 million	3.58%	

Figure 8: Inception-v3 architecture (Adapted from Szegedy et al. 2016)

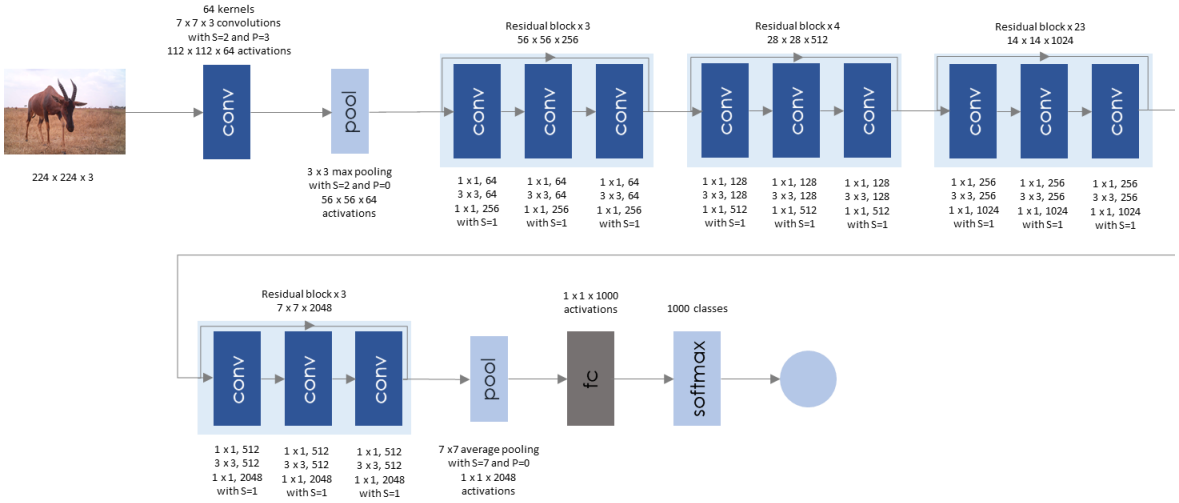
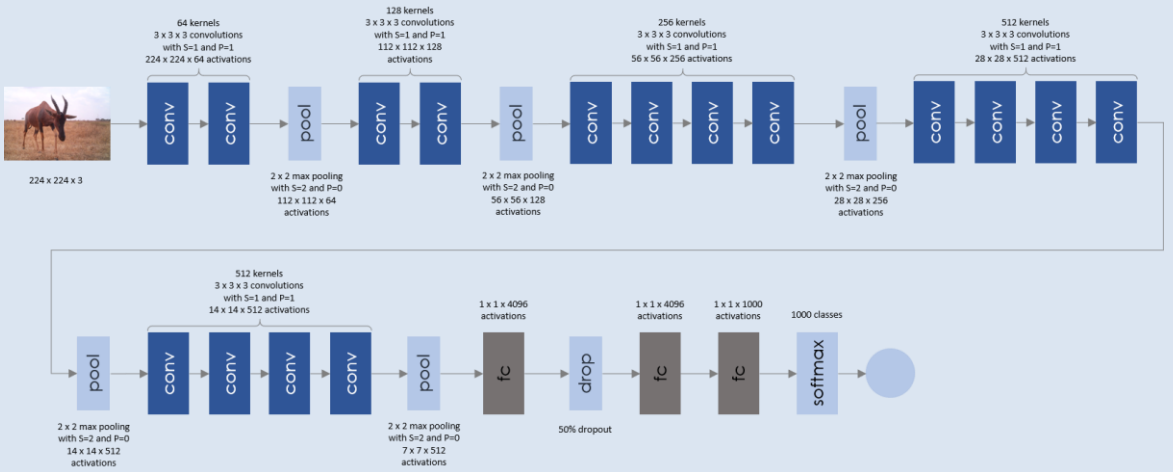
ResNet (He et al. 2016)	2015	11 million	3.57% (ensemble model)	

Figure 9: ResNet architecture (Adapted from He et al. 2016)

VGG-19 (Simonyan & Zisserman 2015)	2014	138 million	7.5%	 Figure 10: VGG-19 architecture (Adapted from Garcia-Gasulla et al. 2017)

2.3.4 Optimisation

Optimisation is one of the core components of machine learning. When training machine learning models, a loss function needs to be defined in order to measure the difference between the predicted and target values. The aim of most machine learning algorithms is to build an optimisation model and learn the parameters to maximise the objective function from the given data and minimise the loss function.

Optimisation methods can be divided into three main categories (Sun et al. 2019):

1. First-order optimisation methods – an algorithm that requires at least one first derivative or gradient, such as stochastic gradient descent methods;
2. High-order optimisation methods – an algorithm used to address problems where the objective function is non-linear; and
3. Heuristic derivative-free optimisation methods – algorithms that do not use derivative information to find an optimal solution as the derivative of the objective function may not exist or may not be easy to calculate.

Furthermore, optimisation can be divided into two types: convex optimisation which has one global minimum and non-convex optimisation which has 'valleys' of local minima that are not the global minimum. Non-convex optimisation has been one of the challenges in deep neural networks as it is prone to obtain locally optimal solutions rather than global optimal solutions (Sun et al. 2019).

The development of optimisation can be divided into three steps (Sun 2019):

1. Convergence - start running the algorithm and converge to a reasonable solution;
2. Convergence speed - make the algorithm converge as fast as possible; and
3. Global quality - ensure the algorithm converges to a solution with a low objective value i.e. the global minima.

Unfortunately, there is no consensus on choosing the "right" optimisation algorithm (Goodfellow et al. 2016). Although Schaul et al. (2014) presented a comparison of optimisation algorithms across a range of learning tasks, no single best algorithm emerged. Research conducted by Dogo et al. (2018) compared seven optimisation algorithms on three image classification datasets and their findings revealed that the

performance of each optimiser varied from each dataset confirming the effect of data type and data size on performance.

Although there are challenges associated with optimisation algorithms, over the years gradient descent optimisation algorithms have been developed for deep learning applications to mitigate challenges associated with 'vanilla' gradient descent methods (Dogo et al. 2018). In addition, the process of SGD usually finds a good set of weights quickly when compared to other optimisation techniques (Bottou & Bousquet 2007).

For the purposes of this research, first-order optimisation methods, specifically gradient descent will be considered further.

Gradient-descent variants

The idea of gradient descent methods is that variables update iteratively in the opposite direction of the gradients of the objective function, gradually converging to the optimal value (Sun et al. 2019). The learning rate determines the step size of each iteration and influences the number of iterations to reach the optimal value (Ruder 2016).

During the model execution, weights are initialised to obtain a starting point for convergence to occur. Gradient descent algorithms provide a means to identify which direction brings about the steepest decline in the value of a loss function. Optimally tuning the weights are key to producing accuracy in classification and prediction (Dogo et al. 2018). However, tuning and adjusting weights is dependent on setting the weights with the lowest loss function and the direction of the change in slope, also known as backpropagation.

The backpropagation procedure to calculate the gradient of an objective function with respect to the weights of a stack of layers is a practical application of the chain rule (LeCun et al. 2015). This derivative, or gradient, of the objective with respect to the input of the layer can be computed by working backwards from the gradient with respect to the output of that layer (LeCun et al. 2015). This method has proven to be simple and inexpensive as it simultaneously computes the partial derivates using one forward pass through the network followed by one backward pass through the network, backpropagating the error from the output (Nielsen 2015).

There are three variants of gradient descent (Dogo et al. 2018):

1. Batch gradient descent (BGD) which uses the entire dataset for training at the same time and then adjusts the weights;
2. Stochastic gradient descent (SGD) which uses a single training dataset at a time and iterates weight adjustment for each dataset; and
3. Mini-batch gradient descent which is a hybrid of BGD and SGD using more than one training example at a time.

However, there are challenges with 'vanilla' mini-batch gradient descent. The impact of learning rates including learning rate schedules and when to update the learning rate can result in slow convergence or divergence as well as numerous suboptimal local minima (Ruder 2016).

In addition, SGD oscillates across slopes while making hesitant progress along the bottom towards the local minimum (Sutton 1986). By adding a momentum term, the SGD can be accelerated in the relevant direction and oscillations can be dampened. The momentum term increases the dimensions whose gradients pull in the same direction and reduces updates for dimensions whose gradients change directions, resulting in faster convergence and reduced oscillation (Ruder 2016). According to Sun (2019), the benefit of momentum only applies when using batch methods i.e. all samples used at each iteration. The results in an algorithm called stochastic gradient descent with momentum, or SGDm.

The reasons for using SGDm rather than standard gradient descent methods are the memory constraints and faster convergence (Sun 2019). A single CPU or GPU cannot load all samples into memory for calculating the full gradient, therefore, loading a mini batch of samples at each iteration is more reasonable.

2.3.5 Training & learning

Although optimisation aims to reduce the loss of the objective function, improving the overall accuracy and performance of the model can be achieved through several methods.

Data optimisation

Many datasets related to medical diagnoses, natural phenomena or demographics are naturally imbalanced, where the number of samples in one class is significantly higher compared to the other classes (Al-Stouhi & Reddy 2016). In the case of wildlife

classification, if the model only predicts frequent classes such as gazelle, zebra and wildebeest, the model might still achieve an overall high accuracy without learning rare classes. When class imbalance exists within the training data due to increased probability of frequent instances, over-classification of the majority group can occur. Methods for handling class imbalance can be grouped into three categories (Johnson & Khoshgoftaar 2019):

1. *Data-level techniques* attempt to reduce the level of imbalance through data sampling methods. These techniques include over-sampling and under-sampling, which both result in changing the training data distribution. Under-sampling voluntarily discards data to reduce the total amount of information that the model has to learn from by randomly deleting entries in the majority class. Over-sampling aims to increase the size of the training dataset by randomly duplicating examples in the minority classes. However, these techniques have been found to cause underfitting and overfitting respectively, which can be addressed through regularisation. This will be discussed in the next section: Overfitting & regularisation. In the study conducted by Norouzzadeh et al. (2017), emphasis sampling was used, which is a technique used where the samples that the network fails are presented to the network to increase the probability that the network will retrain on them and make changes to learn.
2. *Algorithm-level methods* which are implemented using weight or cost schema aim to reduce bias towards the majority group. Unlike data sampling methods, algorithm methods do not alter the training data distribution. This technique considers a penalty or weight to reduce the bias toward the negative class. This technique will be discussed in more detail in the section: Overfitting & regularisation.
3. *Hybrid approaches* combine data-level and algorithm-level methods by reducing class noise and imbalance and then applying cost-sensitive learning or weight initialisation to further reduce the bias toward the majority group.

Overfitting & regularisation

In general, regularisation techniques prevent the model from becoming too complex and familiar with the training data, thus reducing the tendency to overfit the data (Williams 2017). Techniques to address over-fitting include the following:

- *Dropout* is a technique in which a random set of neurons from each hidden layer is excluded from classification and during updating the training pass through the data (Williams 2017). By dropping a unit out, it is temporarily removed from the network, with its incoming and outgoing connections. A fixed fraction, or "dropout", is specified so that each output of a hidden neuron with a probability greater than the dropout value will not contribute to the forward pass and will not participate in backpropagation (Krizhevsky et al. 2012). This ensures that every time an input is presented, the network will sample a different architecture, although all architectures will share the weights. This technique discourages co-adaptations in which neurons rely on the presence of other neurons and the remaining neurons are forced to learn more robust features that are useful together with other random subsets of other neurons (Krizhevsky et al. 2012).
- *L2 regularisation* uses the squared sum of the weights as a penalty to the loss function (Williams 2017). By adding an extra term to the loss function, called a regularisation term, this term penalises large weight values so that the model prefers to learn small weights, while still minimising the loss function (Nielsen 2015). The optimisation algorithm function will contain two terms: a loss term, to measure how well the model fits the data, and a regularisation term, a measure of model complexity. If the model's complexity is a function of weights, feature weights with larger values are more complex than feature weights with low values, hence outlier weights will have a huge impact on model complexity (Chollet 2017).
- *Artificially expanding the training data* can reduce overfitting by using data augmentation techniques on the dataset. A generally accepted notion is that bigger datasets result in better deep learning and by using intelligent over-sampling techniques to create new instances of data rather than random over-sampling, the size of the dataset can be increased (Shorten & Khoshgoftaar 2019). Methods of data augmentation include the following (Mikolajczyk & Grochowski 2018):

- Traditional data augmentation techniques including geometric distortions or deformations including cropping, rotation, padding, flipping and colour modifications.
- Generative adversarial networks (GAN) are ML models in which two neural networks compete with one another, with one generator maximising the loss function and one minimising it. It is still unclear on how to quantitively evaluate these models as well as other limitations including generating images that are too flat and not conforming to reality.
- Texture transfer applies contouring, shading, lines, strokes and other visual attributes to a new image based on the original image. However, the difficulty of this method is applying a given texture to the new image in such a way that the visual attributes of the original image are still visible.

Traditional techniques are more widely used and have proven to be more successful in improving model accuracy than techniques such as GAN or texture transfers (Wang & Perez 2017).

Hyper-parameter improvement

Hyperparameters are values that must be initialised in the network and that are used to control the learning process. They are important for machine learning algorithms as they directly control the behaviour of training algorithms and significantly affect the performance of ML models (Wu, J. et al. 2019). Hyperparameters differ from internal model parameters as they cannot be learned from the data during the model training phase and instead, are initialised prior to the training phase (Wu, J. et al. 2019). The process of hyperparameter improvement or tuning aims to find the set of hyperparameter values which can achieve the best performance in a reasonable amount of time.

Although there are no direct methods to identifying the best set of hyperparameters, hyperparameters are determined by finding the optimal value for a model across multiple datasets by either using manual search or automatic search methods (Wu, J. et al. 2019).

Below is a list of hyperparameters used in SGD algorithms:

- *Learning rate*s inform the optimiser the step size at each iteration while moving toward the global minima. If the learning rate is too low, training may be more reliable, but optimisation will take longer as the step towards the minimum will

be too small. On the other hand, a learning rate that is too high may not converge at all and may diverge, as the optimiser might overshoot the minimum (Bengio 2012).

- *Loss functions*, in its simplest terms, are used to evaluate how well the algorithm models the dataset. Commonly used in classification problems, log loss via logistic regression can be used to reduce model complexity through L2 regularisation or by early stopping, which limits the number of training sets or the learning rate.
- *Batch size* refers to the number of training examples used in one iteration before updating the model's internal parameters. An epoch consists of one or more batches and defines the number of complete passes through the training dataset. According to Bengio (2012), this hyperparameter affects training time and not so much test performance and choice of batch size is relatively independent of other hyperparameters (Nielsen 2015).
- *Activation functions* are required when using SGD with backpropagation to model nonlinear functions and complex relationships. The prediction accuracy of the model is defined by the type of activation function used. Rectified linear units (ReLU) are more efficient than other functions because all neurons are not activated at the same time but rather only a certain number of neurons are activated at a time (Sharma et al. 2020). Recent research on image recognition have found considerable benefit in using ReLU units through the network (Nair & Hinton 2010; Nielsen 2015). In addition, deep convolutional neural networks with ReLUs train several times faster than their equivalents (Krizhevsky et al. 2012). The scope of this project considers the ReLU activation function during model development.

Ensemble algorithms

Ensemble algorithms use multiple machine learning algorithms to improve the predictive performance of models, without using any one specific machine learning algorithm. These algorithms have been known to outperform individual classifiers as the tendency of any single model to overfit the data is reduced (Becherer et al. 2017).

Evaluation metrics

In typical classification problems, evaluation metrics are employed in two stages, namely during the training stage and the testing stage. During the training stage, evaluation metrics are used to optimise the classification algorithm, while during the testing stage, evaluation metrics are used to measure the effectiveness of the classification algorithm (Hossin & Sulaiman 2015).

Torrey and Shavlik (2009) identified three common measures by which transfer might improve learning:

1. The initial performance achievable in the target task using only transferred knowledge;
2. The amount of time it takes to learn the target task using transferred knowledge compared to the time to learn it from scratch; and
3. The final performance level achievable in the target task compared to the performance without transfer.

Accuracy of error rate is one of the most common metrics used in practice for classification algorithms (Hossin & Sulaiman 2015). However, using this evaluation metric on its own can produce less distinctive and discriminable values, being less informative and favouring the minority class instances (Chawla et al. 2004; Garcia & Herrera 2008; Hossin et al. 2011; Rawana et al. 2006; Wilson 2001 as cited in Hossin & Sulaiman 2015).

When selecting and discriminating the optimal solution during classification training, the significance trade-off between classes is elemental to ensure every class is represented and this becomes more important when the classes in the dataset are imbalanced (Hossin & Sulaiman 2015).

A confusion matrix is one such evaluation metric. A binary confusion matrix, shown in Table 2, will be used for illustration purposes.

Table 2: Confusion matrix for binary classification (Hossin & Sulaiman 2015)

	Actual Positive Class	**Actual Negative Class**
Predicted Positive Class	True positive (TP)	False negative (FN)
Predicted Negative Class	False positive (FP)	True negative (TN)

Although there are a number of threshold metrics for evaluating classification problems as described by Hossin and Sulaiman (2015), the metrics used in this study are shown in Table 3. This is consistent with the metrics used in studies conducted by Norouzzadeh et al. (2017).

Table 3: Evaluation metrics

Metric	**Formula**	**Description**
Accuracy (acc)	$acc = \frac{TP + TN}{TP + FP + TN + FN}$	The ratio of correct predictions over the total number of instances
Error Rate (err)	$err = \frac{FP + FN}{TP + FP + TN + FN}$	The ratio of incorrect predictions over the total number of instances
Precision (p)	$p = \frac{TP}{TP + FP}$	A measure of the positive patterns that are correctly predicted from the total predicted patterns in a positive class (Hossin & Sulaiman 2015)
Recall (r)	$r = \frac{TP}{TP + TN}$	A measure of the fraction of positive patterns that are correctly classified (Hossin & Sulaiman 2015)
F-Measure (FM)	$FM = \frac{2\ x\ p\ x\ r}{p + r}$	The mean between recall and precision values

In addition to these evaluation metrics, the top-N accuracy metric is another measure of accuracy used in classification problems. This metric is a common choice in image recognition tasks, such as the ILSVRC (Deng et al. 2009). The top-N accuracy is a measure of how often the predicted class falls within the top N values of the softmax distribution.

The top-1 accuracy rate is the ratio of images whose ground truth labels are exactly the prediction class with the highest probability while the top-5 accuracy indicates the ratio of images whose ground truth category is within the top five prediction categories, sorted by probability (Liu et al. 2017).

This method of accuracy provides insight into how the model works. For example, a low top-1 accuracy might suggest that the model has not learned enough from the training dataset. However, if the top-N accuracy increases it might suggest that fine-tuning could result in a higher top-1 accuracy.

In addition to the mathematical evaluation metrics, graphical methods will also be used in the form of training and validation learning and loss curves. The loss is an indication of how bad the model is able to make predictions on a single example, or batch. The goal is to have weights and biases set to minimise the overall loss. The loss is calculated on both the training and validation dataset. The loss is the sum of errors made in a particular dataset. The accuracy of the model is determined by the model's parameters and the accuracy percentage is the fraction of misclassified examples in the particular dataset. These curves can provide an indication if the model is overfitting, underfitting or has a good fit (Brownlee 2019b).

In this study, accuracy in the validation set was used as a performance metric. The top-1 and top-5 performance metrics were used to determine how well the models were able to correctly classify the species from the test dataset.

2.3.6 Understanding CNN's

While there have been considerable improvements in the knowledge of creating architectures and learning algorithms, the understanding of how these neural networks operate has lagged behind due to the large number of interacting, non-linear components that they consist of (Yosinski et al. 2015; Mahendran & Vedaldi 2015; Zeiler & Fergus 2014). Larger, deep networks are even more difficult to study due to their size and the number of parameters they consist of. Visualising convolution layers provide intuitions about their effects and functions (Yosinski et al. 2015).

Understanding what computations are performed at each layer in a CNN can be done by studying each layer as a group and investigating the type of computation performance by

a set of neurons on each layer (Yosinski et al. 2015; Mahendran & Vedaldi 2015). This is informative because the neurons in each layer interact with each other to pass information on to higher layers, contributing to the entire function of the layer (Yosinski et al. 2015). A second approach to try to interpret the function computed by the neurons in the various layers is to display images from the training and test dataset that cause high or low activations for each individual unit (Yosinski et al. 2015) or by highlighting the portion of the image that is responsible for firing each neuron (Zeiler & Fergus 2014). Visualising features to gain intuition about the network has become more common, however, this is limited to shallower layers (Zeiler & Fergus 2014). In deeper layers, however, this is not possible, as there are limited methods of interpreting activity (Erhan et al. 2009; Zeiler & Fergus 2014).

By visualising a combination of neurons instead of a single neuron, one can see what the network understands of the input image as a whole, instead of what the network detects at each position (Olah et al. 2018). Versions of the data corresponding to different layers are referred to as representations, and at every layer the network discovers a new representation of the input (Olah et al. 2018). Working across the layers provides an indication of how the network's understanding evolves from detecting edges to more sophisticated shapes and object parts.

Each layer of the network deals with features of different levels of abstraction. For example, lower-level layers produce strokes and ornament-like patterns, as those layers are sensitive to basic features such as edge detection and orientations. Higher level layers identify more complex features or whole objects (Mordvintsev et al. 2015). Feature maps allow one to visually identify what the high-level 'ideas' were that lead to the final output classification. Although this allows one to visualise what the network detects, it does not provide an explanation as to how the network assembles these concepts that lead to the final decision.

Research conducted by Simonyan et al. (2014), Nguyen et al. (2015), Szegedy et al. (2014) and Goodfellow et al. (2016) shows that the optimisation process tends to produce images that do not resemble natural images, and instead, consist of a collection of 'hacks' that happen to cause high or low activations, extreme pixel values, high frequency patterns and copies of common themes without any structure (Yosinski et al. 2014). This

can result in images being misclassified due to imperceptibly small changes producing ‘fooling examples’ (Nguyen et al. 2015).

Unlike activation maps that are difficult to interpret due to the abstract vectors which are contained within them, feature visualisation is expressed using descriptions from a ‘semantic dictionary’ (Olah et al. 2018). This allows activations to be mapped to iconic representations, that are similar to human ideas such as “fur”, “pointy ears” and “snout” (Olah et al. 2018). Feature visualisations aim to answer questions about what a network, or parts of the network, are looking for and what neurons react to, how they interact and how to improve optimisation (Yosinski et al. 2015; Olah et al. 2017).

This topic is beyond the scope of this research and will not be discussed further in detail.

2.4 Related Work on Wildlife Imaging, Monitoring, and Identification

Tracking animal footprints and the presence of dung, nests, trails, and calls are some of the oldest methods of identifying an animal’s presence (Silveira et al. 2003). Over the past couple of years, the popularity of non-invasive camera traps has grown in the field of population studies of species. Historically, such technology was used to track rare species in understudied areas (Silveira et al. 2003). However, advancements in computing and technology have allowed this tool to be used in various applications.

2.4.1 Traditional approaches

Citizen science, or the use of volunteers who collect and/or process data as part of a scientific study, plays an important role in ecology and environmental sciences (Silverton 2009). Volunteer participation in ecological studies has been the basis of research aimed at conservation of biodiversity and is currently having the greatest influence when monitoring biodiversity on large geographical scales (Dickinson et al. 2010).

Through the advent of technology and developments in digital techniques and computing, volunteers can remotely participate in these projects and collect or process data into central databases (Dickinson et al. 2010). Supervised machine learning algorithms require large amounts of labelled data, hence human-labelled datasets are valuable resources (Nguyen et al. 2017).

However, challenges associated with citizen science data include error and bias in observer quality. Specifically, with ecological studies, observer quality, or lack of

experience and training, can lead to errors in classification of data (Dickinson et al. 2010). Studies have also shown that the age of the observers can affect the data quality, with older observers being able to correctly identify more species than younger observers. In general, there is a wider need for data validation and data quality which raises the need for professional training, task training, experience with the task, observer age, training duration, training method and variation in species detection (Dickinson et al. 2010). Therefore, standard methods and protocols need to be developed for citizen science projects. Ecologists working on long-term monitoring data and projects also need to consider issues such as whether cognitive bias leads to chronic data bias, whether certain attributes or species are particularly difficult to measure or tell apart or how much experience is required before the data is deemed reliable (Dickinson et al. 2010).

While the need for large amounts of data is ever-increasing, there are four main challenges with traditional approaches to wildlife cataloguing and classification (Harris et al. 2010):

1. Image identification lags image acquisition as cataloguing images is a slow process;
2. User entry is tedious and causes errors;
3. Inconsistent filing and naming conventions complicate data retrieval and use; and
4. The rate of image acquisition and management from existing camera traps slows the deployment of new camera traps in the field.

The challenges associated with citizen science data and the lack of standard protocols and methods in this field further emphasize the need for ML techniques in wildlife classification problems.

2.4.2 Challenges with camera trapping

The challenges associated with camera trapping can be classified into three main categories: environmental conditions, animal behaviour-related and hardware limitations (Villa et al. 2017):

1. Environmental conditions refer to how these conditions affect the quality of the camera trap image. As camera traps remain in the wild for long periods of time, many objects such as plants, fallen trees and raindrops on the camera lens, can

occlude the animal. Day and night illumination conditions also cause overexposure on the images, as shown in Figure 11.

2. Although a common approach in camera trapping is to place the camera pointing towards the natural path of where the animals are expected to pass through, the behaviour of the animals is often unpredictable resulting in context occlusion. This includes animals being too close to the camera which hide important features used to recognise the species and animal movement, as seen in Figure 12. The general behaviour and movement of animals often result in multiple species being present in a single frame as well as some images having no animals in the frame at all, as seen in Figure 13. In addition, intra-species appearance, as shown in Figure 14 exists among similar species, especially when partial or unexpected poses are in the frame. Images with multiple animal species also exists as symbiotic relationships occur between species during activities such as grazing and movement. This can be seen in Figure 15.



Figure 11: Images with different exposure levels and day and night-time images

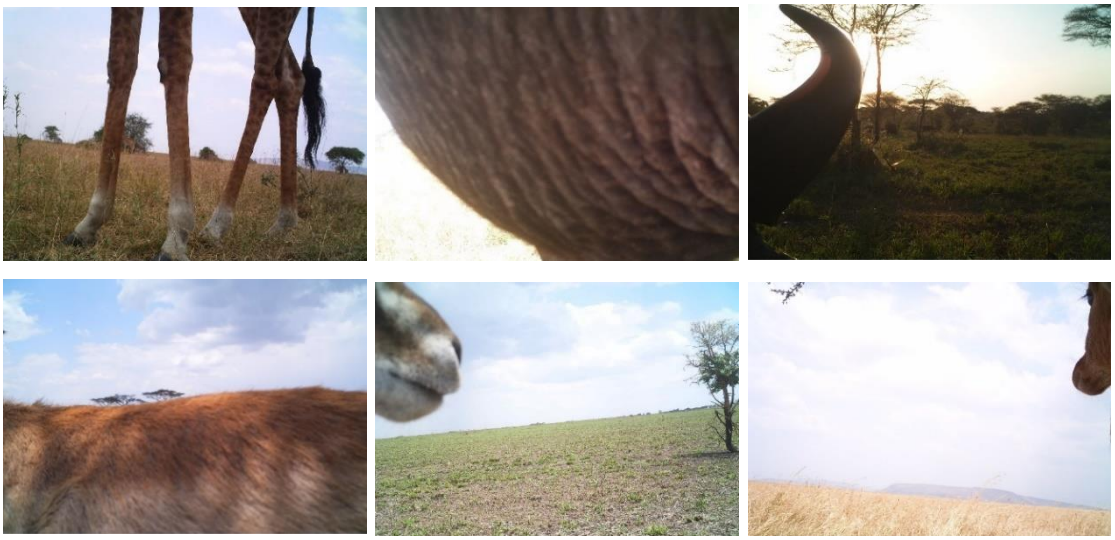


Figure 12: Images with partial of the animal species in the frame



Figure 13: Empty images with no animal species in the frame



Figure 14: Intra-species images with animals from the same species in a frame



Figure 15: Multiple animal species appear in a single frame

3. Camera trapping hardware and specifications include camera resolution, trigger sensors, frames per second and night illumination. The hardware and selected parameters affect the clarity of the images taken.

Images with the entire body of the animal are a rare case, as shown in Figure 16, however, an automatic image recognition system must be able to deal with any of the abovementioned conditions.



Figure 16: Full-body images of wildlife species

2.4.3 Deep learning approaches

Research conducted by Nguyen et al. (2017) confirmed the time consuming and costly nature of analysing wildlife images captured from camera traps. Wildlife monitoring citizen science projects such as Snapshot Serengeti and Wildlife Spotter recruited volunteers to assist in analysing and identifying these images. However, despite 96.6% of images being accurately identified by volunteers on the Snapshot Serengeti project, thousands of photos were inconsistently labelled, taking more than two months to interpret images gathered over a six-month period with a group of 28 000 registered and 40 000 unregistered volunteers (Swanson & Kosmala 2015).

Snapshot Serengeti (SS) is one of the world's largest collections of camera-trap images that has been generated (Swanson & Kosmala 2015). This project deployed 225 camera traps across a 1 125 km^2 area in the Serengeti National Part in Tanzania, from 2010 to 2013, resulting in over 1.2 million image sets, with each image set containing between one and three photographs taken in a single burst (Swanson & Kosmala 2015). Similarly, another project, Wildlife Spotter, contains a collection of millions of wildlife images in regions of Australia. However, large amounts of data and varying image quality all affect the analysis process (Nguyen et al. 2017). According to Nguyen et al. (2017), image quality can also refer to images with a partial body of the animal in the frame, the animal being too far from the camera, images in grayscale as well as images with no animals in the frame.

Previous research conducted by Swinnen et al. (2014), proposed a method to distinguish between different camera-trap recordings by detecting variations in the pixel values from frame to frame. In research conducted by Yu et al. (2013), images were manually cropped and only images that contained the whole animal body were selected. In addition, all empty frames, or images without animals, were removed from the dataset. Using this conditional dataset, they obtained an accuracy of 82% for classifying 18 animal species consisting of 7 196 images.

Chen et al. (2014) used a deep CNN to classify 20 animal species consisting of approximately 20 000 images. Their data augmentation used an automatic segmentation algorithm for cropping animals from the images. However, this model obtained an accuracy of only 38.31%. Research conducted by Villa et al. (2017) removed all species from the SS dataset that had a small number of images to address the issue of unbalanced datasets. Of the 48 classes available in the SS dataset, Villa et al. (2017) used 26. This experiment obtained an accuracy of 57% (Norouzzadeh et al. 2017). In another experiment, Villa et al. (2017), used a simplified version of the SS dataset containing only 33 000 images that were manually cropped and specifically chosen to have animals in the foreground.

Although Norouzzadeh et al. (2017) used transfer learning as part of the various experiments that were conducted, their results indicated that transfer learning made little difference on the task when training on the full dataset and hence did not use it for simplicity of their experiments.

For the purposes of this research, a deep learning solution that does not use manual intervention in data augmentation and preparation will be presented.

3 RESEARCH METHODOLOGY

This chapter highlights the research methodology employed in this study.

3.1 Introduction

According to Goodfellow et al. (2016), correct application of an algorithm depends on mastering a simple methodology. Deciding whether to gather more data, increase model capacity, add or remove regularising features, improving model optimisation or debugging methods are important operations which need to be considered and implemented in a systematic fashion, rather than through speculating or guessing.

3.2 Theoretical Foundation

The research methodology used in this study followed the approach outlined in Goodfellow et al. (2016) as adapted from Ng (2015). The practical design process is depicted in Figure 17:

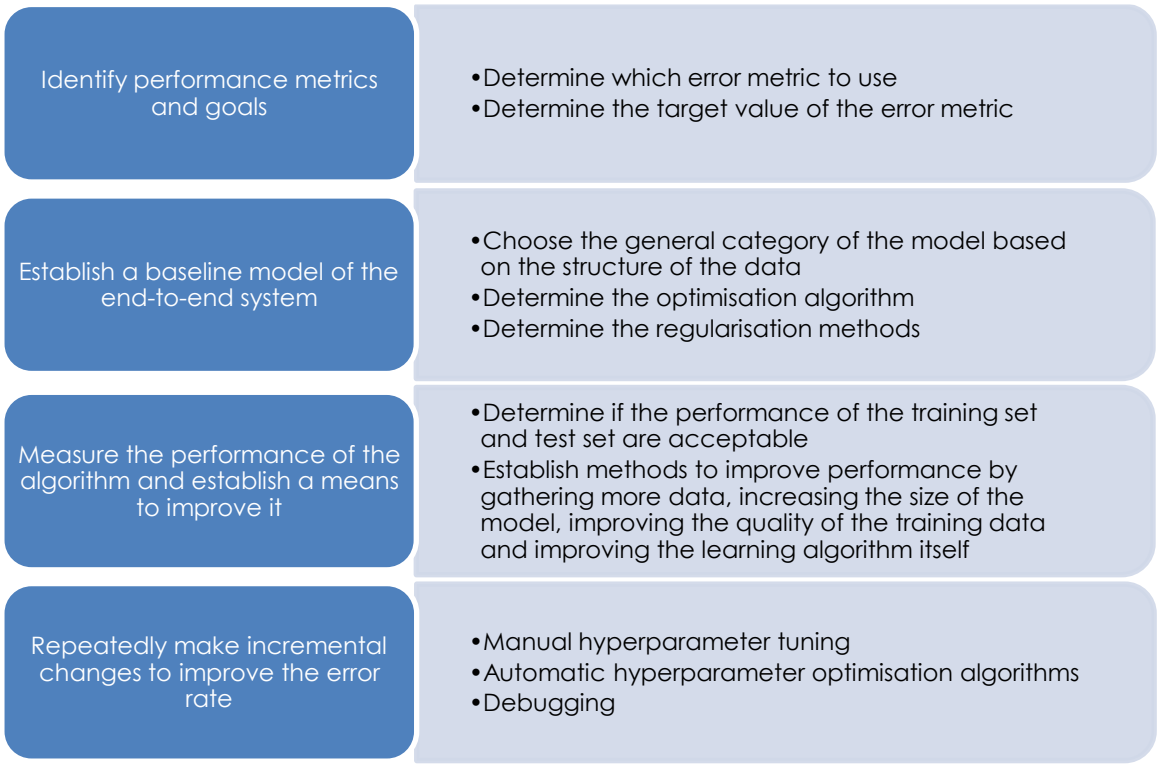


Figure 17: Practical design process adapted from Goodfellow et al. (2016)

Using this approach as a guideline, the training and performance metrics were identified, as discussed in section 2.3.5. In this study, accuracy in the validation set was used as a performance metric. The top-1 and top-5 performance metrics were used to determine

how well the models were able to correctly classify the species from the test set. The baseline models were established using pretrained networks. The optimisation, regularisation methods and batch size were kept constant across all the models. Using the accuracy of the validation dataset as a performance metric, the models were deemed to be acceptable and test results were compared across the models. Using additional methods to balance the rare classes, the models were run again, and test results were compared to determine the improvement in prediction accuracy. Details of the baseline models from the pretrained networks as well as the layers which fine-tuned and frozen are described in detail in section 3.3.6.

3.3 Research Design

This section explains the design framework as well as the detail on the datasets and model development.

3.3.1 Research instruments

Large training datasets and network structures require increasingly greater processing power and memory resources. Deep learning is faster when using a high-performance GPU, or graphical processing unit, to accelerate training and utilise parallel computing to train large neural networks (Moulder et al. 2017).

In 2007, NVIDIA corporation released the Computer Unified Device Architecture (CUDA) application programming interface (API) to enable direct programming of their GPUs. GPUs are designed specifically for graphical display, which rely on large amounts of calculations to run in parallel. These processors have many cores, and CUDA takes advantage of the number of cores in GPUs by running highly parallelisable tasks (Williams 2017). Large matrix operations used in deep learning are examples of such tasks. A study conducted by Raina et al. (2009) confirmed that implementations of a GPU are 70 times faster than standard CPU implementations for large models.

For the purposes of this study, two GPU's with CUDA capability and an NVIDIA graphics card were deployed to run the various experiments. A computer with a GeForce GTX 1650 GPU and another computer with a Quadro P5000 GPU were used, with the specifications of each GPU device indicated in Figure 18 and Figure 19, respectively.

```
CUDADevice with properties:

                      Name: 'GeForce GTX 1650'
                     Index: 1
      ComputeCapability: '7.5'
         SupportsDouble: 1
          DriverVersion: 10.1000
         ToolkitVersion: 10.1000
    MaxThreadsPerBlock: 1024
      MaxShmemPerBlock: 49152
    MaxThreadBlockSize: [1024 1024 64]
           MaxGridSize: [2.1475e+09 65535 65535]
             SIMDWidth: 32
           TotalMemory: 4.2950e+09
       AvailableMemory: 2.5098e+09
  MultiprocessorCount: 14
         ClockRateKHz: 1740000
          ComputeMode: 'Default'
  GPUOverlapsTransfers: 1
KernelExecutionTimeout: 1
       CanMapHostMemory: 1
        DeviceSupported: 1
         DeviceSelected: 1
```

Figure 18: GPU device for computer 1

```
CUDADevice with properties:

                      Name: 'Quadro P5000'
                     Index: 1
      ComputeCapability: '6.1'
         SupportsDouble: 1
          DriverVersion: 11
         ToolkitVersion: 10.2000
    MaxThreadsPerBlock: 1024
      MaxShmemPerBlock: 49152
    MaxThreadBlockSize: [1024 1024 64]
           MaxGridSize: [2.1475e+09 65535 65535]
             SIMDWidth: 32
           TotalMemory: 1.7180e+10
       AvailableMemory: 1.6034e+10
  MultiprocessorCount: 20
         ClockRateKHz: 1733500
          ComputeMode: 'Default'
  GPUOverlapsTransfers: 1
KernelExecutionTimeout: 1
       CanMapHostMemory: 1
        DeviceSupported: 1
         DeviceSelected: 1
```

Figure 19: GPU device for computer 2

The model was built using MATLAB, version R2020a. Developed by mathematical computing software company, MathWorks, MATLAB is an interactive programming environment used in many technical fields including data analysis, problem solving, experimentation and algorithm development (Schreiber 2007). Several MATLAB add-on toolboxes were used in the development of this model. MATLAB's Deep Learning Toolbox was used to provide a framework for designing and implementing the deep neural network. MATLAB's Parallel Computing Toolbox was also used which allows users to solve computationally complex and data-intensive problems using multiprocessors or GPUs. Together with parallel computing, MATLAB's Image Processing Toolbox was used for image processing. This toolbox provides functions for image processing, analysis, transformation and enhancements, to name a few.

The various experiments were run simultaneously on these two GPU's with CUDA capability and NVIDIA graphics card over a period of five weeks.

3.3.2 Datasets

Data organisation

A transfer learning algorithm was modelled and applied to pretrained deep learning models with labelled training data obtained from ImageNet. The training and validation datasets were obtained from the SS image dataset. Thereafter, test data, with fewer labelled images, were obtained from the African Wildlife Foundation (AWF) and

Greenpeace datasets. Due the low number of images available in these test datasets, a subset of the Snapshot Serengeti dataset was also used to create the required test dataset of 50 000 images to identify and classify wildlife based on these images. The datasets were split into 48 classes of animal species.

To find the optimal set of model parameters, while still addressing challenges with complexity and overfitting, it is essential to split the data into training and validation. As described by Xu and Goodacre (2018), the training set is used to build the model with multiple parameter settings and the validation set contains samples with known origins, but the classification is not known to the model.

However, Xu and Goodacre (2018) and Harrington (2017) have suggested that a single split of training and test set can provide erroneous estimations of model performance and the use of an additional test set which is not used during the training or validation process can provide a better generalisation performance of the model (Xu & Goodacre 2018). This is further emphasised by Russell and Norvig (2009) who commented that the training dataset used to fit the model can further be split into the training and validation set, and the subset of the training dataset, called the validation set, can be used to estimate the performance of the model.

Three types of data were used in this study and can be defined as follows (Ripley 1996):

1. Training set is a set of examples used for learning that is to fit the parameters of classifier. This is a sample of data used to fit the model.
2. Validation set is a set of examples used to tune the model parameters. This dataset provides an unbiased evaluation of the model fit.
3. Test set is a set of examples used to assess the performance of the model and provides an unbiased evaluation of a final model fit.

A single-label classification approach was used in this paper, in which one class label per image is identified rather than multiple species in an image.

Data collection

Permission from the following organisations has been obtained and wildlife imaging was collected from the sources as listed in Table 4.

Table 5 outlines the process employed to obtain and download the images.

Table 4: Summary of image datasets

Source	Details	Dataset	Number of Images	Image Size (pixels)
ImageNet	A visual dataset which contains 22 000 classes and 15 million labelled images (Deng et al. 2009).	Training dataset	1.5 million labelled images	Average 469 x 387
Snapshot Serengeti (SS)	The world's largest camera-trap project with 225 camera traps positioned in the Serengeti National Park, Tanzania (Swanson & Kosmala 2015).	Training/Validation/Test dataset	729 249 labelled images	Average 2048 x 1536
Greenpeace	A non-governmental organisation which focuses on campaigning on issues pertaining to climate change, deforestation, and overfishing, to name a few (Greenpeace 2016).	Test dataset	571 images	Average 422 x 600
African Wildlife Foundation (AWF)	An international conservation organisation focused on Africa's wildlife and wild lands, with programs focused on conservation strategies and sustainability (African Wildlife Foundation 2018).	Test dataset	3 573 images	Average 2400 x 1796

Table 5: Summary of image datasets download procedure

Source	Description	Procedure
ImageNet	Visual dataset obtained through CNN architectures	Used in pre-trained networks as outlined in section 2.3.3.
Snapshot Serengeti (SS)	Images published in online data release	1. Download the dataset from the online data release of images https://datadryad.org//resource/doi:10.5061/dryad.5pt92 (Data from Swanson et al. 2016). 2. Use Microsoft Excel PowerQuery to merge the spreadsheets and obtain image URL links to download the images and obtain the corresponding animal classification based on the final "consensus" set of labels as determined by the measure of agreement among individual answers. 3. Download images using the predetermined animal classifications in batches of 10 000 images per animal class using a batch downloader from URL links to conduct batch downloads. 4. Create a hierarchical folder structure on local computer by animal classification, which will contain all downloaded images. 5. Separate into pre-defined classes of animal folders as identified in step 3.
Greenpeace	Images published in online library	1. Download datasets from the online library on https://media.greenpeace.org/CS.aspx?VP3=CMS3&VF=Home (Greenpeace Media n.d.). 2. Create a hierarchical folder structure on local computer by animal classification, which will contain all downloaded images.

		3. Separate into pre-defined classes of animal folders as identified in step 2.
African Wildlife Foundation (AWF)	Images uploaded to OneDrive for download	1. Download image datasets made available on OneDrive on local computer. 2. Create a hierarchical folder structure on local computer by animal classification, which will contain all downloaded images. 3. Separate into pre-defined classes of animal folders as identified in step 3.

Pre-processing and data augmentation

The SS dataset contains images that are 2048 x 1536 pixels and images from the Greenpeace and AWF datasets are on average 422 x 600 and 2400 x 1796 pixels, respectively. These varying image sizes are too large for current deep learning models and hence, were rescaled to fit the model.

According to Goodfellow et al. (2016), the standard practice for rescaling images is 256 x 256 pixels, however this is dependent on the pre-trained network's input size requirements. In this study, the input image sizes based on the model requirements were used. Although this may distort the images slightly, it is common practice in deep learning communities (Goodfellow et al. 2016). Each pretrained network requires different input image sizes, as indicated in Table 6.

Table 6: Input image sizes

Model	Input Image Size
AlexNet	227 x 227
GoogLeNet	224 x 224
VGG-19	224 x 224
ResNet-101	224 x 224

All the images in the study are colour images, except for the night images which are grayscale. Each pixel contains three values: one for green, red and blue colour channels, respectively.

Data augmentation was performed randomly on all images. This can include random cropping, flipping and contrast or brightness modification to prevent overfitting of the model. Image rotation in the range of -20 to 20 degrees was performed and reflections along the X and Y axes were randomly performed on all the images. This was done to change the angles of the animals in the images, as all the original images were captured with the animals appearing horizontally in the frame. In addition, pixel values were changed between -30 and 30 pixels to translate the image colour schemes. All images were passed through the augmented image datastore function in MATLAB, which augments images with a random combination of resizing, rotation, reflection and

translation. Figure 20 shows a few examples of augmented images from the SS training dataset.



Figure 20: Examples of augmented images from the SS dataset

3.3.3 Model architecture

As described in section 2.3.2, which provided an overview of deep learning and basic CNN architecture, deep neural networks consist of varying layers of convolutions, pooling, fully connected and classification layers.

Features from the image are extracted during the convolution operation. By sliding filters, or kernels, over the input image, a feature map is produced (Goodfellow et al. 2016). The more filters/kernels present, the more features are extracted from the image and the better the network will be at recognising patterns. By introducing non-linearity, rectifiers are introduced after the convolution operations by allowing the gradient to flow on the path of active neurons (Goodfellow et al. 2016). The pooling layer takes each feature map output from the convolution layer and prepares a condensed feature map (Nielsen 2015). Finally, the output from the convolution and pooling layers denote high-level features of the input image used to classify the image.

The training process can be summarised using an experiment conducted by Liu et al. 2015:

1. Initialisation of all parameters and weights with random values.
2. Using forward propagation in the fully connected layer, the output probabilities of each class are calculated.

3. The total error is then calculated for the output layer.
4. Backpropagation is then used to calculate the gradients of the errors with respect to the weights and using gradient descent to update the filter and weight values to minimise the output error.
5. Weights are then updated in proportion to their contribution to the total error, allowing the network to 'learn' how to classify the input image.

The abovementioned steps were performed for each image that was passed through the network.

The pretrained networks used in the study were AlexNet, GoogLeNet, ResNet-101 and VGG-19. Their detailed architectures were described in section 2.3.3. The AlexNet model is the simplest of all the models in terms of convolution and fully connected layers, consisting of eight learnable layers, while the ResNet-101 model is the most complex consisting of 101 layers. In terms of parameters, VGG-19 is the most complex consisting of 138 million parameters, while the GoogLeNet model consists of only four million parameters. These pretrained models will be described in the subsequent sections using graphical examples of actual training data to further explain activations, features and the training and learning process.

3.3.4 Activation visualisations

As presented in section 2.3.6, while there have been great advances in training large CNN's, the understanding of how these models work, especially at intermediate and deeper layers, remains an area of active research (Yosinski et al. 2015). This section explains the steps of a deep neural network using activation outputs for illustration purposes. This was done using outputs at various stages of the training process across the various models to better understand the architectures. The training images were taken from the SS dataset.

AlexNet

To illustrate the visualisations from the AlextNet model, Figure 21 will be used as an example of an input image.



Figure 21: AlexNet input image example for visualising activations

Figure 22 shows the activations from the first convolution layer for the AlexNet model. The initial layers of CNN models retain most of the input image features, suggesting that initial filters might be detecting primitive edges and lines. For this image, the convolution filters are activated at every part of the image except the three channels which are blank convolution outputs. This suggests that the pattern encoded by the filters were not found in the input image.

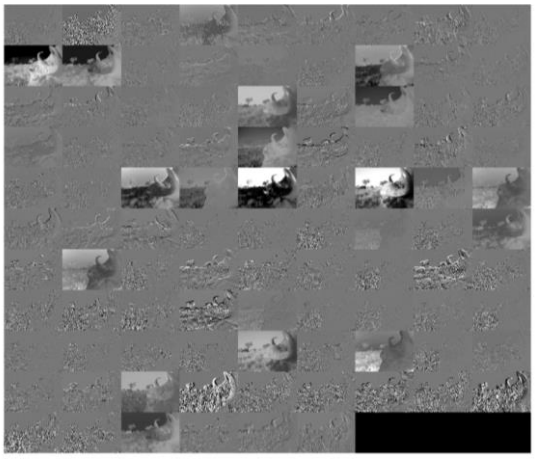


Figure 22: Visualisation of activations extracted from convolution layer one (AlexNet)

Figure 23 shows visualisations from three specific channels to illustrate the activations that each of these channels perform in the first convolutional layer. Each channel detects different types of edges, depending on the output of the kernel at that specific point. More pronounced outlines can be seen in channel 33, compared to channel 34, while colour mapping and contrasting can be seen in channel 45.

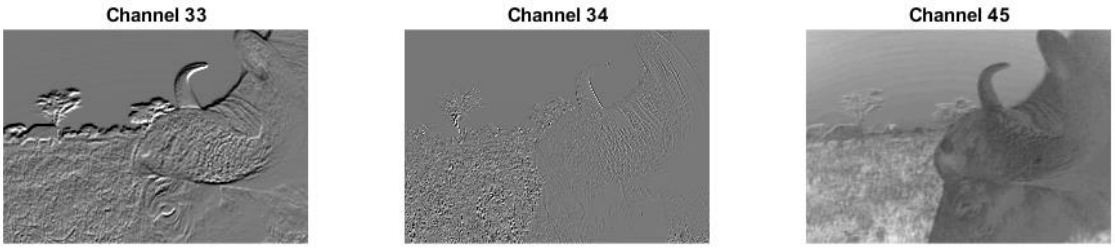


Figure 23: Visualisation of activations extracted from channels 33, 34 and 45 from convolution layer one (AlexNet)

Based on the output from the first convolution layer, the ReLU layer displays the negative and positive values, denoted by black and white respectively as indicated in Figure 24. The dark and light areas represent the weights which have been activated. Figure 25 shows the normalised output from the ReLU layer, which makes the features easier to visualise. With ReLU layers, the activations are denser, however, additional training results in the activations becoming sparser and more localised in deeper layers.

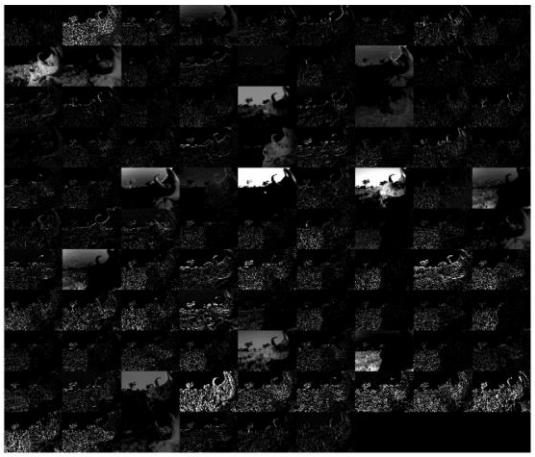


Figure 24: Visualisation of activations extracted from ReLU layer one (AlexNet)

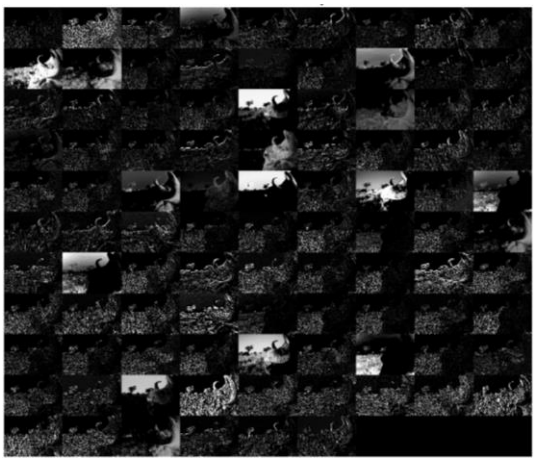


Figure 25: Visualisation of activations extracted from normalisation layer one (AlexNet)

To reduce the amount of detail from previous layers, pooling reduces the size of the feature maps, retaining only the most important information. Figure 26 shows the summarised version of the features detected from the input through the pooled feature maps for the AlexNet model, through max pooling.

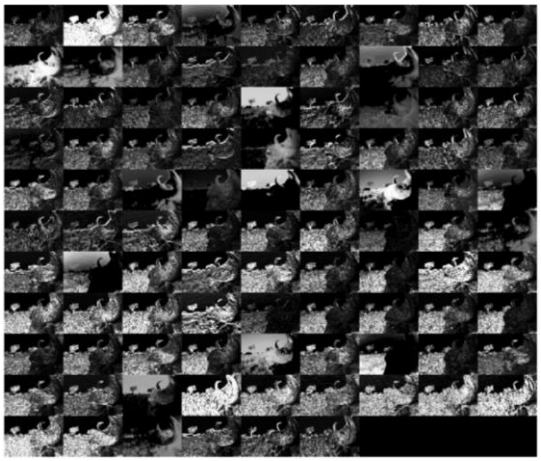


Figure 26: Visualisation of activations extracted from pooling layer one (AlexNet)

Figure 22 to Figure 26 show that activation maps closer to the input capture a lot of fine detail and deeper into the model, the activation maps show less detail. Figure 27 to Figure 29Figure 30 show activations of deeper layers. This allows the model to generalise features from the image to allow the model to perform the classification task (Brownlee 2019a). The pretrained networks use the ImageNet dataset to extract informative features as a starting point to learn new tasks. Even though the ImageNet dataset may not explicitly contain images of buffalo, as the input image in Figure 21, the network has learned to identify these concepts because they represent useful information for making the classification decision later on. For examples, animal and human face detectors, will display the neuron activity as ‘activations’ at various points in the activation image, denoted by the lighter spots on the image, shown in Figure 28. Certain areas of the image indicate dead filters, or activation maps that are zero for the input.

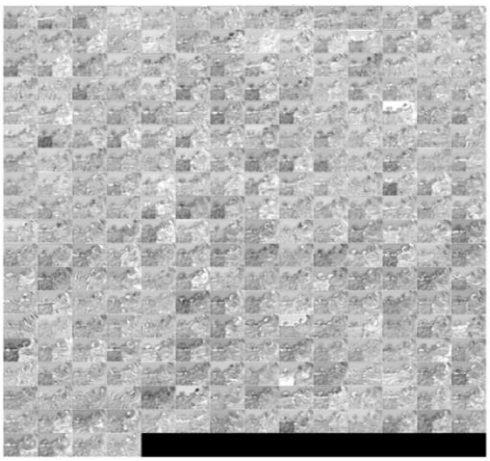


Figure 27: Visualisation of activations extracted from convolution layer five (AlexNet)

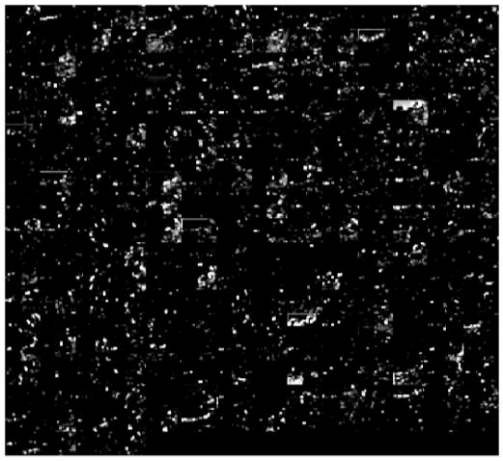


Figure 28: Visualisation of activations extracted from pooling layer five (AlexNet)

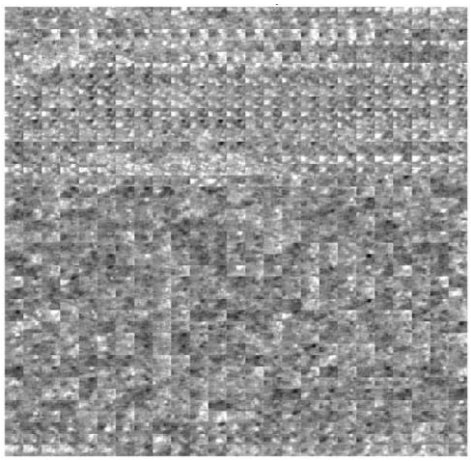


Figure 29: Visualisation of activations extracted from fully connected layer eight (AlexNet)

From the visualisations of deeper convolution layers, it is not evidently clear from the final image in Figure 30 that the model did indeed see a buffalo from the input image in Figure 21, however, the naked eye and human mind do not have the ability to interpret these deeper feature maps (Brownlee 2019a). Unlike features in convolutional layers where most of the original images can be recovered, features from fully connected layers are hard to invert (Wei et al. 2015).

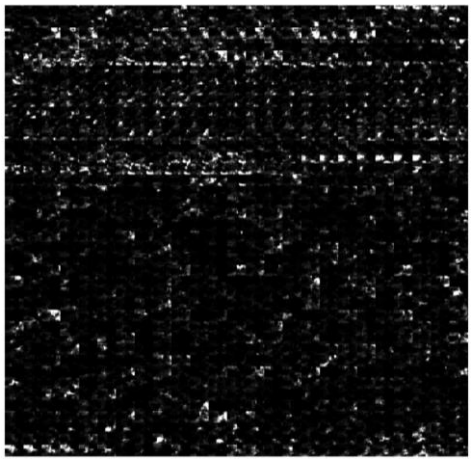


Figure 30: Visualisation of activations extracted from the output layer (AlexNet)

GoogLeNet

To illustrate the activations of various layers of the GoogLeNet model, Figure 31 will be used as an example of an input image.



Figure 31: GoogLeNet input image example for visualising activations

Figure 32 shows the activations from the first convolution layer. Similar to the first convolution layer of the AlexNet model, this layer retains most of the input image features. Because the convolutional network contains a single path from input to output, every layer is a bottleneck through which information must be filtered and passed through to a classification decision (Yosinski et al. 2015).

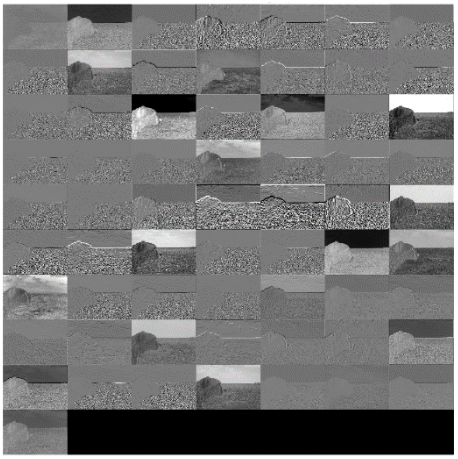


Figure 32: Visualisation of activations extracted from convolution layer 7x7 (GoogLeNet)

The initial layers contain most of the input image features, however, unlike the AlexNet model, there are more blank convolution outputs in the activation map for this image for convolution layer one. This suggests that the input image patterns were not recognisable by the filters. Delving deeper into specific channels, Figure 33 shows the activations of channels 33, 34 and 45. Using the same channels as the AlexNet model, it can be observed that even across models, the channels activate different details, such as edges and outlines.

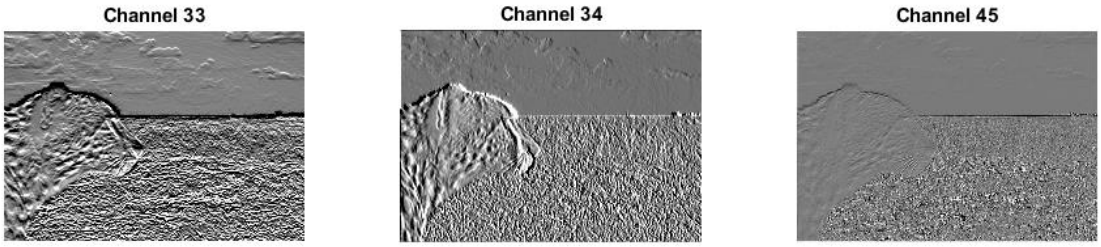


Figure 33: Visualisation of activations extracted from channels 33, 34 and 45 from convolution layer one (GoogLeNet)

Based on the output from the first convolution layer, Figure 34 shows the activation map for the first ReLU layer of the 7x7 convolution. The dark and light areas represent the weights that have been activated. Unlike the AlexNet model, the pooling layer, shown in Figure 35, appears before the normalisation layer, reducing the size of the feature maps before being normalised.

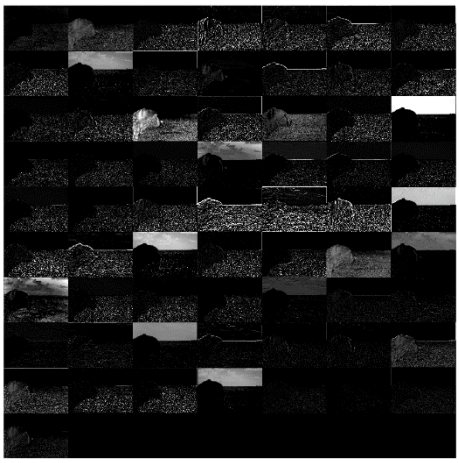


Figure 34: Visualisation of activations extracted from ReLU layer 7x7 (GoogLeNet)

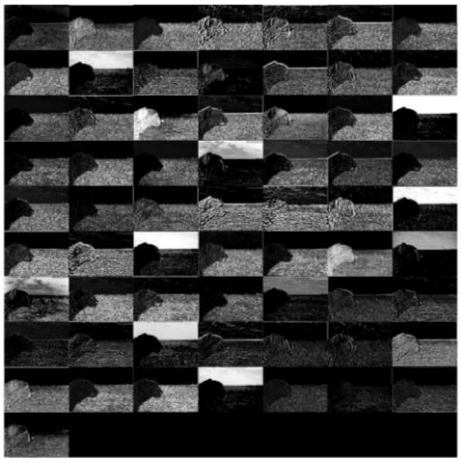


Figure 35: Visualisation of activations extracted from pooling layer one (GoogLeNet)

The normalised activation map, shown in Figure 36 takes the output of the pooling layer and ReLU activations and makes them easier to visualise.

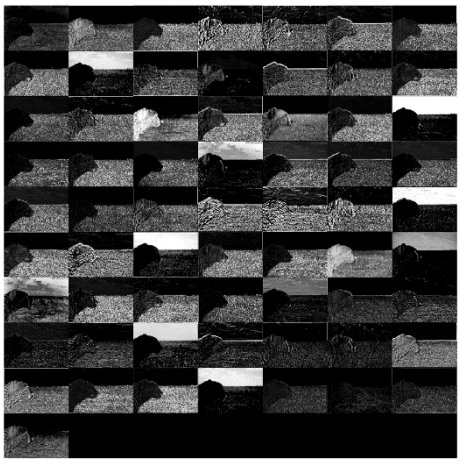


Figure 36: Visualisation of activations extracted from normalisation layer one (GoogLeNet)

The deep convolution layers, known as inception layers, are shown in Figure 37 and Figure 38 for inception 3a 1x1 and 5b 5x5 layer, respectively. Similar to the AlexNet model, the deeper convolution layers are more difficult to interpret. As one goes deeper into the CNN, the network no longer relies on the visual information of the input image and rather converts it to the required output classification domain (Zeiler & Fergus 2014). The activation maps closer to the input image capture more detail than the activation maps of deeper layers.

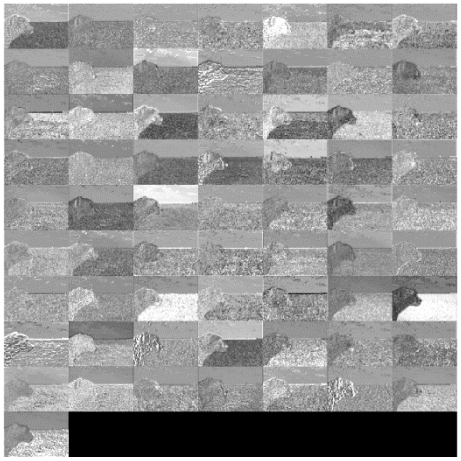


Figure 37: Visualisation of activations extracted from inception layer 3a 1x1 (GoogLeNet)

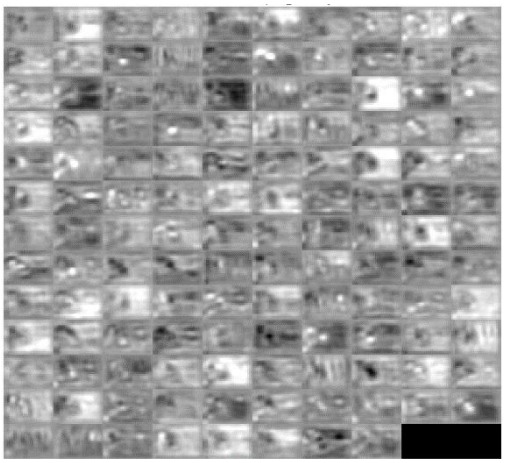


Figure 38: Visualisation of activations extracted from inception layer 5b 5x5 (GoogLeNet)

The original image of the cheetah is no longer visible and becomes more unrecognisable in final output layer, as shown in Figure 39. The neuron activity can be seen in the lighter spots on the output image, implying that certain detectors, present in the pretrained model's learned behaviour, were activated based on the input image of the cheetah while dead filters, or activations of zero value were present on other areas of the image.

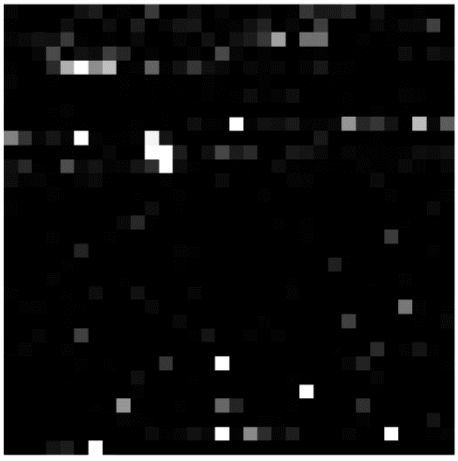


Figure 39: Visualisation of activations extracted from the output layer (GoogLeNet)

ResNet-101

Figure 40 will be used as an example as an input image to visualise the activations of the ResNet-101 model.



Figure 40: ResNet-101 input image example for visualising activations

The activations from the first convolution layer of the ResNet-101 model are shown in Figure 41. Similar to the AlexNet and GoogLeNet models, the first layer retains most of the input image features, depicting primitive edges, lines and outlines of the original input image. There are six channels that have blank convolution outputs, suggesting that these patterns which were present in the object detectors at those points consist of complex shapes which were not present in the input image.

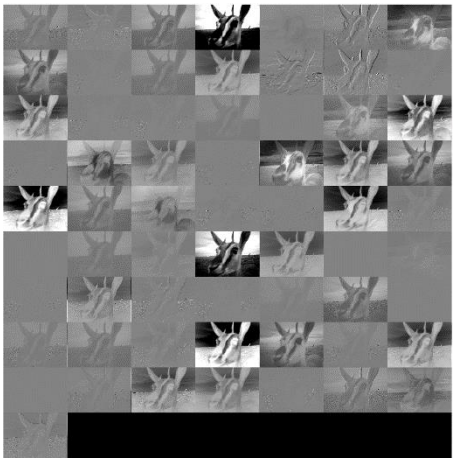


Figure 41: Visualisation of activations extracted from convolution layer one (ResNet-101)

Figure 42 shows the visualisations from three specific channels from convolution layer one, namely channel 33, 34 and 45. Channel 33 detects more of the background area, while in channels 34 and 45, the colour channels and outlines become more pronounced, respectively.



Figure 42: Visualisation of activations extracted from channels 33, 34 and 45 from convolution layer one (ResNet-101)

Based on the output from the first convolution layer, Figure 43 and Figure 44 depict the ReLU and pooling layers after convolution layer one, respectively. The difference in architecture and network parameters between the AlexNet, GoogLeNet and ResNet-101 models indicate that the ReLU and pooling layers identify different activations based on the input image. The light and dark areas depict the neurons that have been activated or that consist of dead filters, respectively.

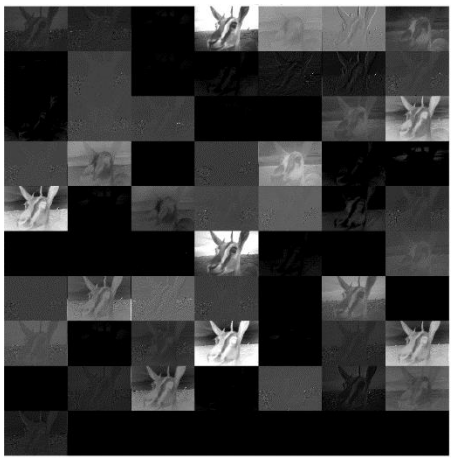


Figure 43: Visualisation of activations extracted from ReLU layer one (ResNet-101)

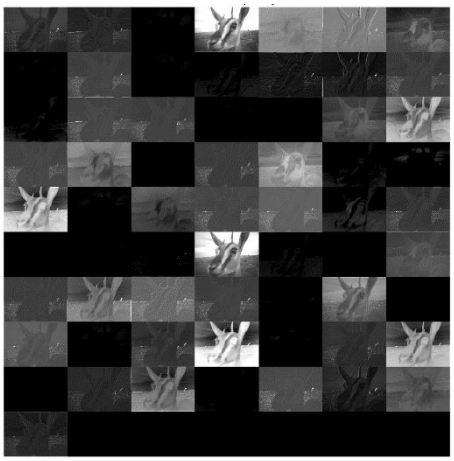


Figure 44: Visualisation of activations extracted from pooling layer one (ResNet-101)

Figure 45 shows the normalised activation map, which is created by taking the output from the ReLU and pooling layers.

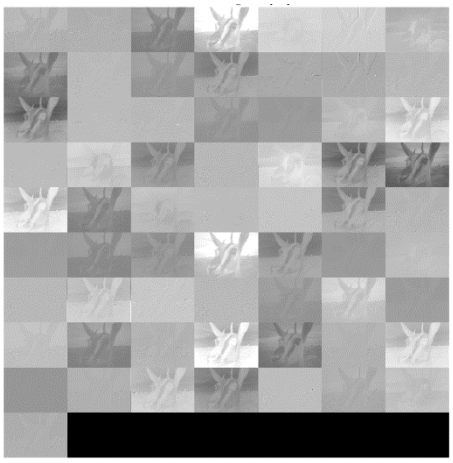


Figure 45: Visualisation of activations extracted from batch normalisation layer one (ResNet-101)

Considerably deeper than the AlexNet and GoogLeNet models, the ResNet-101 model consists of 101 parametrised layers. Figure 46 and Figure 47 show the activations of convolution layers 2b_branch2c and 4b22_branch2c, respectively. As the size of the convolutions change with deeper layers into the model, the activations become less discernible to the naked eye, but are activated across more channels.

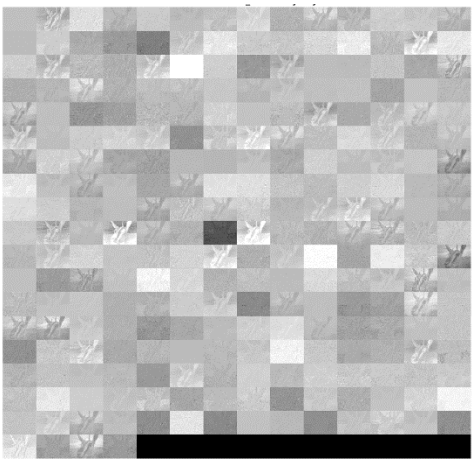


Figure 46: Visualisation of activations extracted from convolution layer 2b_branch2c (ResNet-101)

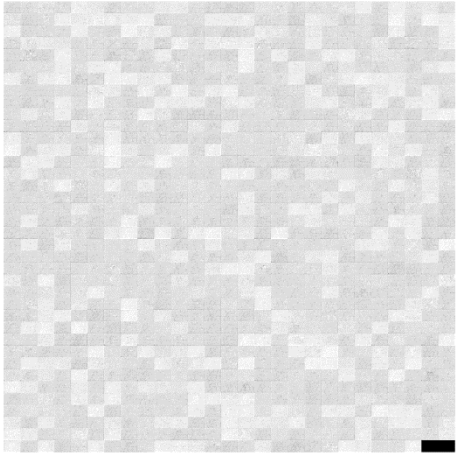


Figure 47: Visualisation of activations extracted from convolution layer 4b22_branch2c (ResNet-101)

Similarly, Figure 48 shows the activations of convolution layer 5c_branch2c. The increase in the number of convolution filters in this layer compared to those in Figure 46 and Figure 47, increase the detail that is learned from the input image and increases the depth of the output image. The number of convolution filters in deeper layers of the ResNet-101 model are more than those of the deeper models in the AlexNet and

GoogLeNet model, which is consistent with the level of accuracy that is observed for deeper models.

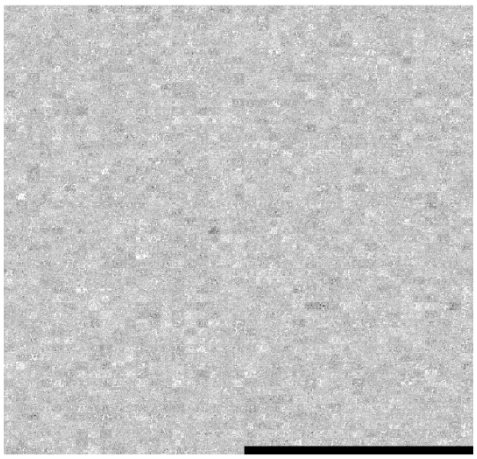


Figure 48: Visualisation of activations extracted from convolution layer 5c_branch2c (ResNet-101)

Figure 49 shows the output layer of the ResNet-101 model. Comparably to the AlexNet and GoogLeNet, it is not clear from the final output layer that the model did indeed see a gazelle Thomson, as shown in input image in Figure 40.

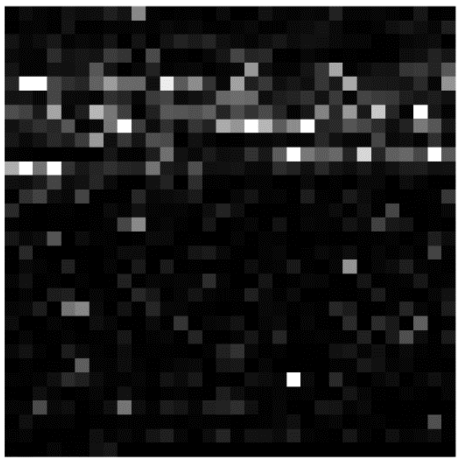


Figure 49: Visualisation of activations extracted from the output layer (ResNet-101)

VGG-19

To illustrate the activations from the VGG-19 model, Figure 50 will be used as an example of an input image.



Figure 50: VGG-19 input image example for visualising activations

Figure 51 shows the activations from the first convolution layer of the VGG-19 model. Similar to the AlexNet, GoogLeNet and ResNet-101 models, the initial convolution layers retain most of the input image features. Like the ResNet-101 model, there are six channels with blank convolution outputs, suggesting that the object detectors consist of complex shapes which might not have been present in the input image.

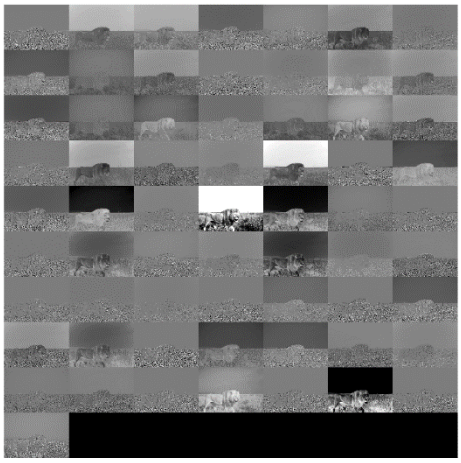


Figure 51: Visualisation of activations extracted from convolution layer one (VGG-19 model)

Figure 52 shows the visualisations from three specific channels from convolution layer one, namely channels 33, 34 and 45. It can be observed that across the various models, the same channels activate different details, including colour contrasts, edges, and outlines.

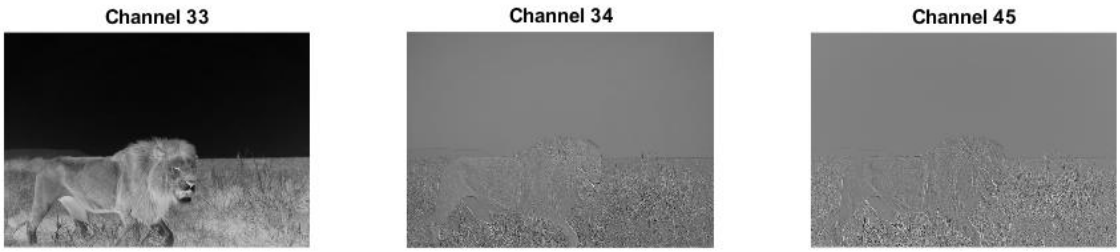


Figure 52: Visualisation of activations extracted from channels 33, 34 and 45 from convolution layer one (VGG-19)

Based on the output of the convolution layer, Figure 53 and Figure 54 show the activation maps for the ReLU and pooling layers, respectively. The dark and light areas depict the areas that have been activated.

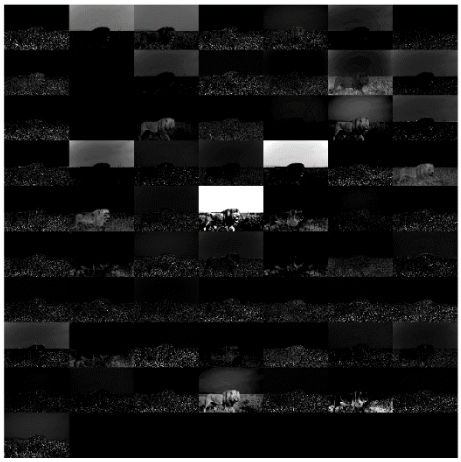


Figure 53: Visualisation of activations extracted from ReLU layer one (VGG-19)

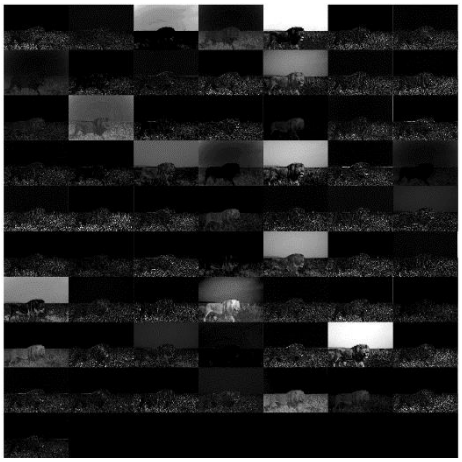


Figure 54: Visualisation of activations extracted from pooling layer one (VGG-19)

Figure 55 and Figure 56 show the activation maps of the deeper convolution layers 4_4 and 5_2, respectively. Similar to the AlexNet, GoogLeNet and ResNet-101 models, the deeper layers are more difficult to interpret. The activation maps closer to the input image capture more detail than deeper layers, converting the visual information into the same domain as the output classification.

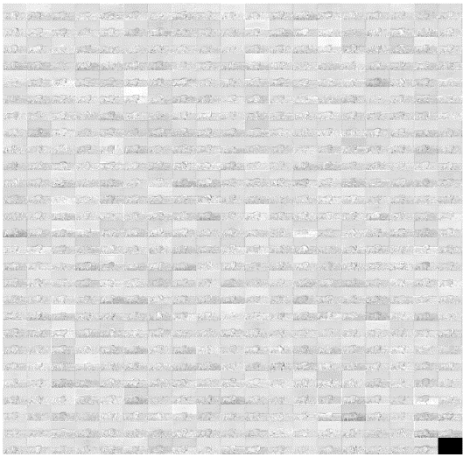


Figure 55: Visualisation of activations extracted from convolution layer 4_4 (VGG-19)

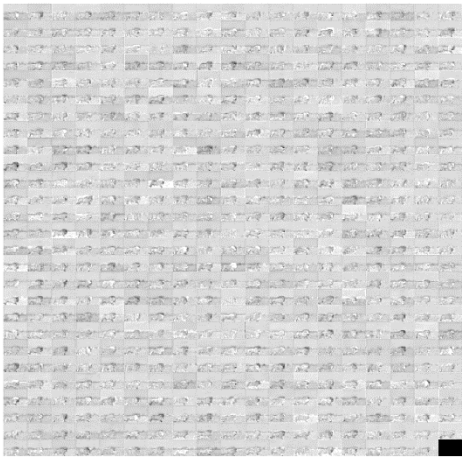


Figure 56: Visualisation of activations extracted from convolution layer 5_2 (VGG-19)

Visualising the fully connected layers of the VGG-19 model, shown in Figure 57 and Figure 58, the 1 000 activation points can be observed. The details of each activation can be seen if one zooms into the image, however, the output of these layers is the input into the final output classification layer. In the fully connected layer seven, shown in Figure 57, there are still dead filters present, which indicate activations of zero value.

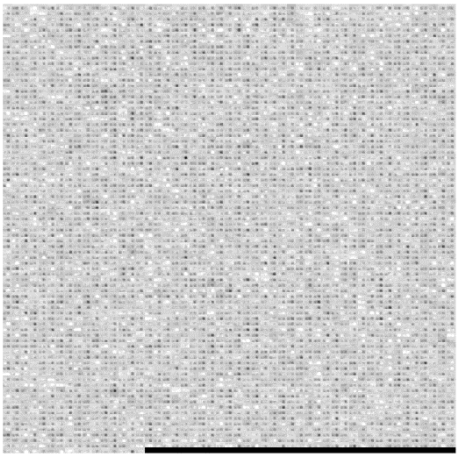


Figure 57: Visualisation of activations extracted from fully connected layer seven (VGG-19)

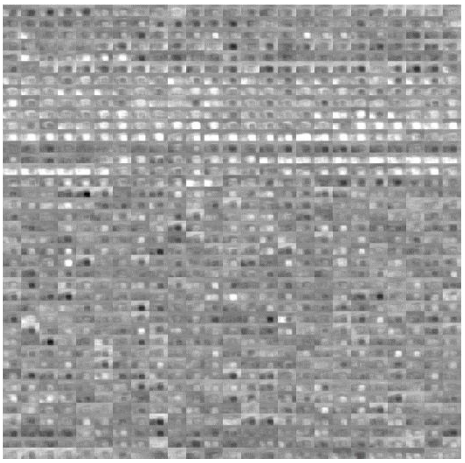


Figure 58: Visualisation of activations extracted from fully connected layer eight (VGG-19)

The output layer of the VGG-19 model, shown in Figure 59, is used to classify the input image. Similar to the AlexNet, GoogLeNet and ResNet-101 models, it is not obvious that

the model did indeed see a lion, based on the input image, however, Wei et al. (2015) and Brownlee (2019a) indicate that these deeper maps are harder to invert and understand.



Figure 59: Visualisation of activations extracted from the output layer (VGG-19)

3.3.5 Feature visualisations

This section explains the steps of a deep neural network using feature outputs for illustration purposes. Unlike activation maps that are difficult to interpret, feature visualisation is described using descriptions from a 'semantic dictionary', allowing for these activations to be mapped to iconic representations, or high-level 'ideas' (Olah et al. 2018). Although this allows one to visualise what the network detects, it does not necessarily provide an explanation as to how the network assembles these concepts that lead to the final decision.

Figure 60, taken from the SS dataset, will be used as an example to illustrate the feature identification and detection process for all four models. This was done using outputs at various stages of the training process across the various models to better understand the architectures and feature representations. The feature visualisations are best viewed electronically, zoomed in.



Figure 60: Input image example for visualising features

AlexNet

Lower-level convolution layers for AlexNet, namely convolution layers one and two are edge detectors and colour filters. These textures combine into increasingly complex patterns in deeper layers. For illustration purposes, only channels one to six have been extracted to visualise the features learned by these layers. Figure 61 and Figure 62 show the low-level features learned by convolutional layers one and two, respectively.



Figure 61: Visualisation of features learned by convolutional layer one (AlexNet)



Figure 62: Visualisation of features learned by convolutional layer two (AlexNet)

Visualisation of deeper layers learn higher-level combinations of features learned during earlier layers. Figure 63 shows the features learned by convolutional layer five in the AlexNet model. This layer starts to provide more detailed filters through pattern and texture overlays.

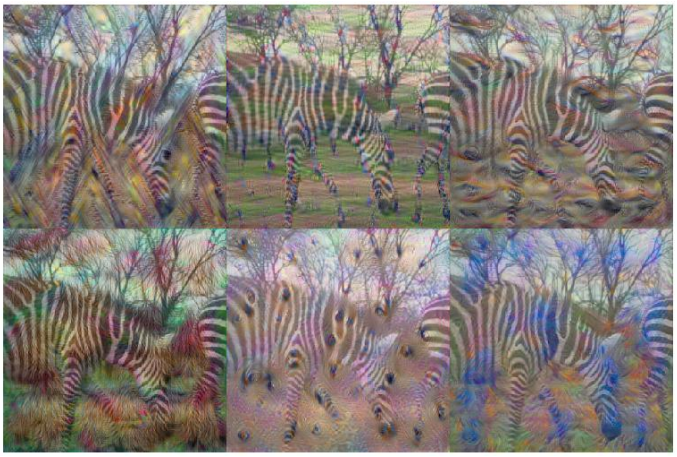


Figure 63: Visualisation of features learned by convolutional layer five (AlexNet)

Figure 64 shows the visualisations of the fully connected layer of the AlexNet model. The initial input image is faintly visible as complex patterns are now at the forefront of the image. Depending on the object detectors which are activated, one can see that detectors for stripes and fur can be seen in Figure 64.

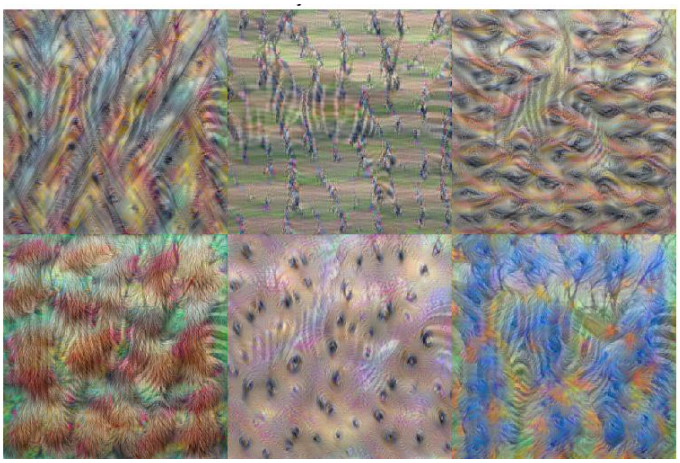


Figure 64: Visualisation of features learned by fully connected six (AlexNet)

Similarly, the fully connected layer seven shown in Figure 65 provides more iconic representations of the visualisations. Using channel two, as shown in Figure 66, as an example, the "snout" object detector, can be seen suggesting that the zebra's nose from the initial input image could have triggered this specific neuron detector.

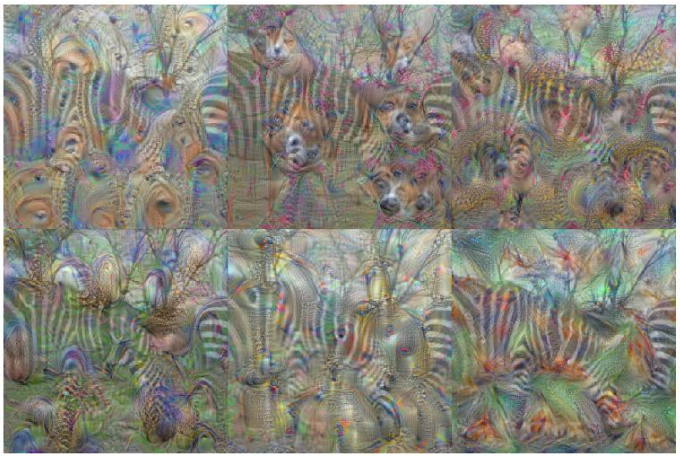


Figure 65: Visualisation of features learned by fully connected seven (AlexNet)



Figure 66: Visualisation of features learned by fully connected layer seven - channel two (AlexNet)

GoogLeNet

Using the GoogLeNet model and the same input image from Figure 60, the feature visualisations of lower-level output convolution operations for inception modules 3b and 4a are shown in Figure 67 and Figure 68. Unlike the first few convolution layers of the AlexNet model, these first inception layer shows various textures while inception layer 4a, which follows a pooling step, sees more complex patterns.

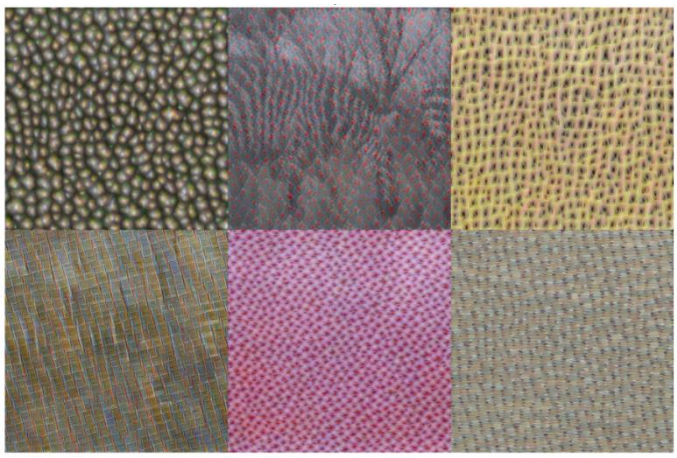


Figure 67: Visualisation of features learned by inception output layer 3b (GoogLeNet)



Figure 68: Visualisation of features learned by inception output layer 4a (GoogLeNet)

Similar to the AlexNet model, the deeper inception layers in the GoogLeNet model such as inception layer 4e shown in Figure 69, depict complex textures that can be related to multiple concepts. If one looks closely at channel six in Figure 69, trees, eyes and the original zebra stripes can be seen.



Figure 69: Visualisation of features learned by inception output layer 4e (GoogLeNet)

Visualisations on the final inception layer 5b shown in Figure 70 illustrate a combination of subjects and patterns. The subjects and patterns are triggered depending on the detectors that are deemed to be important through training. For example, perhaps "snouts" are important when distinguishing zebras, as "snouts" can be seen in channel one in Figure 70. This is consistent with the object detector of "snout" which was observed in the AlexNet model.



Figure 70: Visualisation of features learned by inception output layer 5b (GoogLeNet)

The visualisations show an increase in complexity and variation of higher layers, which are comprised of simpler components from the lower layers. The variation of patterns increases as the depth of the layers increase, indicating that increasingly invariant representations are learned (Yosinski et al. 2015).

ResNet-101

Using the input image in Figure 60 and the ResNet-101 model, Figure 71 shows the feature visualisations of the first convolution layer. Unlike the AlexNet and GoogLeNet models, this layer seems to consist of more colour filters than edge or pattern detectors. This can be seen by the varying shades of the original image i.e. white, sepia and grey.



Figure 71: Visualisation of features learned by convolution layer one (ResNet-101)

Visualising the features of convolution layer 3b_branch2c, shown in Figure 72, the colour detectors become more pronounced, although if one zooms in, slight patterns can be observed in channel two.

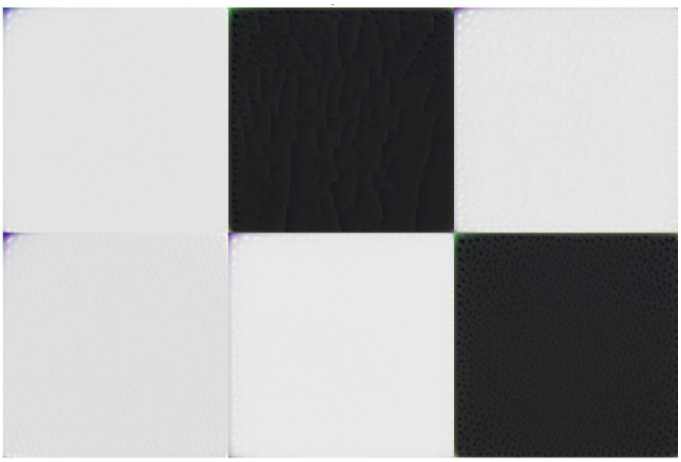


Figure 72: Visualisation of features learned by convolution layer 3b_branch2c (ResNet-101)

Delving into deeper convolution layers 4b10_branch2c and 5c_branch2c, shown in Figure 73 and Figure 74, respectively, more complex patterns are visible.

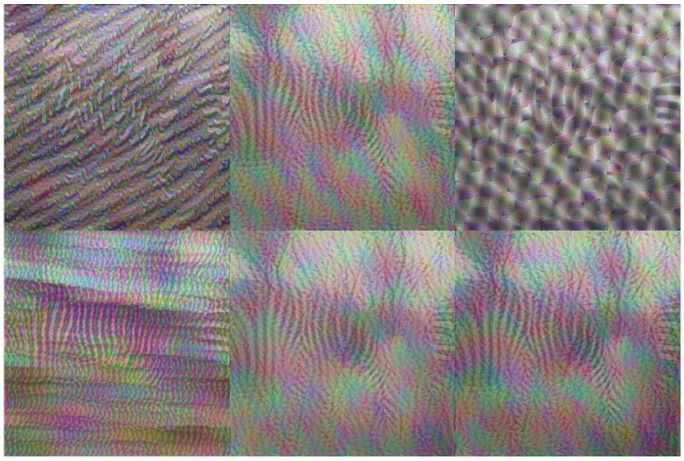


Figure 73: Visualisation of features learned by convolution layer 4b10_branch2c (ResNet-101)

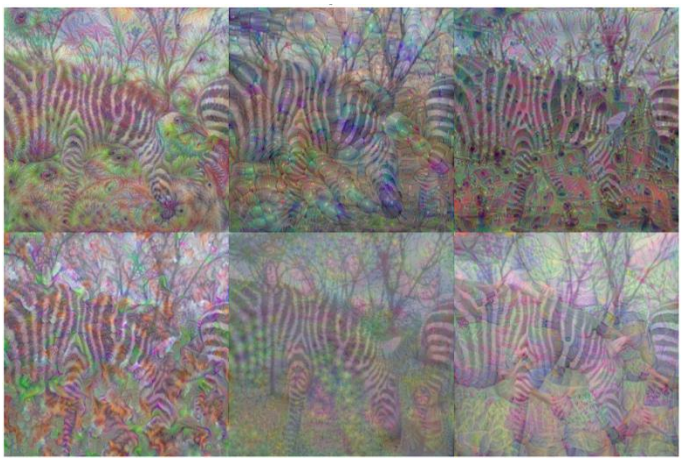


Figure 74: Visualisation of features learned by convolution layer 5c_branch2c (ResNet-101)

Zooming into channel one of the convolution layer 5c_branch2c, shown in Figure 75, object detectors for eyes can be observed. This suggest that the zebra's eye which is visible in the initial input image in Figure 60 triggered this specific detector, assisting in extracting this information from the original image.



Figure 75: Visualisation of features learned by convolution layer 5c_branch2c - channel one (ResNet-101)

Visualising the features of the final output layer are shown in Figure 76. The deeper layers consist of more complex shapes and patterns, which are encoded by the filters. These are then used to extract features such as eyes, noses, ears etc. Zeiler and Fergus (2014) suggested that as feature hierarchies become deeper, they learn increasingly powerful features.



Figure 76: Visualisation of features learned by the output layer (ResNet-101)

VGG-19

Using the VGG-19 model and input image in Figure 60, the feature visualisations of convolution layer one can be observed in Figure 77. Similar to the ResNet-101 model, this layer seems to consist of more colour filters, than pattern detectors. The colour filters of white, grey, and black can be seen in the image.

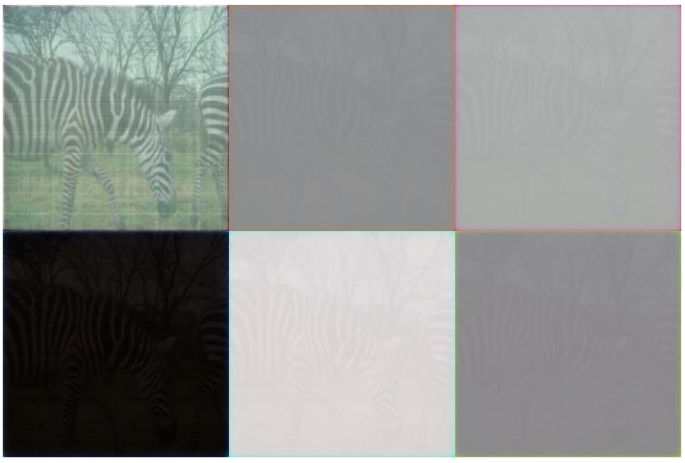


Figure 77: Visualisation of features learned by convolution layer one (VGG-19)

The visualisation of the deeper convolution layer 4_1 can be seen in Figure 78. Although the complex patterns cannot be related to iconic representations which can be described in words, such as “eyes” and “fur”, the patterns identified in this layer will most probably be used by the model to extract specific features.

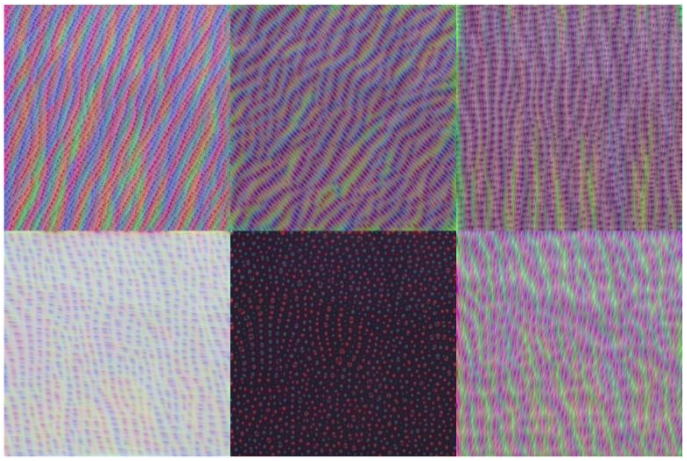


Figure 78: Visualisation of features learned by convolution layer 4_1 (VGG-19)

Unlike the feature visualisations of the AlexNet, GoogLeNet and ResNet-101 models, the fully connected and output layer of the VGG-19 model, shown in Figure 79 and Figure 80, respectively, do not clearly depict patterns and high-level representations. The channels depict low and high frequency patterns, showing the complexity and variation in higher layers, which are comprised of the simpler components from lower layers. It can be observed that the variation of patterns increases as the number of layers increase, indicating that increasingly invariant representations are being learned.

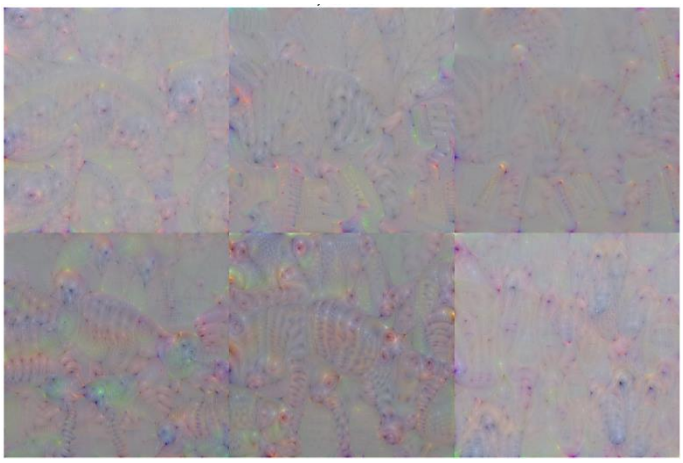


Figure 79: Visualisation of features learned by fully connected layer eight (VGG-19)

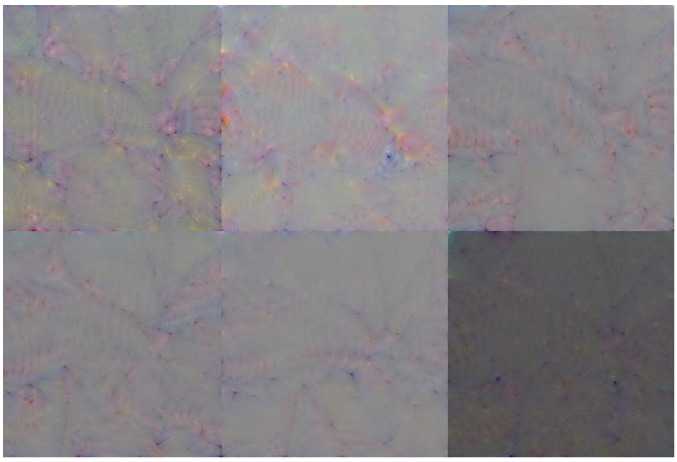


Figure 80: Visualisation of features learned by the output layer (VGG-19)

3.3.6 Model development

As described in section 2.2.3, transfer learning is a deep learning technique in which the network is trained on a large, base dataset and transfers the learned knowledge to a smaller, target dataset (Rehman et al. 2019). There are two main transfer learning techniques: fine-tuning the convolution layers and freezing the convolution layers.

During fine-tuning, pretrained convolution layers are replaced and retrained on the target dataset and the bottom layers are updated during backpropagation. The last fully connected layer classifies the target dataset. When freezing the convolution layers, the last fully connected layers are removed, and the bottom layers are not updated during backpropagation. Both techniques are tested in this study.

When freezing the layers, the weights in earlier layers are 'frozen' and the learning rates in those layers are set to zero. This technique is motivated by the observation that in many deep architectures, the early layers take up the most computation but have the fewest parameters and tend to converge to simple configurations (Brock et al. 2017). This suggests that these layers do not require as much fine tuning as the later layers, where most of the parameters reside.

The 'frozen' layers were identified by considering the layers which contained weights which would change during training to prevent the network from updating the parameters of the frozen layers during the training process. As the gradients of the frozen layers do not need to be computed due to the frozen weights, freezing the layers can speed up the training time. In addition, with a small dataset, freezing the layers can prevent overfitting.

In the pretrained AlexNet model, layers one to ten were frozen and are shown in Figure 81.

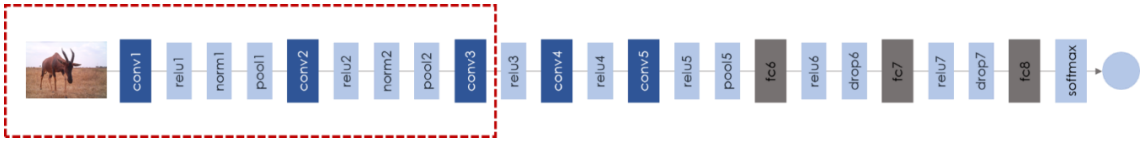


Figure 81: Frozen layers of pretrained AlexNet model (network illustration adapted from Krizhevsky et al. 2012)

In the pretrained GoogLeNet model, layers one to 110 were frozen and are shown in Figure 82.

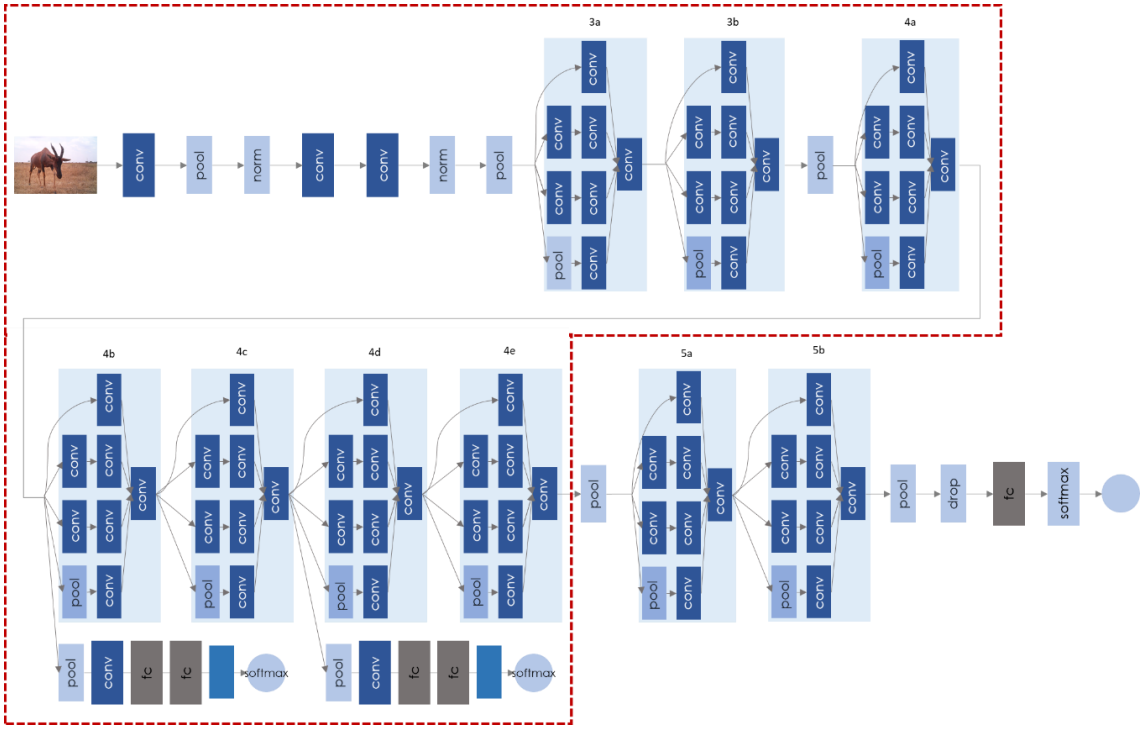


Figure 82: Frozen layers of pretrained GoogLeNet model (network illustration adapted from Szegedy et al. 2015)

In the pretrained ResNet-101 model, layers one to 251 were frozen and are shown in Figure 83. This model consists of 33 residual blocks, and the layers are frozen up to and including the third set of residual blocks of size 14 x 14 x 1024.

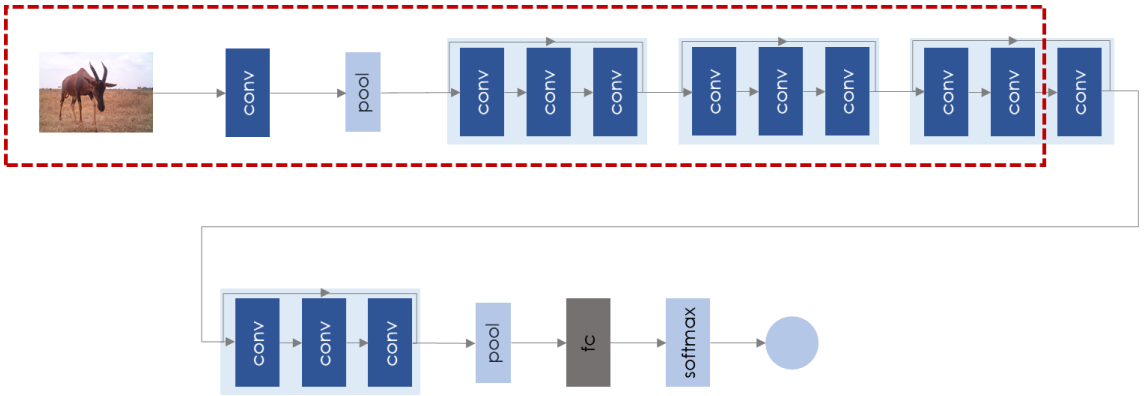


Figure 83: Frozen layers of pretrained ResNet-101 model (network illustration adapted from He et al. 2016)

In the pretrained VGG-19 model, layers one to 22 were frozen and are shown in Figure 84.

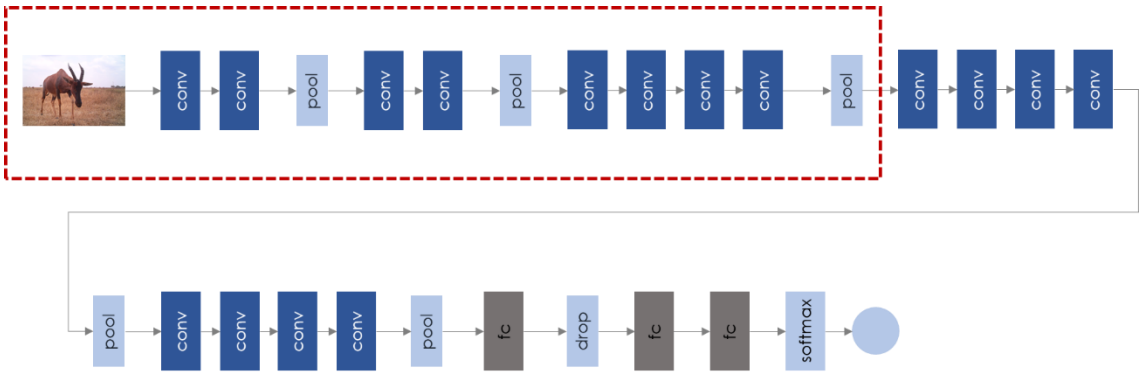


Figure 84: Frozen layers of pretrained VGG-19 model (network illustration adapted from Garcia-Gasulla et al. 2017)

For all pretrained models, the following process was employed for freezing the layers:

1. Load the pretrained model.
2. Examine the network layers and extract the specific layers.
3. Freeze the initial layers to prevent updating of weights.
4. Extract the final output layer and modify the number of classes to the new dataset to match the previous layer using the following line of code (MathWorks 2020b):

$$removeLayers = (lgraph, \{fully\ connected\ layer, probability, output\})$$

$$newLayers = [fullyConnectedLayer \ldots$$

$$(numClasses,' Name',' fc',' WeightLearnRateFactor', X, BiasLearnRateFactor', Y)$$

$$\ldots$$

$$softmaxLayer('name',' softmax')\ldots$$

$$classificationLayer('Name',' classoutput')]$$

where

- the three layers to be replaced are *fully connected layer*, *probability* or *softmax layer and the output, or classification layer;*
- *numClasses* are the number of classes, which in this study is 48 classes; and
- *WeightLearnRateFactor* and *BiasLearnRateFactor* values need to be specified in the X and Y field.

The $newLayers$ are then added to the $lgraph$ and connected to the preceding layers.

5. Create an image datastore with multiple file images from the network folder and label the images using the folder names.
6. Pass the images through the augmented image datastore to implement the data augmentation techniques specified in section 3.3.2.
7. Resize the images with the pretrained network input image requirements.
8. Split the data into training and test datasets and randomise the files in the folder.
9. Specify training options such as optimisation functions, learning rates, mini-batch sizes and number of epochs.
10. Evaluate the training and test performance using various evaluation metrics.

When fine-tuning the model, unfreeze all layers and run the model with adjusted hyperparameters. The same steps as those specified above were used, apart from step 3. Figure 85 to 88Figure 92 show the fine-tuned models which allow for backpropagation and updating of weights.

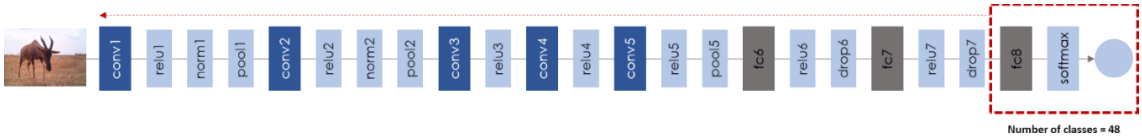


Figure 85: Fine-tuned layers of pretrained AlexNet model (network illustration adapted from Krizhevsky et al. 2012)

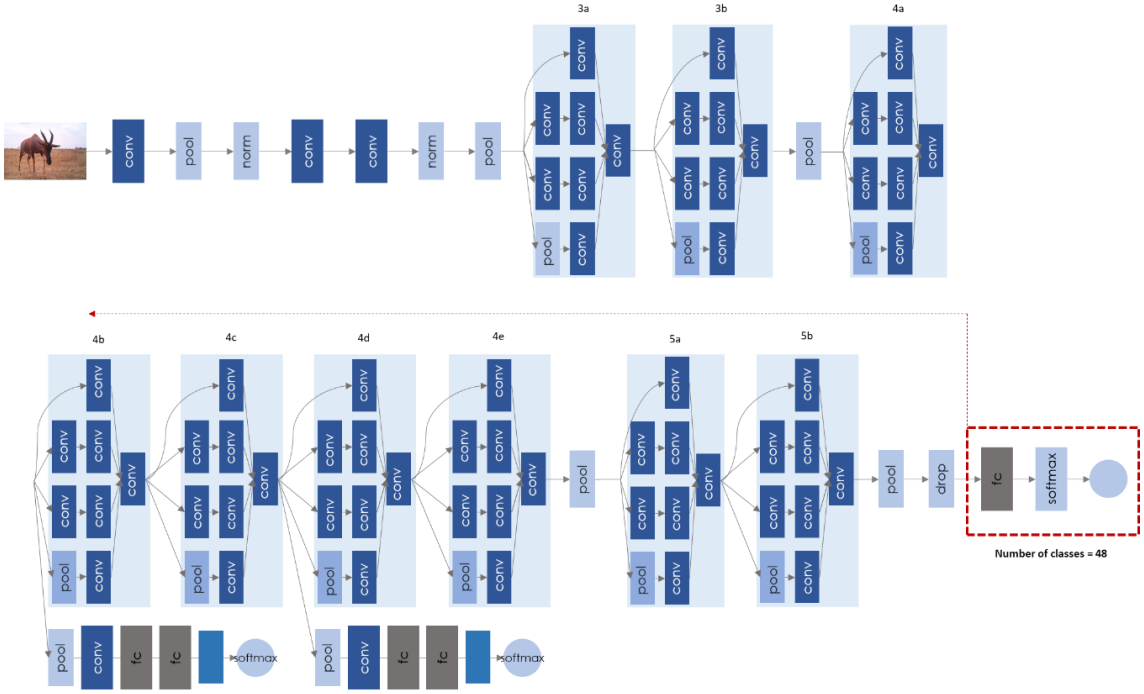


Figure 86: Fine-tuned layers of pretrained GoogLeNet model (network illustration adapted from Szegedy et al. 2015)

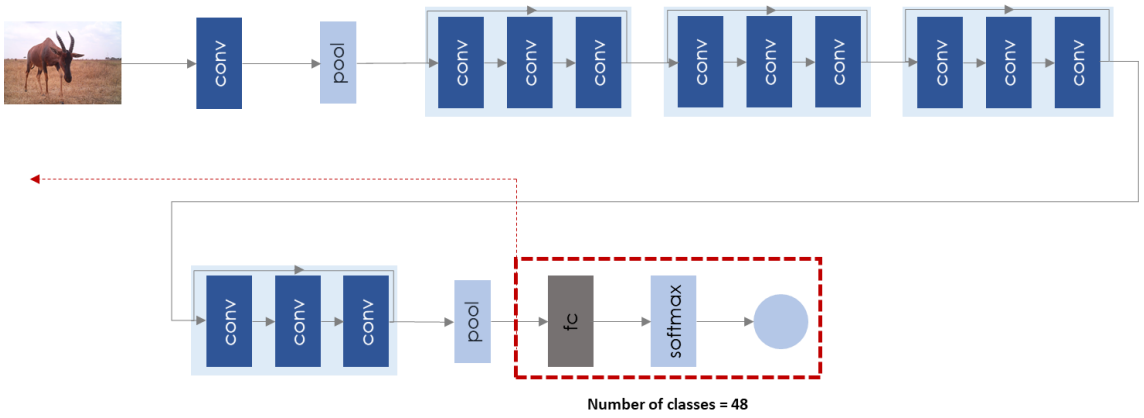


Figure 87: Fine-tuned layers of pretrained ResNet-101 model (network illustration adapted from He et al. 2016)

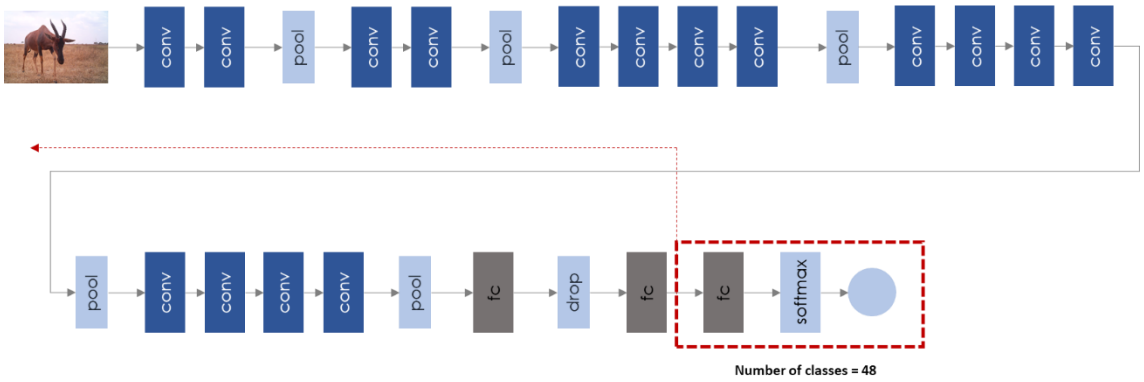


Figure 88: Fine-tuned layers of pretrained VGG-19 model (network illustration adapted from Garcia-Gasulla et al. 2017)

Comparing to the study by Norouzzadeh et al. (2017)

The study conducted by Norouzzadeh et al. (2017) identified the use of transfer learning to automate animal identification as a field for further study. Their implementation of transfer learning followed the methods outlined in previous work conducted by Oquab et al. (2014), Donahue et al. (2014) and Sharif et al. (2014).

In their study, the first test was conducted through transfer learning and training the model on the ImageNet dataset and then further training the model on a smaller simulated camera-trap dataset, which was created by using labelled datasets of different sizes from the SS data. For their experiments, two different transfer learning techniques were used (Norouzzadeh et al. 2017):

1. Further training the entire network on the new smaller, target dataset, and
2. Only further training the last layer of the network.

The second test that they conducted was to determine whether transfer learning would improve performance when fewer labelled images were available.

Norouzzadeh et al. (2017) used the hyperparameters outlined in Table 7. These hyperparameters were used when running the network on transfer learning, rather than training the model from scratch.

Table 7: Network training hyperparameters from Norouzzadeh et al. (2017) study

Hyperparameter	Value
mini batch size	128
momentum	0.9
crop size	224 x 224
number of epochs	40 (30 when training the last layer only)

In this study, transfer learning was performed by using images from three datasets to test the model classification accuracy. Table 8 outlines the experiments that were conducted in this research. The first experiment utilised the technique of freezing layers, while the second experiment used the technique of fine-tuning. Within Experiments 1 and 2, different learning rates were used. In Experiment 3, additional class balancing techniques were used to determine the impact on the accuracy of rare classes.

Table 8: Experiments for testing

Experiment	Description
1	Transfer learning on the full SS training dataset and using a subset of the SS dataset, AWF and Greenpeace datasets for testing using the technique of freezing layers
2	Transfer learning on the full SS training dataset and using a subset of the SS dataset, AWF and Greenpeace datasets for testing using the technique of fine tuning
3	Transfer learning on the full SS training dataset and using a subset of the SS dataset, AWF and Greenpeace datasets for testing using additional class balancing techniques

Improving the accuracy of rare classes

The Snapshot Serengeti image dataset is highly imbalanced dataset with the smallest animal class containing 29 images the largest class containing over 200 000 images, as

shown in Figure 89. To improve the accuracy of rare classes, all three techniques for overfitting were used in this study: dropout, regularisation and over and under sampling.

Each pretrained network uses the dropout technique to randomly exclude a set of neurons from updating during the training pass so that it will not participate in backpropagation, allowing a different architecture to be sampled every time an input is presented. AlexNet, VGG-19 and GoogLeNet have dropout layers built into their networks of 50%, 50% and 40%, respectively. In some cases, such as with Resnet-101, instead of dropout layers, batch normalisation is used which is a technique to normalise the inputs to the layer, resulting in a regularisation effect. In addition, L2 regularisation or weight decay was used when specifying the training options. A weight decay policy similar to the one employed in the study by Norouzzadeh et al. (2017) was used and kept constant at a value of 0.005 throughout the training and learning process.

For Experiment 3, an emphasis sampling technique was used. This was done manually by oversampling the rare classes after the network misclassified them.

3.3.7 Experimental framework

There are several data splitting methods which can be categorised into the following (Xu & Goodacre 2018):

1. Cross-validation divides the data into *k*-different parts, or folds, and one part is held out for validation. The model is trained on *k*-1 folds and applied to the validation set. This process is repeated *k*-times so each part has been used a validation once;
2. Randomly selecting a proportion of samples for training and validation; and
3. Systematically selecting a number of representative samples from the datasets and using the remaining samples as validation.

For the purposes of this study, a random proportion of samples was selected for the training, validation and test dataset. In order to test the theory as outlined in the research objectives in section 1.6, the data was split into 86%, 7% and 7%, for training, validation and test datasets, respectively. This was in line with the train-test split ratios used in the studies conducted by Villa et al. (2017) and Norouzzadeh et al. (2017). Table 9 outlines

the number of images used for training and validation. The same training and test data split were used for Experiments 1, 2 and 3.

Table 9: Training and test data split

Training/validation dataset	682 055
Test dataset	51 338

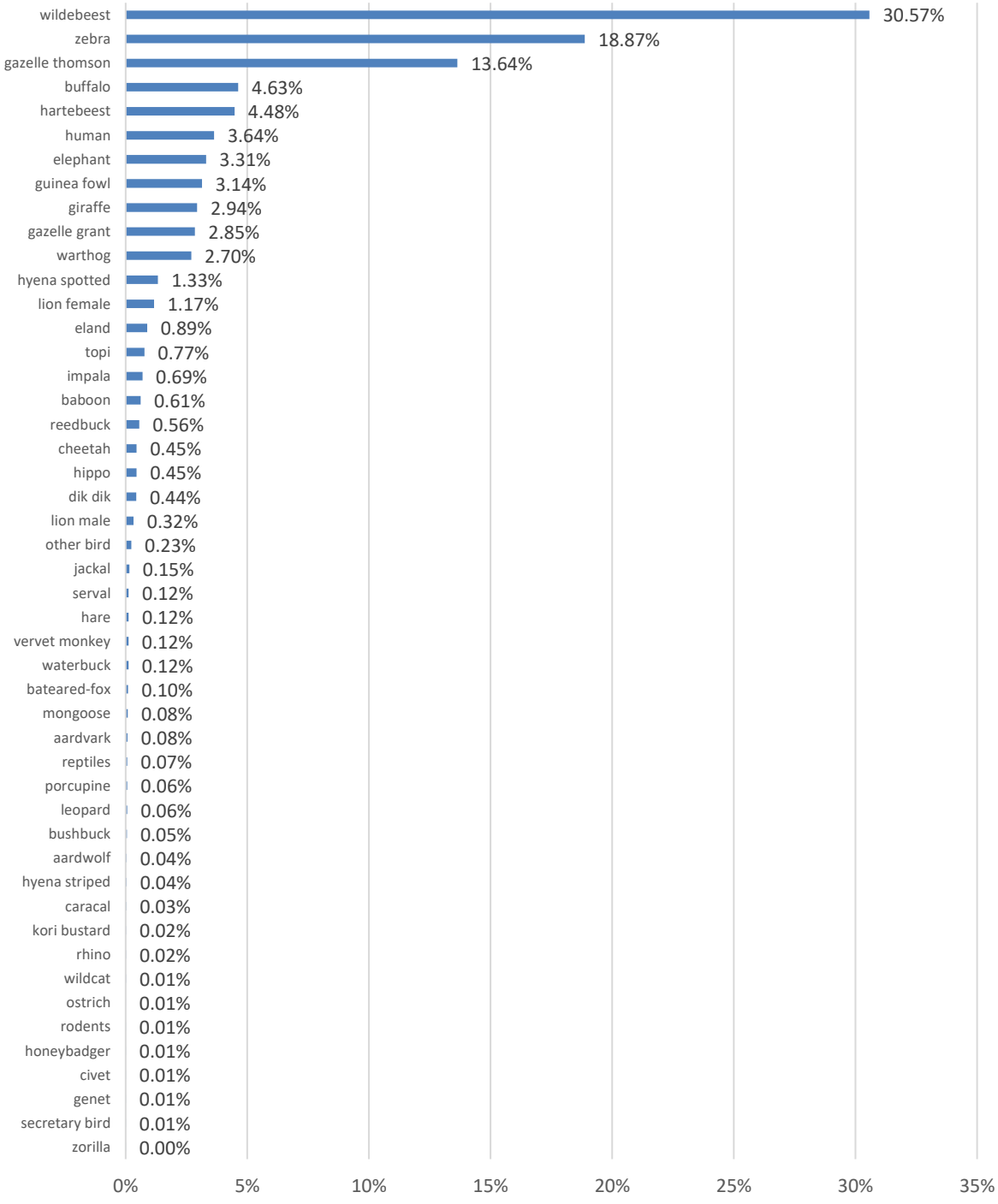


Figure 89: Class imbalance of Snapshot Serengeti dataset

Training

All pretrained models were trained on a subset of ImageNet dataset. The SS dataset was used for training across all four models. The mini-batch sizes were kept constant at 128 and the number of epochs were kept constant at ten for all the experiments. Preliminary

models that were run for longer epochs, i.e. 20 and 30 showed no significant improvement in accuracy for longer training times. Hence, the number of epochs was kept constant at ten epochs. In addition, this reduced the computational time and resources required to run the various models.

All models were run using stochastic gradient descent optimisation algorithms with a constant momentum of 0.09 and L2 regularisation factor of 0.0005. This was consistent with the parameters used in the study conducted by Norouzzadeh et al. (2017). The learning decay policy was constant at 10 for all models. The training parameters for each pretrained model for Experiments 1, 2 and 3 are shown in Table 10, Table 11 and Table 12, respectively.

Table 10: Experiment 1 with training parameters

		Parameters						
	Model	Drop-out	Initial learning rate	Weighted learning factor	Bias learning factor	Learning drop rate factor	Learn rate drop factor	**Technique**
Experiment 1A	AlexNet	0.4	0.005	10	10	4	0.2	Freeze: layers 1-10
	GoogLeNet	0.5						Freeze: layers 1-110
	ResNet-101	batch norm.						Freeze: layers 1– 251
	VGG-19	0.5						Freeze: layers 1-22
Experiment 1B	AlexNet	0.4	0.001	10	10	4	0.2	Freeze: layers 1-10
	GoogLeNet	0.5						Freeze: layers 1-110
	ResNet-101	batch norm.						Freeze: layers 1– 251
	VGG-19	0.5						Freeze: layers 1-22

Table 11: Experiment 2 with training parameters

Model	**Parameters**

		Drop-out	Initial learning rate	Weighted learning factor	Bias learning factor	Learning drop rate factor	Learn rate drop factor	Technique
Experiment 2A	AlexNet	0.4	0.005	20	20	4	0.2	Fine-tuning
	GoogLeNet	0.5						
	ResNet-101	batch norm.						
	VGG-19	0.5						
Experiment 2B	AlexNet	0.4	0.001	20	20	4	0.2	Fine-tuning
	GoogLeNet	0.5						
	ResNet-101	batch norm.						
	VGG-19	0.5						

Table 12: Experiment 3 with training parameters

		Parameters						
	Model	Drop-out	Initial learning rate	Weighted learning factor	Bias learning factor	Learning drop rate factor	Learn rate drop factor	**Technique**
Experiment 3	AlexNet	0.4	0.001	20	20	4	0.2	Fine-tuning
	GoogLeNet	0.5						
	ResNet-101	batch norm.						
	VGG-19	0.5						

The number of iterations for each model was kept consistent with the balanced training dataset that was created using the techniques outlined in section 3.3.2. The calculation for the number of iterations was as follows:

$$size\ per\ mini\ batch = \frac{number\ of\ training\ images}{mini\ batch\ size}$$

$$number\ of\ iterations = size\ per\ mini\ batch\ x\ number\ of\ epochs$$

The learning rate was reduced every four epochs by a factor of 0.2 for Experiment 1 and by a factor of 0.1 for Experiments 2 and 3. Below is a sample calculation for the learn rate calculation:

$$initial\ learning\ rate = 0.01$$

$$learn\ rate\ drop\ factor = 0.1$$

$$learning\ drop\ rate\ factor = 4\ epochs$$

$$number\ of\ epochs = 10$$

$For\ epochs\ 1 - 4$:

$$learning\ rate\ 1 = initial\ learning\ rate$$

$$learning\ rate\ 1 = 0.01$$

$For\ epochs\ 5 - 8$:

$$learning\ rate\ 2 = learning\ rate\ 1\ x\ learn\ rate\ drop\ factor$$

$$learning\ rate\ 2 = 0.01\ x\ 0.1 = 0.001$$

$For\ epochs\ 9 - 10$:

$$learning\ rate\ 3 = learning\ rate\ 2\ x\ learn\ rate\ drop\ factor$$

$$learning\ rate\ 3 = 0.001\ x\ 0.1 = 0.0001$$

Validation

Validation was calculated using mini-batch accuracy. This corresponds to the accuracy of that mini batch at each iteration. The validation dataset was taken from the SS dataset using 7% of the training dataset.

Testing

The test dataset was comprised of images from the SS, AWF and Greenpeace datasets. The top-1 and top-5 performance metrics were used to determine how well the models were able to correctly classify the species from the test set. Using additional methods to

balance the rare classes, the models were run again, and test results were compared to determine the improvement in prediction accuracy.

4 RESULTS

This chapter highlights the research findings and results.

4.1 Model Experiments

The results will be organised as follows. First, the training and validation results will be shown for all experiments. This will be followed by test results for the experiments. Each experiment was run on the AlexNet, GoogLeNet, ResNet-101 and VGG-19 pretrained networks. Table 13 summarises the hyperparameters used in Experiments 1, 2 and 3.

Table 13: Summary of hyperparameters for Experiments 1, 2 and 3

	Experiment				
Parameters	**1A**	**1B**	**2A**	**2B**	**3**
initial learning rate	0.005	0.001	0.005	0.001	0.001
momentum	0.009				
L2 reg	0.0005				
learning decay	10				
weighted learning factor	10	10	20	20	20
bias learning factor	10	10	20	20	20
learning drop rate factor	4				
freeze layers	YES	YES	NO	NO	NO
learn rate drop factor	0.2	0.1	0.2	0.1	0.1

4.2 Training & Validation Results

This section will explain the training and validation results by experiment. The SS dataset was used for training and validation.

4.2.1 Experiment 1 results

Experiment 1 used the technique of freezing layers during the model development.

Experiment 1A

Experiment 1A used a learning rate of 0.005 and a learn rate drop factor of 0.2 for every four epochs. The momentum and L2 regularisation were kept constant at 0.09 and 0.0005, respectively.

AlexNet

Figure 90 shows the accuracy curve for the AlexNet model, which peaked at an accuracy of 88.5%.

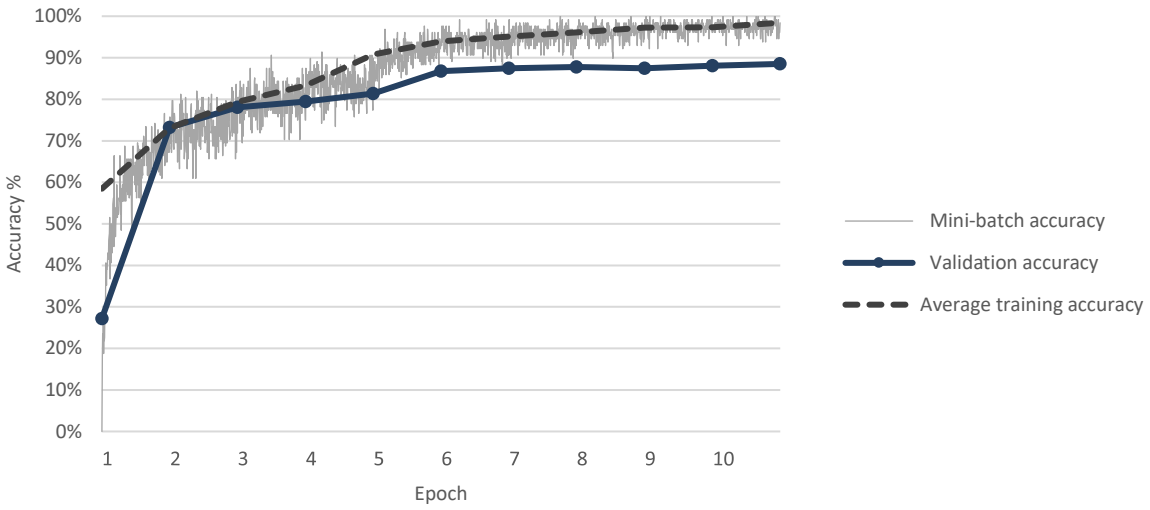


Figure 90: Experiment 1A - AlexNet accuracy curve

The loss curve for the AlexNet model is shown in

Figure 91.

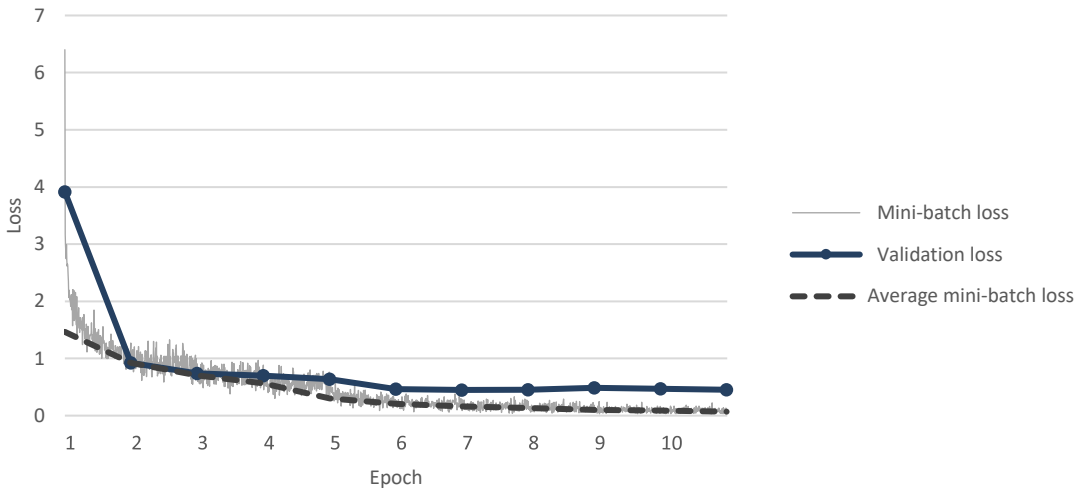


Figure 91: Experiment 1A - AlexNet loss curve

GoogLeNet

The overall validation accuracy for the GoogLeNet model for Experiment 1A was 82% as displayed in Figure 92.

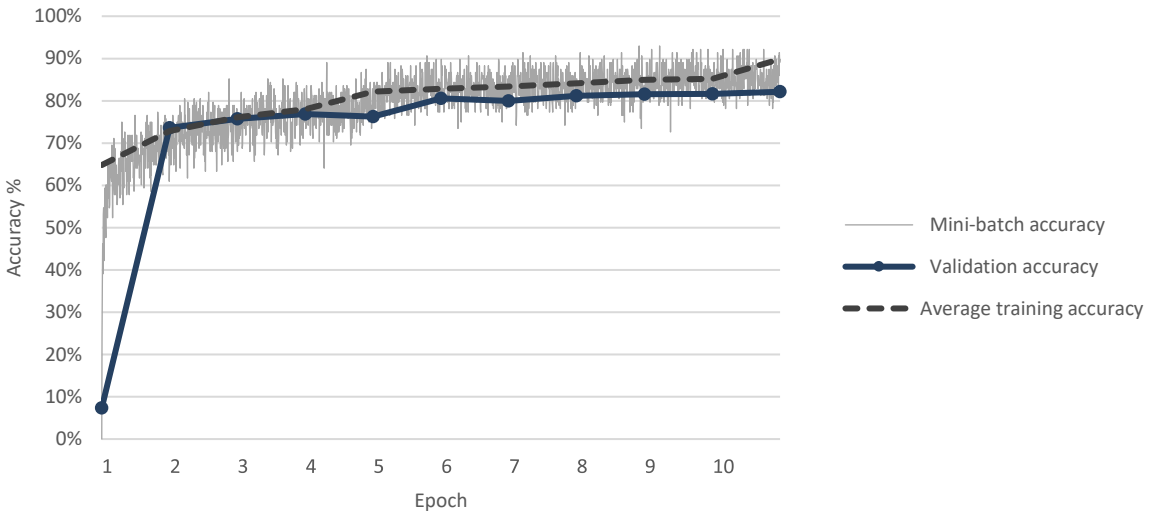


Figure 92: Experiment 1A - GoogLeNet accuracy curve

The loss curve is shown in Figure 93 for the GoogLeNet model.

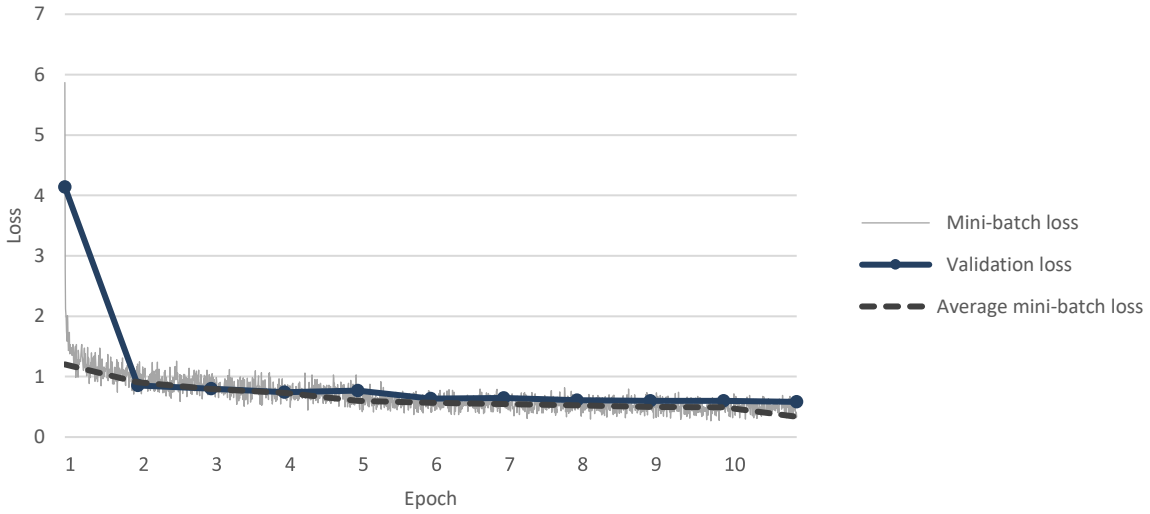


Figure 93: Experiment 1A - GoogLeNet loss curve

ResNet-101

The overall validation accuracy for the ResNet-101 model in this experiment was 92.9%. Figure 94 displays the accuracy curve over the ten epochs.

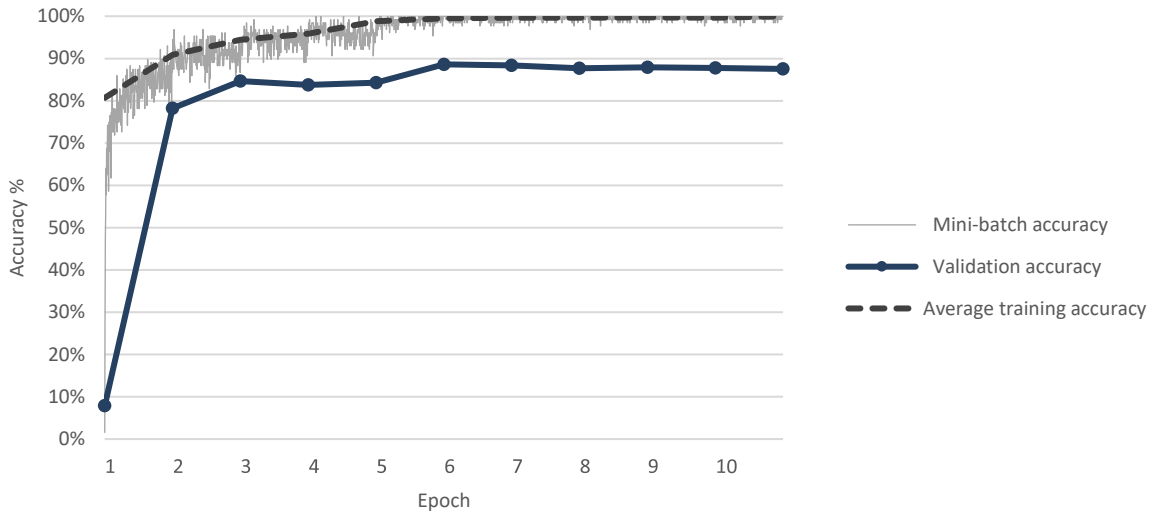


Figure 94: Experiment 1A - ResNet-101 accuracy curve

The loss curve for this model is shown in Figure 95.

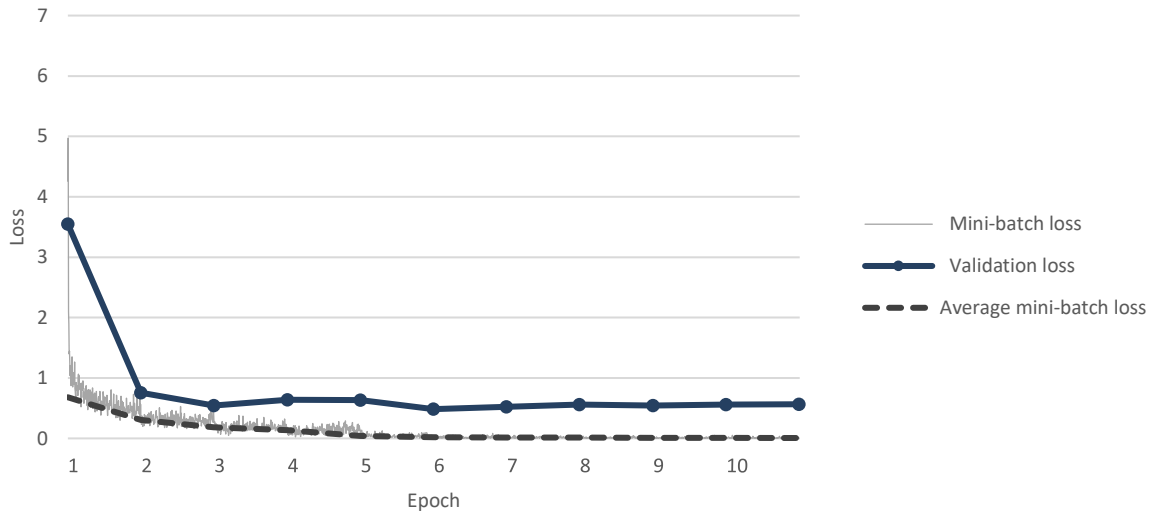


Figure 95: Experiment 1A - ResNet-101 loss curve

VGG-19

This model did not converge to a result and presented an output which was undefined as not a number, or NaN.

Experiment 1B

Experiment 1B used a learning rate of 0.001 and a learn rate drop factor of 0.1 for every four epochs. The momentum and L2 regularisation were kept constant at 0.09 and 0.0005, respectively.

AlexNet

The AlexNet model achieved an overall validation accuracy of 86.8%. Figure 96 illustrates the accuracy curve for the model.

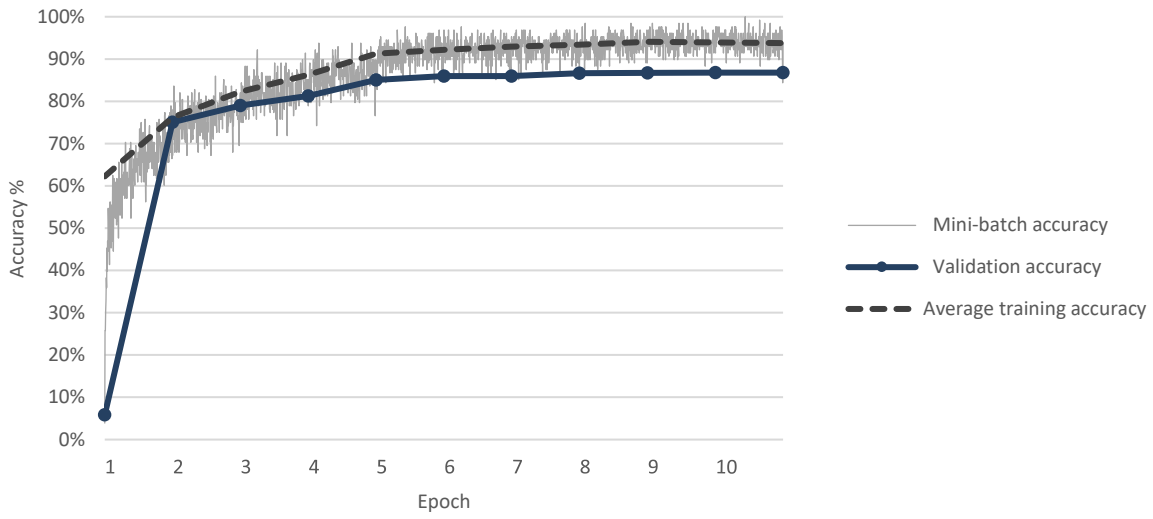


Figure 96: Experiment 1B - AlexNet accuracy curve

The loss curve is shown in Figure 97.

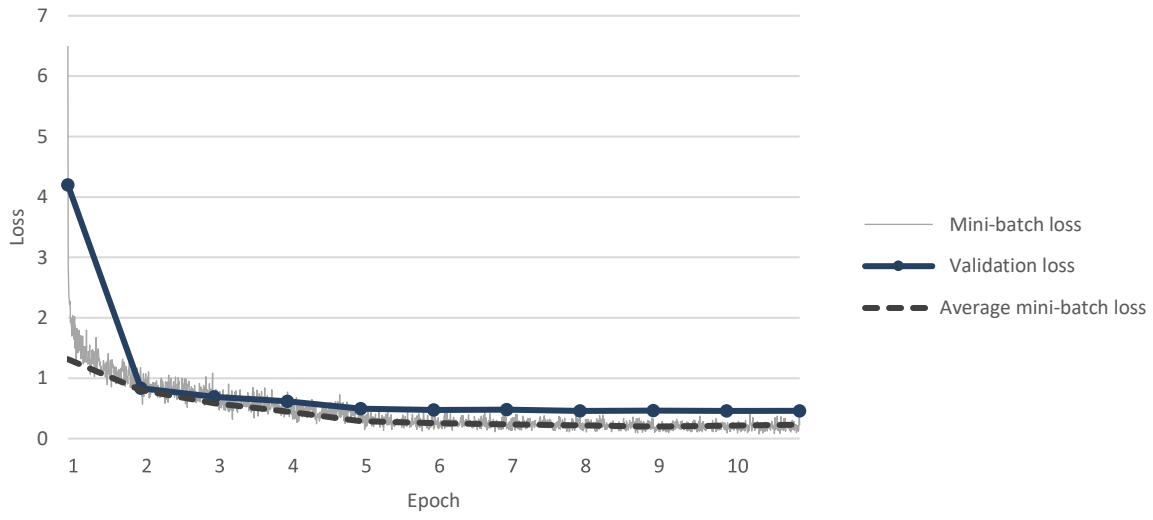


Figure 97: Experiment 1B - AlexNet loss curve

GoogLeNet

The overall validation accuracy for the GoogLeNet model for Experiment 1B was 76.3%. Figure 98 displays the accuracy curve over the ten epochs.

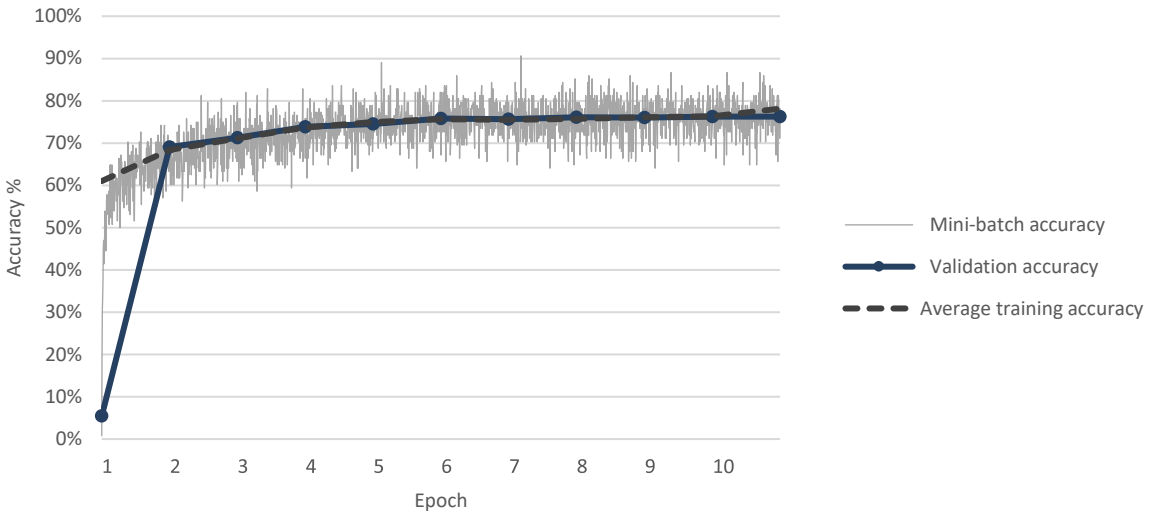


Figure 98: Experiment 1B - GoogLeNet accuracy curve

The loss curve in Figure 99 is shown for the GoogLeNet model.

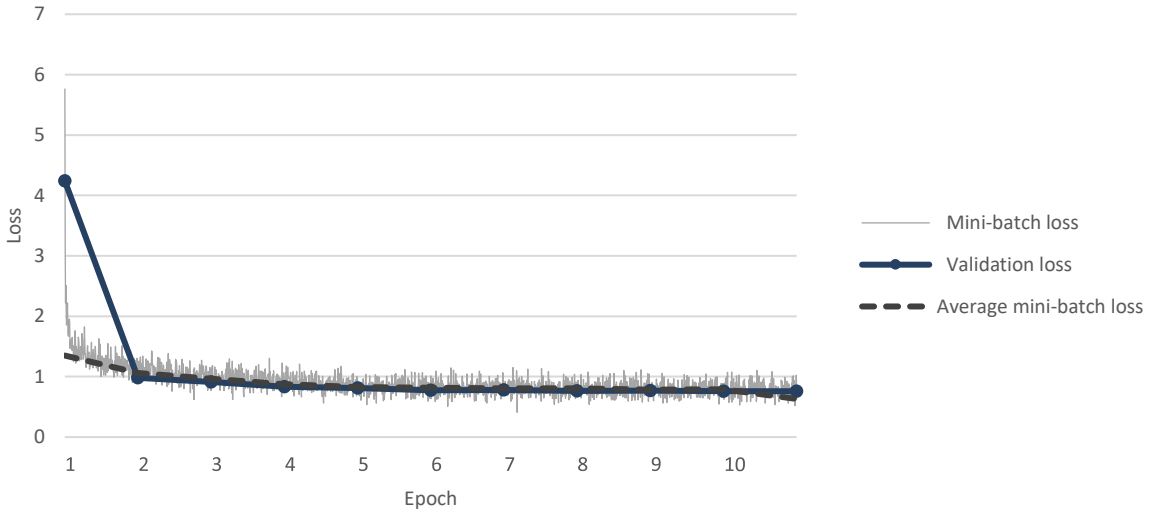


Figure 99: Experiment 1B - GoogLeNet loss curve

ResNet-101

The accuracy curve in Figure 100 shows that the model achieved an overall validation accuracy of 85.8%.

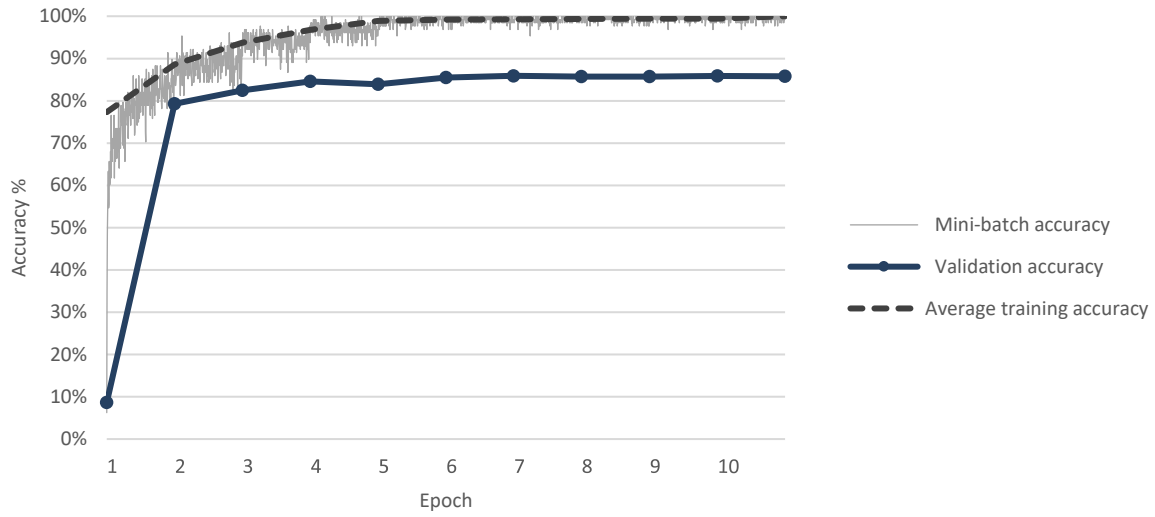


Figure 100: Experiment 1B - ResNet-101 accuracy curve

The loss curve is shown in Figure 101.

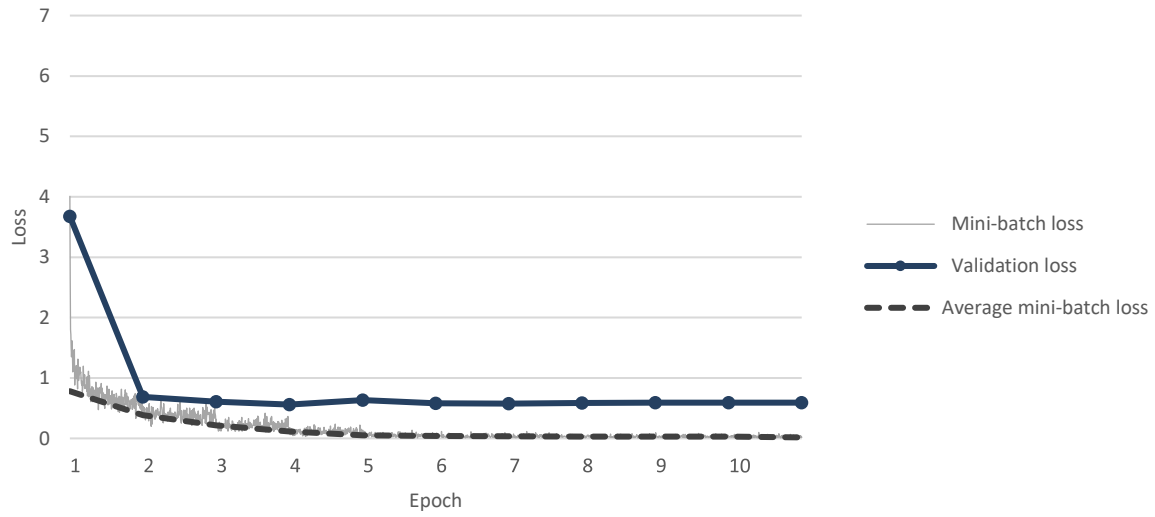


Figure 101: Experiment 1B - ResNet-101 loss curve

VGG-19

The overall validation accuracy for the VGG-19 model for this experiment was 92.7%. Figure 102 displays the accuracy curve over the ten epochs.

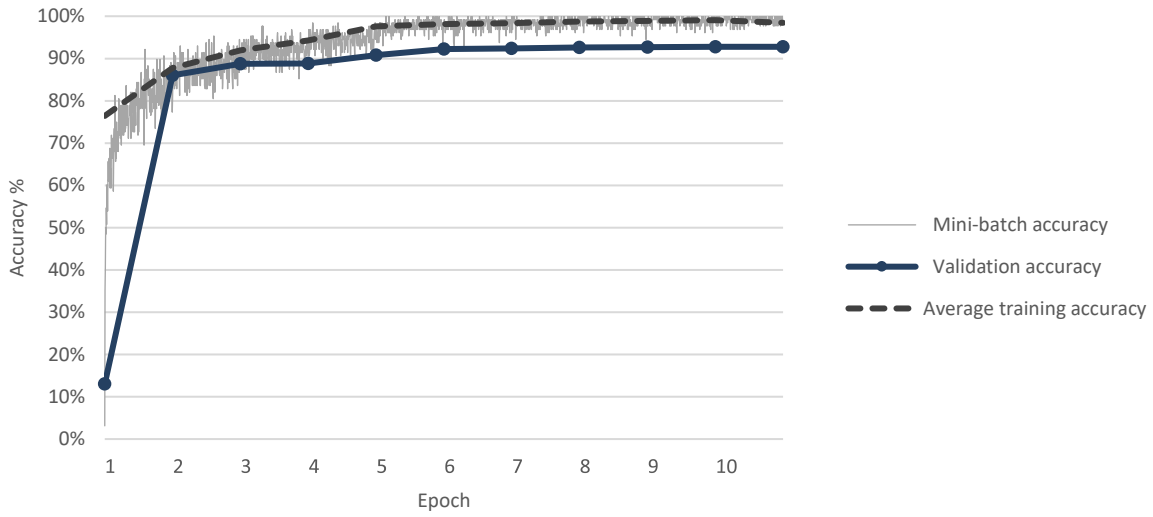


Figure 102: Experiment 1B - VGG-19 accuracy curve

The loss curve for the VGG-19 model is shown in Figure 103.

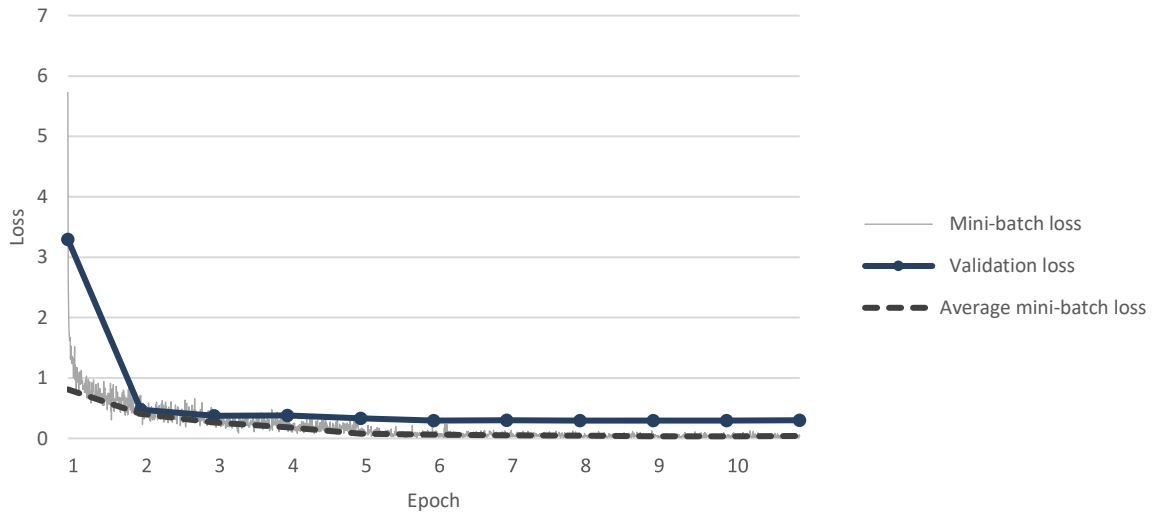


Figure 103: Experiment 1B - VGG-19 loss curve

4.2.2 Experiment 2 results

Experiment 2 used the technique of fine-tuning layers during the model development.

Experiment 2A

Experiment 2A used a learning rate of 0.005 and a learn rate drop factor of 0.2 for every four epochs. The momentum and L2 regularisation were kept constant at 0.09 and 0.0005, respectively.

AlexNet

This model did not converge to a result and presented an output which was undefined as not a number, or NaN.

GoogLeNet

The overall validation accuracy for the GoogLeNet model for Experiment 2A was 91.2% and is shown in Figure 104.

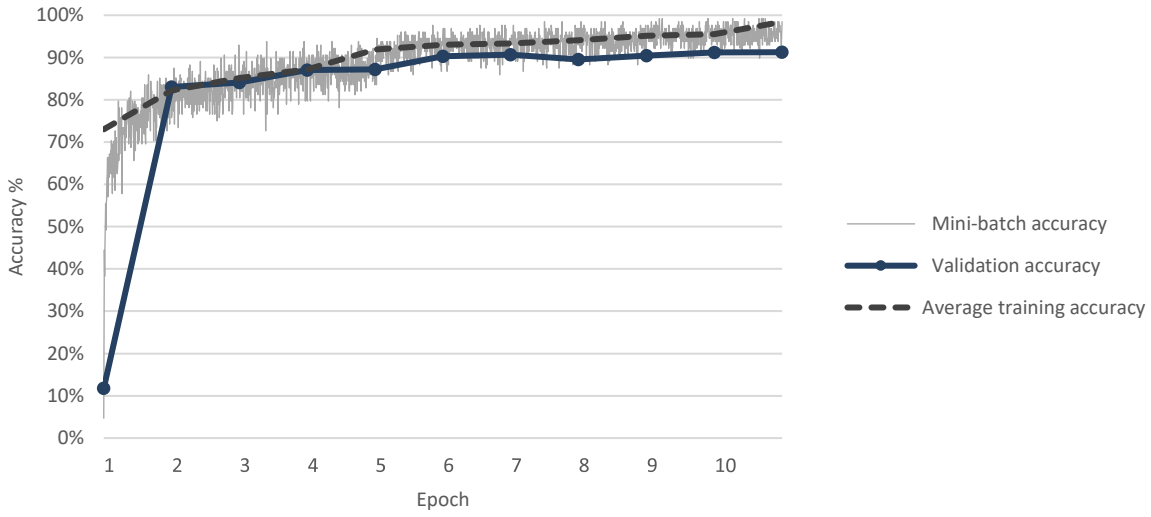


Figure 104: Experiment 2A - GoogLeNet accuracy curve

Figure 105 shows the loss curve for this model.

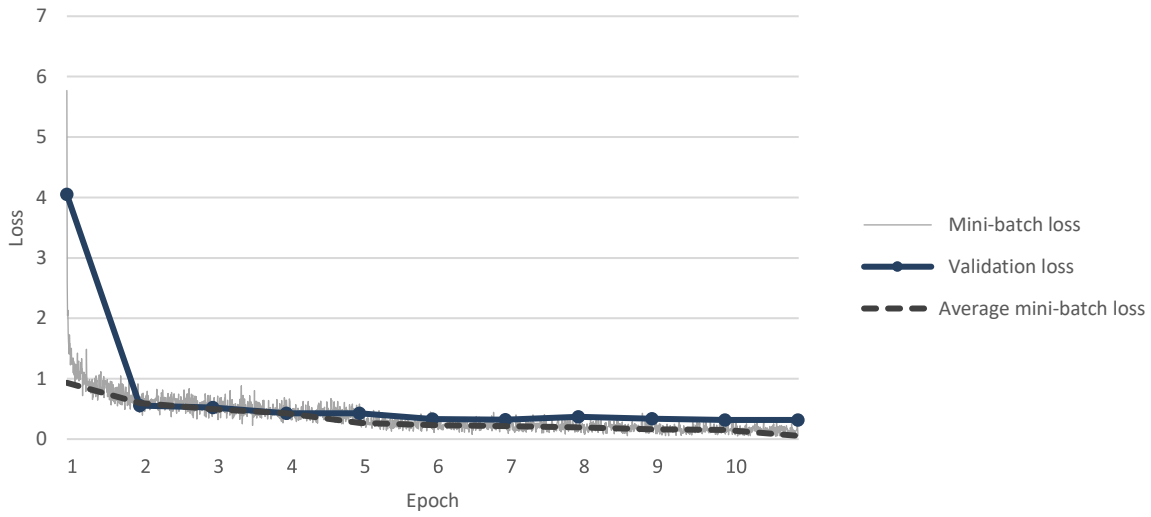


Figure 105: Experiment 2A - GoogLeNet loss curve

ResNet-101

The accuracy curve in Figure 106 shows that the model achieved an overall validation accuracy of 90.1%.

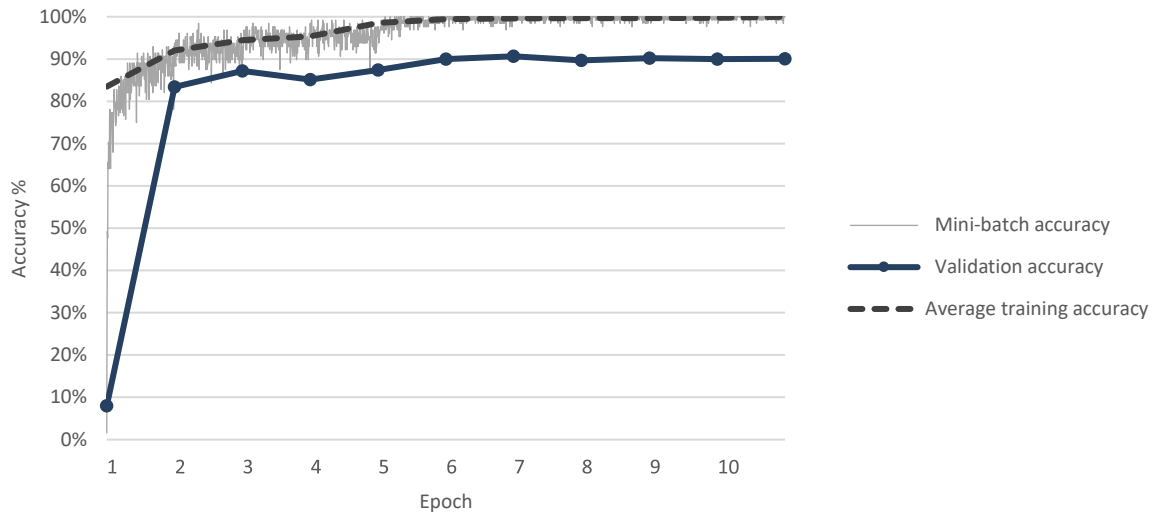


Figure 106: Experiment 2A - ResNet-101 accuracy curve

The loss curve for the ResNet-101 model is shown in Figure 107.

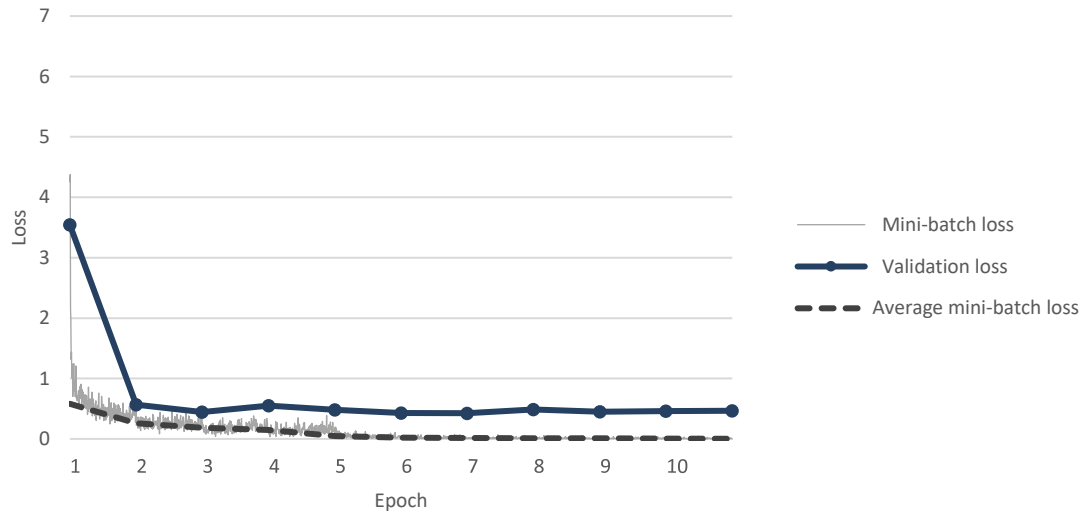


Figure 107: Experiment 2A - ResNet-101 loss curve

VGG-19

The accuracy curve for this model is shown in Figure 108.

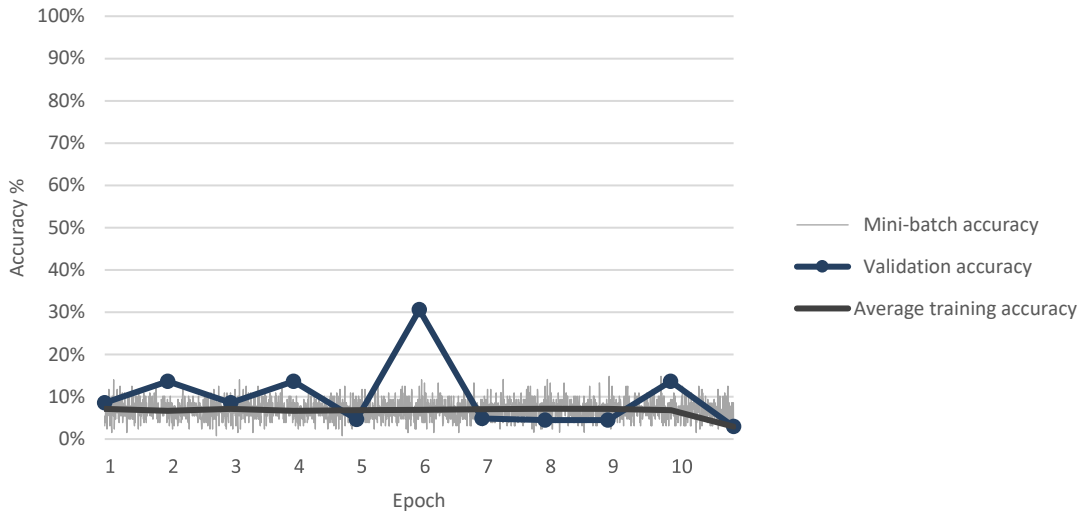


Figure 108: Experiment 2A - VGG-19 accuracy curve

The loss curve for this model is shown in Figure 109.

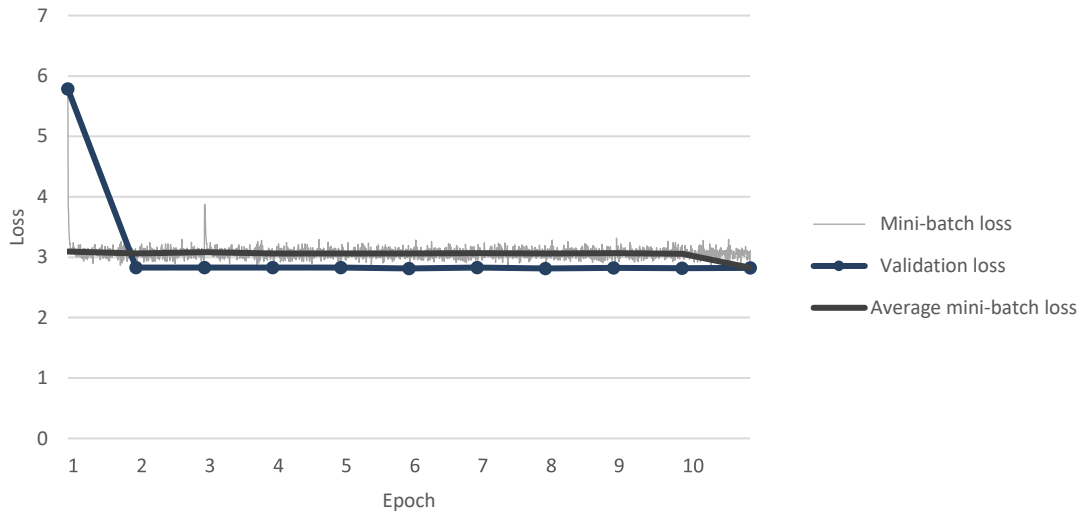


Figure 109: Experiment 2A - VGG-19 loss curve

Experiment 2B

Experiment 2B used a learning rate of 0.001 and a learn rate drop factor of 0.1 for every four epochs. The momentum and L2 regularisation were kept constant at 0.09 and 0.0005, respectively.

AlexNet

The AlexNet model achieved an overall validation accuracy of 88.5%, as shown in Figure 110.

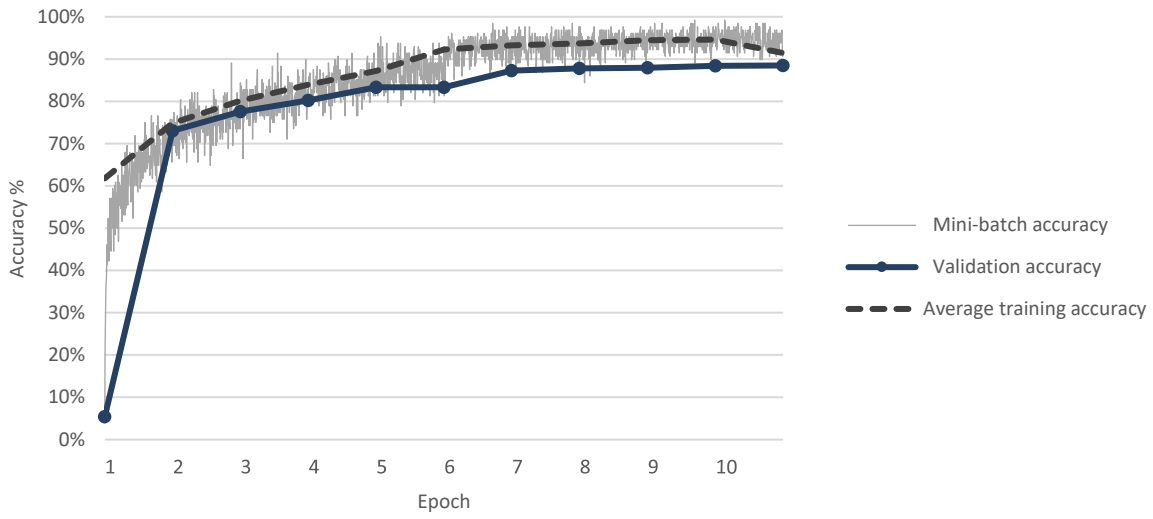


Figure 110: Experiment 2B - AlexNet accuracy curve

The loss curve in Figure 111 illustrates the validation loss achieved.

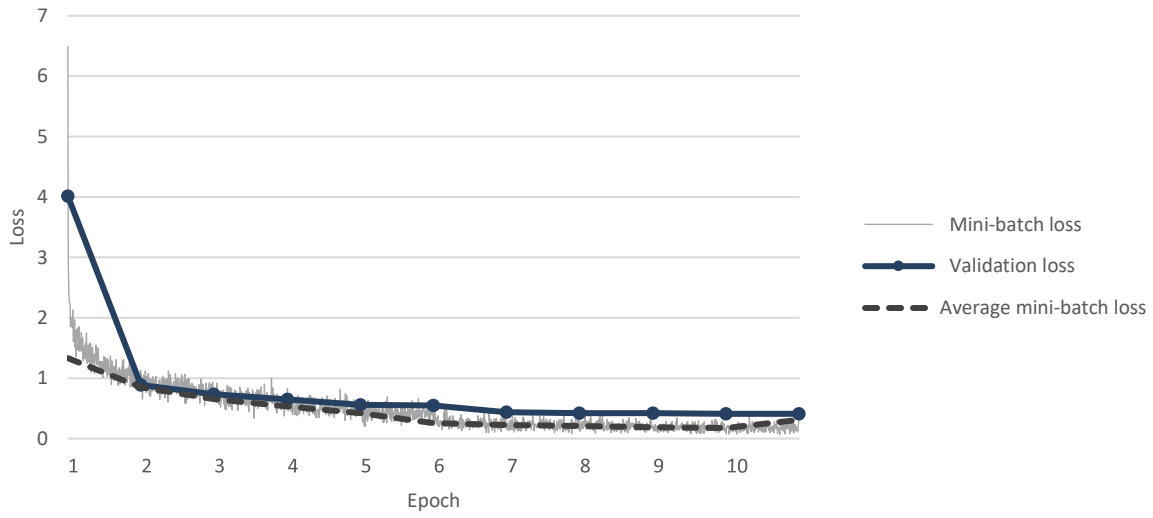


Figure 111: Experiment 2B - AlexNet loss curve

GoogLeNet

The overall validation accuracy for the GoogLeNet model was 88.3%, as shown in Figure 112.

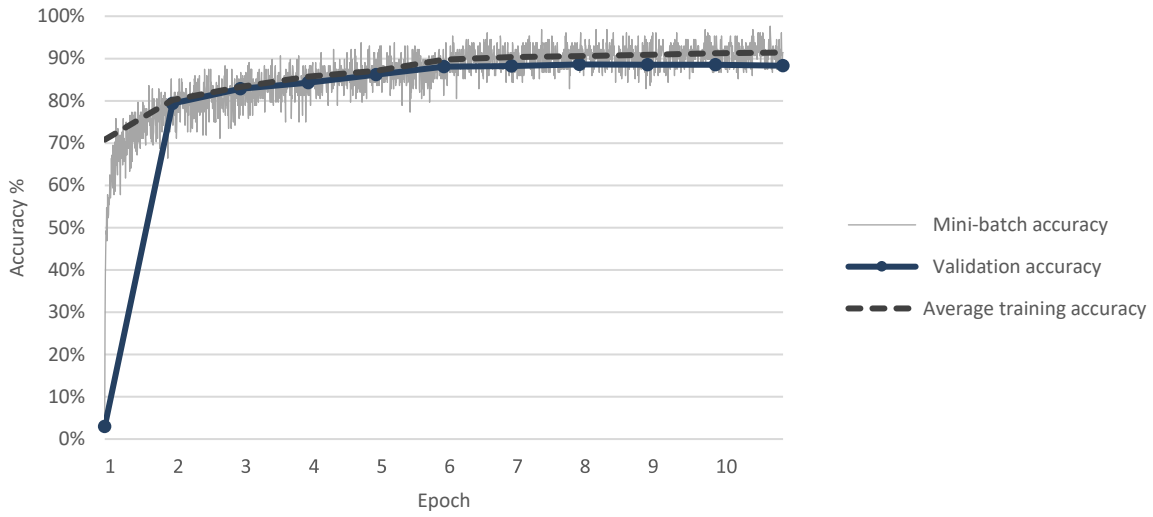


Figure 112: Experiment 2B - GoogLeNet accuracy curve

The loss curve in Figure 113 illustrates the validation loss achieved.

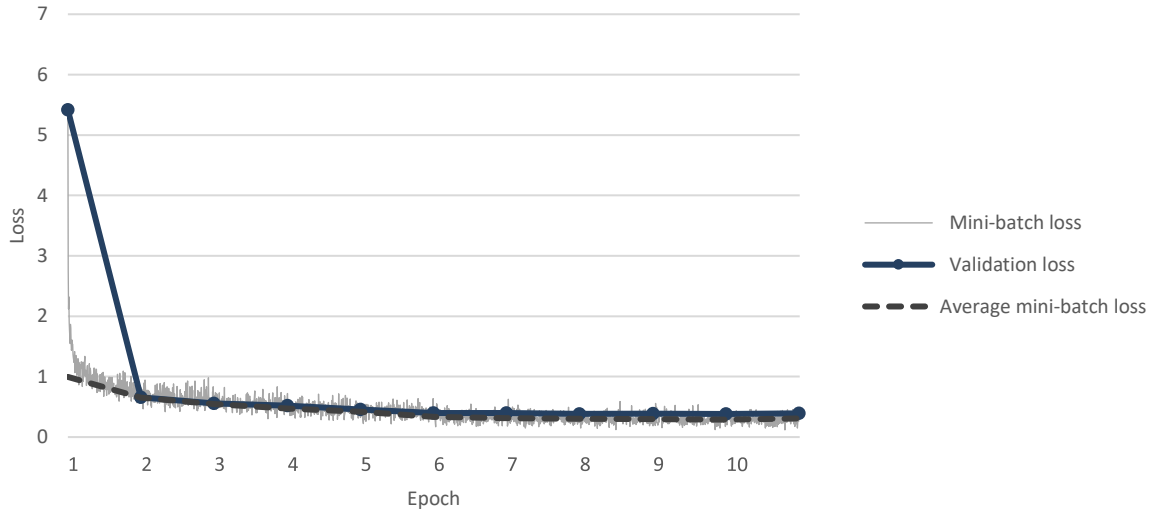


Figure 113: Experiment 2B - GoogLeNet loss curve

ResNet-101

The overall validation accuracy for the ResNet-101 model was 91.9%, as shown in Figure 114.

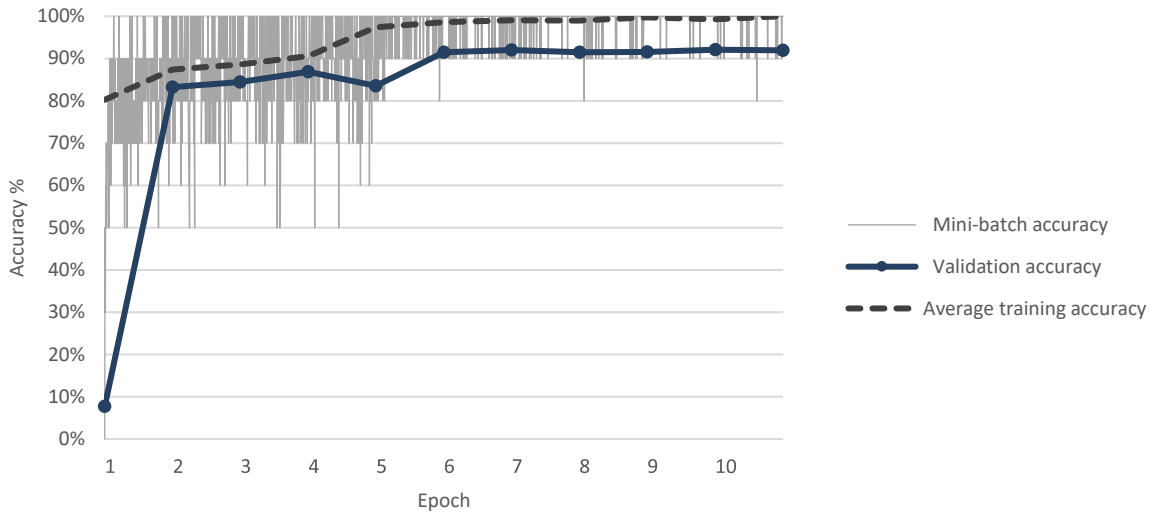


Figure 114: Experiment 2B - ResNet-101 accuracy curve

Figure 115 shows the loss curve for the ResNet-101 model.

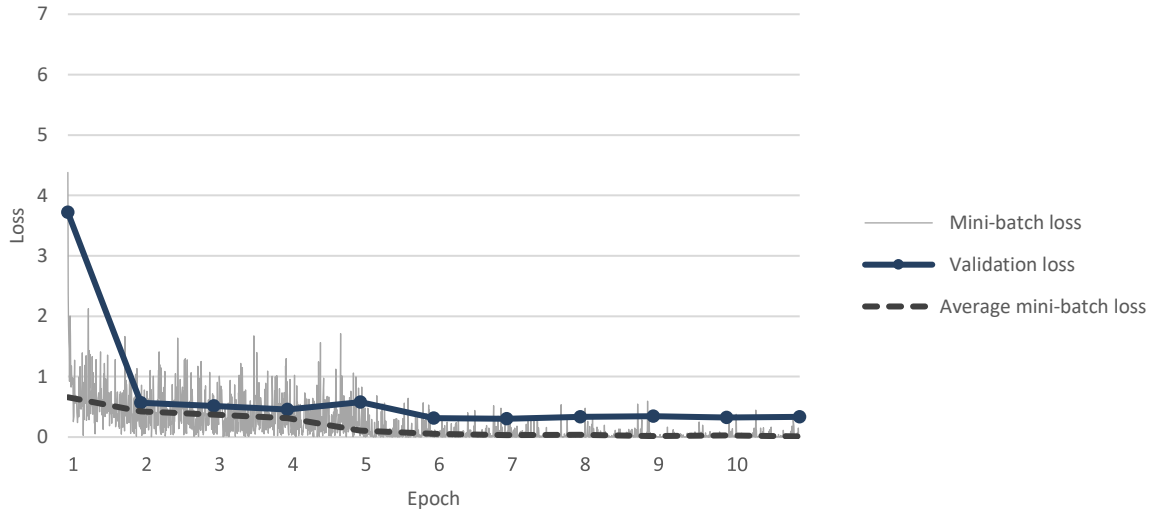


Figure 115: Experiment 2B - ResNet-101 loss curve

VGG-19

The overall validation accuracy for the VGG-19 model was 93.2%, as shown in Figure 116.

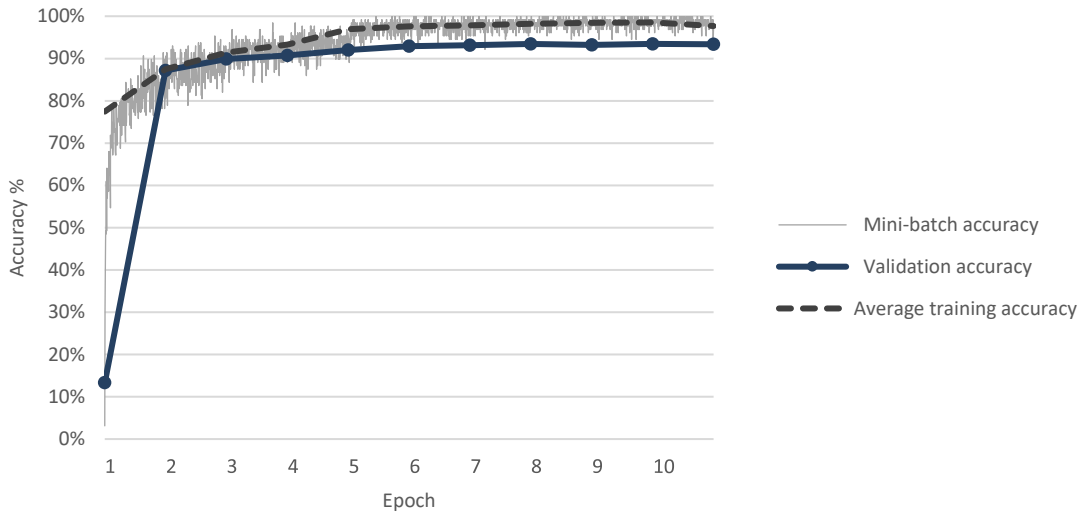


Figure 116: Experiment 2B - VGG-19 accuracy curve

Figure 117 shows the loss curve for the VGG-19 model.

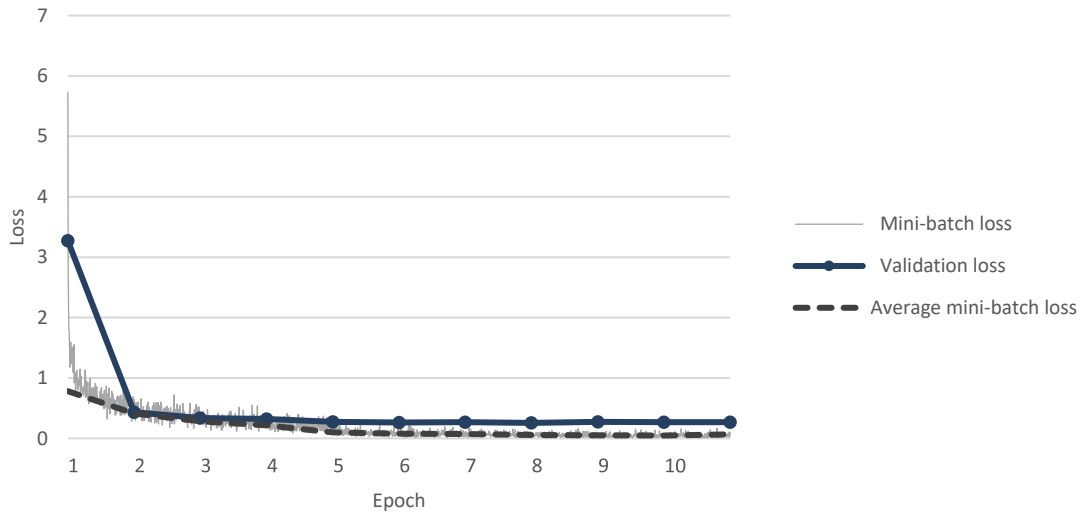


Figure 117: Experiment 2B - VGG-19 loss curve

4.2.3 Experiment 3 results

The models in Experiment 3 used a learning rate of 0.001 and a learn rate drop factor of 0.1 for every four epochs. The momentum and L2 regularisation were kept constant at 0.09 and 0.0005, respectively. Additional class balancing was done by using emphasis sampling of rare classes.

AlexNet

The accuracy curve for this model is shown in Figure 118. The overall validation accuracy for this model was 87.5%.

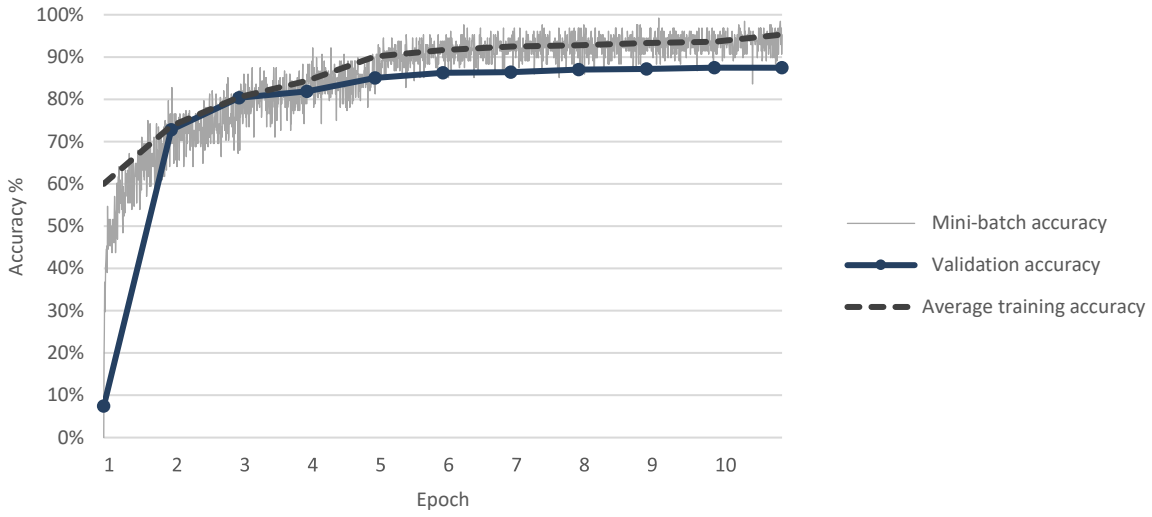


Figure 118: Experiment 3 - AlexNet accuracy curve

The loss curve in Figure 119 shows the loss curve for this model.

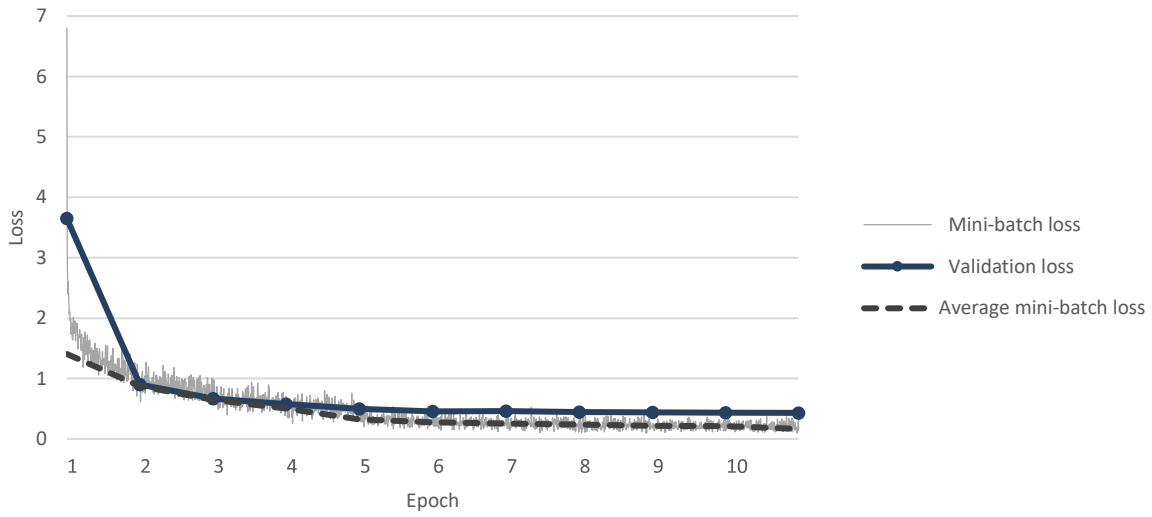


Figure 119: Experiment 3 - AlexNet loss curve

GoogLeNet

The GoogLeNet model achieved an overall validation accuracy of 88.3%, as shown in Figure 120.

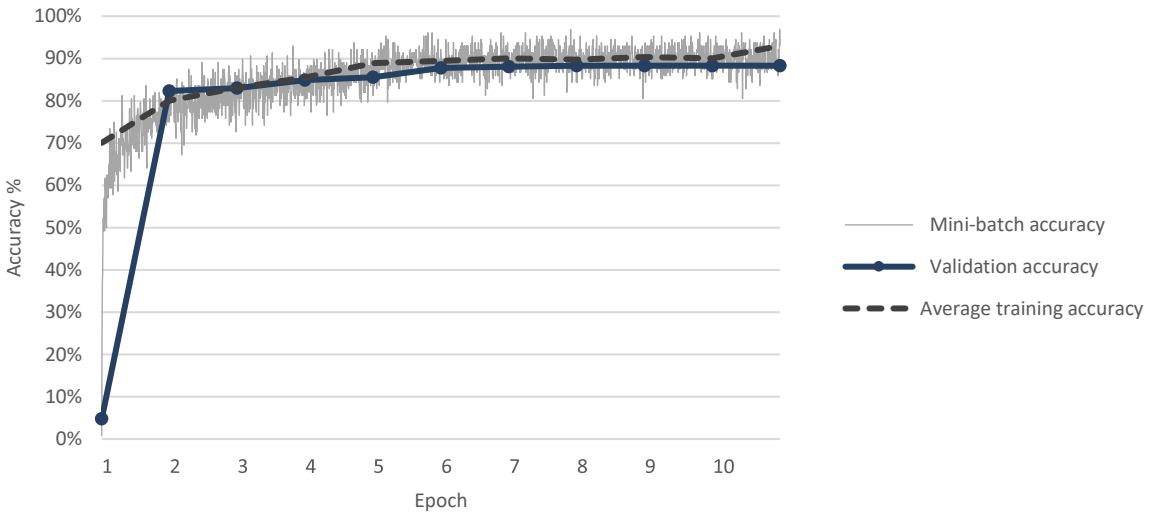


Figure 120: Experiment 3 - GoogLeNet accuracy curve

The loss curve in Figure 121 shows a similar trend as the accuracy curve.

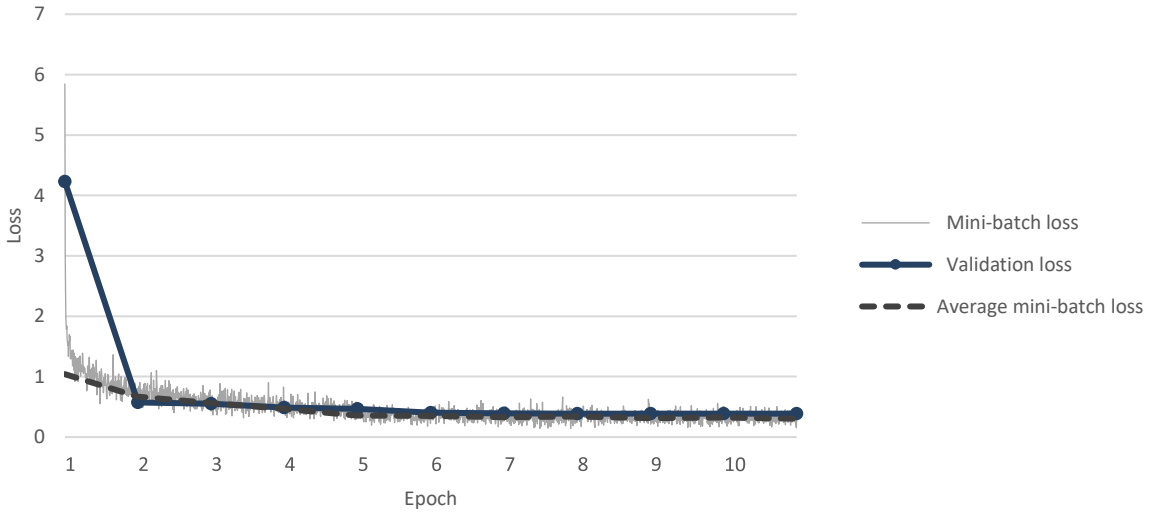


Figure 121: Experiment 3 - GoogLeNet loss curve

ResNet-101

The accuracy curve for this model is shown in Figure 122.

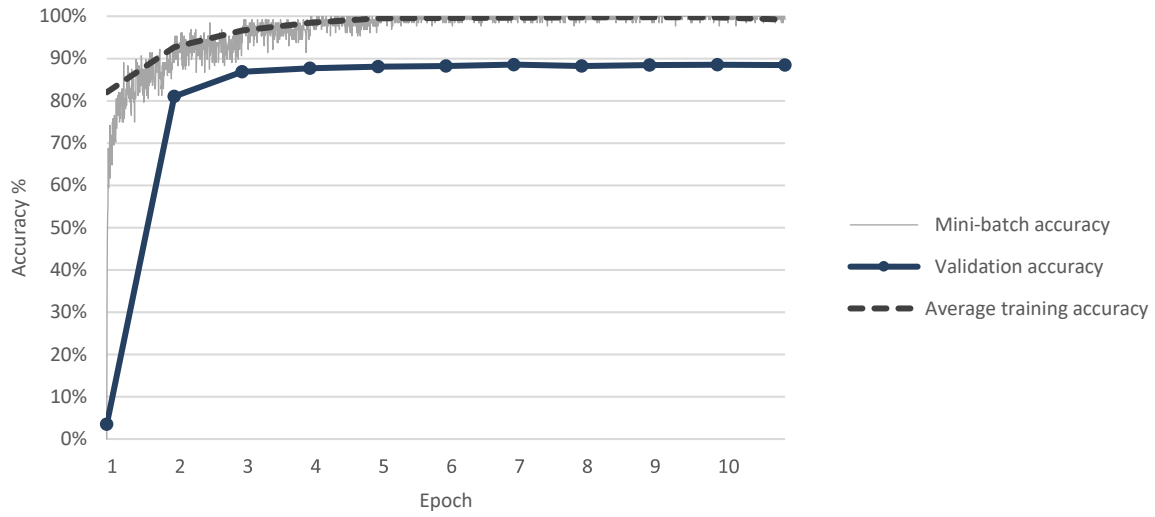


Figure 122: Experiment 3 - ResNet-101 accuracy curve

The loss curve is shown in Figure 123.

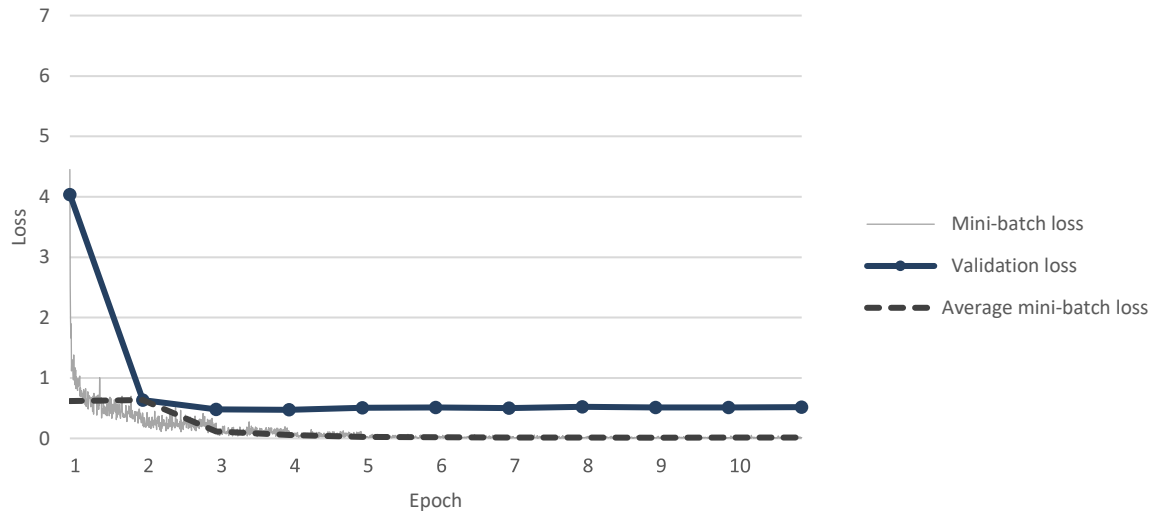


Figure 123: Experiment 3 - ResNet-101 loss curve

VGG-19

The VGG-19 model achieved a model accuracy of 93.7%, as shown in Figure 124.

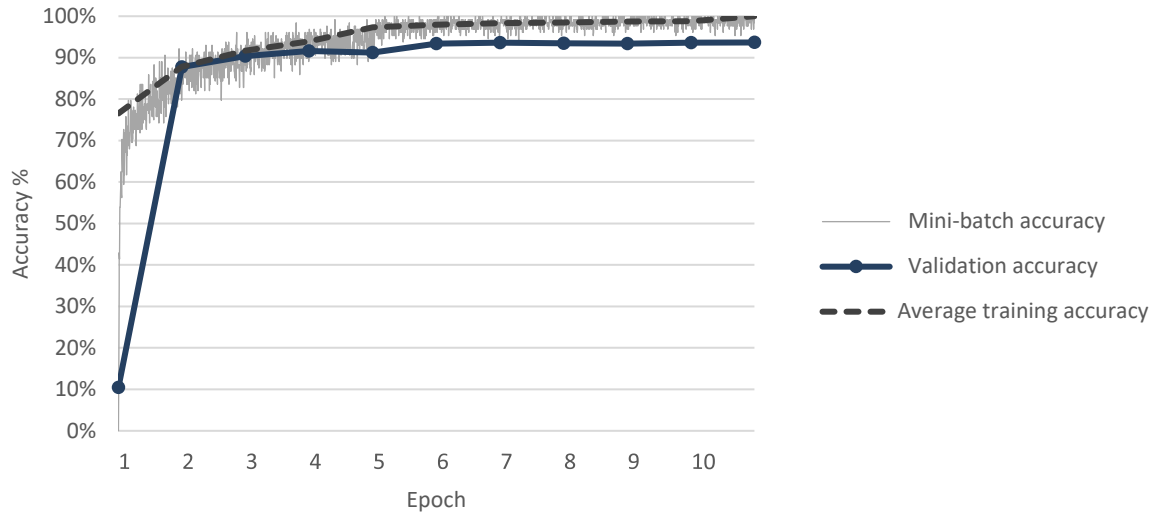


Figure 124: Experiment 3 - VGG-19 accuracy curve

The loss curve in Figure 125 corresponds to the accuracy curve trend.

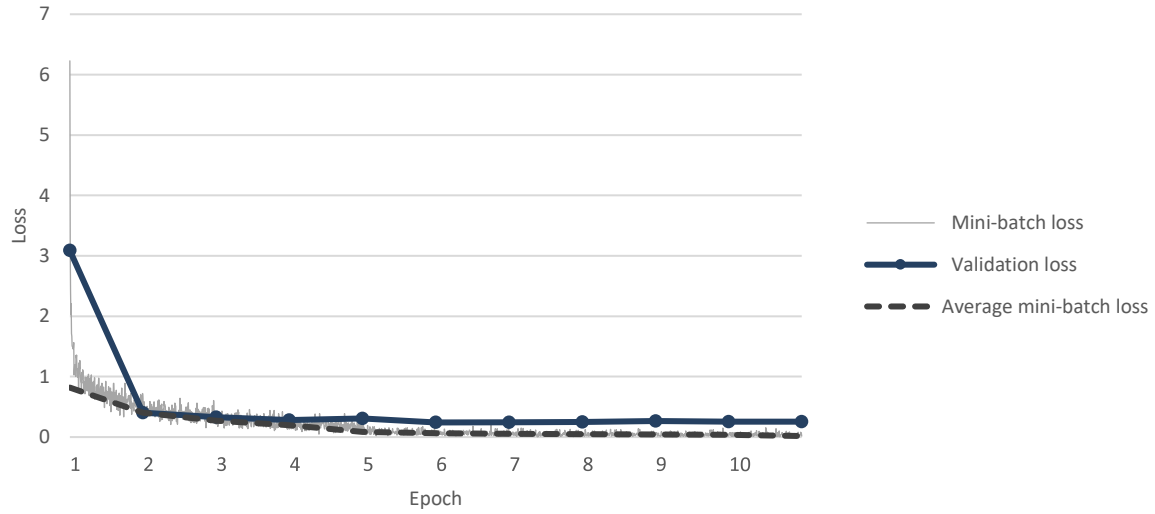


Figure 125: Experiment 3 - VGG-19 loss curve

4.3 Test Results

This section will explain the test results by experiment. The SS, AWF and Greenpeace datasets were used for testing.

4.3.1 Experiment 1

Experiment 1 used the technique of freezing layers during the model development.

Experiment 1A

AlexNet

The top-1 and top-5 accuracy of the AlexNet model were 85.4% and 89.6%, respectively. The precision and recall values were 65.0% and 53.3%, respectively. Figure 126 shows four examples of the true versus predicted classifications and the confidence levels for the AlexNet model. The complete set of examples can be found in Appendix A Figure 146.

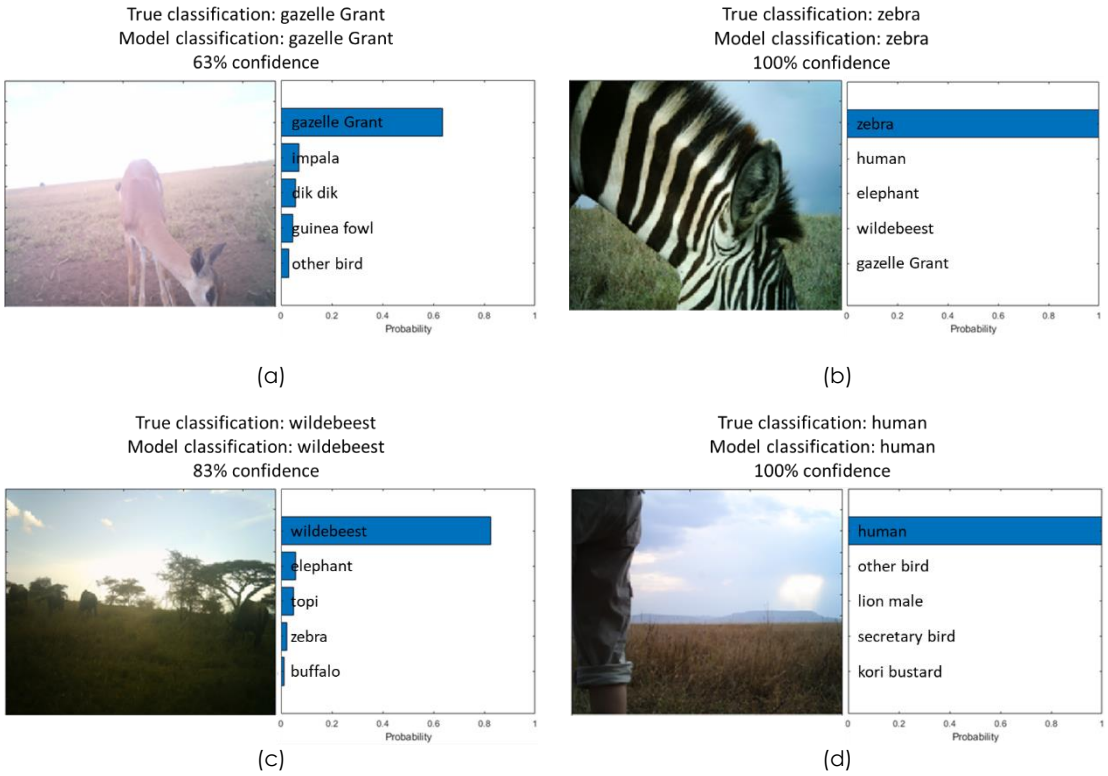


Figure 126: Confidence levels for classification examples (Experiment 1A - AlexNet)

The weighted calculation of precision and recall resulted in an f-score of 0.6645. Table 14 provides a summary of the results for the AlexNet model.

Table 14: Results for Experiment 1A - AlexNet

Top-1 Accuracy	85.4%
Top-5 Accuracy	89.6%
Precision	65.0%
Recall	68.0%
F-score	0.6645

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 164.

GoogLeNet

Results from the GoogLeNet model had a top-1 accuracy of 79.2% and a top-5 accuracy of 87.5%, respectively. The precision and recall values were 53.3% and 61.6%, respectively. Examples of true versus predicted classifications and the confidence levels for the GoogLeNet model are shown in Figure 127. The complete set of examples can be found in Appendix A Figure 147.

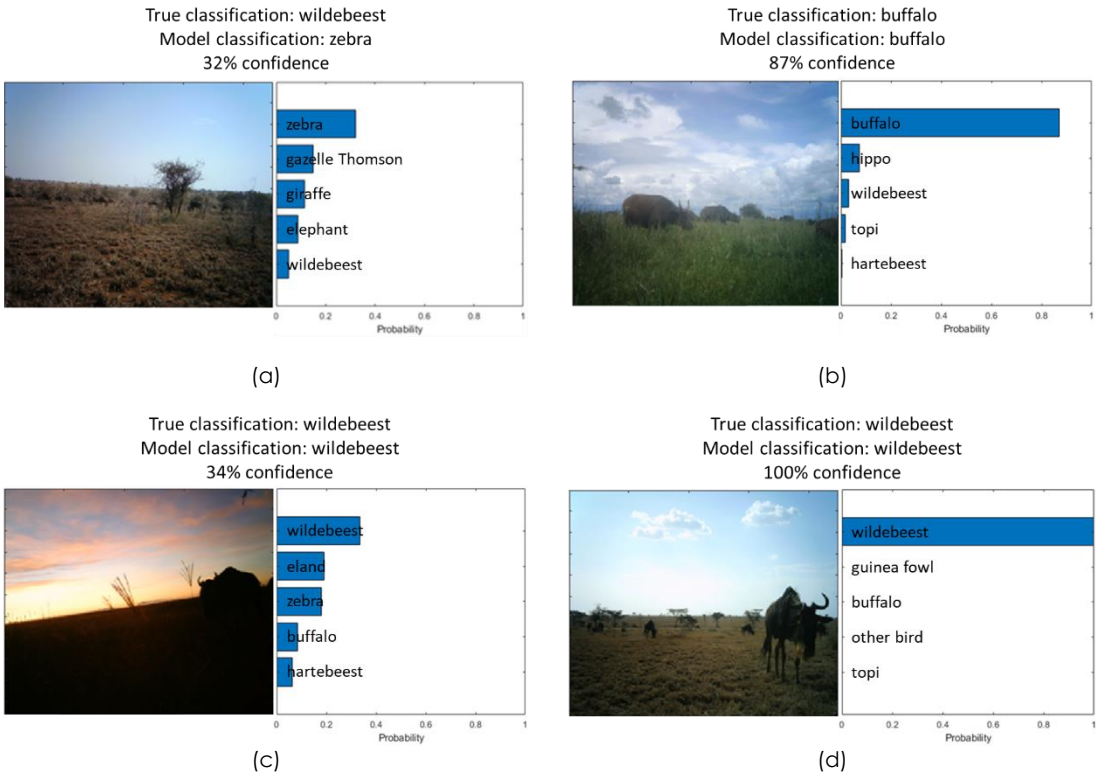


Figure 127: Confidence levels for classification examples (Experiment 1A - GoogLeNet)

The weighted calculation of precision and recall resulted in an f-score of 0.5717. Table 15 provides a summary of the results for the GoogLeNet model.

Table 15: Results for Experiment 1A - GoogLeNet

Top-1 Accuracy	79.2%
Top-5 Accuracy	87.5%
Precision	53.3%
Recall	61.6%
F-score	0.5717

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 165.

ResNet-101

Results from the ResNet-101 model had a top-1 and top-5 accuracy of 100%, respectively. The precision and recall values were 81.5% and 82.6%, respectively. Figure 128 shows examples of classification and the confidence levels. The complete set of examples can be found in Appendix A Figure 148.

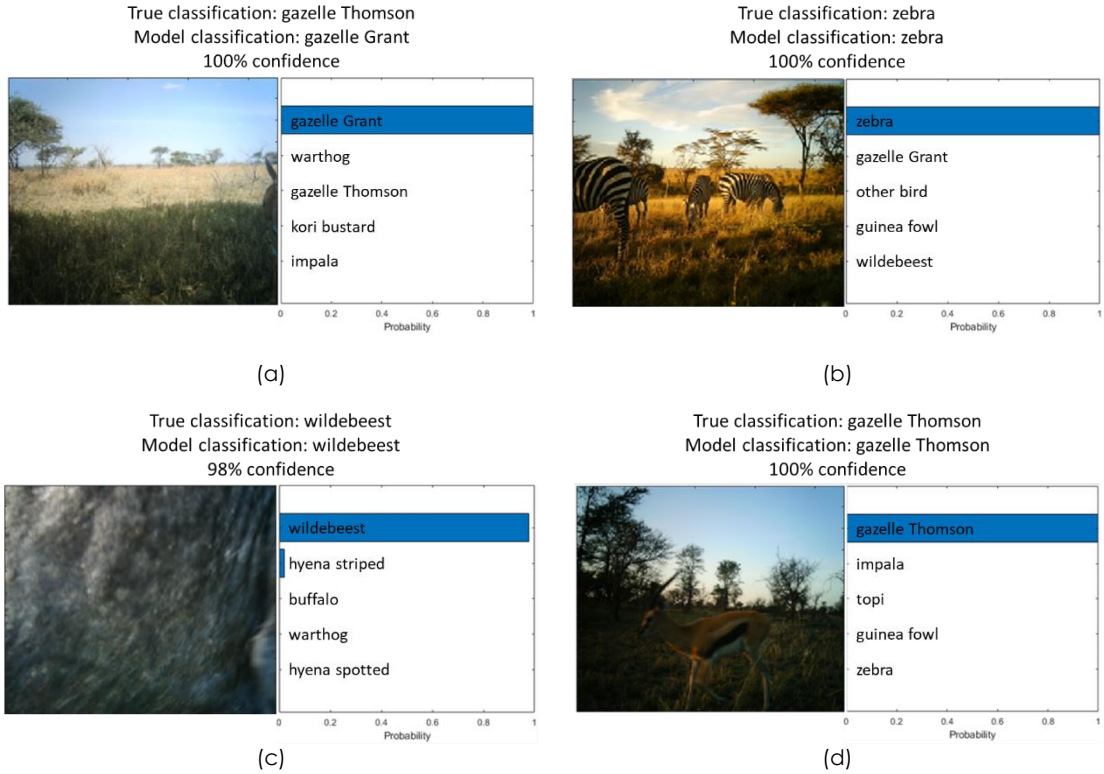


Figure 128: Confidence levels for classification examples (Experiment 1A - ResNet-101)

The weighted calculation of precision and recall resulted in an f-score of 0.8205. Table 16 provides a summary of the results for this model.

Table 16: Results for Experiment 1A - ResNet-101

Top-1 Accuracy	100.0%
Top-5 Accuracy	100.0%
Precision	81.5%
Recall	82.6%
F-score	0.8205

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 166.

VGG-19

This model did not converge to a result and presented an output which was undefined as not a number, or NaN. There are several reasons which might have caused this including the learning rate, optimiser or input data. This will be discussed in detail in section 6.

Experiment 1B

AlexNet

This model achieved a top-1 and top-5 accuracy of 83.3% and 87.5%, respectively. The precision and recall values were 83.3% and 87.5%, respectively. Examples of the true versus predicted animal classes in Figure 129. The complete set of examples can be found in Appendix A Figure 149.

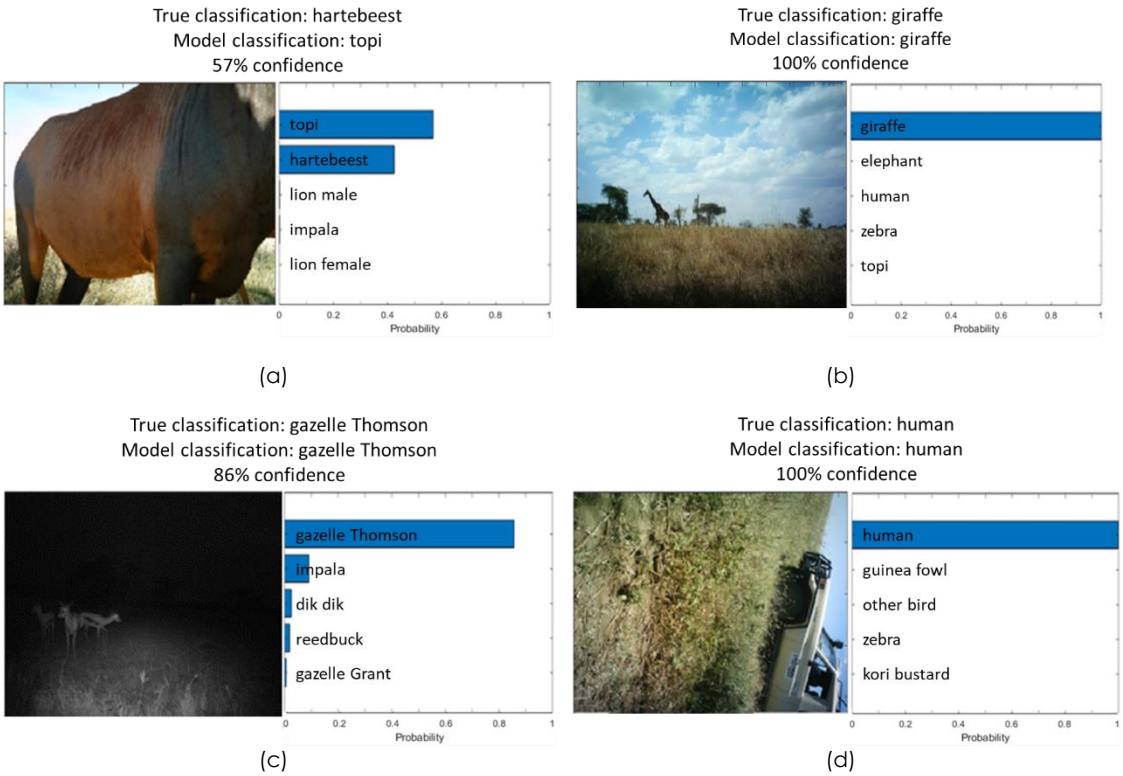


Figure 129: Confidence levels for classification examples (Experiment 1B - AlexNet)

Table 17 provides a summary of the results for the AlexNet model. The weighted calculation of precision and recall resulted in an f-score of 0.6326.

Table 17: Results for Experiment 1B - AlexNet

Top-1 Accuracy	83.3%
Top-5 Accuracy	87.5%
Precision	61.8%
Recall	64.8%
F-score	0.6326

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 167.

GoogLeNet

The top-1 and top-5 accuracy were 66.7% and 81.3%, respectively. In addition, the precision and recall figures were 45.1% and 54.7%, respectively. Examples of the true versus predicted classifications and confidence levels for the GoogLeNet model can be seen in the examples in Figure 130. The complete set of examples can be found in Appendix A Figure 150.

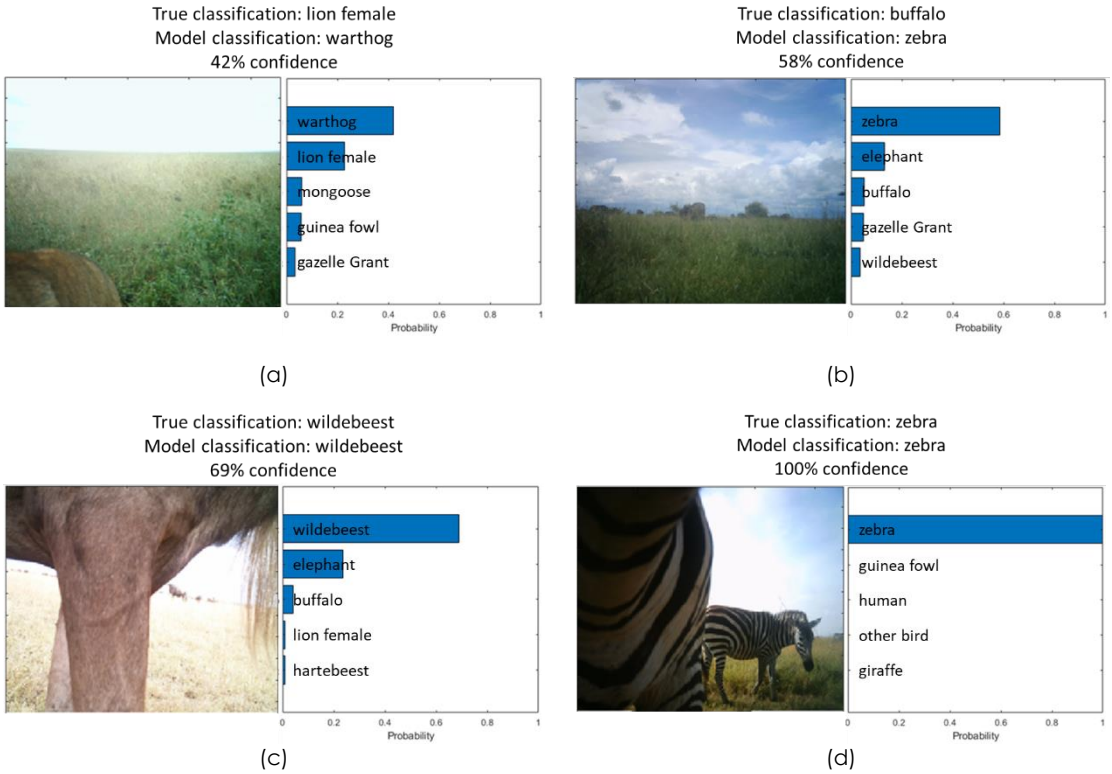


Figure 130: Confidence levels for classification examples (Experiment 1B - GoogLeNet)

The weighted calculation of precision and recall resulted in an f-score of 0.4941. Table 18 provides a summary of the results for this model.

Table 18: Results for Experiment 1B - GoogLeNet

Top-1 Accuracy	66.7%
Top-5 Accuracy	81.3%
Precision	45.1%
Recall	54.7%
F-score	0.4941

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 168.

ResNet-101

The top-1 and top-5 accuracy were 93.8% and 97.9%, respectively. The precision and recall values were 74.5% and 74.6%, respectively, with an f-score 0.7458. Figure 131 shows four examples of true versus predicted classifications and confidence levels for the ResNet-101 model. The complete set of examples can be found in Appendix A Figure 151.

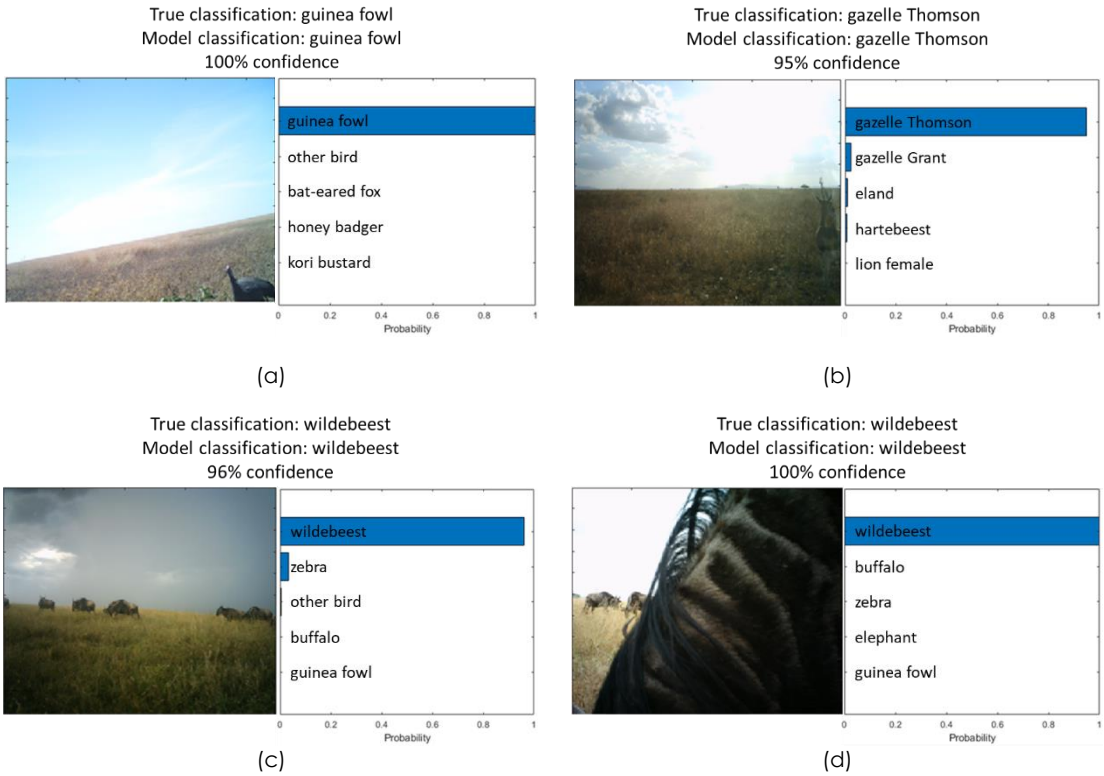


Figure 131: Confidence levels for classification examples (Experiment 1B - ResNet-101)

Table 19 provides a summary of the results for the ResNet-101 model.

Table 19: Results for Experiment 1B - ResNet-101

Top-1 Accuracy	93.8%
Top-5 Accuracy	97.9%
Precision	74.5%
Recall	74.6%
F-score	0.7458

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 169.

VGG-19

The top-1 and top-5 accuracy were 95.8% and 97.9%, respectively, which was the highest of all the models in Experiment 1B. The precision and recall figures were 76.8% and 79.0%, respectively, which were also the highest of all the models in this experiment. Figure 132 shows the examples of true versus predicted classifications and confidence levels for the VGG-19 model. The complete set of examples can be found in Appendix A Figure 151.

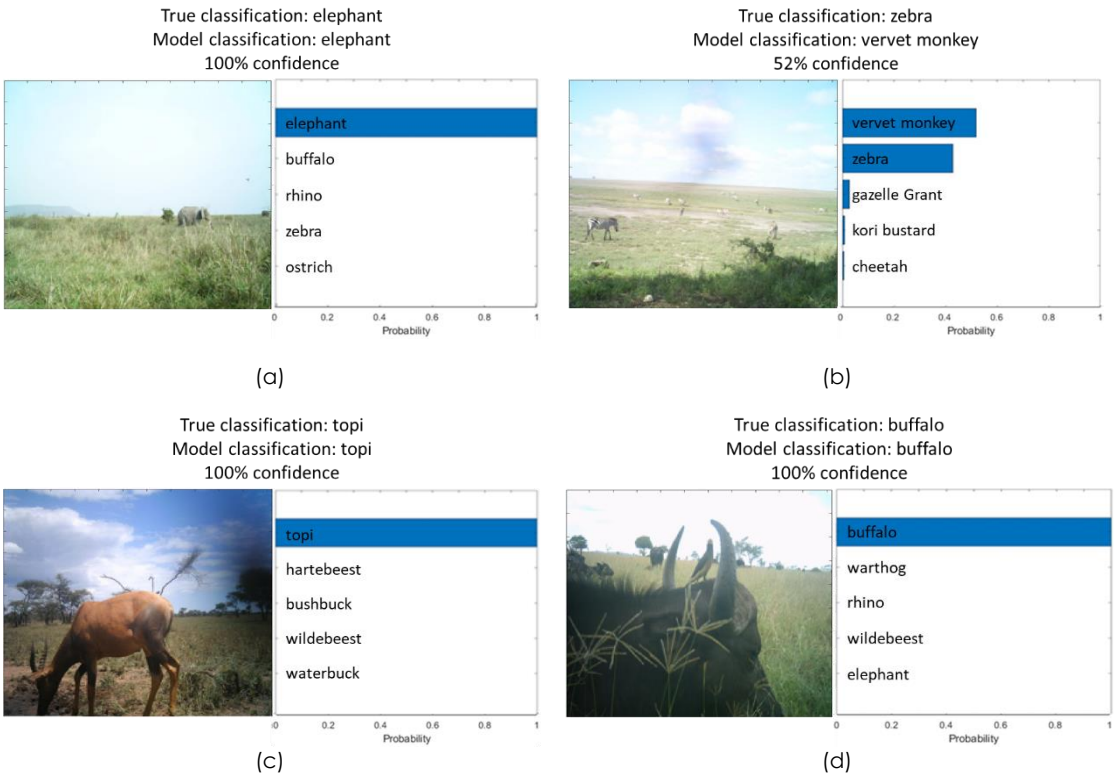


Figure 132: Confidence levels for classification examples (Experiment 1B - VGG-19)

Table 20 provides a summary of the results for the ResNet-101 model.

Table 20: Results for Experiment 1B - VGG-19

Top-1 Accuracy	95.8%
Top-5 Accuracy	97.9%
Precision	76.8%
Recall	79.0%
F-score	0.7792

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 170.

4.3.2 Experiment 2

Experiment 2 used the technique of fine-tuning layers during the model development.

Experiment 2A

AlexNet

Similar to the VGG-19 model in Experiment 1A, this model did not converge to a result and presented an output which was undefined as not a number, or NaN. There are several reasons which might have caused this including the learning rate, optimiser or input data. This will be discussed in detail in section 6.

GoogLeNet

The model achieved a top-1 accuracy of 83.3% and a top-5 accuracy of 93.8%. The precision and recall values were 68.6% and 69.3%, respectively. Examples of the true versus predicted classifications and confidence levels for the GoogLeNet model can be seen in Figure 133. The complete set of examples can be found in Appendix A Figure 153.

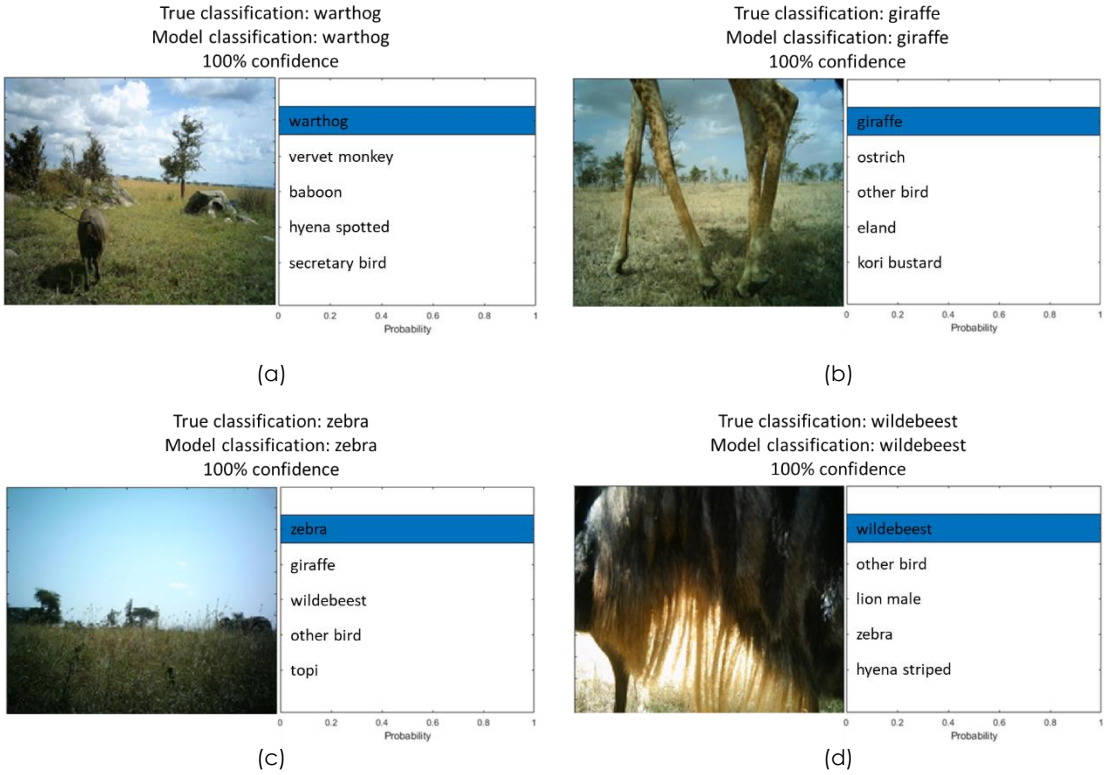


Figure 133: Confidence levels for classification examples (Experiment 2A - GoogLeNet)

Table 21 provides a summary of the results for the GoogLeNet model.

Table 21: Results for Experiment 2A - GoogLeNet

Top-1 Accuracy	83.3%
Top-5 Accuracy	93.8%
Precision	68.6%
Recall	69.3%
F-score	0.6899

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 171.

ResNet-101

The ResNet-101 model in this experiment had a top-1 and top-5 accuracy of 100%, which was the same result as the accuracy from the ResNet-101 model in Experiment 1A. This model had a precision and recall value of 82.6% and 82.5%, respectively. Figure 134 shows examples of the true versus predicted classifications and confidence levels of this model. The complete set of examples can be found in Appendix A Figure 154.

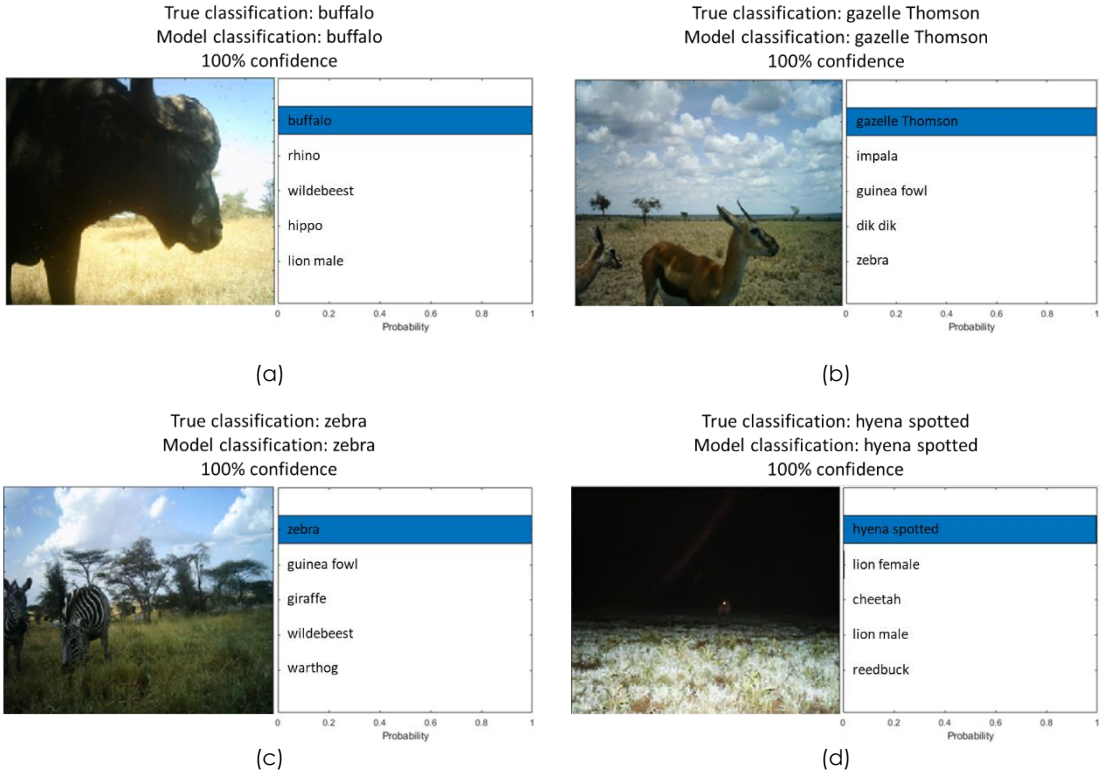


Figure 134: Confidence levels for classification examples (Experiment 2A - ResNet-101)

In addition to the high precision and recall values, f-score was 0.8252. Table 22 provides a summary of the results for this model.

Table 22: Results for Experiment 2A - ResNet-101

Top-1 Accuracy	100.0%
Top-5 Accuracy	100.0%
Precision	82.6%
Recall	82.5%
F-score	0.8252

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 172.

VGG-19

This model achieved a top-1 and top-5 accuracy of 2.1%. The precision and recall values were 2.1% and 0.1%, respectively. Examples of the true versus predicted classifications and confidence levels are shown in Figure 135. The complete set of examples can be found in Appendix A Figure 155Figure 153.

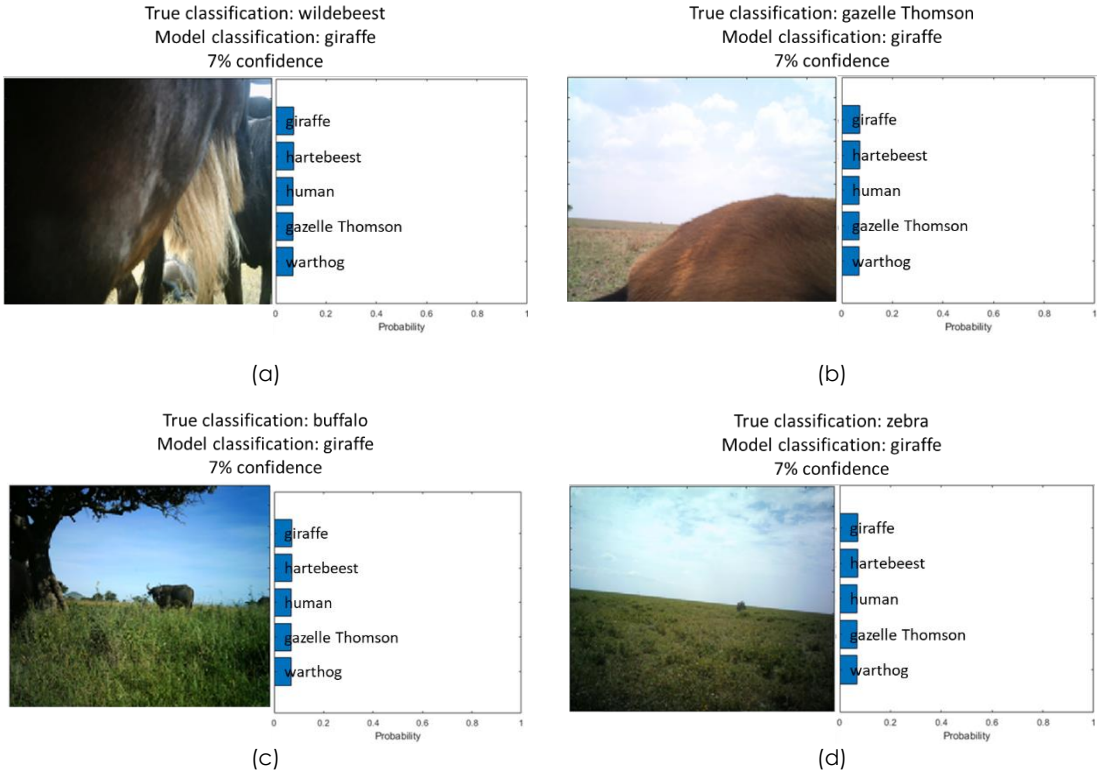


Figure 135: Confidence levels for classification examples (Experiment 2A - VGG-19)

Table 23 provides a summary of the results for this model.

Table 23: Results for Experiment 2A - VGG-19

Top-1 Accuracy	2.1%
Top-5 Accuracy	2.1%
Precision	2.1%
Recall	0.1%
F-score	0.0012

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 173.

Experiment 2B

AlexNet

The model achieved a top-1 accuracy of 87.5% and a top-5 accuracy of 95.8%. The precision and recall values were 63.2% and 65.1%, respectively. Examples of true versus

predicted classifications and confidence levels for this model can be seen in Figure 136. The complete set of examples can be found in Appendix A Figure 156.



Figure 136: Confidence levels for classification examples (Experiment 2B - AlexNet)

Table 24 provides a summary of the results for the AlexNet model.

Table 24: Results for Experiment 2B - AlexNet

Top-1 Accuracy	87.5%
Top-5 Accuracy	95.8%
Precision	63.2%
Recall	65.1%
F-score	0.6412

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 174.

GoogLeNet

This model achieved a top-1 accuracy of 87.5% and a top-5 accuracy of 91.7%. The precision and recall values were 64.2% and 70.1%, respectively. Figure 137 shows

examples of the true versus predicted classifications and confidence levels for this model. The complete set of examples can be found in Appendix A Figure 157.

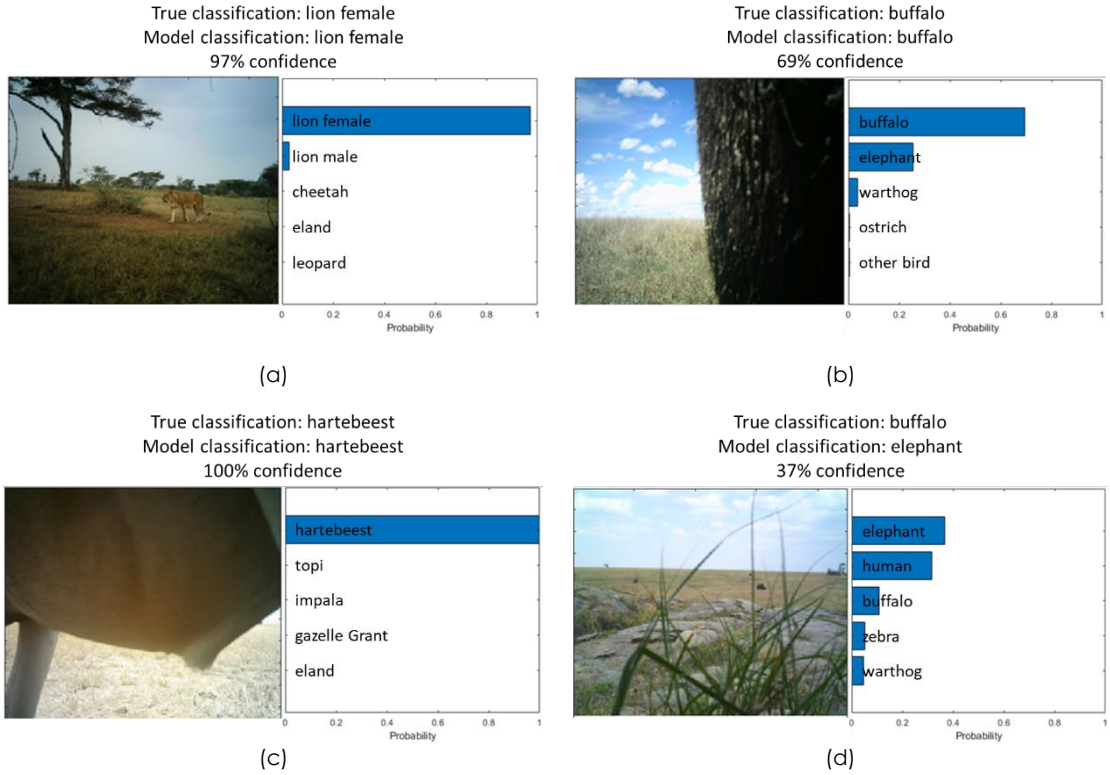


Figure 137: Confidence levels for classification examples (Experiment 2B - GoogLeNet)

Table 25 provides a summary of the results for the GoogLeNet model.

Table 25: Results for Experiment 2B - GoogLeNet

Top-1 Accuracy	87.5%
Top-5 Accuracy	91.7%
Precision	64.2%
Recall	70.1%
F-score	0.6701

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 175.

ResNet-101

This model achieved the same top-1 and top-5 accuracy of 100%, which was the same as the ResNet-101 models in Experiment 1A and 2A. The precision and recall values of this

model were 82.2% and 72.4%, respectively. Figure 138 shows examples of the true versus predicted classifications and confidence levels for this model. The complete set of examples can be found in Appendix A Figure 158.

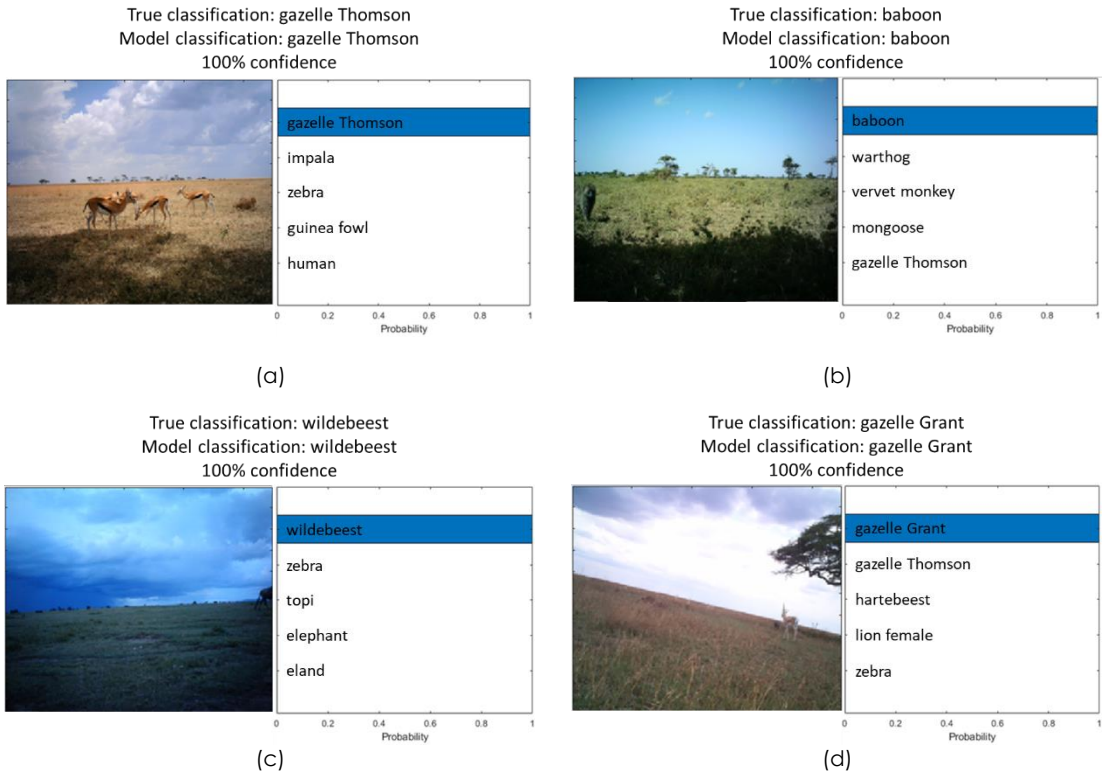


Figure 138: Confidence levels for classification examples (Experiment 2B - ResNet-101)

Table 26 provides a summary of the results for the ResNet-101 model.

Table 26: Results for Experiment 2B - ResNet-101

Top-1 Accuracy	100.0%
Top-5 Accuracy	100.0%
Precision	82.2%
Recall	72.4%
F-score	0.7700

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 176.

VGG-19

The VGG-19 model in this experiment achieved a top-1 and top-5 accuracy of 95.8% and 97.9%, respectively. The precision and recall values were both 77.9%. Figure 139 shows examples of the true versus predicted classifications and confidence levels for this model. The complete set of examples can be found in Appendix A Figure 159.

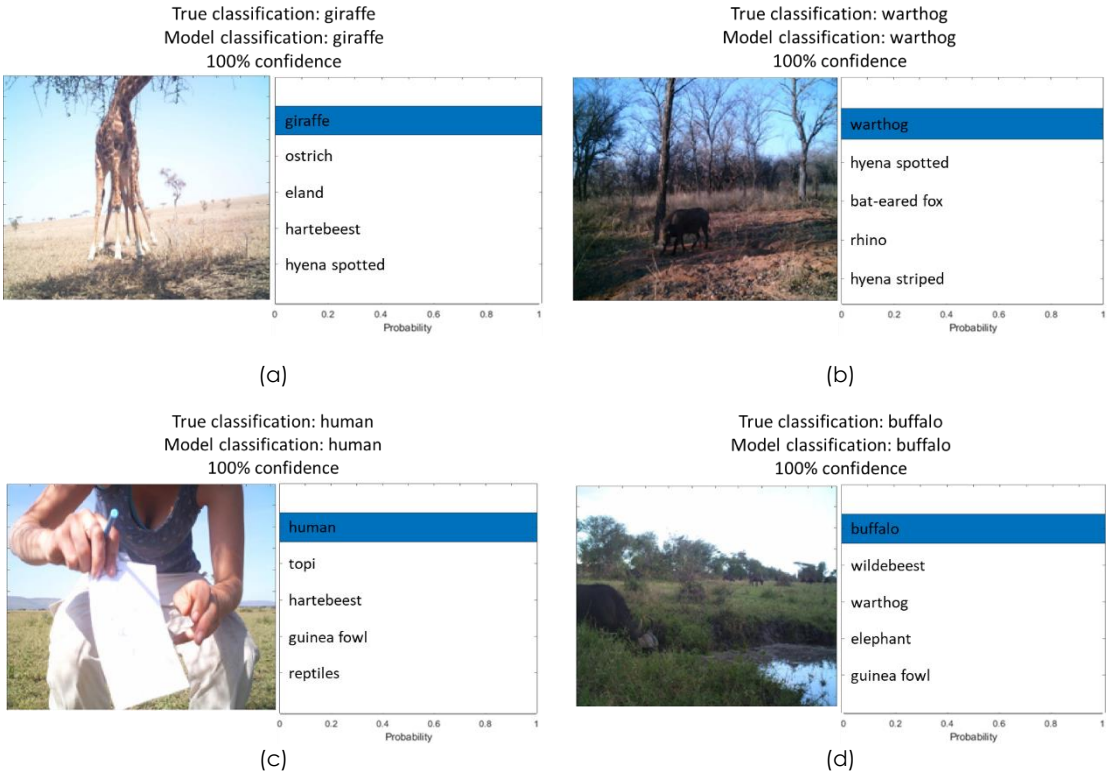


Figure 139: Confidence levels for classification examples (Experiment 2B - VGG-19)

Table 27 provides a summary of the results for this model.

Table 27: Results for Experiment 2B - VGG-19

Top-1 Accuracy	95.8%
Top-5 Accuracy	97.9%
Precision	77.9%
Recall	77.9%
F-score	0.7788

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 177.

4.3.3 Experiment 3

AlexNet

The AlexNet model in this experiment achieved a top-1 accuracy of 93.8% and a top-5 accuracy of 97.9%. The precision and recall values were 72.0% and 64.0%, respectively. Figure 140 shows the examples of the true versus predicted classifications and confidence levels for this model. The complete set of examples can be found in Appendix A Figure 160.

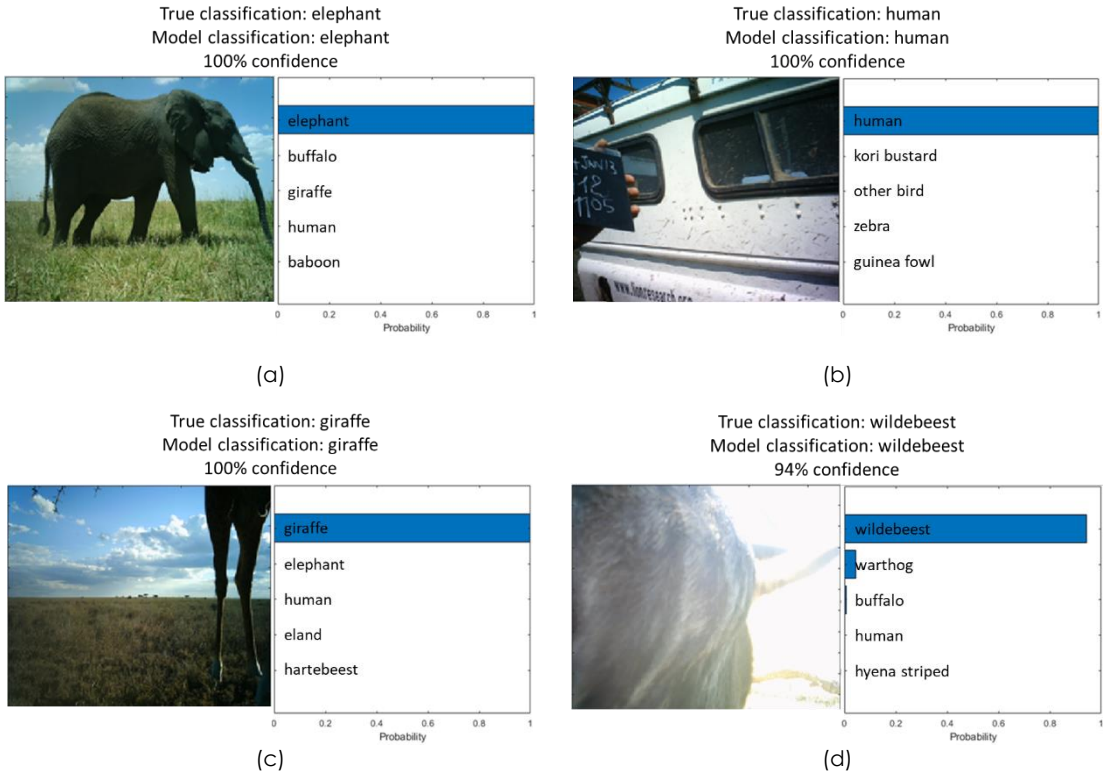


Figure 140: Confidence levels for classification examples (Experiment 3 - AlexNet)

Table 28 provides a summary of the results for this model.

Table 28: Results for Experiment 3 - AlexNet

Top-1 Accuracy	93.8%
Top-5 Accuracy	97.9%
Precision	72.0%
Recall	64.0%
F-score	0.6752

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 178.

GoogLeNet

This model achieved a top-1 and top-5 accuracy of 97.9% and 97.9%, respectively. The precision and recall values were 71.0% and 67.0%, respectively. Figure 141 shows the examples of the true versus predicted classifications and confidence levels for this model. The complete set of examples can be found in Appendix A Figure 161.

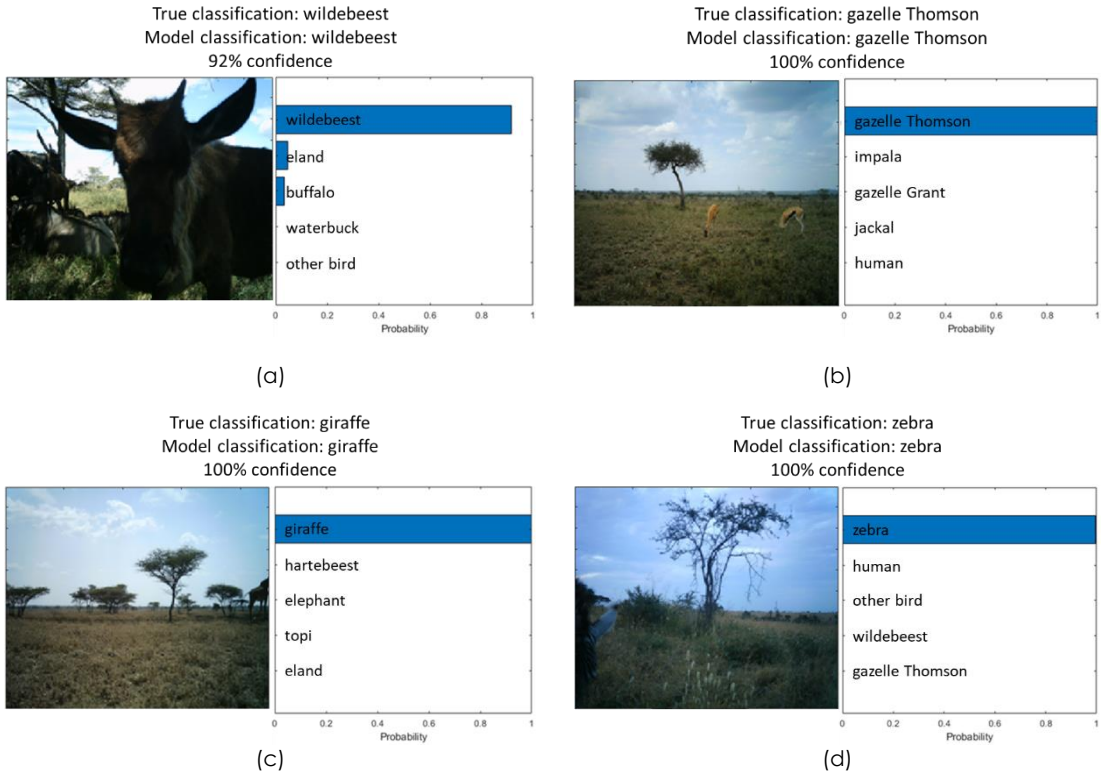


Figure 141: Confidence levels for classification examples (Experiment 3 - GoogLeNet)

Table 29 provides a summary of the results for this model.

Table 29: Results for Experiment 3 - GoogLeNet

Top-1 Accuracy	97.9%
Top-5 Accuracy	97.9%
Precision	71.0%
Recall	67.0%
F-score	0.6894

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 179.

ResNet-101

This model achieved a top-1 accuracy of 97.9% and a top-5 accuracy of 100%. The precision value of 83% was the highest of all the models across all the experiments and the recall value of 84% was the second highest of all the models. Examples of true versus predicted classifications and confidence levels for the model are shown in Figure 142. The complete set of examples can be found in Appendix A Figure 162.

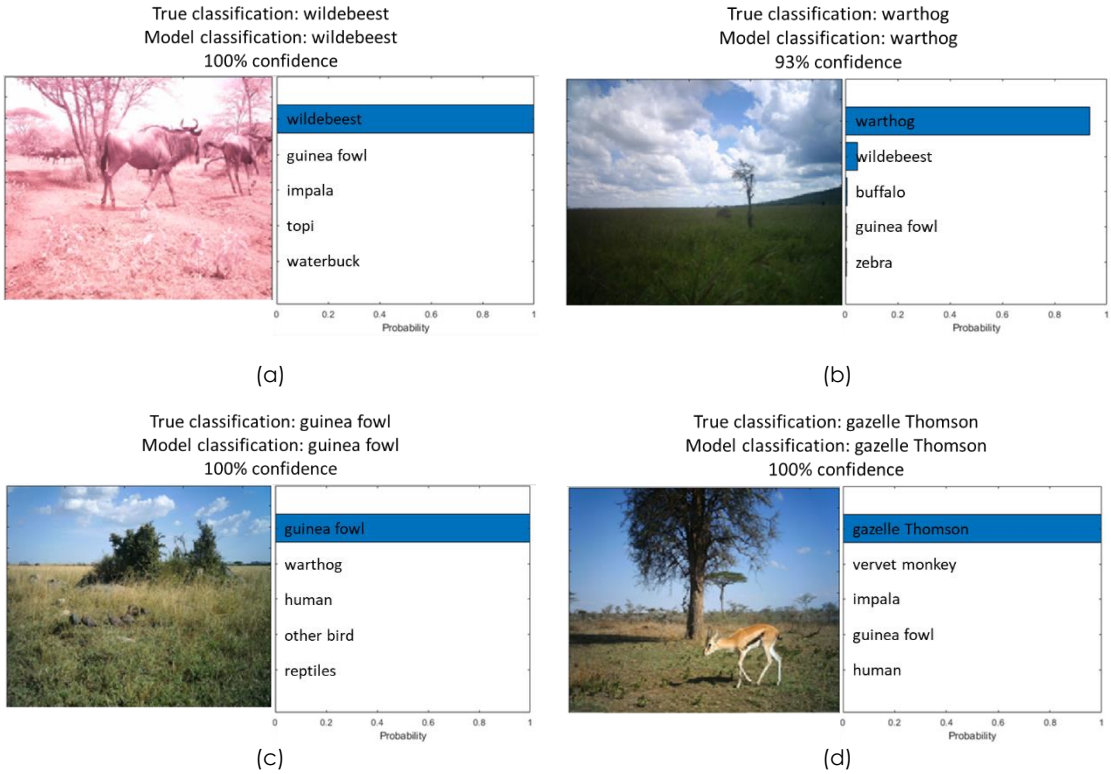


Figure 142: Confidence levels for classification examples (Experiment 3 - ResNet-101)

A summary of these results can be found in Table 30.

Table 30: Results for Experiment 3 - ResNet-101

Top-1 Accuracy	97.9%
Top-5 Accuracy	100.0%
Precision	83.0%
Recall	84.0%
F-score	0.8352

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 180.

VGG-19

This model achieved a lower top-1 and top-5 accuracy of 95.8% and 97.9%, respectively, compared to the ResNet-101 models. The precision and recall values were 84.3% and 81.1%, respectively. Examples of true versus predicted classifications and confidence levels for the model are shown in Figure 143. The complete set of examples can be found in Appendix A Figure 163.

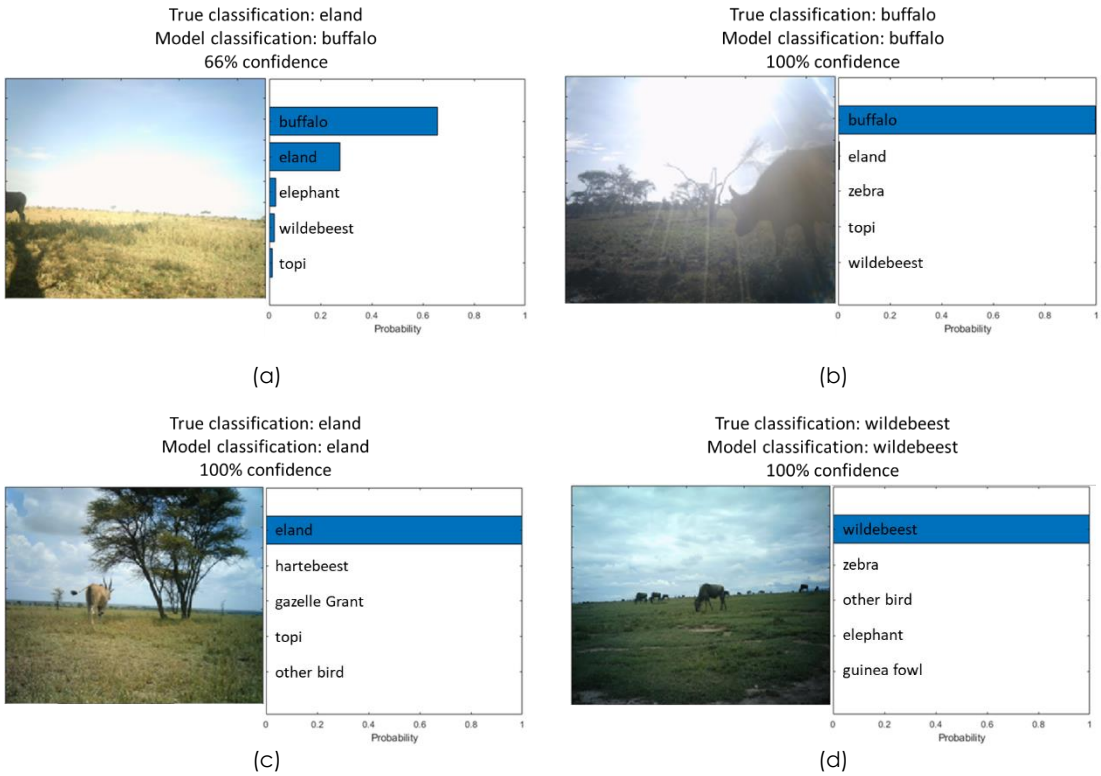


Figure 143: Confidence levels for classification examples (Experiment 3 - VGG-19)

The summary of these results can be found in Table 31.

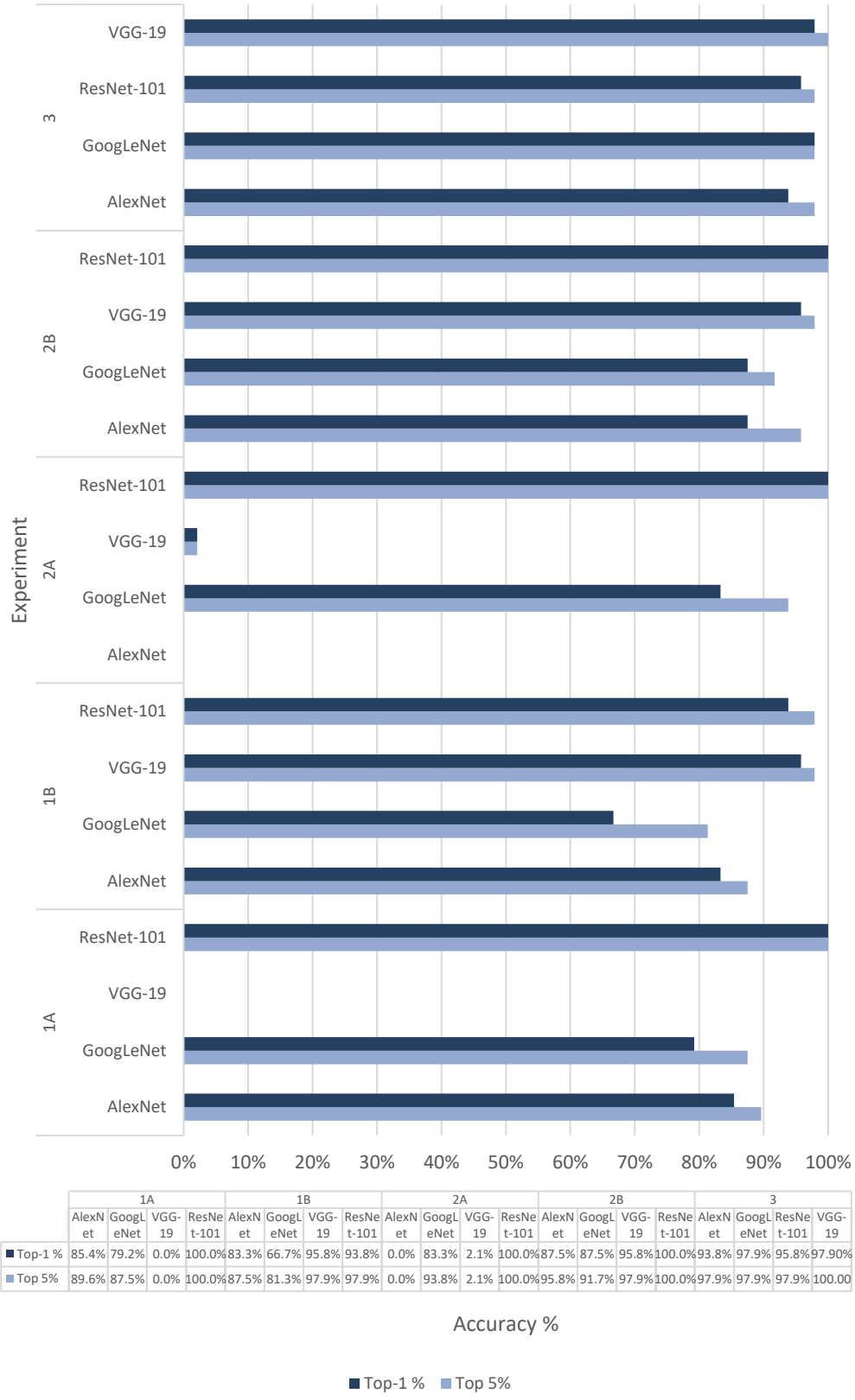
Table 31: Results for Experiment 3 - VGG-19

Top-1 Accuracy	95.8%
Top-5 Accuracy	97.9%
Precision	84.3%
Recall	81.1%
F-score	0.8268

The accuracies, precision and recall were calculated using the confusion matrix, which can be found in Appendix B Figure 181.

4.4 Summary

The top-1 and top-5 results for all experiments and models are shown in Figure 144.



	1A				1B				2A				2B				3			
	AlexNet	GoogLeNet	VGG-19	ResNet-101	AlexNet	GoogLeNet	VGG-19	ResNet-101	AlexNet	GoogLeNet	VGG-19	ResNet-101	AlexNet	GoogLeNet	VGG-19	ResNet-101	AlexNet	GoogLeNet	ResNet-101	VGG-19
Top-1 %	85.4%	79.2%	0.0%	100.0%	83.3%	66.7%	95.8%	93.8%	0.0%	83.3%	2.1%	100.0%	87.5%	87.5%	95.8%	100.0%	93.8%	97.9%	95.8%	97.90%
Top 5%	89.6%	87.5%	0.0%	100.0%	87.5%	81.3%	97.9%	97.9%	0.0%	93.8%	2.1%	100.0%	95.8%	91.7%	97.9%	100.0%	97.9%	97.9%	97.9%	100.00

Figure 144: Summary of experiments by top-1 and top-5 accuracy

The comparison of training times in minutes for the models are shown in Figure 146.

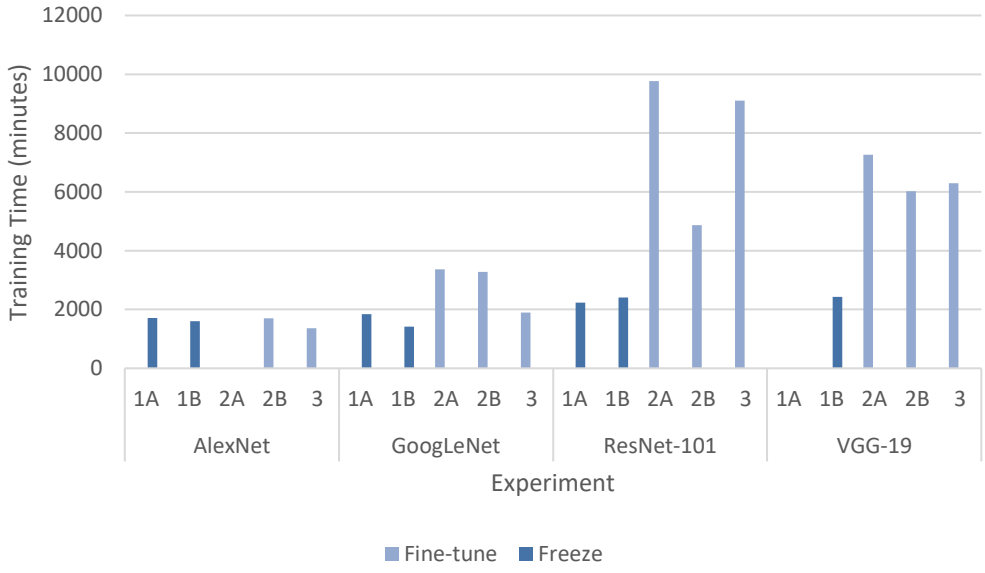


Figure 145: Summary of training time in minutes by experiment

5 ANALYSIS

This chapter provides an analysis of the results.

5.1 Training & Validation Results Analysis

This section will provide an analysis of the training and validation results by experiment. The SS dataset was used for training and validation.

5.1.1 Experiment 1

Experiment 1 used the technique of freezing layers during the model development. Experiment 1A used a learning rate of 0.005 and Experiment 1B used a learning rate of 0.001.

Experiment 1A

AlexNet

From the accuracy curve in Figure 90, the starting validation accuracy was 27%. The change in learning rate at epoch five resulted in a 6% improvement in validation accuracy. The second change in learning rate at epoch nine, resulted in a 2% improvement in validation accuracy from epoch eight which was at a smaller learning rate. This model maintained the accuracy improvement trajectory and did not show any regression in validation accuracy at any point during the ten epochs. From the loss curve for the AlexNet model, shown in Figure 91, it is evident that the loss plateaued from epoch six onwards.

GoogLeNet

The overall validation accuracy for the GoogLeNet model for Experiment 1A was 82%. Figure 92 displays the accuracy curve over the ten epochs. A steep improvement from epoch one to two can be seen in training and validation accuracy as the model reached 74% at the end of epoch one, however, it is only from epoch five that there is a noticeable improvement in validation accuracy of 4%. This is consistent with the change in learning rate which occurred at epoch five. The change in learning rate at epoch nine did not show any noticeable improvement in validation accuracy as from epoch five to ten the validation accuracy improved from 81% to 82%, however, between epochs five and six, there was a regression in validation accuracy from 81% to 80%.

According to Brownlee (2019b), the loss of the training dataset should always be lower than the validation dataset, which is known as the 'generalisation gap'. From the accuracy curve in Figure 92 and the loss curve in Figure 93 for the GoogLeNet model, the validation loss decreased to a point of stability from epoch seven onwards.

ResNet-101

The overall validation accuracy for the ResNet-101 model in this experiment was 92.9%. Figure 94 displays the accuracy curve over the ten epochs. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 78% at the end of epoch one. At epoch five, there is a noticeable improvement of 4% in validation accuracy which remained constant to the seventh epoch. This is consistent with the change in learning rate which occurred at the end of the fourth epoch.

Similarly, the validation loss decreased to the point of stability at the sixth epoch and did not improve any further. This can be seen in the loss curve in Figure 95.

VGG-19

This model did not converge to a result and presented an output which was undefined as not a number, or NaN.

Experiment 1B

AlexNet

The AlexNet model achieved an overall validation accuracy of 86.8%, as shown in Figure 96. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 75% at the end of epoch one. The change in learning rate at epoch five and epoch nine did not result in a significant change in validation accuracy. The validation accuracy plateaued at 84% from epochs four to six and only increased by 1% to 86.8% by the end of epoch ten.

The loss curve is shown in Figure 97. The change in learning rates did not result in improvements in the loss values of the model and remained constant from epoch seven onwards.

GoogLeNet

The overall validation accuracy for the GoogLeNet model for Experiment 1B was 76.3%. Figure 98 displays the accuracy curve over the ten epochs. At the learning rate change at

epoch five, the validation accuracy improved by less than 1% from 73.9% to 74.5%, and with the last learning rate change at epoch nine, there was no change in validation accuracy. The change in learning rate did not seem to have a drastic impact on the validation accuracy as the accuracy remained consistent above 73% throughout the training duration.

The loss curve in Figure 99 indicates that although the validation loss fit was good compared to the training loss as the gap between the two was minimal, the model had a high overall loss suggesting that the accuracy of the model was compromised.

ResNet-101

The accuracy curve in Figure 100 shows that the model achieved an overall validation accuracy of 85.8%. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 79.2% at the end of epoch one. The change in learning rate at epoch five resulted in a 2% improvement in validation accuracy, however, the accuracy remained constant at 85% thereafter.

The loss curve in Figure 101 shows a similar trend as the accuracy curve, with the loss values starting to plateau at epoch six.

VGG-19

The overall validation accuracy for the VGG-19 model for this experiment was 92.7%. Figure 102 displays the accuracy curve over the ten epochs. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 86.0% at the end of epoch one. At the learning rate change in epoch five, the validation accuracy improved by 2%, however, the change in learning rate at epoch nine did not show any improvement in validation accuracy. The validation accuracy remained constant at 92% from epoch five onwards.

The loss curve, shown in Figure 103, indicates a similar trend as the accuracy curve. The loss value at epoch six regressed slightly but improved and remained constant from epoch seven onwards.

5.1.2 Experiment 2

Experiment 2 used the technique of fine-tuning layers during the model development. Experiment 2A used a learning rate of 0.005 and Experiment 2B used a learning rate of 0.001.

Experiment 2A

AlexNet

This model did not converge to a result and presented an output which was undefined as not a number, or NaN.

GoogLeNet

The overall validation accuracy for the GoogLeNet model for Experiment 2A was 91.2%. From the accuracy curve in Figure 104, this model showed the steepest accuracy improvement from epoch one to two, to 83.0%. However, the validation accuracy peaked at epoch six at 91% and did not improve thereafter. The change in learning rates at epochs five and nine did not result in any significant improvement in validation accuracy and between epochs seven and eight, the validation accuracy dropped to 90.5% but recovered to 91.3% by epoch ten.

Figure 105 shows the loss curve which followed a similar trend to the accuracy curve.

ResNet-101

The accuracy curve in Figure 106 shows that the model achieved an overall validation accuracy of 90.1%, the accuracy regressed at epoch four from 87% to 85%, however, improved to 90% at epoch five. This is consistent with the change in learning rate which occurred at epoch five. There was a slight decrease in accuracy at epoch eight, but the accuracy improved to 90% and remained constant to the end of the tenth epoch.

The loss curve in Figure 107 shows that the loss value remained constant from epoch five onwards.

VGG-19

The accuracy curve for this model, shown in Figure 108, indicated a constant mini-batch accuracy, but the validation accuracy seemed to remain low throughout the training duration. This model failed to achieve a high accuracy and although there seemed to be a

slight recovery in epoch six, this reduced to below 10% at the seventh epoch. Similarly, there seemed to be an increase in accuracy at the start of the tenth epoch, but the final accuracy achieved was 2.9%.

The loss curve in Figure 109 shows the high loss value of 2.8 which remained constant over the training duration.

Experiment 2B

AlexNet

The AlexNet model achieved an overall validation accuracy of 88.5%. From the accuracy curve in Figure 110, a steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 73% at the end of epoch one. The change in learning rate at epoch five resulted in an improvement of 4%, and thereafter, the model accuracy plateaued at 88%. The second change in learning rate did not have any further improvement on the accuracy.

The loss curve in Figure 111 illustrates the low validation loss achieved. The loss plateaued at 0.410 in the last two epochs. This is consistent with the trend observed in the accuracy curve.

GoogLeNet

The overall validation accuracy for the GoogLeNet model was 88.3%, as shown in Figure 112. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 79.4% at the end of epoch one. The model accuracy peaked at 88.6% at epoch seven, however, this reduced to 88.3% by the end of epoch ten. The change in learning rates which occurred at epochs five and nine did not show any significant improvement in the model accuracy.

The loss curve in Figure 113 illustrates the low validation loss achieved. However, this model had a high starting loss at epoch 0 of 5.4 which reduced to 0.6 by the start of epoch two.

ResNet-101

The overall validation accuracy for the ResNet-101 model was 91.9%, as shown in Figure 114. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 83.2% at the end of epoch one. The model

accuracy increased to 86.9% at the end of epoch four but regressed to 83.5% in epoch five. The accuracy improved to 91% by epoch seven, increased to 92% at epoch nine and ended at 91.9% at the end of epoch ten. These changes were consistent with the change in learning rate at epoch five and epoch nine. The mini-batch accuracy showed fluctuations in accuracy from epoch one to epoch five but became more stable after epoch five.

Figure 115 shows the loss curve for the ResNet-101 model. Similar to the accuracy curve, the loss regressed at epoch five, but showed an improvement from epoch six onwards.

VGG-19

The overall validation accuracy for the VGG-19 model was 93.2%, as shown in Figure 116. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 87.1% at the end of epoch one. The model accuracy peaked at 93.0% at epoch five and remained constant until the end of the tenth epoch. The change in learning rate at epoch five was consistent with the improvement in accuracy, however, the second change in learning rate at epoch nine, did not show any improvement in accuracy.

Figure 117 shows the loss curve for the VGG-19 model. This follows a similar trend to the accuracy curve and the loss value plateaus at epoch five.

5.1.3 Experiment 3

Experiment 3 used the technique of fine-tuning layers during the model development. This experiment used a learning rate of 0.001 with additional class balancing techniques.

AlexNet

The accuracy curve for this model is shown in Figure 118. The overall validation accuracy for this model was 87.5%. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 72.8% at the end of epoch one. The change in learning rate at epoch five showed a 1% improvement in accuracy. Thereafter, the validation accuracy plateaued at 87% from epoch seven onwards. The second change in learning rate did not have any further improvement on the accuracy.

The loss curve in Figure 119 shows a similar trend as the accuracy curve. The loss value remained constant at 0.4 from epoch four onwards to the end of the tenth epoch.

GoogLeNet

The GoogLeNet model achieved an overall validation accuracy of 88.3%, as shown in Figure 120. A steep improvement from epoch one to two can be seen in the training and validation accuracy, as the model reached 82.3% at the end of epoch one. The change in learning rate at epoch five resulted in a 2% improvement from 85% to 87%. The second change in learning rate, however, did not result in an improvement in validation accuracy and the accuracy remained constant from epoch seven to the end of the tenth epoch at 88%.

The loss curve in Figure 121 shows a similar trend as the accuracy curve. The loss value remained constant at 0.6 from epoch six onwards.

ResNet-101

The accuracy curve for this model, shown in Figure 122, indicates that this model had a steep increase of 78%, from epoch one to epoch two. The validation accuracy plateaued at 88% from epoch four for the remainder of the training duration but showed a slight regression at epoch seven. The final validation accuracy peaked at 88.4% at the end of the tenth epoch.

The loss curve in Figure 123 indicates that the loss plateaued at 0.5 from epoch seven and slightly improved by the end of epoch ten to 0.5.

VGG-19

The VGG-19 model achieved a model accuracy of 93.7%. The accuracy curve in Figure 124 indicates that after the fifth epoch, the training accuracy remained constant above 93%. This model had a starting validation accuracy of 10.5% and achieved a validation accuracy after the first epoch of 87.7%.

The loss curve in Figure 125 corresponds to the accuracy curve trend, decreasing to a point of stability.

5.2 Test Results Analysis

This section will provide an analysis of the test results by experiment. The SS, AWF and Greenpeace datasets were used for testing.

5.2.1 Experiment 1

Experiment 1 used the technique of freezing layers during the model development. Experiment 1A used a learning rate of 0.005 and Experiment 1B used a learning rate of 0.001.

Experiment 1A

AlexNet

The top-1 and top-5 accuracy of the AlexNet model were 85.4% and 89.6%, respectively. For the AlexNet model, seven of the 48 classes were incorrectly classified using the top-1 accuracy metric, however, this reduced to five classes when using the top-5 accuracy metric. The misclassified classes were honey badger, striped hyena, civet, genet, secretary bird, wildcat and rhino. This suggests that similarities between certain species due to characteristic features, such as stripes or horns, resulted in the misclassification. For example, the rhino was misclassified as a buffalo, which suggests that perhaps the 'horn', which is iconic of the rhino but also present on the buffalo, could have resulted in the misclassification.

The precision and recall values were 65.0% and 53.3%, respectively. The lower recall values suggest that more of the predicted labels were incorrect when compared to the true labels. Figure 126 shows examples of the true versus predicted classifications and the confidence levels for the AlexNet model. Intra-species, which exists among animals of similar species such as gazelles and impala appear, can be noted in the five predicted results when classifying either of these animal species. Although the model correctly classifies the images, in some instances, it does not appear to do so with high confidence levels.

GoogLeNet

Results from the GoogLeNet model had a top-1 accuracy of 79.2% and a top-5 accuracy of 87.5%. This model had the lowest top-1 and top-5 accuracy of all models in this experiment. Ten of the 48 classes were incorrectly classified using the top-1 accuracy

metric. These included honey badger, striped hyena, genet, secretary bird, wildcat, rhino and aardwolf, to name a few. Using the top-5 accuracy metric, this resulted in a classification accuracy improvement of 9% and the number of incorrectly classified classes reduced from ten to six. In addition, six of the seven incorrectly classified image classes were the same classes misclassified in the AlexNet model.

The precision and recall values of 53.3% and 61.6% respectively, indicate that the model returned more irrelevant results than relevant ones based on the low precision value and higher recall value. Despite some animal classes having low precision values, the high recall values for those same classes suggested that although the complete set of true labels were not returned, the prediction and relevancy of the results that were returned, were correctly classified. For example, the leopard class had a precision value of 35% and a recall of 89% and the model returned a correct result for this prediction.

From the examples of true versus predicted classifications and the confidence levels for the GoogLeNet model shown in Figure 127, it is evident that although the model correctly classifies the image, it does so with a low confidence level. In addition, it is noticeable that for animal species that have similar features i.e. buffalo and wildebeest both have horns, the model is less confident about the result and both species are often included in top five results.

ResNet-101

Results from the ResNet-101 model had a top-1 and top-5 accuracy of 100%, respectively. Of the five ResNet-101 models that were run across Experiments 1, 2 and 3, three of these models achieved a top-1 and top-5 accuracy of 100%. This indicates that all the probabilities resulted in the true classification of the animal class being predicted. Figure 128 shows examples of classification and the confidence levels.

The precision and recall values were 81.5% and 82.6%, respectively. These scores were among the highest achieved by any of the models, suggesting that the model architecture has an impact on the classification accuracy. 50% of all the predictions had precision values of greater than 90%. Animal classes that had lower precision values had recall values that were equal to or greater than their equivalent precision values, suggesting that the model was able to find the balance of accuracy within the dataset. For example, the striped hyena had a precision value of 32% with a recall value of 55%, suggesting that

although more than half of all the relevant results were retrieved, very few relevant results were returned from the retrieved results.

VGG-19

This model did not converge to a result and presented an output which was undefined as not a number, or NaN. There are several reasons which might have caused this including the learning rate, optimiser or input data. This will be discussed in detail in section 6.

Experiment 1B

AlexNet

This model achieved a top-1 and top-5 accuracy of 83.3% and 87.5%, respectively. Eight of the 48 classes were misclassified resulting in the top-1 accuracy of 83.3%. These included seven of the same misclassified classes as the AlexNet and GoogLeNet in Experiment 1A, namely, honey badger, genet, secretary bird, wildcat, kori bustard, civet, striped hyena and rhino. This reduced to six misclassified classes to reach the top-5 accuracy of 87.5%, and excluded striped hyena and rhino, which were misclassified in the top-1 metric. Similar to the AlexNet model in Experiment 1A, further examination into the misclassified classes indicated that characteristic features between certain species could have resulted in the incorrect classification i.e. rhino and buffalo, and wildcat and bat-eared fox.

The precision and recall values were 83.3% and 87.5%, respectively. Further investigation into specific animal classes, suggests that the characteristics of the animal species could have been the reason for this result. This is confirmed in the examples of true versus predicted animal classes in Figure 129. Although the predicted classifications were correct, the confidence levels were low, suggesting that the learned features among similar animal species resulted in a greater degree of confusion among the animal predictions.

GoogLeNet

The top-1 and top-5 accuracy were 66.7% and 81.3%, respectively, which was lowest of all models in this experiment. In addition, the precision and recall figures were also the second lowest of all the models that were run across Experiments 1, 2 and 3, resulting in a low f-score of 0.4941. The low top-1 score was as a result of 16 animal classes being misclassified using this accuracy metric. Although these were the same classes that were

misclassified in the other experiments (secretary bird, genet and honey badger, to name a few), this model also misclassified animal classes such as leopards, mongoose and other birds.

Examples of the true versus predicted classifications and confidence levels for the GoogLeNet model confirm the low precision and recall values. This can be seen in the examples in Figure 130. Despite the clarity of some of the images that displayed the full animal within the image, the model was not able to correctly classify the image, and in some cases, although the model classification was correct, the confidence level was low and contained results of similar species in the five probabilities displayed.

ResNet-101

The top-1 and top-5 accuracy were 93.8% and 97.9%, respectively, which was the second highest accuracies of all the models in Experiment 1B. In addition, the precision and recall figures were also the second highest of all the models that were run in this experiment. The top-1 accuracy was due to the misclassification of the following classes: honey badger, hyena striped and zorilla. These were the same classes that were misclassified by the GoogLeNet model and two of these three classes were also misclassified by the AlexNet model. The top-5 accuracy was due to misclassification of the honey badger and striped hyena classes. Figure 131 shows the examples of true versus predicted classifications and confidence levels for the ResNet-101 model.

The precision and recall values were 74.5% and 74.6%, respectively, with an f-score 0.7458. Six of the 48 animal classes had precision values of less than 40%. These were the same animal classes that the AlexNet and GoogLeNet model had classified with either 0% precision or very low precision values. These classes include zorilla, civet, honey badger, striped hyena, caracal and secretary bird. The low recall values of these classes also indicate that the model returned less relevant results.

VGG-19

The top-1 and top-5 accuracy were 95.8% and 97.9%, respectively, which was the highest of all the models in Experiment 1B. The top-1 accuracy was due to the misclassification of the classes striped hyena and wildcat. These were the same classes which were misclassified as the AlexNet and GoogLeNet models and only one animal class corresponded with the ResNet-101 misclassified classes. The misclassified class that

resulted in the top-5 accuracy of 97.9% was wildcat. Figure 132 shows the examples of true versus predicted classifications and confidence levels for the VGG-19 model.

The precision and recall figures were 76.8% and 79.0%, respectively, which were also the highest of all the models in this experiment. Of all the models in this experiment, this was the only model that did not have any animal classes with 0% precision value. In addition, this model had the highest number of animal classes with recall and precision values of greater than 80%, respectively, of all the models in this experiment.

5.2.2 Experiment 2

Experiment 2 used the technique of fine-tuning layers during the model development. Experiment 2A used a learning rate of 0.005 and Experiment 2B used a learning rate of 0.001.

Experiment 2A

AlexNet

Similar to the VGG-19 model in Experiment 1A, this model did not converge to a result and presented an output which was undefined as not a number, or NaN. There are several reasons which might have caused this including the learning rate, optimiser or input data. This will be discussed in detail in section 6.

GoogLeNet

The model achieved a top-1 accuracy of 83.3% and a top-5 accuracy of 93.8%. Although the AlexNet model from Experiment 1A achieved a higher top-1 accuracy, this model achieved a higher top-5 accuracy. The ResNet-101 and VGG-19 models in Experiment 1, however, outperformed both the AlexNet and GoogLeNet models in terms of top-1 and top-5 accuracy. This model misclassified eight of the animal classes, which were the same classes which were misclassified in the models in Experiment 1. This suggests that the characteristic features of the species in those classes as well as the class imbalance which might still be present, could have influenced the results.

The precision value of 68.6% was higher than the AlexNet and GoogLeNet models in Experiment 1. Although the recall value of 69.3% was not the highest that was achieved, the overall high precision values are an indication that fine-tuning resulted in higher accuracies than freezing layers. This suggests that valuable learnings might be lost when

freezing layers as overall lines, edges and shapes are learned in the earlier layers of the model architecture.

Examples of the true versus predicted classifications and confidence levels for the GoogLeNet model can be seen in Figure 133. Of the random examples that were generated by the model, it is evident that this model can classify the images with greater confidence, and the top five predicted results often resulted in 100% confidence for some of the images. In addition to the high precision and recall values, the high f-score implies that a higher number of predicted labels were classified correctly when compared to the true labels.

ResNet-101

The ResNet-101 model in this experiment had a top-1 and top-5 accuracy of 100%, which was the same result as the accuracy from the ResNet-101 model in Experiment 1A. This was the highest of all the models from Experiment 1. Figure 134 shows examples of the true versus predicted classifications and confidence levels of this model.

In addition to the high top-1 and top-5 accuracy metrics, this model also achieved the highest precision values of all the models in Experiment 1A, 1B and 2A. All animal classes achieved precision values of 40% or greater, with more than half of all 48 classes achieving precision values of greater than 80%. The animal classes with low precision values were similar to the misclassified classes from the GoogLeNet model in this experiment. This includes the civet, wildcat, genet and zorilla. The recall values for these classes were also low, suggesting that although the model was able to correctly classify the image, the level of confidence may not have been high.

In addition to the high precision and recall values, the high f-score of 0.8252, which was the highest of all the models in Experiment 1A, 1B and 2A, implies that a higher number of predicted labels were classified and were correct when compared to the true labels.

VGG-19

The VGG-19 model in this experiment had the lowest top-1, top-5, precision and recall values of all the models in Experiments 1, 2 and 3. This excludes any of the models that achieved an undefined result, namely the AlexNet and VGG-19 model from Experiment 1A and 2A, respectively. This model achieved a top-1 and top-5 accuracy of 2.1%,

misclassifying all but one class: the giraffe. From the precision and recall values, all animal classes achieved a precision and recall of 0%, apart from the giraffe which had a precision of 100% but a recall of only 3%. This suggests that of all the results returned when classifying this specific animal, the model did not return many relevant results. This is confirmed in the examples of the true versus predicted classifications and confidence levels shown in Figure 135. From these examples, the confidence levels reflect the model's behaviour and inability to distinguish between the various classes.

Experiment 2B

AlexNet

The model achieved a top-1 accuracy of 87.5% and a top-5 accuracy of 95.8%. This was the highest top-5 accuracy achieved by any of the AlexNet and GoogLeNet models in Experiments 1 and 2. However, although the model achieved a high top-5 accuracy, the six classes which were misclassified were the same as the misclassified classes from the AlexNet and GoogLeNet models in Experiment 1, namely wildcat, civet, genet and honey badger, to name a few. The ResNet-101 and VGG-19 models in Experiments 1 and 2, however, still achieved greater accuracies and precision results.

The precision value of 63.2% was also lower than the AlexNet model from Experiment 1A and the GoogLeNet model from Experiment 2A. The low precision value can be seen by the low precision achieved in several animal classes. Examples of true versus predicted classifications and confidence levels for this model can be seen in Figure 136. Similar to the GoogLeNet model, even with partial views of the animal in the image, the model is able to correctly classify the image. However, as shown in the examples, the model also incorrectly classifies the image, despite the high confidence level that it displays, suggesting that intra-species might have caused a greater degree of confusion during the classification task.

GoogLeNet

This model achieved a top-1 accuracy of 87.5% and a top-5 accuracy of 91.7%. Although the top-1 accuracy was the same as the AlexNet model from this experiment, the AlexNet model achieved a higher top-5 accuracy of 95.8%. The six misclassified animal classes were consistent with the classes which were incorrectly classified in the previous models, namely honey badger, secretary bird, wildcat and rhino.

The precision value of 64.2% was lower than the AlexNet model from Experiment 1 and the GoogLeNet model from Experiment 2A. Furthermore, the GoogLeNet and AlexNet models in this experiment both had 14 classes that achieved precision values of less than 50%. The recall value of 70.1%, however, was the highest achieved by any of the AlexNet and GoogLeNet models in Experiments 1 and 2. As with the results from Experiment 1, the higher recall value than precision value suggests that the predicted labels were incorrectly classified when compared to the training labels. However, despite the high recall value, the lower precision value is an indication that the accuracy per class was very low for several animal classes.

Figure 137 shows examples of the true versus predicted classifications and confidence levels for this model. It is evident that even though some of the images include partial views of the animals, the model is still able to correctly classify the animal species with a relatively high confidence level among the five probabilities displayed.

ResNet-101

This model achieved the same top-1 and top-5 accuracy of 100%, which was the same as the ResNet-101 models in Experiment 1A and 2A. The only ResNet-101 model which did not perform as well was the model in Experiment 1B. However, this model still achieved better results than the AlexNet and GoogLeNet models in Experiments 1 and 2.

The precision and recall values of this model were 82.2% and 72.4%, respectively. Although the precision values were similar to those of the ResNet-101 model in Experiment 2A, the recall values were much lower than the previous ResNet-101 and VGG-19 models in the other experiments. This can be observed in animal classes that achieved high precision but low recall values and vice versa. For example, the impala had a precision of 87% and a recall value of 55%, suggesting that the model was not able to identify all the relevant data for this specific animal class.

VGG-19

The VGG-19 model in this experiment achieved the same top-1 and top-5 accuracy of 95.8% and 97.9%, respectively, as the VGG-19 model in Experiment 1B. However, the misclassified classes in this experiment were striped hyena and civet, whereas the misclassified classes for the VGG-19 model in Experiment 1B was striped hyena and wildcat. This model misclassified the striped hyena as an aardwolf, which suggests that

due to the similar features between these two animals, the model was not able to differentiate between the two. The misclassified class which resulted in the top-5 accuracy of 97.9% in this model was civet. This was also different from the VGG-19 model in Experiment 1B, which had a top-5 misclassification due to the wildcat animal class.

The fine-tuned models in Experiments 1 and 2 yielded the best results in terms of top-1 and top-5 accuracy as well as precision and recall values. The models which were trained on higher learning rates returned precision and recall values that were less accurate than those that were trained on lower learning rates. In addition, more of the models that used the fine-tuning technique, rather than freeze-layer technique, achieved better top-1 and top-5 accuracies. Hence, to further improve the accuracy and classification, Experiment 3 was conducted using fine-tuning and a modified version of emphasis sampling was used to further balance the rare classes. This will be discussed in the subsequent section.

5.2.3 Experiment 3

Experiment 3 used the technique of fine-tuning layers during the model development. This experiment used a learning rate of 0.001 with additional class balancing techniques.

AlexNet

The AlexNet model in this experiment achieved a top-1 accuracy of 93.8% and a top-5 accuracy of 97.9%. This was the same results as the ResNet-101 model in Experiment 1B. There were three misclassified classes that resulted in a top-1 accuracy of 93.8%. These were civet, striped hyena and aardwolf. This reduced to one class that was misclassified (civet) when using the top-5 accuracy metric. Despite the additional fine-tuning and class balancing, this model did not perform as well as the ResNet-101 and VGG-19 models in Experiments 1 and 2, suggesting that model architecture influences the accuracy of the prediction. Figure 140 shows the examples of the true versus predicted classifications and confidence levels for this model.

The precision and recall values were 72.0% and 64.0%, respectively. Compared to the AlexNet models in the previous experiments, over 65% of the animal classes had precision values greater than 70%. This suggests that although the model did not achieve a high top-1 and top-5 accuracy compared to the other models, the fine-tuning and class balancing did have an impact on the precision values. However, the recall value was

considerably lower and only had better recall value than the GoogLeNet model in Experiments 1A and 1B and the VGG-19 model in Experiment 2A.

GoogLeNet

This model achieved a top-1 and top-5 accuracy of 97.9% and 97.9%, respectively. This was the highest accuracy of all the GoogLeNet models in Experiments 1, 2 and 3. The animal class zorilla was misclassified and resulted in the top-1 and top-5 accuracy of 97.9%. This was the same class that was misclassified by the GoogLeNet and ResNet-101 models in Experiment 1B and the VGG-19 model in Experiment 2A. The precision and recall values of 71.0% and 67.0%, respectively, however, were lower than the AlexNet model in Experiment 3. 56% of the animal classes in this model achieve precision of greater than 70%, compared to the 65% from the AlexNet model.

Figure 141 shows the examples of the true versus predicted classifications and confidence levels for this model.

ResNet-101

This model achieved a top-1 accuracy of 97.9% and a top-5 accuracy of 100%. There was one animal class (genet) that was misclassified which resulted in the top-1 accuracy of 98%.

Examples of true versus predicted classifications and confidence levels for the model are shown in Figure 142. From these examples, it can be seen that even though the animals in two of the three examples shown are far in the distance, the model is able to correctly classify the image, however the impact that the surrounding environment has on the model's ability to classify the image is not known and should be further investigated in future studies.

The precision value of 83% was the highest of all the models across all the experiments and the recall value of 84% was the second highest of all the models, with the VGG-19 model in Experiment 3 achieving a higher recall value. The resulting f-score was 0.8352, indicating that many, correct predictions were returned by the model.

VGG-19

This model achieved a lower top-1 and top-5 accuracy of 95.8% and 97.9%, respectively, compared to the ResNet-101 models. There were two animal classes that were

misclassified (aardwolf and zorilla) which resulted in the top-1 accuracy of 95.8%. The precision value of 84.3% was the highest of all the models that were run, however, the recall value was lower than the ResNet-101 model from this experiment.

Examples of true versus predicted classifications and confidence levels for the model are shown in Figure 143. The high precision value can be noticed in two of the three examples. However, the impact of the surrounding environment and the partial view of the animal in the image suggests that despite the changes in architecture and learning rate, the model is less confident when it is presented with less information from which to make a decision.

5.3 Summary

The ResNet-101 models in Experiments 1A, 2A and 2B achieved the highest top-1 and top-5 accuracy of 100%, respectively. This was followed by the ResNet-101 and GoogLeNet models in Experiment 3 which had top-1 accuracies of 97.9%. Comparing the different models in the same experiment suggests that the transfer layer technique and model architecture might have affected the accuracy. This is especially noticeable when comparing the AlexNet and GoogLeNet model accuracy within the same experiment, as the models that were trained on higher learning rates with the freeze layer technique achieved lower accuracies than the models that were fine-tuned with lower learning rates. The top-1 and top-5 results are shown in Figure 144.

The VGG-19 model in Experiment 3 had the overall highest precision of 84.3% while the VGG-19 from Experiment 2A and GoogLeNet model in Experiments 1A and 1B achieved the lowest precision values of 2.1%, 45.1% and 53.3%, respectively. This is consistent with the low top-1 and top-5 accuracies achieved from these models. This comparison excludes the AlexNet and VGG-19 models from Experiment 1A and 2A, respectively, which did not converge to a result. Table 32 in Appendix C displays the precision values per class for all experiments.

The ResNet-101 model in Experiment 3 had the highest recall value of 84.0% while the VGG-19 and GoogLeNet models in Experiments 1A, 2A and 1B, respectively, had the lowest recall values of 0.1%, 54.75 and 61.6%, respectively. This is consistent with the low precision values obtained in these models. This comparison excludes the AlexNet

and VGG-19 models from Experiment 2A and 1A, respectively, which did not converge to a result. Table 33 in Appendix C displays the recall values per class for all experiments.

The ResNet-101 models in Experiment 1A, 2A, 2B and 3 and the VGG-19 model in Experiment 1B were the only models to have no 0% precision values. There were eight models that returned 0% precision values for at least one of the following animal classes: civet, honey badger and wildcat. At least eight other models returned 0% precision values for animal classes: genet, ostrich and secretary bird. Comparatively, for the models in Experiments 1 and 2 that utilised the same learning rate and transfer learning technique, the AlexNet and GoogLeNet models returned higher precision values.

Fine-tuning of the models by changing the learning rate and removing the freeze on certain layers had a positive impact on the classification accuracy of the networks. This is evident in the comparison of the two GoogLeNet models in Experiments 1A and 2A, that used the same learning rate but different transfer learning techniques. There was a 15% improvement in the overall precision values between these models, highlighting the impact of fine-tuning and the corresponding learned behaviour. A similar trend can be observed when comparing the AlexNet models and ResNet-101 models across the experiments.

Similarly, the models with fine-tuning achieved higher recall values than the freeze-layered models, suggesting that the freeze-layer transfer learning technique, although can be beneficial in terms of computational resources, does not result in better precision and recall values. The impact on all the classification accuracy metrics needs to be considered. The freeze layers technique did not seem to yield the best results in terms of all the classification accuracy metrics and hence was not considered further for experiment 3.

The results from Experiment 3 indicated that additional class balancing to improve the accuracy of rare classes resulted in a precision improvement, on average, of 21.4% across all the models. Specifically, with the rare classes, there was a 29% improvement in precision for the civet animal class, 57% improvement in precision for the secretary bird class and 30% improvement in precision for the genet class.

The results from the test dataset across all three experiments illustrate that ML models can perform classification tasks in the absence of large, labelled datasets for testing, through the use of transfer learning. By using pretrained networks and training the model

using existing labelled data, the test dataset consisting of approximately 50 000 images was used and resulted in positive, meaningful classification results.

The comparison of training times in minutes is shown in Figure 145. The training times for the freeze-layer models overall, were less than the fine-tuned models, confirming that freezing layers can accelerate training. By ‘freezing out’ layers, they are excluded from the backward pass.

The training times for the freeze-layer models across all experiments were similar, while the training time for the fine-tuned model experiments varied widely. This was due to the model architecture and depth which differed across the models. The depth of the architecture also had an impact on the training times. For deeper architectures, such as ResNet-101 and VGG-19, the training time for the fine-tuned models was significantly higher than the freeze-layer models for the same architectures.

6 DISCUSSION

This chapter provides an explanation and interpretation of the results and findings.

6.1 Introduction

The classification of wildlife has been studied through a variety of experiments and techniques (Swinnen et al. 2014; Yu et al. 2013; Chen et al. 2014; Villa et al. 2017; Norouzzadeh et al. 2017). In this study, various scenarios were run across three experiments to determine the best performing models in terms of network architecture and hyperparameter selection. Through the use of a smaller test dataset, the results indicated that ML models were able to perform classification tasks in the absence of large, labelled datasets for testing. The impact that transfer learning technique, learning rate, architecture and class balance has on model performance will be discussed in the following sections through the analysis of several classification accuracy metrics.

6.2 Effect of transfer learning techniques

In this study, the two techniques of transfer learning were tested: fine-tuning the CNN and freezing the layers of the CNN. The results from the models that utilised the freezing layer technique had lower classification accuracies than the models that utilised the fine-tuning technique. When comparing the top-1, top-5, precision and recall figures, these models did not perform as well as the fine-tuned models. The low precision and recall values suggested that freezing layers might have resulted in the model that was not able to learn the features necessary for recognising the various animal classes with greater accuracy.

The pretrained models used in this study (AlexNet, GoogLeNet, ResNet-101 and VGG-19) all used the ImageNet dataset as a starting point for initialising the weights and biases in the CNN. The ImageNet dataset contains more than 20 000 categories of everyday objects and scenes, suggesting that freezing earlier layers which might have assisted the model in detecting basic features of the animals such as shapes and outlines, might have hindered the model's ability to generalise to the SS training and validation dataset, affecting the model's performance and precision of the results.

As different layers in deep learning networks capture a different set of features depending on the depth and architecture of the network, feature selection by improving generalisability and removing irrelevant input features, might have improved the model's performance. Freezing the layers in the network reduces the number of trainable parameters as well as utilises the knowledge of the existing architecture used to make classification predictions. In general, the early layers of the CNN learn low-level image features that are applicable to a variety of visual recognition problems, whereas the later layers learn high-level features which are specific to the application at hand (Khan et al. 2016).

From the examples of the confidence levels of the predicted classes for Experiment 1, it is evident that the AlexNet and GoogLeNet models that used the freeze layer technique, had the lowest classification accuracies. Despite applying the same technique on the ResNet-101 and VGG-19 models, the difference in architecture, trainable parameters and learning rate had an impact on the accuracy that these models achieved, suggesting that the layers which were frozen affected the learned behaviour by the model. Despite the high top-1 and top-5 accuracies achieved by the ResNet-101 and VGG-19 model in Experiment 1, the fine-tuned models in Experiments 2A, 2B and 3 achieved higher precision values when comparing all models across all the experiments.

The point at which features are being transferred i.e. from the bottom, middle or top of the network, can influence which of these issues appear dominantly. The study conducted by Yosinski et al. (2014) on the transferability of features in deep neural networks found that transferability is negatively affected by specialisation of high-level neurons to their original task and co-adapted neurons. Although the regularisation techniques, such as dropout and batch normalisation were implemented to prevent co-adaptation, which can cause overfitting, the depth and number of parameters of the AlexNet and GoogLeNet models compared to the ResNet-101 and VGG-19 models, and the overall architecture differ widely suggesting that the number of layers which were 'frozen' could have affected the model's ability to classify the images. Furthermore, freezing the layers might have inhibited the high-level neurons from learning the necessary features to accurately perform classification to the task at hand.

When comparing the same models across the various experiments, for example, the ResNet-101 models in Experiments 1, 2 and 3, with the exception of the VGG-19 model in Experiment 2A, all the fine-tuned models performed better in terms of top-1, top-5 and precision values.

The test dataset consisted of images from three datasets, namely SS, AWF and Greenpeace. The images which were obtained from the AWF dataset introduced more variety into the test dataset in terms of grayscale versus colour images as well as surrounding environment. Although the model was trained on the large, labelled SS dataset, the surrounding environment and the colour of the images from the test dataset suggests that the model might have become biased at classifying these animal classes. Certain animals are only captured at night and hence these images are in grayscale.

For example, the majority of aardvark and aardwolf images were taken at night and hence, the model's prediction of grayscale could have influenced the classification prediction, rather than the characteristic of the animal itself. This is evident in the precision values for the AlexNet and GoogLeNet models, which were the lowest for the models that utilised the freeze-layer technique. Although the VGG-19 model in Experiment 3 returned the highest precision value, the low top-1 accuracy due to two misclassified classes confirms this bias. The aardwolf was misclassified in this model as a striped hyena and the zorilla was misclassified as a bat-eared fox. The majority of images for both these animals were taken at night and the characteristic features of these animals make them look similar in the night-time light settings. For example, the low-quality night-time images might have caused the model to 'learn' the noise, making it a concept of the model, which resulted in a negative impact on accuracy. This was more evident in the models that utilised the freeze-layer technique.

Although transfer learning, in this case, was achieved by using pretrained models, the technique of transfer has affected the overall classification accuracy, suggesting that utilisation of transfer without fine-tuning can result in suboptimal results. This is of particular importance when one considers the application of the model in real-life scenarios. The challenges with camera trapping and their resultant images cannot be ignored and the model must be robust enough to learn enough about the characteristic features of the animal class to ensure that it is provided with enough information to

perform the classification task. The impact of image quality is further affected if these images form part of the rare animal classes. The impact of class balancing on the accuracy will be discussed in the subsequent section.

The impact of transfer learning technique on training time was also observed. The freeze-layer technique had lower training times than the fine-tuned models. The training times for the freeze-layer models overall, were less than the fine-tuned models, confirming that freezing layers can accelerate training. However, despite the lower training times for the models which used the freeze-layer technique, the fine-tuned models achieved higher accuracies overall, thus suggesting that freezing layers can affect the accuracy.

6.3 Effect of class imbalance

Several techniques to address the class imbalance in the image dataset were employed. These included over sampling, under sampling and data augmentation. Despite these techniques, the animal classes which originally had a smaller number of images per class, returned lower precision values and affected the top-1 accuracy metric.

For all models with top-1 accuracy misclassifications, the animal classes that originally had fewer images prior to oversampling, returned incorrect results. Across all the experiments, except the ResNet-101 models that achieved 100% in the top-1 and top-5 accuracy and those that did not converge to a result, at least half of all the models misclassified striped hyena, honey badger and wildcat. In addition, at least seven models incorrectly classified genet, rhino, civet and secretary bird. These animal classes all had image counts of less than 300 prior to implementing over sampling techniques. However, there were animal classes with low image counts such as zorilla, rodents, ostrich and caracal that were correctly classified in at least 15 of the 20 model experiments that were conducted. This suggests that although class imbalance might still affect model accuracy even after applying over sampling techniques, the characteristics and features of the animal affect the model's ability to learn with greater confidence. In addition, the technique of freezing layers or fine-tuning can impact the degree to which the model can learn the task at hand.

For example, the misclassification of honey badger as a zebra in the GoogLeNet model in Experiment 2B is indicative of the feature similarity that might have resulted in the incorrect prediction. The honey badger and zebra both have characteristic black and white

stripes which distinguish them from other animals but can cause the model to confuse the two animals. Most of the honey badger images are also night-time images, which can cause the model to confuse the characteristics of the honey badger with that of a zebra in those specific light settings. A similar comparison can be done with the misclassification of the wildcat and bat-eared fox in the GoogLeNet model in experiment 2A, due to the distinct pointy ears which are characteristic of both animals. These misclassifications become more evident with rare classes such as honey badgers and wildcats.

Despite the additional emphasis sampling techniques used in Experiment 3, the ResNet-101 model misclassified genet as a serval resulting in a top-1 accuracy of 97.9%, which was the second highest achieved from all the models. The characteristic similarity between the genet and serval (both animals are of similar colour and have characteristic spotted coats), may have attributed to the misclassification. The genet animal class had a very low number of images prior to implementing any class balancing, suggesting that the model might have become biased when classifying the genet as a serval, which comparatively had a higher number of images. A similar observation can be made in the misclassification of striped hyena as an aardwolf in the VGG-19 model in Experiment 2B confirm this observation.

This was also evident in the precision and recall values for these same classes. For example, half of all the models in the experiments that were conducted, returned precision values for wildcat and honey badger that were below 20%, implying that the model did not return any correct results for these animal classes. The recall values of these classes indicated that although the precision values were low, suggesting that the model correctly classified the few results which were returned.

The further technique of emphasis sampling used in Experiment 3 to further improve the classification predictions and reduce any overfitting, showed significant improvements in the top-1, top-5, precision and recall values. This suggests that a single technique to address the issue of class imbalance is not sufficient to ensure that the model learns features from rare classes as well as those of classes with more images. The risk of such sampling techniques, however, is that the model might begin to overfit to the data when over sampling and under sampling might result in loss of important information which the model could have learned from.

These results suggest that although the class balancing techniques aimed to improve the accuracy per class, in some cases where the quality of the image and the characteristics of the animal are dominant features of the image, the results do not provide the correct predictions, specifically in the cases of rare animal classes. In addition, the techniques of balancing rare classes, such as those described in Mikolajczyk & Grochowski (2018) and Johnson et al. (2019) do not improve the accuracy of all the rare classes. Fine-tuning of the learning rate and employing additional class balancing techniques with deeper architectures, improved the top-1, top-5, precision and recall values of these models.

6.4 Effects of learning rates

According to Wu, Y. et al. (2019), learning rates are global hyperparameters that determine the size of the steps which the gradient descent optimiser takes along the direction of the slope derived from the loss function to reach a global minimum. Small learning rates will slow down the training process, but can achieve greater accuracies, however, large learning rates may cause the training to get stuck at local minima, preventing the model from improving, resulting in low accuracies or models that do not converge (Wu, Y. et al. 2019). Typically, finding a good learning rate is achieved through trial and error. Smaller learning rates require more training epochs given the smaller changes in weights while larger learning rates require less training epochs due to the rapid changes made in weights of the model.

The initial learning rate for the models in Experiments 1A and 2A were based on the study conducted by Norouzzadeh et al. (2017) and initialised at 0.005. The models that used the freeze-layer techniques and the learning rate of 0.005 had lower top-1, top-5, precision and recall values compared to the models that used the same learning rate with fine-tuning. This implies that apart from the learning rate, the transfer learning technique also affected the classification accuracy.

The most notable improvement from the change in learning rate could be seen in the top-1 and top-5 accuracy metrics when comparing the same models. All models that utilised the smaller learning rate of 0.001 achieved higher top-1 and top-5 accuracies, irrespective of the transfer learning technique that was applied. The same result is observed when comparing the models using the precision and recall values. The fine-tuned models that were initialised with learning rates of 0.005 in Experiment 2A, showed

better classification accuracy results than the freeze- layer models in Experiment 1A, except for the ResNet-101 model. This suggests that the architecture of the ResNet-101 model and the number of layers that were frozen could have contributed to its higher accuracy.

In almost all the models, the reduction in learning rates during the training and validation process resulted in slight improvements in accuracy, however, further changes in learning rates were not beneficial to model accuracy and in some cases, caused the model to plateau and get stuck at a suboptimal result, resulting in no further improvements.

All the experiments used the same learning rate policy, which reduced the learning rate every four epochs during the training process. It was observed that during the training and validation process, the change in learning rates did not show any significant improvements in accuracy and in most of the experiments, remained constant after the sixth epoch. This suggests that the lower learning rate might have resulted in the model being stuck a local minimum, and not showing any further improvement in loss.

The AlexNet and VGG-19 models in Experiment 2A and 1A, respectively, were the only two models to returned undefined results. As all other hyperparameters were kept constant, the high learning rate seemed to have caused this result. This was confirmed when the same fine-tuned models were run with the lower learning rate of 0.001 for the AlexNet and VGG-19 models in Experiments 2B and 1B, respectively. This was indicative of the fact that higher learning rates can cause the model to overshoot the global minimum and not converge to a result at all. In addition, the VGG-19 model in Experiment 1A, that used the freeze layers technique, returned the lowest top-1 and top-5 accuracies of all the experiments, at 2.1%. This model was also initialised at a learning rate of 0.005, indicating that at this learning rate, the VGG-19 model was not able classify the images, irrespective of the transfer learning technique that was applied.

Despite the learning rate changes, the learning rate policy might have impacted the accuracy of the results. All models were run at ten epochs; however, it is suggested that for future work, the models be run for longer epochs with different learning rates policies to identify the impact that training duration had on accuracy.

6.5 Effect of architecture

Recent studies have shown that network depth is beneficial to classification accuracy (Szegedy et al. 2015; He et al. 2016; Alom et al. 2018). Four different pretrained models were utilised in this study to monitor the effect of network depth and architecture on accuracy.

Each model utilised in this study had a varying number of convolution and fully connected layers. The AlexNet model is the simplest of all the architectures used in this study. It consists of the fewest convolution and fully connected layers. The VGG model showed that depth of a network is critical in achieving better recognition and classification accuracy. The GoogLeNet model increased the width of the model through inception modules, rather than the depth. The ResNet-101 model was developed to address the problem of degradation by using skip connections.

Using the top-1 and top-5 accuracy metrics for comparison, the ResNet-101 and VGG-19 fine-tuned models achieved the highest accuracy. This was consistent with the literature that indicated that depth of a network is critical in achieving better recognition and classification accuracy. However, the AlexNet model in Experiment 3 that used a lower learning rate and additional class balancing, was among the top seven models in terms of top-1 and top-5 accuracy. This suggests that the shallower AlexNet model can result in the same validation accuracy as more complex architectures such as ResNet-101, even when the learning rate is kept constant.

However, the depth of the network can result in saturation of accuracy, which can further result in a degradation of model performance, and accuracy. He et al. (2016) noted that such degradation may not be caused by overfitting but rather by adding more layers to the deep models, causing a higher training error.

Subsequent fine-tuning of the deeper and more complex ResNet-101, VGG-19 and GoogLeNet models resulted in the highest top-1 and top-5 accuracies, confirming that deeper convolutional neural networks that comprise of a greater number of trainable parameters, can create abstract representations of the data, allowing the model to learn features and improving the overall accuracy.

The impact of the model architecture also resulted in varying training times. The deeper, more complex ResNet-101 and VGG-19 architectures had longer training times for both transfer learning techniques. Despite the freeze-layer technique having significantly lower training times, as the depth of the architectures increased, the training times increased.

7 CONCLUSION & RECOMMENDATIONS

This chapter describes the conclusions and recommendations based on the research findings.

This study aimed to provide a transfer learning approach to wildlife classification and identification by using different transfer learning techniques, changing the learning rate and using different pretrained models. Although transfer learning is useful in the cases where limited training data is available, the choice of learning rate and class balancing methods are of critical importance to successful classification. The use of fine-tuning can result in more accurate results, despite the adjustment of learning rates that can result in suboptimal results.

Extensive experiments were conducted to identify wildlife in 48 species through images from Snapshot Serengeti, African Wildlife Foundation and Greenpeace datasets. These experiments were run on four state-of-the-art pretrained models, namely AlexNet, GoogLeNet, ResNet-101 and VGG-19. Results indicated that deeper, fine-tuned networks with lower learning rates achieved higher classification accuracies. In particular, the fine-tuned ResNet-101 model achieved the highest top-1 and top-5 accuracy of 100%. The experiments were conducted using two transfer learning techniques, namely fine-tuning and freezing layers, by adjusting the learning rates and learning rate policy. The results from these experiments revealed that it is possible to obtain high classification accuracies from camera-trap images by utilising the suitable algorithm, learning capacity and data quantity through class balancing.

The findings of this study revealed that it is possible to improve the training results, resulting in higher classification accuracies by fine-tuning the model through lower learning rates. However, in the case of pretrained models, the choice of model needs to be considered to the application at hand. This includes the number of learnable layers and regularisation techniques. Furthermore, this study showed that through fine-tuning techniques and adjustment of learning rates, deep neural networks can perform well on image datasets, however the problem of class imbalance is still prevalent.

The study adds to literature by demonstrating that transfer learning can provide computationally efficient ways of dealing with the problem of labelling images from camera-traps, especially in the absence of large labelled datasets. This will assist

researchers and volunteers in extracting valuable information about the environment and its inhabitants, which can assist in conservation efforts.

To further improve the results obtained when performing wildlife classification through transfer learning, the following is recommended:

- The impact of training duration on classification accuracy needs to be considered by changing the number of epochs in each experiment.
- The image dataset for each sample needs to consider the impact of empty images as well as the various other views, including partial and full-body images, day and night poses and multi-class images. This will ensure that each sample contains all possible image varieties which will allow the model to become more robust. This is especially important in the case of rare classes.
- In order to improve the accuracy of rare classes and address the challenge of class imbalance, other techniques of class balancing, which do not affect the distribution of the animal classes, need to be considered. This can include algorithmic approaches or hybrid approaches, which consist of traditional and algorithm-based techniques, to address class imbalance.

REFERENCE LIST

African Wildlife Foundation (2018) *African Wildlife Foundation* [online] Available from < https://www.awf.org/strategic-vision-2020-2030> [Accessed 2 April 2018].

Al-Saffar, A.A.M., Tao, H. & Talab, M.A. (2017) 'Review of Deep Convolution Neural Network in Image Classification'. *2017 International Conference on Radar, Antenna, Microwave, Electronics and Telecommunications (ICRAMET)*. Jakarta, Indonesia, 26-31.

Al-Stouhi, S. & Reddy, C.K. (2016) 'Transfer Learning for Class Imbalance Problems with Inadequate Data'. *Knowledge Information Systems* 48(1), 201-228.

Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Van Essen, B.C., Awwal, A. A. S. and Asari, V. K. (2018) 'The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches'. *ArXiv* [online] Available from <https://arxiv.org/abs/1803.01164> [Accessed 20 November 2019].

Becherer, N., Pecarina, J., Nykl, S. & Hopkinson, K. (2017) 'Improving optimisation of convolutional neural networks through parameter fine-tuning'. *Neural Computing and Applications* 31, 3469-3479.

Bengio, Y. (2012) 'Practical recommendations for gradient-based training of deep architectures'. In *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg, 437-478.

Benko, A. & Lanyi, C.S. (2009) *History of Artificial Intelligence.* Encyclopaedia of Information Science and Technology, 2nd edition, IGI Global, 1759 – 1762.

Bottou, L. & Bousquet, O. (2007) 'The tradeoffs of large scale learning'. *Proceedings of the 20th International Conference on Neural Information Processing Systems* 20, 161–168.

Britz, D. (2015) 'Understanding Convolutional Neural Networks for NLP' *WILDML Artificial Intelligence, Deep Learning and NLP* [online] Available from <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/> [Accessed 20 November 2020].

Brock, A., Lim, T., Ritchie, J. & Weston, N. (2017) 'FreezeOut: Accelerate Training by Progressively Freezing Layers. *ArXiv* [online] Available from <<https://arxiv.org/abs/1706.04983> [Accessed 1 October 2021].

Brownlee, J. (2019a) 'How to Visualize Filters and Feature Maps in Convolutional Neural Networks'. *Machine Learning Mastery* [online] Available from <https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/> [Accessed 29 November 2020].

Brownlee, J. (2019b) 'How to use Learning Curves to Diagnose Machine Learning Model Performance' *Machine Learning Mastery* [online] Available from < https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/> [Accessed 20 December 2020].

Brownlee, J. (2020) 'How Do Convolutional Layers Work in Deep Learning Neural Networks?' *Machine Learning Mastery* [online] Available from <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks> [Accessed 18 November 2020].

Buchanan, B.G. (2005) 'A (Very) Brief History of Artificial Intelligence'. *AI Magazine* (26), 53-60.

Buchanan, B.G. & Shortliffe, E.H. (1984) *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley.

Canziani, A., Culurciello, E. & Paszke, A. (2016) 'An Analysis of Deep Neural Network Models for Practical Applications'. *ArXiv* [online] Available from <https://arxiv.org/abs/1605.07678> [Accessed 29 November 2019].

Chang, H., Han, J., Zhong, C., Snijders, A. M. and Mao, J. H. (2018) 'Unsupervised Transfer Learning via Multi-Scale Convolutional Sparse Coding for Biomedical Applications'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (5), 1182-1194.

Chawla, N.V., Japkowicz, N. & Kolcz, A. (2004) 'Editorial: Special issue on learning from imbalanced data sets'. *SIGKDD Explorations* 6, 1-6.

Chen, G., Han, T.X., He, Z, Kays, R. & Forrester, T. (2014) 'Deep convolutional neural network based species recognition for wild animal monitoring'. *IEEE International Conference on Image Processing (ICIP)*, 858-862.

Chollet, F. (2017) *Deep Learning with Python*. New York: Manning Publications.

Cicero, F.M., Oliveira, A.H.M. & Botelho, G.M. (2016) 'Deep Learning and Convolutional Neural Networks in the Aid of the Classification of Melanoma'. *Workshop of the Undergraduate Works (WUW) in the 29th Conference on Graphics, Patterns and Images (SIBGRAPI '16)*. Sao Paulo, Brazil.

Cook, D., Feuz, K.D. & Narayanan, C.K. (2013) 'Transfer Learning for Activity Recognition: A Survey'. *Knowledge and Information Systems* 36(3), 537-556.

Copeland, B.J. (2000) 'The Turing Test*' *Minds and Machines* 10, 519-539.

Deng, J., Dong, W., Socher, R., Li, L., Li., K. and Fei-Fei, L. (2009) 'ImageNet: A large-scale hierarchical image database'. *IEEE Conference on Computer Vision and Pattern Recognition.* Miami, FL, 248-255.

Dickinson, J. L., Zuckerberg, B. & Bonter, D.N. (2010) 'Citizen science as an ecological research tool: Challenges and benefits'. *Annual Review of Ecology, Evolution, and Systematics* 41, 149–172.

Dogo, E., Nwulu, N., Afolabi, O. & Twala, B. (2018) 'A Comparative Analysis of Gradient Descent-Based Optimisation Algorithms on Convolutional Neural Networks'. *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, 92-99.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. & Darrell, T. (2014) 'DeCaf: a deep convolutional activation feature for generic visual recognition'. *Proceedings of the International Conference on Machine Learning* 32, 647-655.

Dumoulin, V. & Visin, F. (2016) 'A guide to convolution arithmetic for deep learning'. *ArXiv* abs/1603.0728.

Erhan, D., Bengio, Y., Courville, A. & Vincent, P. (2009) *Visualizing Higher-Layer Features of a Deep Network*. Paper No. 1341. June 9th 2009, Canada, University of Montreal.

Garcia, S. & Herrera, F. (2008) 'Evolutionary training set selection to optimize C4.5 in imbalance problems'. *Proceedings of 8th International Conference on Hybrid Intelligent Systems (HIS 2008)*. Held at Washington DC, USA, 567-572.

Garcia-Gasulla, D., Parés, F., Vilalta, A., Moreno, J., Ayguadé, E., Labarta, J., Cortés, U. & Suzumura, T. (2017) 'On the Behaviour of Convolutional Nets for Feature Extraction'. *Journal of Artificial Intelligence Research* 61 (1).

Gershenson, C. (2003) 'Artificial neural networks for beginners'. *ArXiv* [online] Available from < https://arxiv.org/abs/cs/0308031> [Accessed 2 April 2018].

Gilchrist, G., Mallory, M. & Merkel, F. (2005) 'Can local ecological knowledge contribute to wildlife management? Case studies of migratory birds'. *Ecology and Society* 10(1).

Gitay, H., Suarez, A., Dokken, D.J. & Watson, R.T. (2002) 'Climate change and biodiversity'. *Intergovernmental panel on climate change*. Geneva, Switzerland.

Glorot, X., Bordes, A. & Bengio, Y. (2011) 'Deep Sparse Rectifier Neural Networks'. *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Fort Lauderdale, FL, USA, 315 – 323.

Greenpeace (2016). *Greenpeace* [online] Available from <https://www.greenpeace.org/usa/> [Accessed 2 April 2018].

Greenpeace Media (n.d.). *Greenpeace Media* [online] Available from <https://media.greenpeace.org/CS.aspx?VP3=CMS3&VF=Home> [Accessed 2 April 2018].

Goldstein, I. & Papert, S. (1977) 'Artificial intelligence, language, and the study of knowledge'. *Cognitive Science* 1, 84-123.

Goodfellow, I., Bengio, Y. & Courville, A. (2016) *Deep Learning*. MIT Press.

Hannah, L., Midgley, G., Hughes, G. & Bomhard, B. (2005) 'The view from the Cape: Extinction risk, protected areas, and climate change'. *BioScience* 55(3), 231-242.

Harrington P.D. (2017) 'Multiple versus single set validation of multivariate models to avoid mistakes'. *Critical Reviews in Analytical Chemistry* 48, 33–46.

Harris, G., Thompson, R., Childs, J.L. & Sanderson, J.G. (2010) 'Automatic Storage and Analysis of Camera Trap Data'. *Bulletin of the Ecological Society of America* 91(3), 352-360.

He, K., Zhang, X., Ren, S. & Sun, J. (2015) 'Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification'. *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. USA, 1026-1034.

He, K., Zhang, X., Ren, S. & Sun, J. (2016) 'Deep Residual Learning for Image Recognition'. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

Hinton, G.E., Osindero, S. and The, Y. (2006) 'A Fast Learning Algorithm for Deep Belief Nets'. *Neural Computation* 18(7), 1527-1554.

Hossin, M. & Sulaiman, M.N. (2015) 'A Review on Evaluation Metrics for Data Classification Evaluations'. *International Journal of Data Mining & Knowledge Management Process* 5(2).

Hossin, M., Sulaiman, M.N., Mustapha, A., Mustapha, N. and Rahmat, R.W. (2011) 'A Hybrid Evaluation Metric for Optimizing Classifier'. *2011 3rd Conference on Data Mining and Optimization (DMO)*, 165-170.

Huang, Z., Pan, Z. & Lei, B. (2017) 'Transfer learning with deep convolutional neural network for SAR target classification with limited labeled data'. *Remote Sens* 9(907), 1-21.

James, M., Michael, C., Brad, B. & Jacques, B. (2011) *Big Data: The Next Frontier for Innovation, Competition, and Productivity*. McKinsey Global Institute, New York, NY.

Jarret, K. Kavukcuoglu, K., Ranzato, M. & LeCun, Y. (2009) 'What is the Best Multi-Stage Architecture for Object Recognition?'. *2009 IEEE 12th International Conference on Computer Vision*. Kyoto, Japan, 2146-2153.

Johnson, J. & Khoshgoftaar, T. (2019) 'Survey on deep learning with class imbalance'. *Journal of Big Data* 6(27).

Khan, N.M., Abraham, N. & Hon, M. (2016) 'Transfer Learning with intelligent training data selection for prediction of Alzheimer's Disease'. *IEEE Access* 4.

Kononenko, I. (2001) ‘Machine learning for medical diagnosis: history, state of the art and perspective’. *Artificial Intelligence in Medicine* 23, 89–109.

Krizhevsky, A., Sutskever, I. & Hinton, G.E. (2012) ‘ImageNet Classification with Deep Convolutional Neural Networks’. *Advances in neural information processing systems* 25(2).

L’Heureux, A., Grolinger, K. & Capretz, M.A.M. (2017) ‘Machine Learning with Big Data: Challenges and Approaches’. *IEEE Access* (5), 7776-7797.

LeCun, Y., Bengion, Y. & Hinton, G. (2015) ‘Deep Learning’. *Nature* (521), 436-444.

Levin, K. (2007) ‘Protecting biodiversity in a changing climate: the state of adaptation policies dedicated to enhancing the resiliency of biota’. *2007 George Wright Society Conference*, 257-262.

Lindsay, R.K., Buchanan, B.G., Feigenbaum, E.A. & Lederberg, J. (1980) *Applications of Artificial Intelligence for Chemical Inference: The DENDRAL Project*. New York: McGraw-Hill.

Liu, T., Fang, S., Zhao, Y., Wang, P. & Zhang, J. (2015) ‘Implementation of Training Convolutional Neural Networks’. *ArXiv* [online] Available from <https://arxiv.org/abs/1506.01195> [Accessed 24 October 2020].

Liu, Y., Yin, B., Yu, J. & Wang, Z. (2017) ‘Image classification based on convolutional neural networks with cross-level strategy’. *Multimed Tools Appl* 76 11065-11079.

Lobova, T.I., Lankin,, Y. (2007) ‘Assessing the anthropogenic impact on lake Shira from antibiotic resistance of heterotrophic bacteria by neural network methods’. *Mikrobiologiia* 76 (2), 263-270.

Lundstrom, D. (2017) ‘Data-efficient Transfer Learning with Pre-trained Networks’. *Master’s Dissertation*. Linkoping University, Sweden.

Mahendran, A. & Vedaldi, A. (2015) ‘Understanding Deep Image Representations by Inverting Them’. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Held 7-12 June, 2015. Boston, Massachusetts.

MathWorks (2020a) *Pretrained Deep Neural Networks* [online] Available from <https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks> [Accessed 20 September 2020].

MathWorks (2020b) *Transfer Learning Using AlexNet* [online] Available from <https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html> [Accessed 19 April 2019].

Maruster, L. (2003) *A machine learning approach to understand business processes.* PhD thesis. Eindhoven University of Technology.

Mawdsley, J.R., O'Malley, R. & Ojima, D.S. (2008) 'A review of climate-change adaptation strategies for wildlife management and biodiversity conservation'. *Conservation Biology* 23 (5), 1080-1089.

McAfee, A. & Brynjolfsson, E. (2012) 'Big Data: The Management Revolution'. *Harvard Business Review* [online] October 2012. Available from <https://hbr.org/2012/10/big-data-the-management-revolution> [Accessed 7 June 2020].

McCulloch, W.S. & Pitts, W. (1943) 'A Logical Calculus of the Idea Immanent in Nervous Activity'. *Bulletin of Mathematical Biophysics* 5, 115–133.

Mikołajczyk, A. & Grochowski, M. (2018) 'Data augmentation for improving deep learning in image classification problem'. *International Interdisciplinary PhD Workshop (IIPhDW)*. Swinoujście, 117-122.

Minsky, M. & Papert, S. (1969) *Perceptron.* M.I.T. Press, Oxford, England.

Mohri, M., Rostamizadeh, A. & Talwalkar, A. (2012) *Foundations of Machine Learning*. London, England: The MIT Press.

Mordvintsev, A., Olah, C. & Tyka, M. (2015) 'Inceptionism: Going Deeper into Neural Networks' *Google AI Blog* [online] Available from <https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html> [Accessed 2 December 2020].

Moulder, S., Sheridan, T., Cavallo, P. & Rossini, G. (2017) *Deep Learning with MATLAB and Multiple GPUs* [online] Available from

<https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/d/Deep_Learning_in_Cloud_Whitepaper.pdf> [Accessed 10 June 2020].

Movshovitz-Attias, Y., Yu, Q., Stumpe, M., Shet, V., Sacha, A. & Yatziv, L. (2015) 'Ontological supervision for fine grained classification of Street View storefronts'. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1693-1702.

Nair, V. & Hinton, G.E. (2010) 'Rectified Linear Units Improve Restricted Boltzmann Machines'. *Proceedings of the 27 the International Conference on Machine Learning*, Haifa, Israel.

Ng, A. (2015) *Advice for applying machine learning* [online lecture] Available from <https://see.stanford.edu/materials/aimlcs229/ML-advice.pdf> [Accessed 4 July 2020].

Nguyen, A., Yosinski, J. & Clune, J. (2015) 'Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images'. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Held 7-12 June, 2015. Boston, Massachusetts.

Nguyen, H., Maclagan, S.J., Nguyen, T.D., Nguyen, T., Flemons, P., Andrews, K., Ritchie, E.G. & Phung, D. (2017) 'Animal Recognition and Identification with Deep Convolutional Neural Networks for Automated Wildlife Monitoring'. *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Tokyo, 40-49.

Nielsen, M. (2015) *Neural Networks and Deep Learning.* Determination Press.

Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M., Packer, C. & Clune, J. (2017) 'Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning'. *CoRR*, 1-17.

Olah, C., Mordvintsev, A. & Schubert, L. (2017) 'Feature Visualisation'. *Distill* [online] Available from < https://distill.pub/2017/feature-visualization/> [Accessed 2 December 2020].

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K. & Mordvintsev, A. (2018) 'The Building Blocks of Interpretability' *Distill* [online] Available from <https://distill.pub/2018/building-blocks/> [Accessed 2 December 2020].

Oquab, M., Bottou, L., Laptev, I. & Sivic, J. (2014) 'Learning and transferring mid-level image representations using convolutional neural networks'. *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Columbus, Ohio, 1717 -1724.

Pan, S.J. & Yang, Q. (2010) 'A survey on transfer learning'. *IEEE Transactions on Knowledge and Data Engineering* 22(10) 1345-1359.

Pan, S.J., Zheng, V. W., Yang, Q. & Hu, D.H. (2008) 'Transfer learning for wifi-based indoor localisation'. *Proceedings of the Workshop on Transfer Learning for Complex Task of the 23rd AAAI Conference on Artificial Intelligence*, Chicago, Illinois.

Parmesan, C. (2006) 'Ecological and evolutionary responses to recent climate change'. *Annual Review of Ecology, Evolution and Systematics* 37, 637-669.

Pechyony, D. (2008) *Theory and practice of transductive learning*. PhD Dissertation, Israel Institute of Technology.

Provost, F. & Fawcett, T. (2013) 'Data Science and its Relationship to Big Data and Data-Driven Decision Making'. *Big Data* 1(1), 51-59.

Raina, R., Madhavan, A. & Ng, A.Y. (2009) 'Large-scale deep unsupervised learning using graphics processors'. *Proceedings for the 26th International Conference on Machine Learning*, 1-9.

Rehman, A., Naz, S., Razzak, M.I., Akram, F. & Imran, M. (2019) 'A Deep-Learning-Based Framework for Automatic Brain Tumors Classification Using Transfer Learning'. *Circuits, Systems, and Signal Processing* 39, 757-775.

Ripley, B. (1996) *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

Rosenblatt, F. (1958) 'The perceptron: a probabilistic model for information storage and organization in the brain'. *Psychological Review* 65(6), 386–408.

Ruder, S. (2016) 'An overview of gradient descent optimization algorithms'. *ArXiv* [online] Available from <https://arxiv.org/abs/1609.04747> [Accessed 6 December 2020].

Russell, S. & Norvig, P. (2009) *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey.

Sathya, R. & Abraham, A. (2013) 'Comparison of supervised and unsupervised learning algorithms for pattern classification'. *International Journal of Advanced Research in Artificial Intelligence* 2(2), 34-38.

Shavlik, J.W. & Dietterich, T.G. (1990) *Readings in Machine Learning.* Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Schaul, T., Antonoglou, I. & Silver, D. (2014) 'Unit tests for stochastic optimization'. *International Conference on Learning Representations,* 306.

Schreiber, R., 2007, *MATLAB* [online] Available from <http://www.scholarpedia.org/article/MATLAB> [Accessed 11 July 2020].

Schroff, F., Kalenichenko, D. & Philbin, J. (2015) 'FaceNet: A unified embedding for face recognition and clustering'. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 815-823.

Sharma, S., Sharma, S. & Athaiya, A. (2020) 'Activation Functions in Neural Networks'. *International Journal of Engineering Applied Sciences and Technology* 4(12), 2455-2143.

Shorten, C. & Khoshgoftaar, T.M. (2019) 'A survey on Image Data Augmentation for Deep Learning'. *Journal of Big Data* 6(60).

Silveira, L., Jacomo, A.T. & Diniz-Filho, J.A.F. (2003) 'Camera trap, line transect census and track surveys: a comparative evaluation'. *Biological Conservation* 114, 351-355.

Silverton, J. (2009) 'A new dawn for citizen science'. *Trends in Ecology & Evolution* 24(9), 467-471.

Simonyan, K., Vedaldi, A. & Zisserman, A. (2014) 'Deep inside convolutional networks: Visualising image classification and saliency maps'. *International Conference on Learning Representations (ICLR),* 1-8.

Simonyan, K. & Zisserman, A. (2015) 'Very Deep Convolutional Networks for Large-Scale Image Recognition'. *International Conference on Learning Representations (ICLR).*

Sun, R. (2019) 'Optimization for deep learning: theory and algorithms'. *ArXiv* [online] Available from < https://arxiv.org/abs/1912.08957> [Accessed 20 June 2020].

Sun, S., Cao, Z., Zhu, H. & Zhao, J. (2019) 'A Survey of Optimsisation Methods from a Machine Learning Perspective'. *IEEE Transactions on Cybernetics*, 50(8), 3668-3681..

Sutton, R.S. (1986) 'Two problems with backpropagation and other steepest-descent learning procedures for networks'. *Proceedings of the Eight Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Sutton, R.S. & Barto, A.G. (1998) *Reinforcement Learning*. Cambridge, Massachusetts: The MIT Press.

Swanson, A. & Kosmala, M. (2015) 'Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna'. *Scientific Data*, 1-14.

Swanson, A., Kosmala, M., Lintott, C.J., Simpson, R.J., Smith, A. & Craig, P. (2016) 'Data from: Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna'. *Dryad* [online] Available from: <https://doi.org/10.5061/dryad.5pt92>.

Swinnen, K.R.R., Reijniers, J., Breno, M. & Leirs, H. (2014) 'A novel method to reduce time investment when processing videos from camera trap studies'. *PLoS One* 9(2).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. & Fergus, R. (2014) 'Intriguing properties of neural networks'. *International Conference on Learning Representations (ICLR).* Banff, Canada.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015) 'Going Deeper with Convolutions'. *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 1-9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016) 'Rethinking the Inception Architecture for Computer Vision'. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Las Vegas, NV, 2818 – 2826.

Thenmozhi, K. & Reddy, U.S. (2019) 'Crop pest classification based on deep convolutional neural network and transfer learning'. *Computers and Electronics in Agriculture* 164 (2019).

Torrey, L. & Shavlik, J. (2009) 'Transfer learning'. In *Handbook of research on machine learning applications and trends*. Ed. by Soria, E., Martin, J., Magdalena, R., Martinez, M., Serrano, A., IGI Global, 242–264.

Turing, A.M. (1947), 'Lecture to the London Mathematical Society on 20 February 1947'. *Carpenter and Doran (1986).*

Turing, A.M. (1950) 'Computing Machinery and Intelligence' *Mind* lix (236), 433-460.

Urban, G., Geras, K.J., Kahou, S.E., Aslan, O., Wang, S., Mohamed, A., Philipose, M., Richardson, M. & Caruana, R. (2017) 'Do Deep Convolutional Nets Really Need to be Deep and Convolutional?'. *International Conference on Learning Representations (ICLR)*. Held 24-26 April, 2017, Toulon, France.

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

Vilalta, R., Giraud-Carrier, C., Brazdil, P. and Soares, C. (2011) 'Inductive Transfer', in Sammut, C. & Webb, G.I. eds. *Encyclopaedia of Machine Learning*. Springer, Boston, MA.

Wang, J. & Perez, L. (2017) 'The Effectiveness of Data Augmentation in Image Classification using Deep Learning'. *ArXiv* [online] Available from <https://arxiv.org/abs/1712.04621> [Accessed 18 May 2020].

Wei, D., Zhou, B., Torrabla, A. & Freeman, W. (2015) 'Understanding Intra-Class Knowledge Inside CNN'. *ArXiv* [online] Available from <https://arxiv.org/abs/1507.02379> [Accessed 23 December 2020].

Williams, J. M. (2017) *Deep Learning and Transfer Learning in the Classification of EEG Signals.* Master of Science Thesis. Department of Computer Science and Engineering, University of Nebraska-Lincoln.

Wilson, S. W. (2001) 'Mining oblique data with XCS'. *P.L. Lanzi, W. Stolzmann and S.W. Wilson (Eds.) Advances in Learning Classifier Systems: Third International Workshop (IWLCS 2000).* Held at Berlin, Heidelberg: Springer-Verlag, 283-290.

Wu, J., Chen, X, Zhang, H., Xiong, L., Lei, H. & Deng, S. (2019) 'Hyperparameter Optimisation for Machine Learning Models Based on Bayesian Optimisation'. *Journal of Electronic Science and Technology* 17(1), 26-40.

Wu, Y., Liu, L., Bae, J., Chow, K., Iyengar, A., Pu, C., Wei, W., Yu, L. & Zhang, Q. (2019) 'Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks'. *2019 IEEE International Conference on Big Data (Big Data).* Los Angeles, California, 1971-1980.

Xu, Y. & Goodacre, R. (2018) 'On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning'. *Journal of Analysis and Testing* 2, 249-262.

Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. (2014) 'How transferable are features in deep neural networks?'. *Advances in Neural Informational Processing Systems 27 (NIPS '14)*. NIPS Foundation.

Yosinski, J., Clune, J. Nguyen, A, Fuchs, T. & Lipson, H. (2015) 'Understanding Neural Networks Through Deep Visualisation'. *31st International Conference on Machine Learning: Deep Learning Workshop*. Lille, France.

Yu, X., Wang, J., Kays, R., Jansen, P.A., Wang, T. & Huang, T. (2013) Automated identification of animal species in camera trap images'. *Journal on Image and Video Processing,* 52.

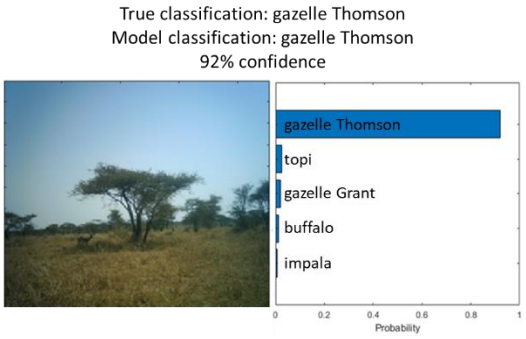
Zeiler, M.D. & Fergus, R. (2014) 'Visualising and Understanding Convolutional Networks'. In Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) *Computer Vision – ECCV 2014*, 818-833.

Appendix A

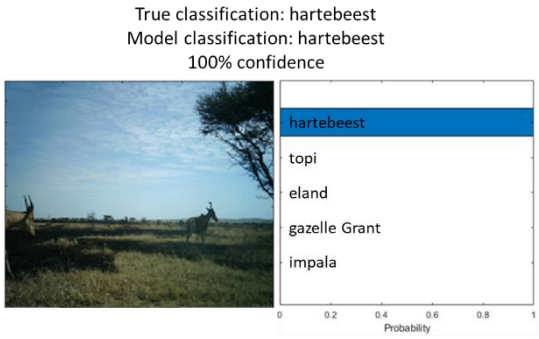
This section provides the complete set of examples by experiment, illustrating the true versus predicted classifications and confidence levels for the images that are displayed.

Experiment 1A

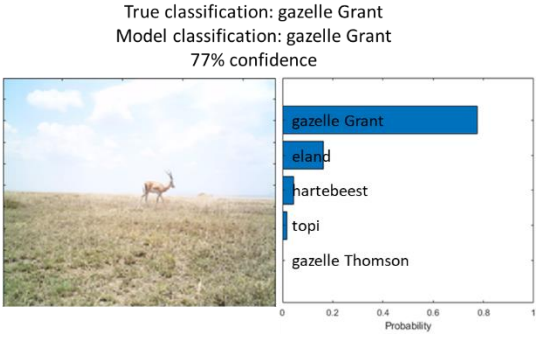
AlexNet



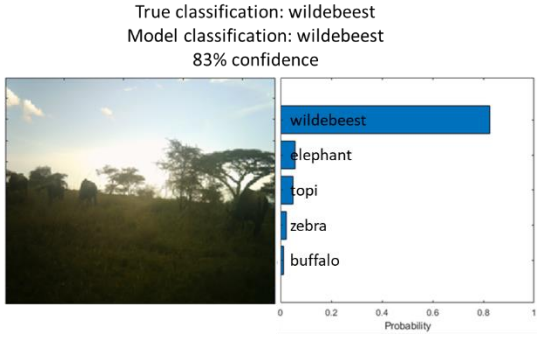
(a)



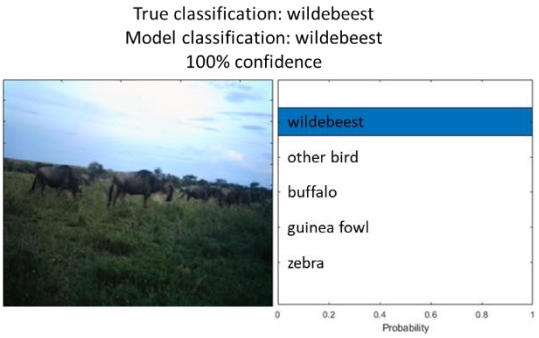
(b)



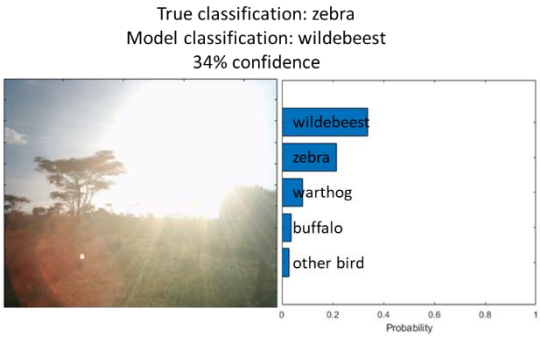
(c)



(d)



(e)



(f)

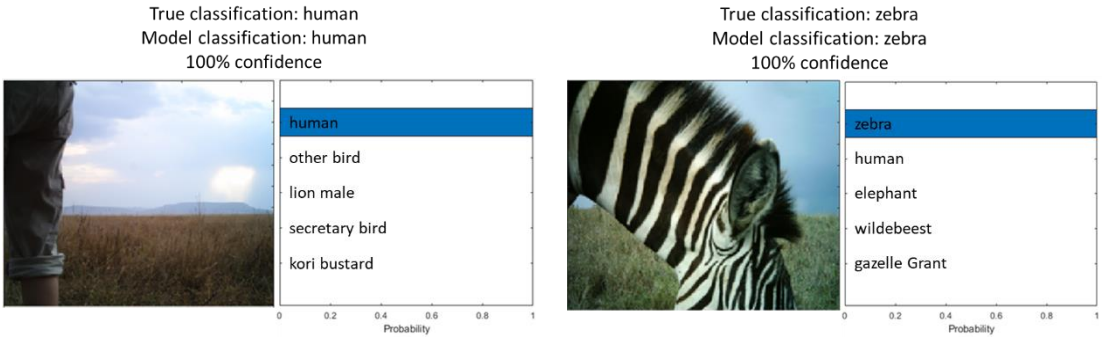


Figure 146: Model predictions with confidence level (Experiment 1A - AlexNet)

GoogLeNet

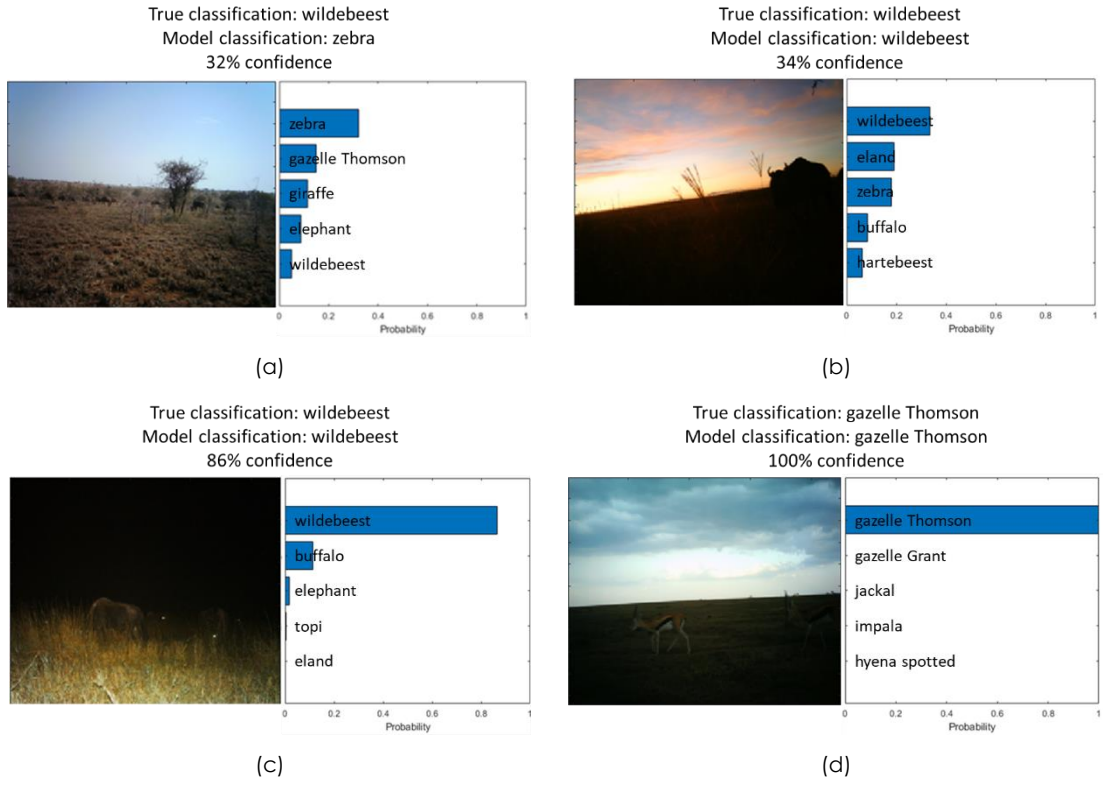
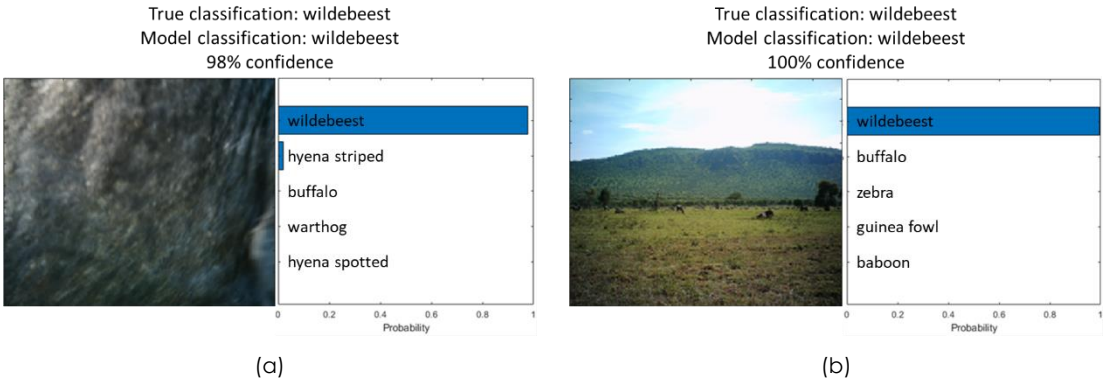




Figure 147: Model predictions with confidence level (Experiment 1A - GoogLeNet)

ResNet-101



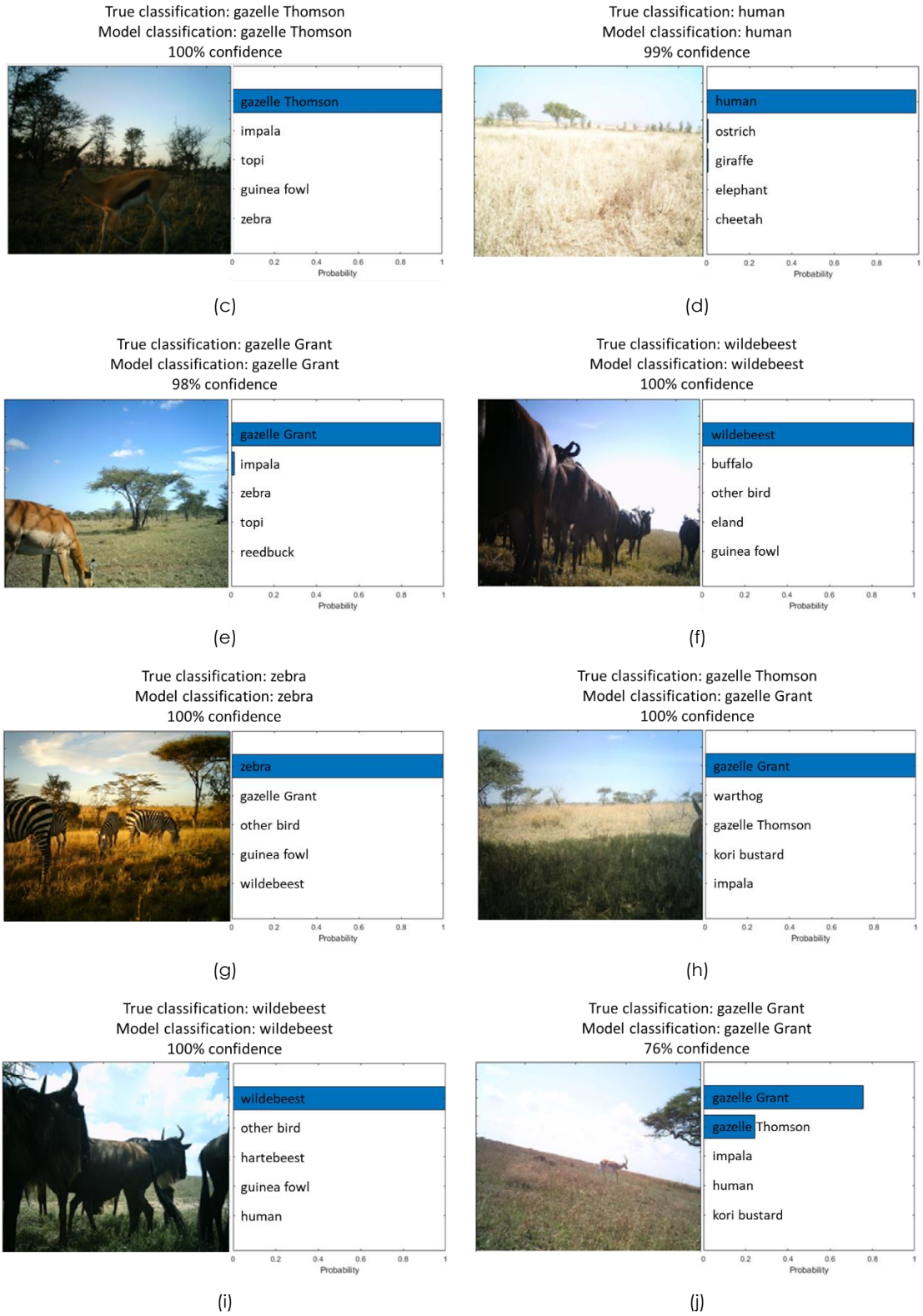
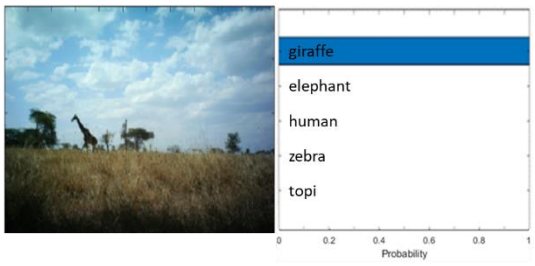


Figure 148: Model predictions with confidence level (Experiment 1A - ResNet-101)

Experiment 1B

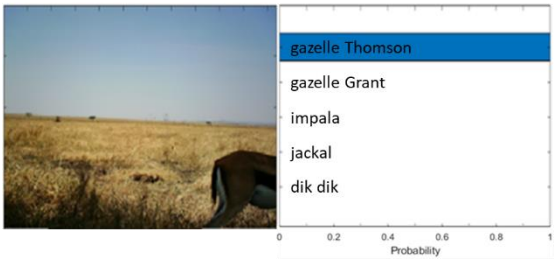
AlexNet

True classification: giraffe
Model classification: giraffe
100% confidence



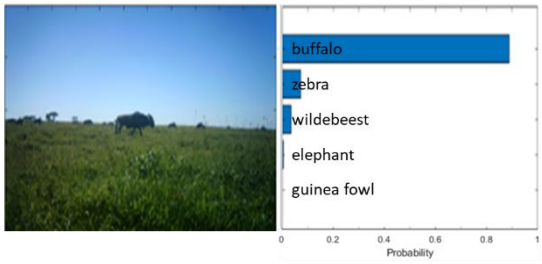
(a)

True classification: gazelle Thomson
Model classification: gazelle Thomson
100% confidence



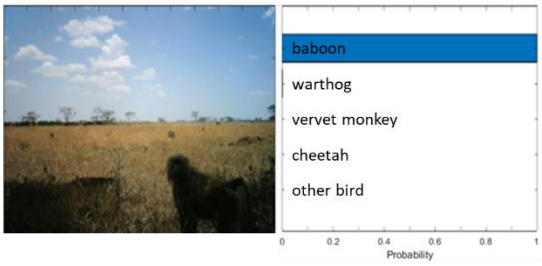
(b)

True classification: wildebeest
Model classification: buffalo
89% confidence



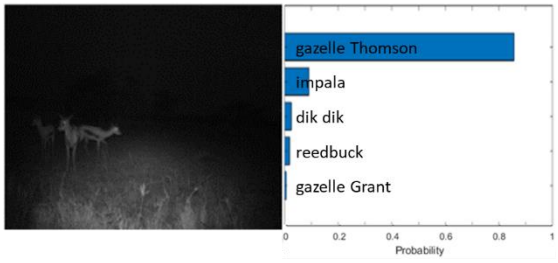
(c)

True classification: baboon
Model classification: baboon
100% confidence



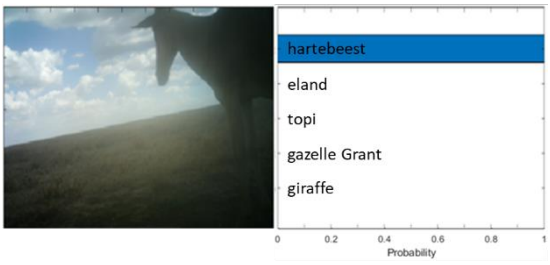
(d)

True classification: gazelle Thomson
Model classification: gazelle Thomson
86% confidence



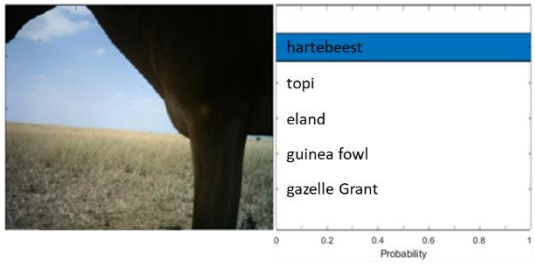
(e)

True classification: hartebeest
Model classification: hartebeest
100% confidence



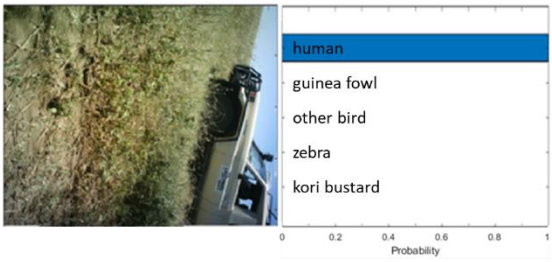
(f)

True classification: hartebeest
Model classification: hartebeest
100% confidence

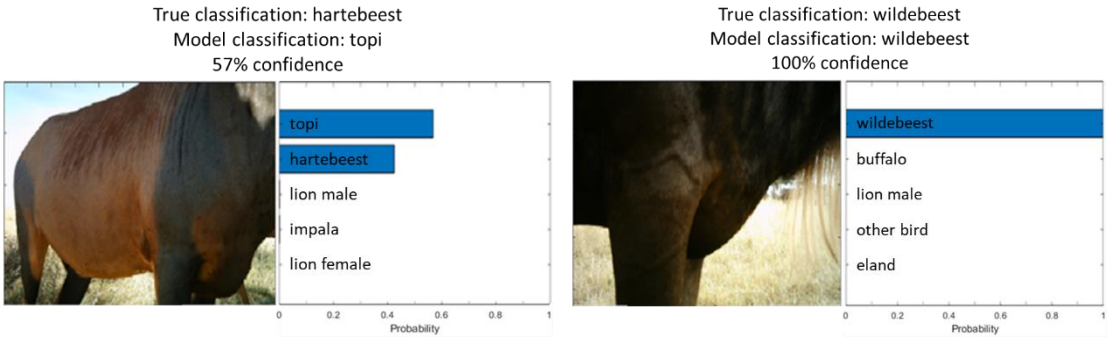


(g)

True classification: human
Model classification: human
100% confidence



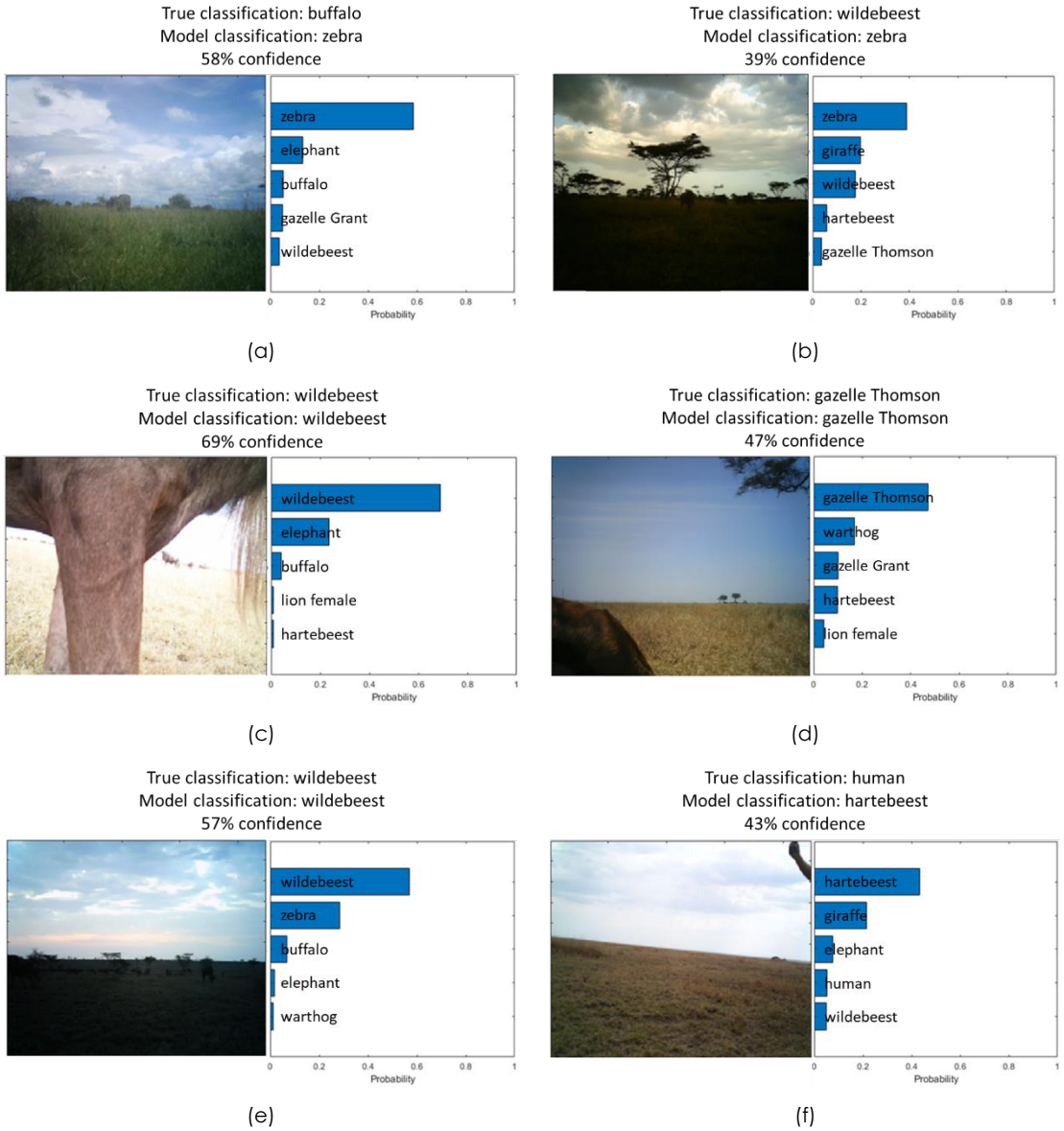
(h)



(i) (j)

Figure 149: Model predictions with confidence level (Experiment 1B - AlexNet)

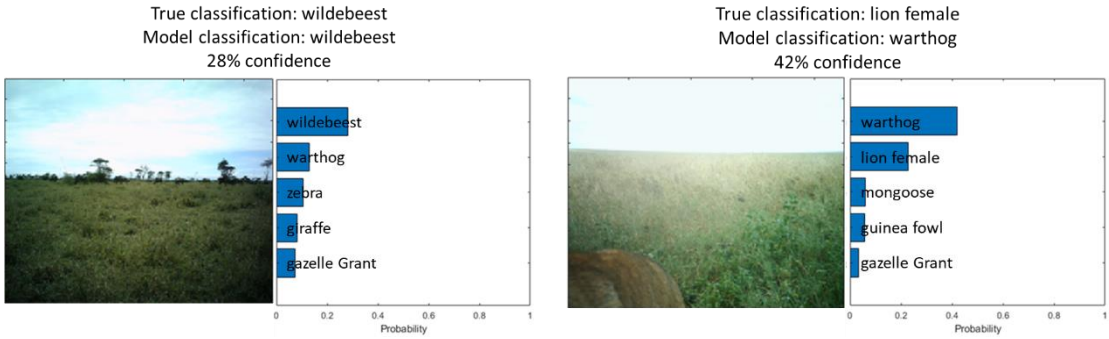
GoogLeNet



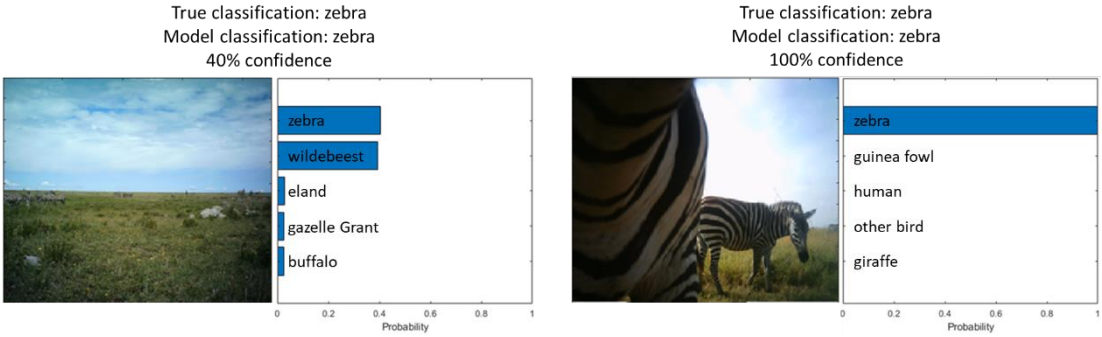
(a) (b)

(c) (d)

(e) (f)



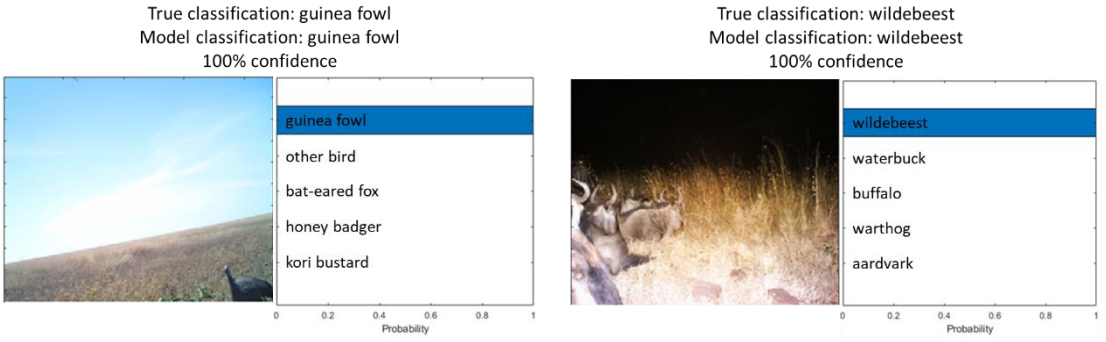
(g) (h)



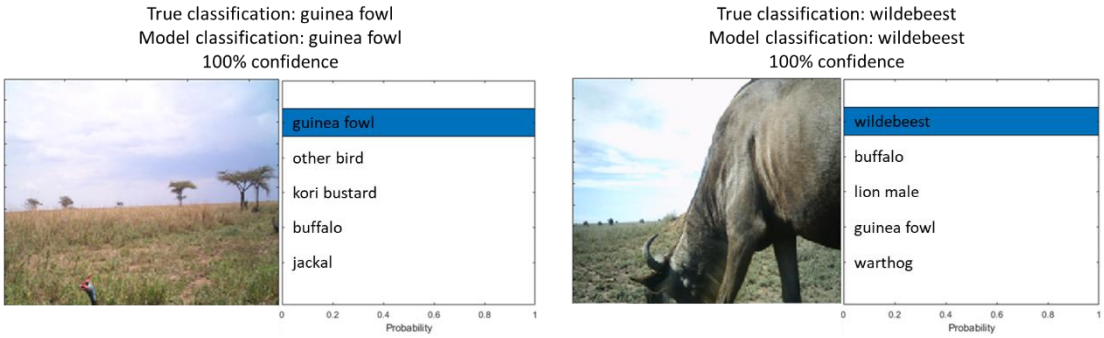
(i) (j)

Figure 150: Model predictions with confidence level (Experiment 1B - GoogLeNet)

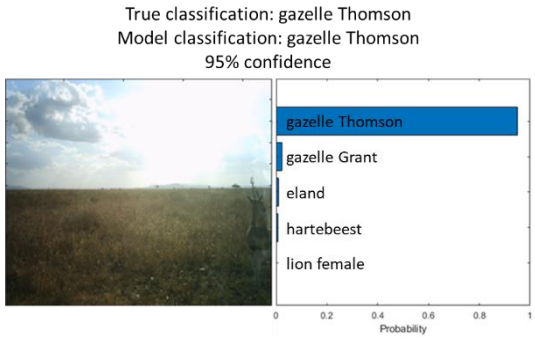
ResNet-101



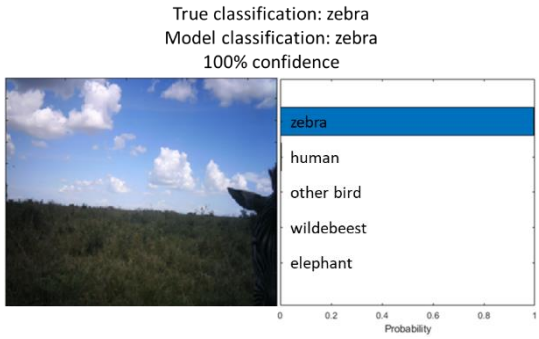
(a) (b)



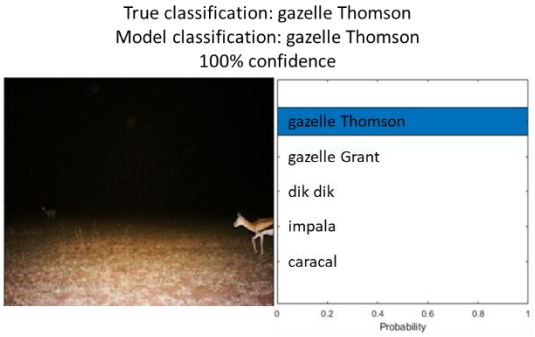
(c) (d)



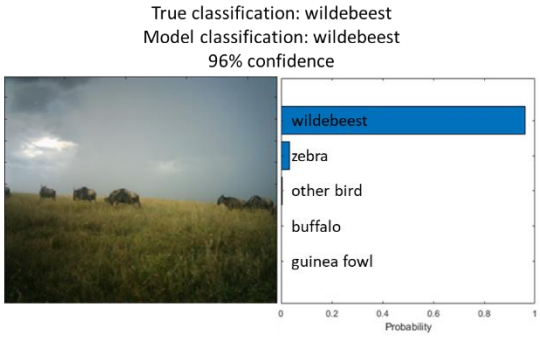
(e)



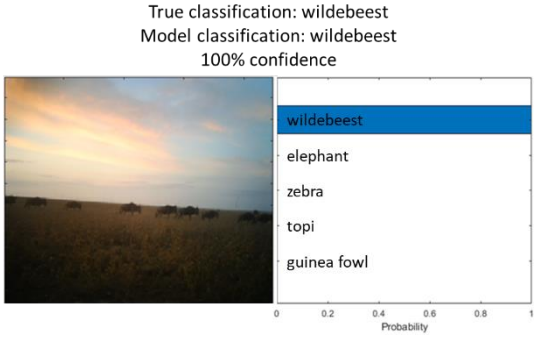
(f)



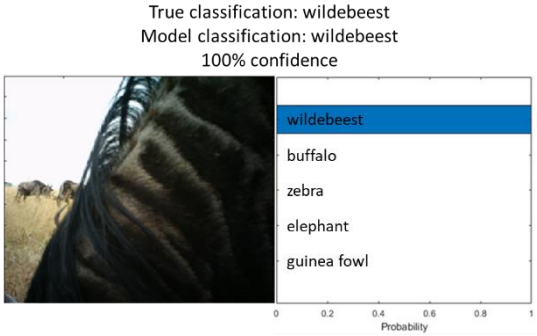
(g)



(h)



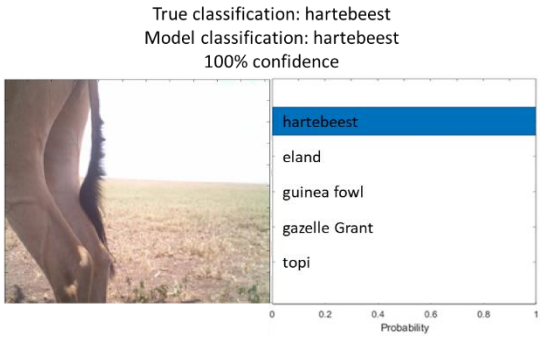
(i)



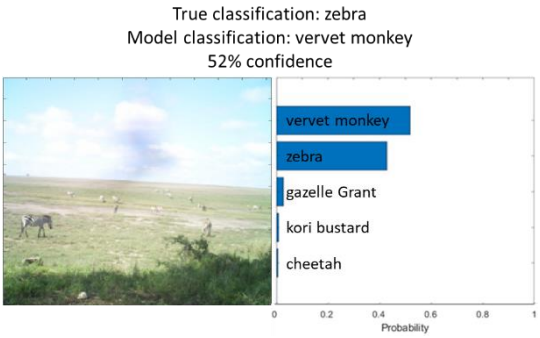
(j)

Figure 151: Model predictions with confidence level (Experiment 1B - ResNet-101)

VGG-19



(a)



(b)

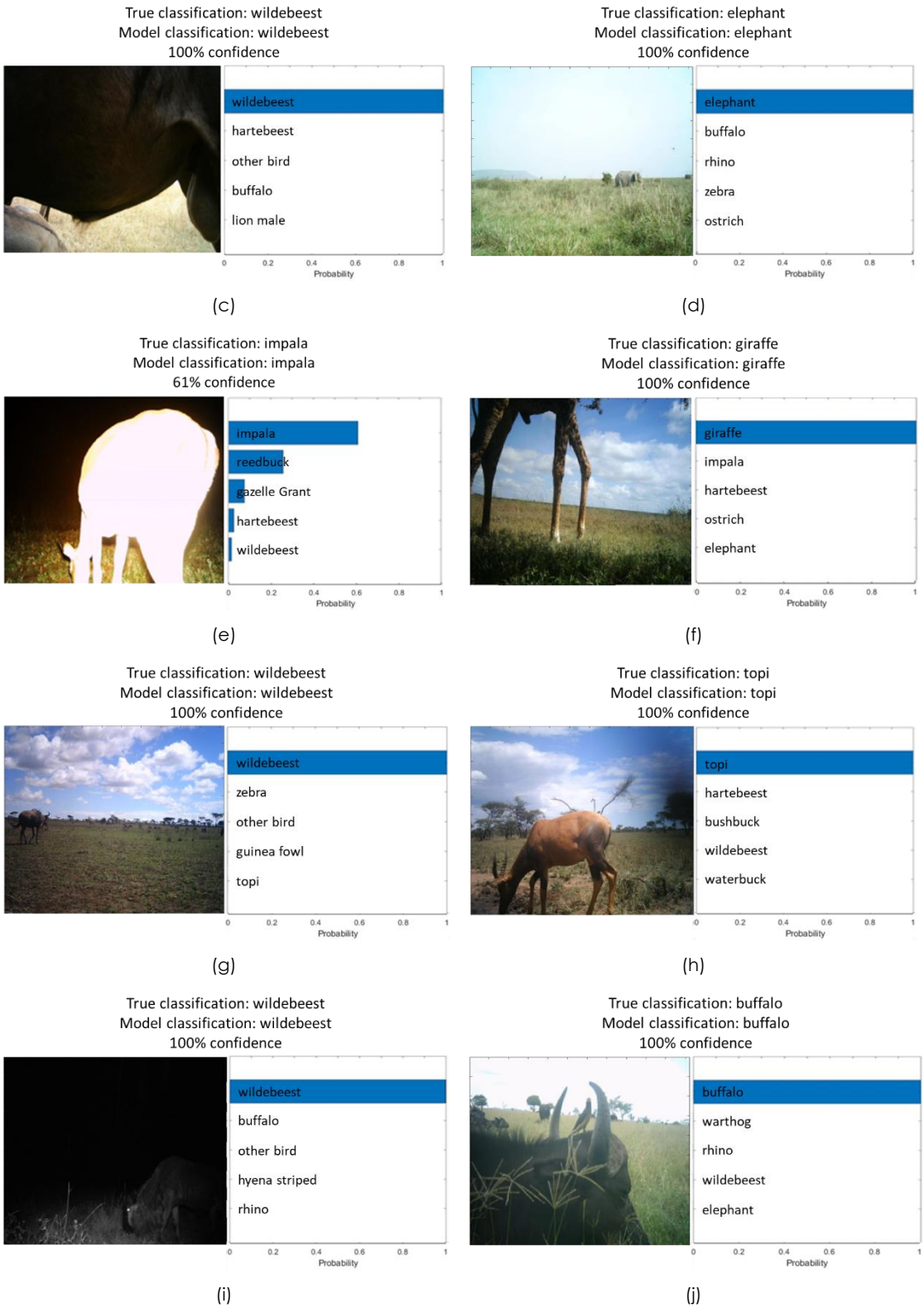
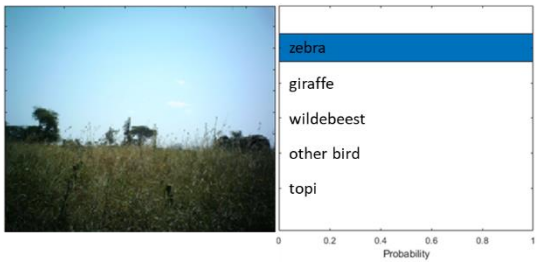


Figure 152: Model predictions with confidence level (Experiment 1B - VGG-19)

Experiment 2A

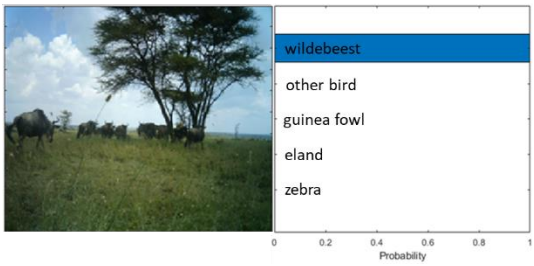
GoogLeNet

True classification: zebra
Model classification: zebra
100% confidence



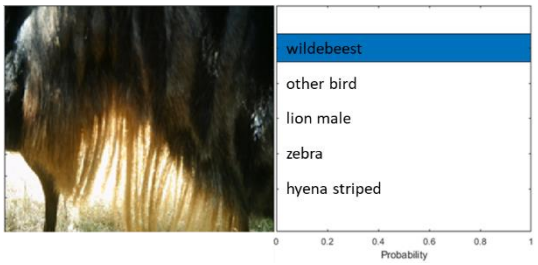
(a)

True classification: wildebeest
Model classification: wildebeest
100% confidence



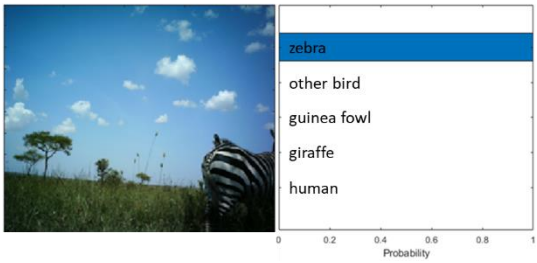
(b)

True classification: wildebeest
Model classification: wildebeest
100% confidence



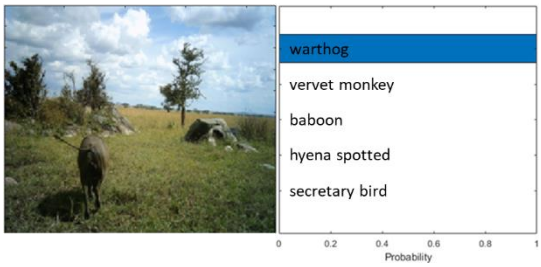
(c)

True classification: zebra
Model classification: zebra
100% confidence



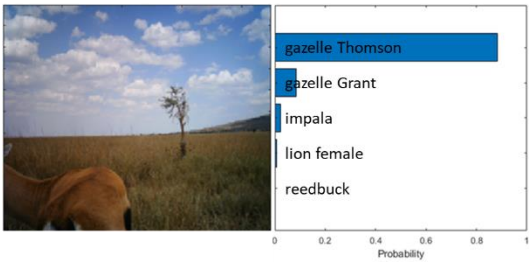
(d)

True classification: warthog
Model classification: warthog
100% confidence



(e)

True classification: gazelle Thomson
Model classification: gazelle Thomson
88% confidence



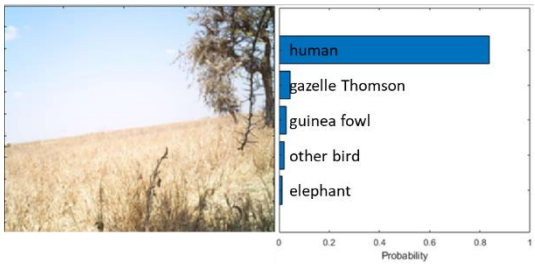
(f)

True classification: gazelle Thomson
Model classification: gazelle Thomson
100% confidence

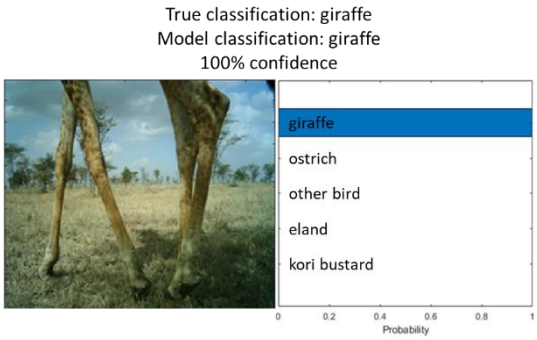


(g)

True classification: human
Model classification: human
84% confidence



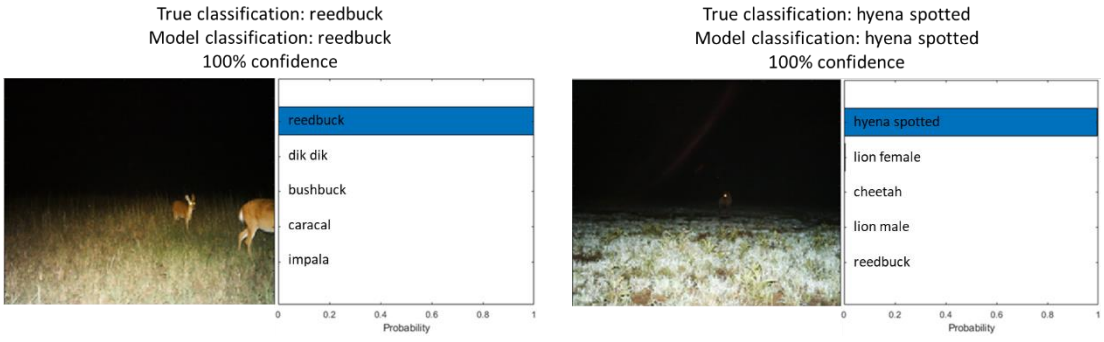
(h)



(i)

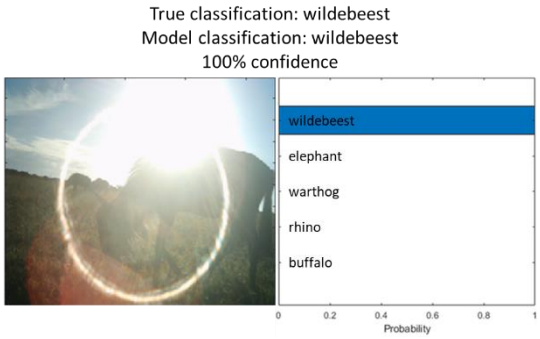
Figure 153: Model predictions with confidence level (Experiment 2A - GoogLeNet)

ResNet-101

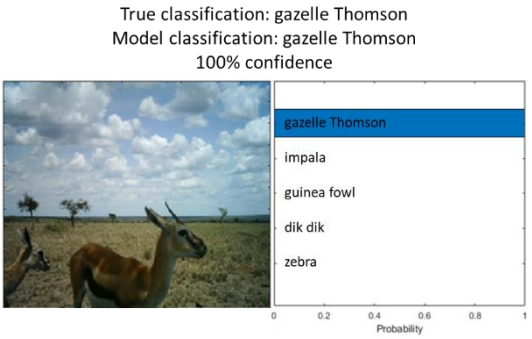


(a)

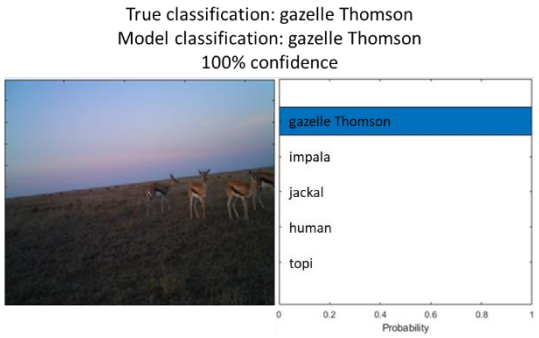
(b)



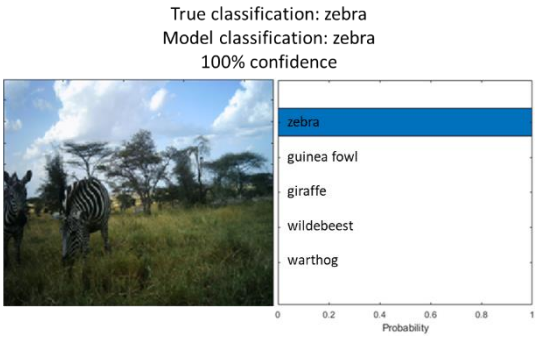
(c)



(d)



(e)



(f)

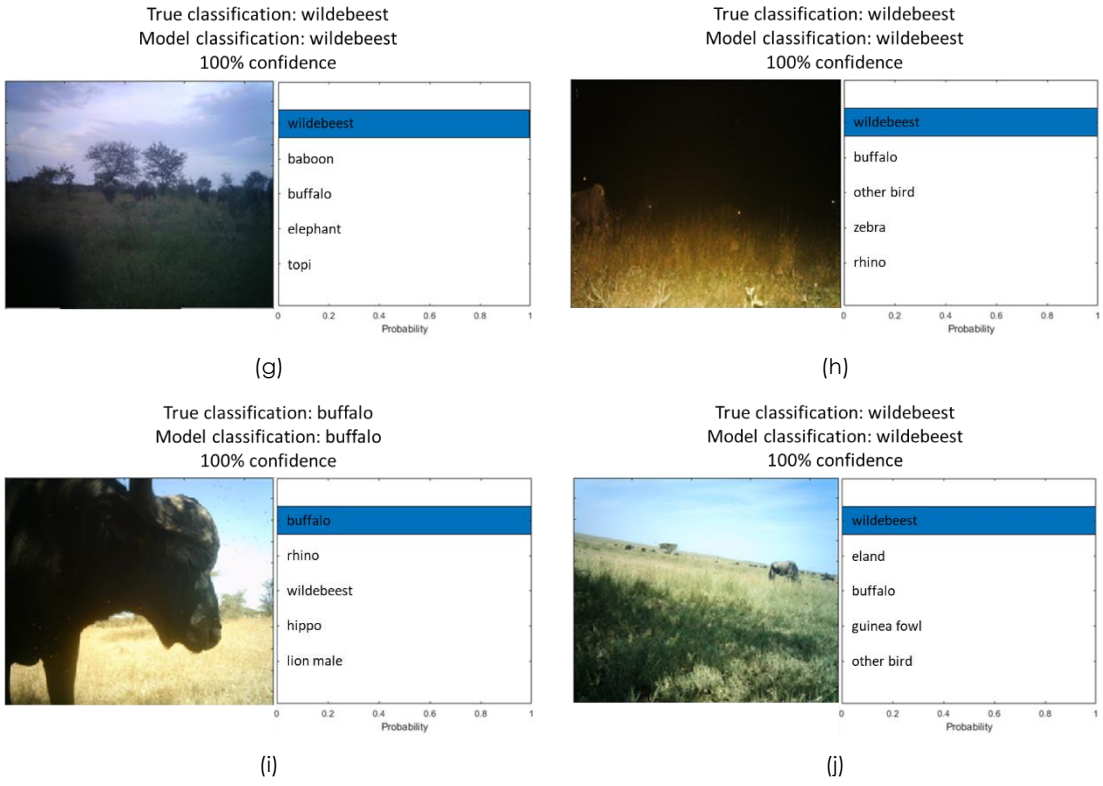


Figure 154: Model predictions with confidence level (Experiment 2A - ResNet-101)

VGG-19

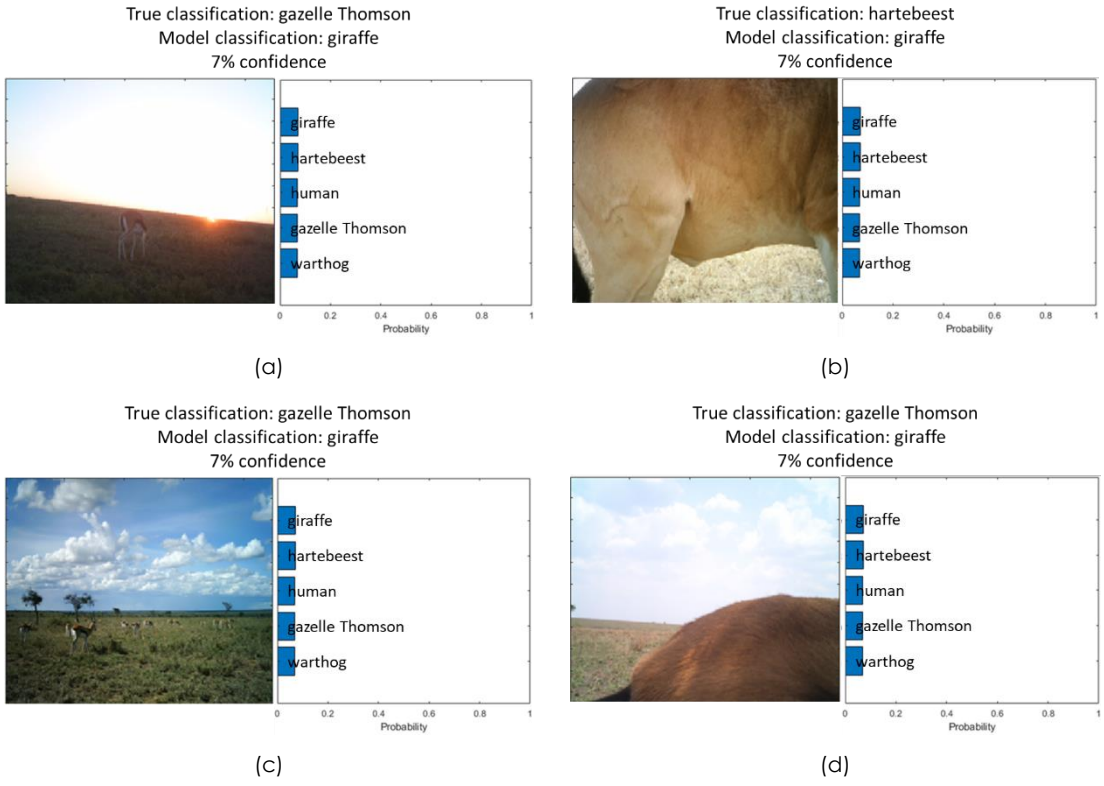
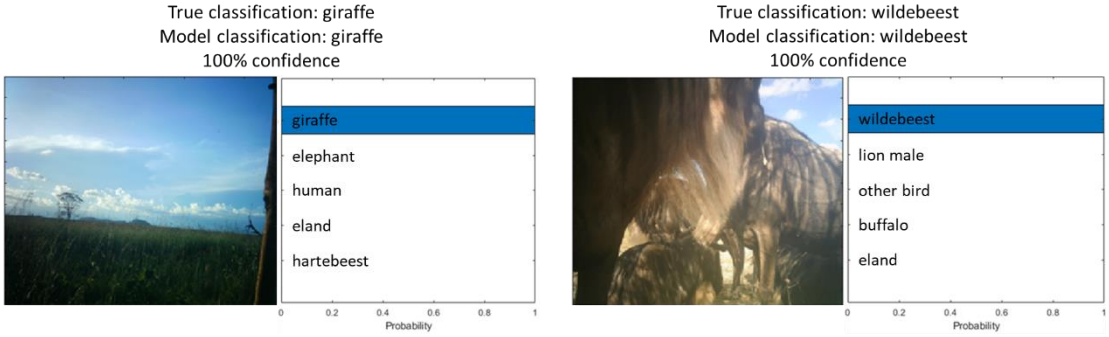




Figure 155: Model predictions with confidence level (Experiment 2A - VGG-19)

Experiment 2B

AlexNet



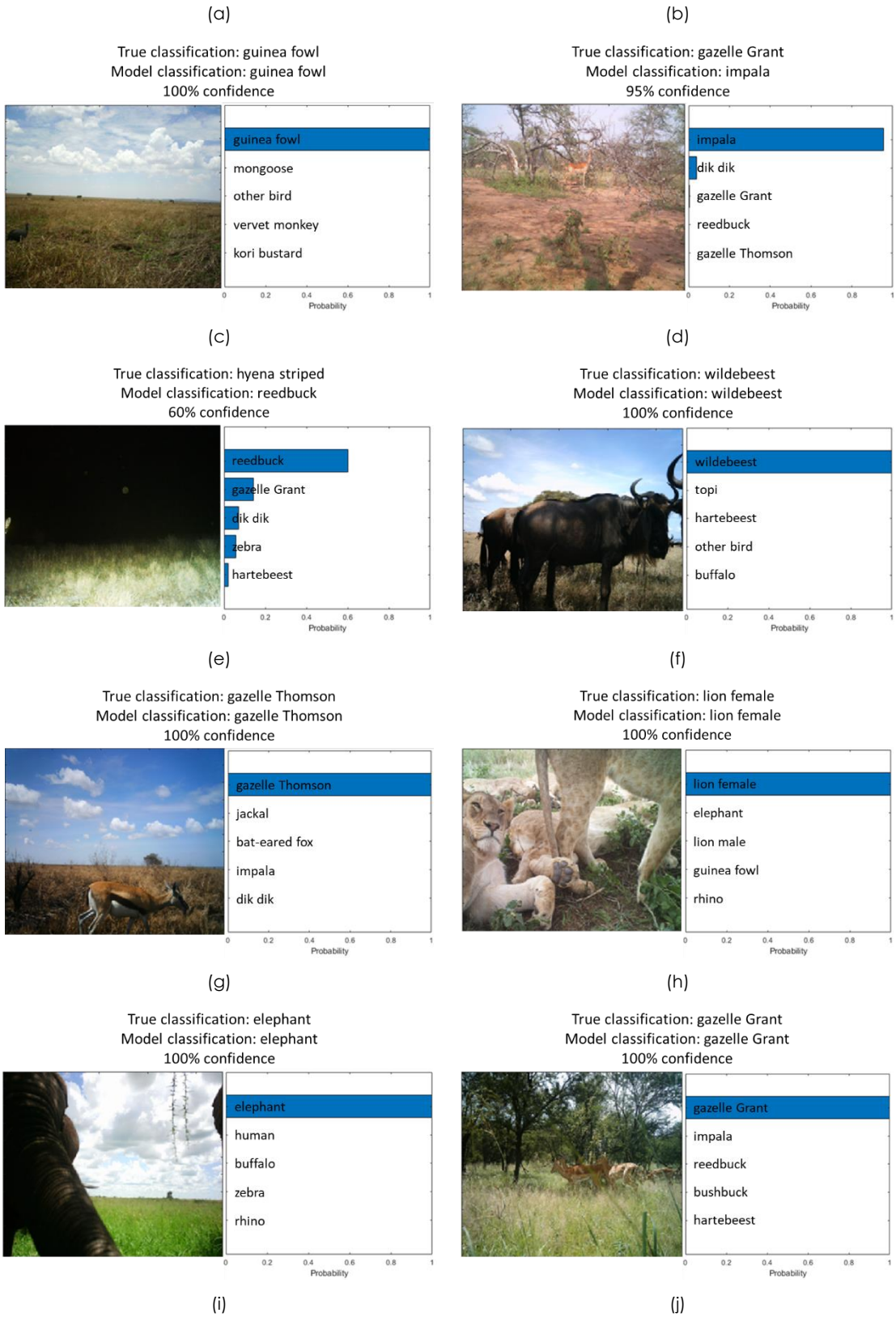
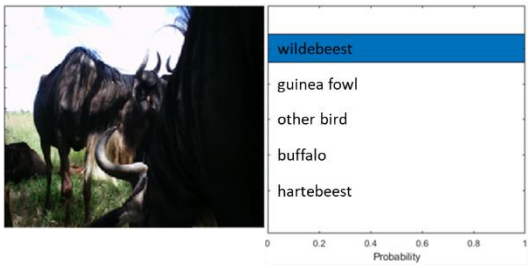


Figure 156: Model predictions with confidence level (Experiment 2B - AlexNet)

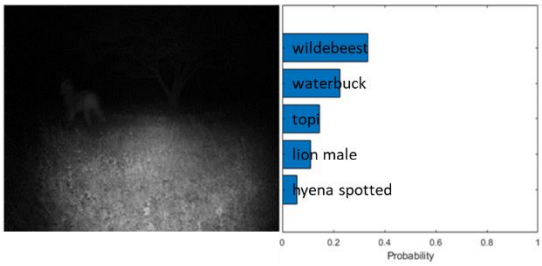
GoogLeNet

True classification: wildebeest
Model classification: wildebeest
100% confidence



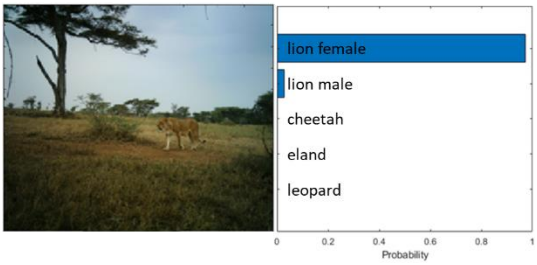
(a)

True classification: caracal
Model classification: wildebeest
33% confidence



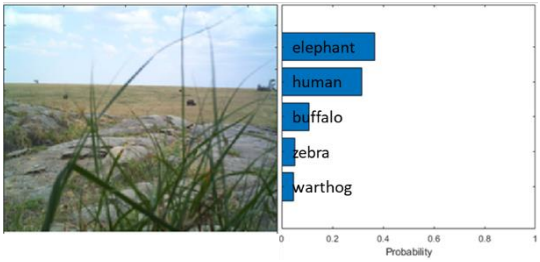
(b)

True classification: lion female
Model classification: lion female
97% confidence



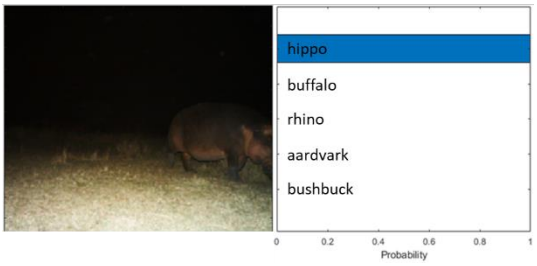
(c)

True classification: buffalo
Model classification: elephant
37% confidence



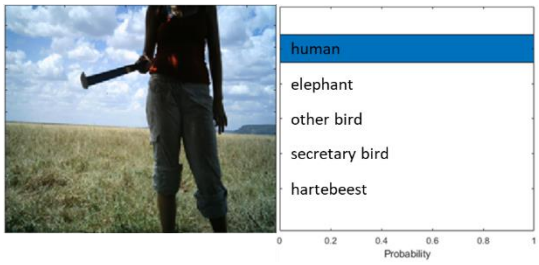
(d)

True classification: hippo
Model classification: hippo
100% confidence



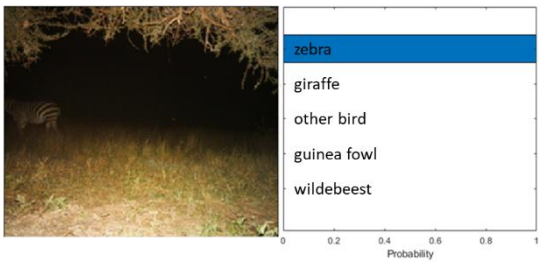
(e)

True classification: human
Model classification: human
100% confidence



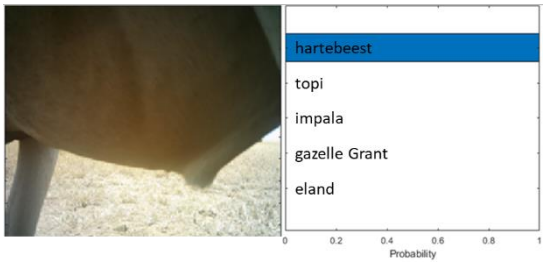
(f)

True classification: zebra
Model classification: zebra
100% confidence

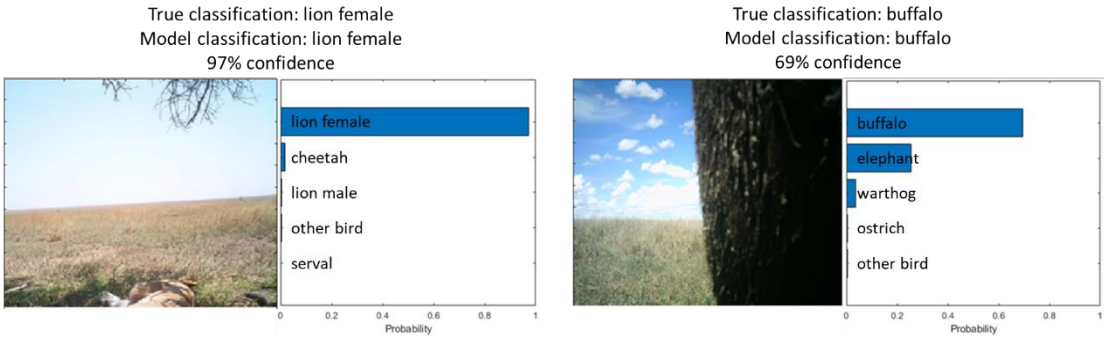


(g)

True classification: hartebeest
Model classification: hartebeest
100% confidence



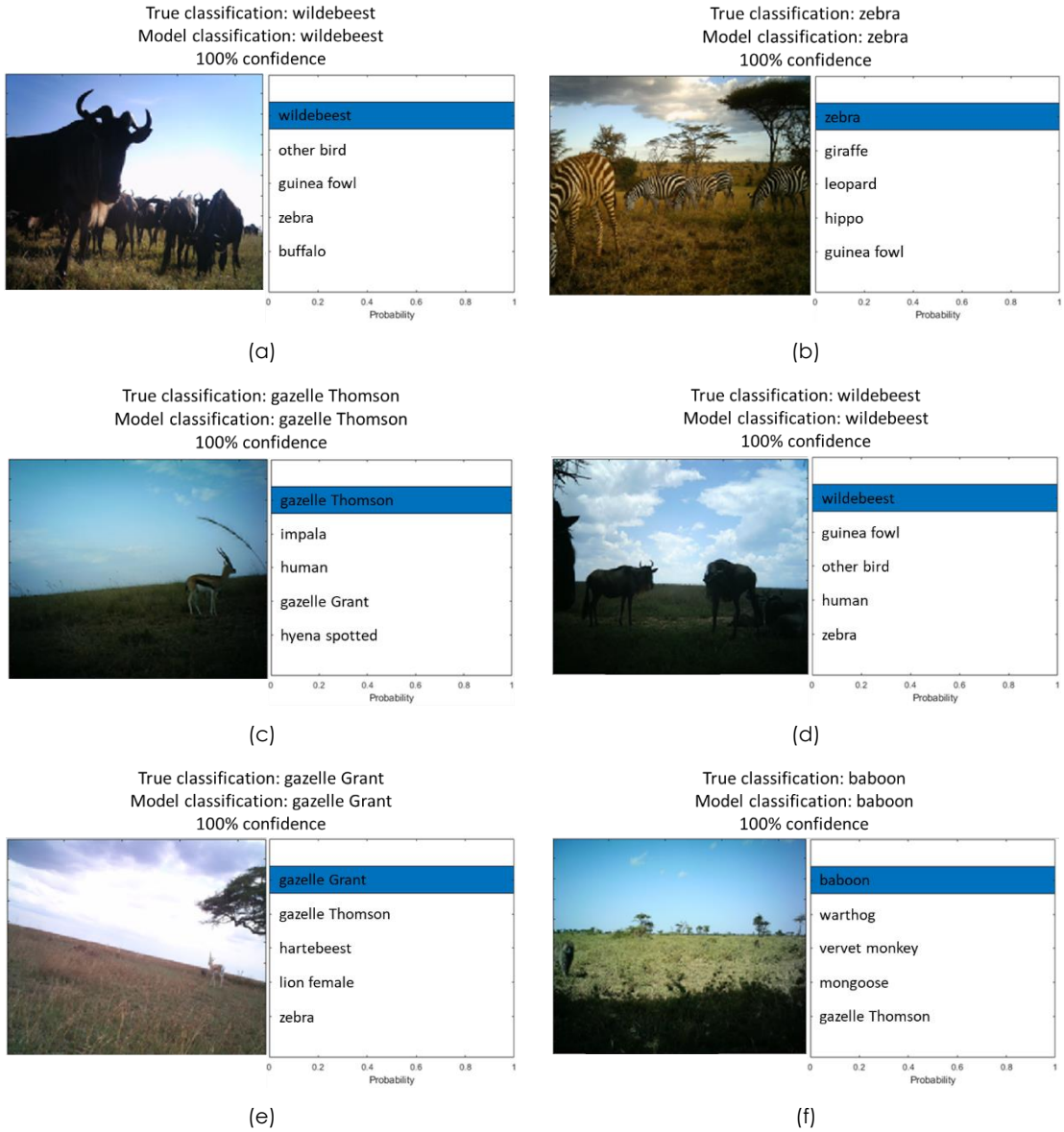
(h)



(i) (j)

Figure 157: Model predictions with confidence level (Experiment 2B - GoogLeNet)

ResNet-101



(a) (b)

(c) (d)

(e) (f)

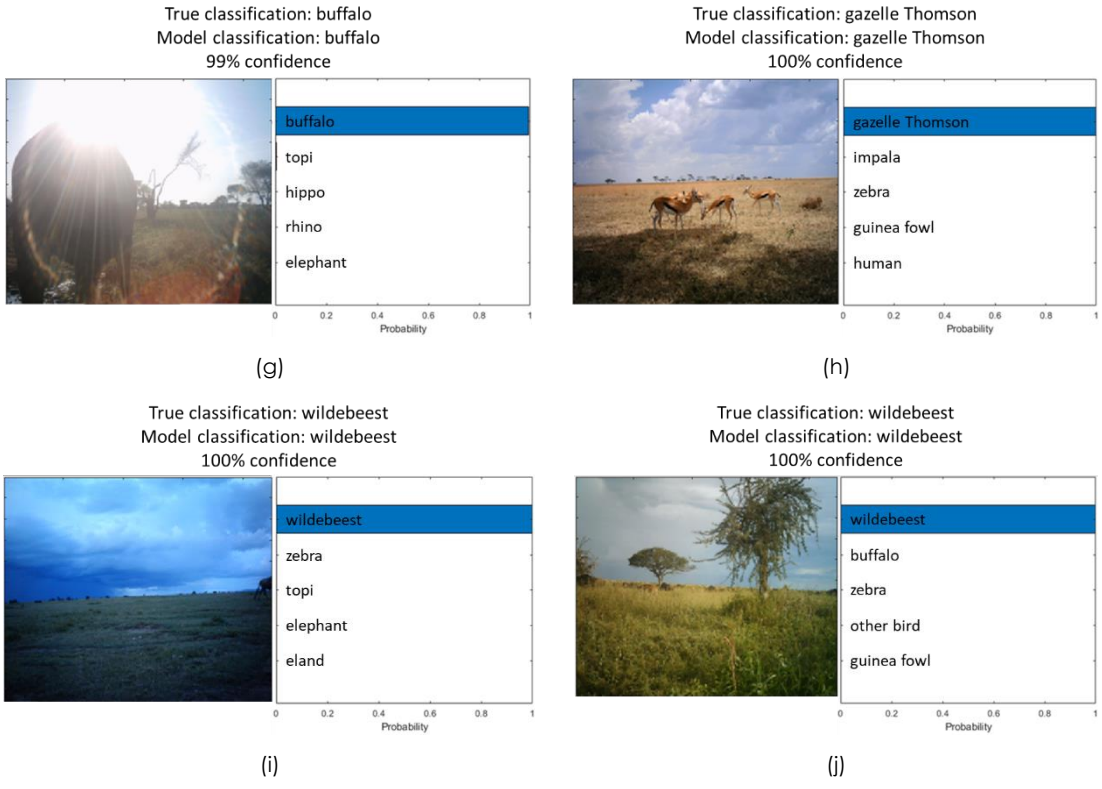
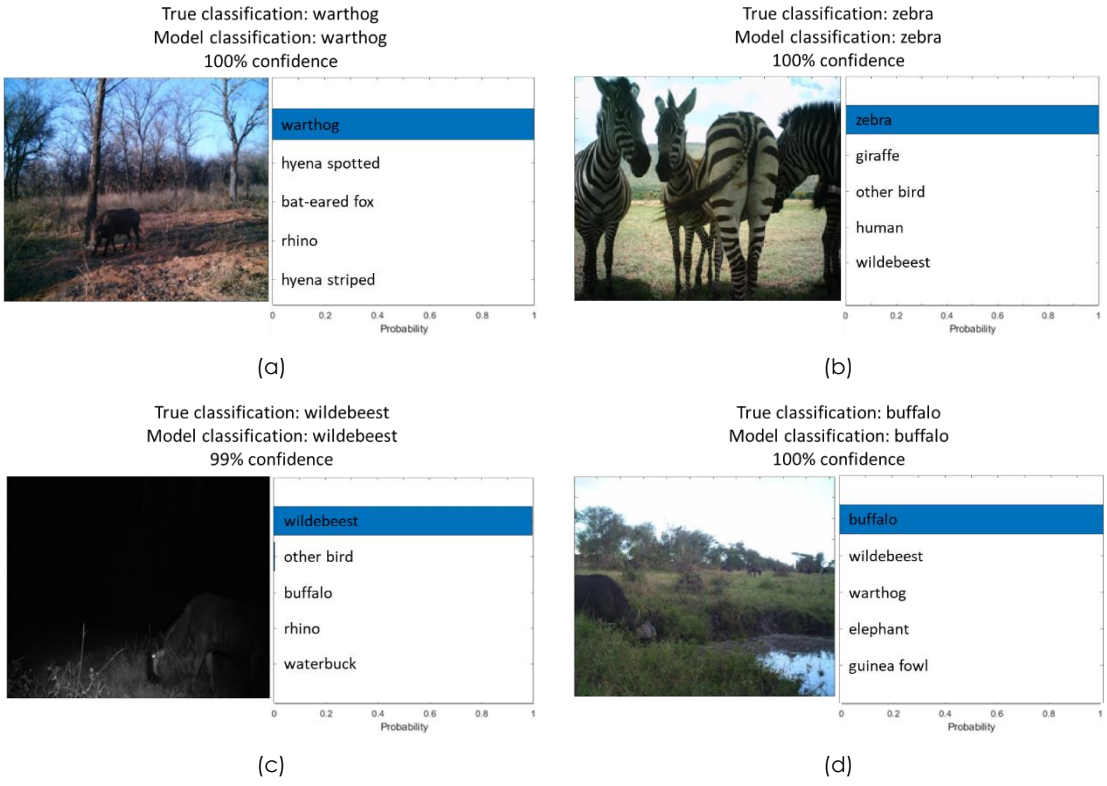


Figure 158: Model predictions with confidence level (Experiment 2B - ResNet-101)

VGG-19



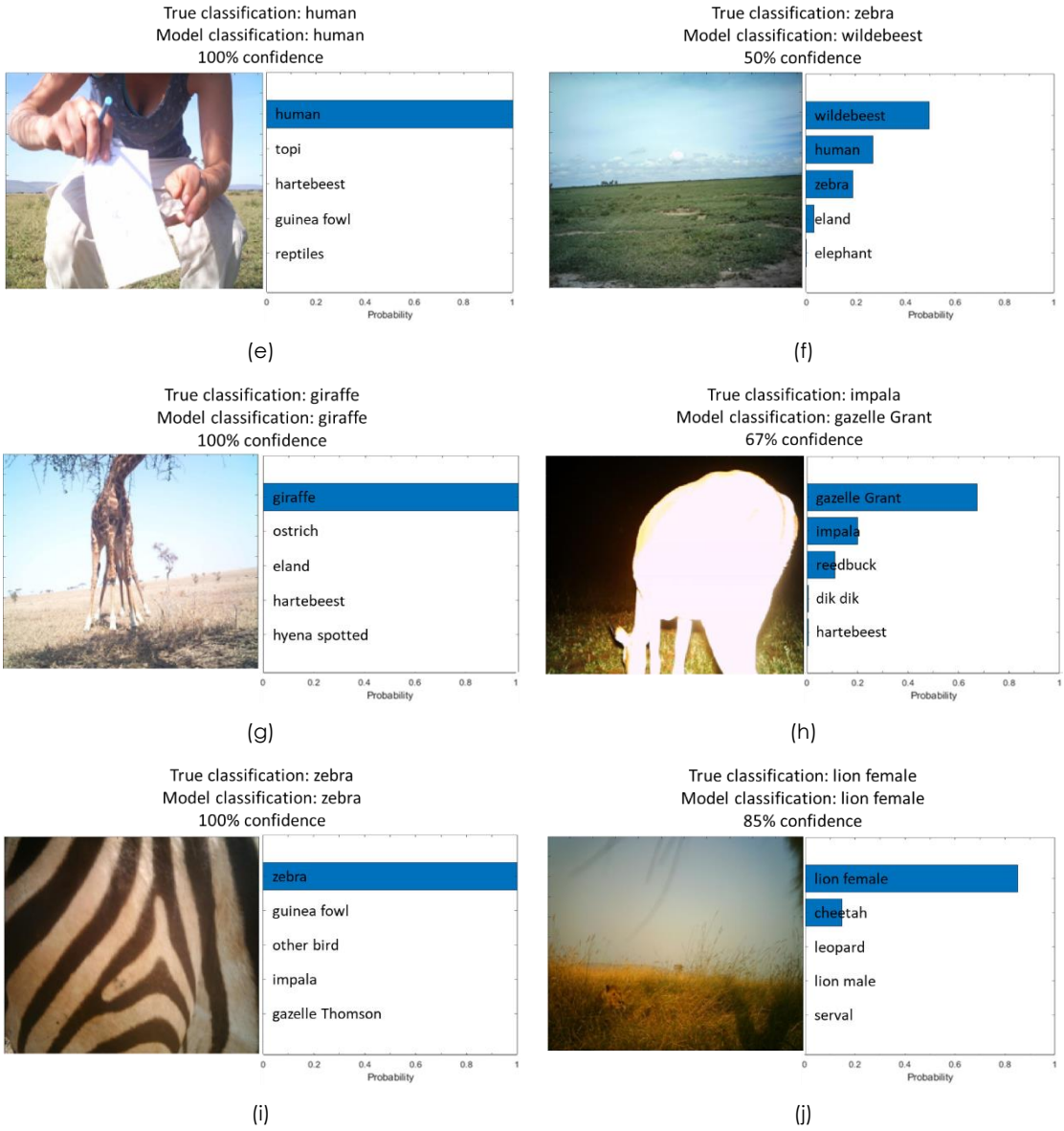
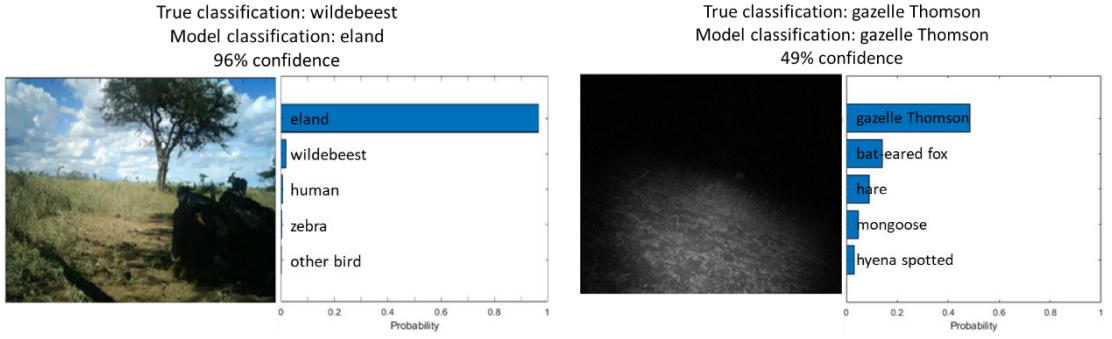


Figure 159: Model predictions with confidence level (Experiment 2B - VGG-19)

Experiment 3

AlexNet



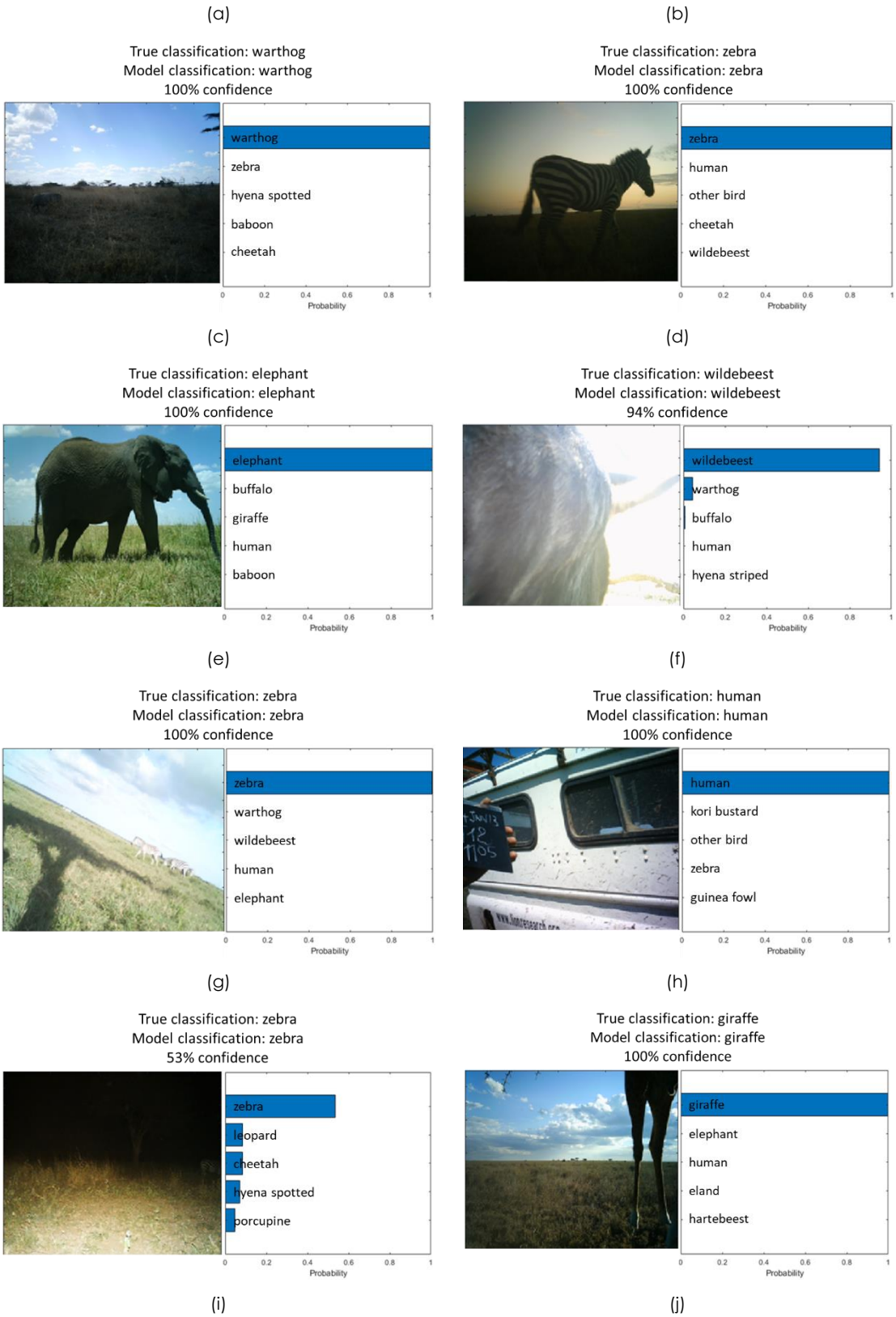
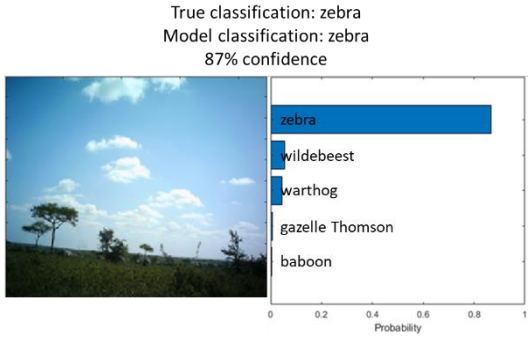
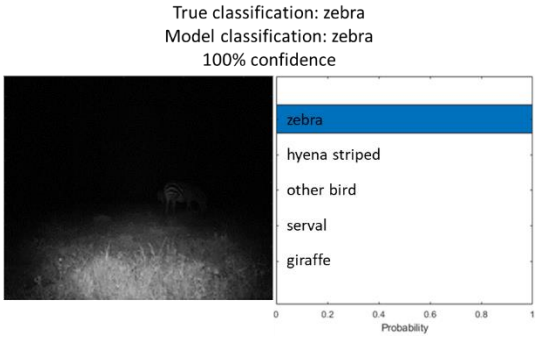


Figure 160: Model predictions with confidence level (Experiment 3 - AlexNet)

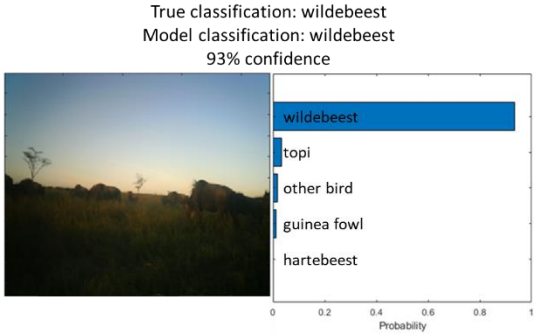
GoogLeNet



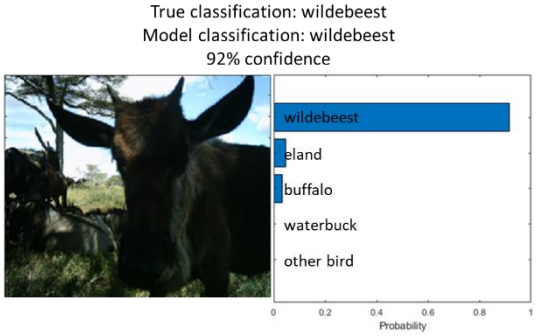
(a)



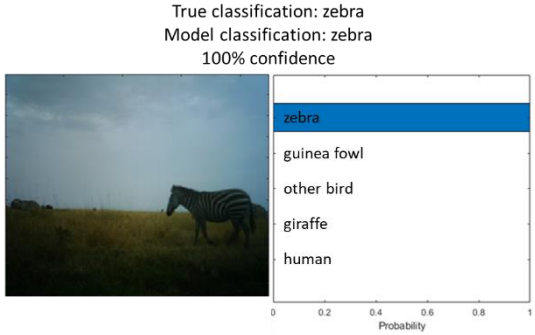
(b)



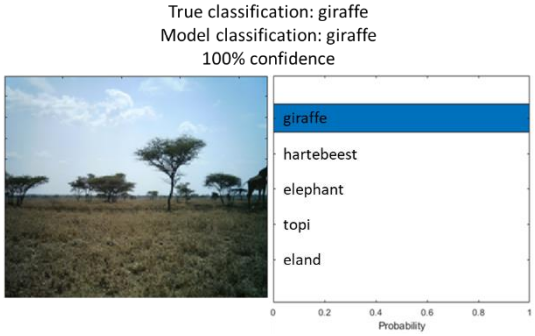
(c)



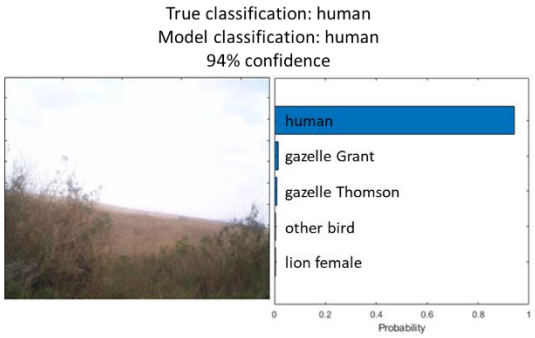
(d)



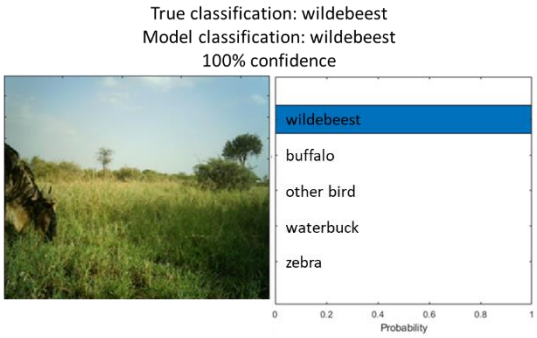
(e)



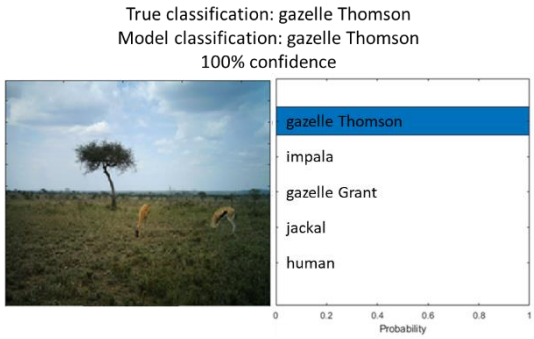
(f)



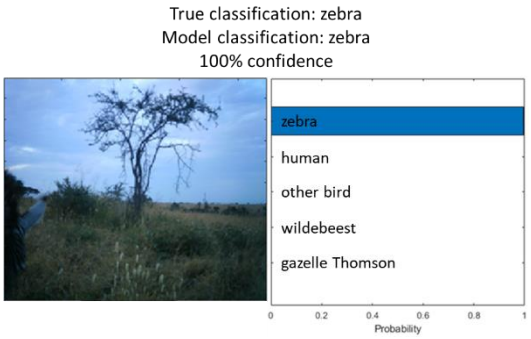
(g)



(h)



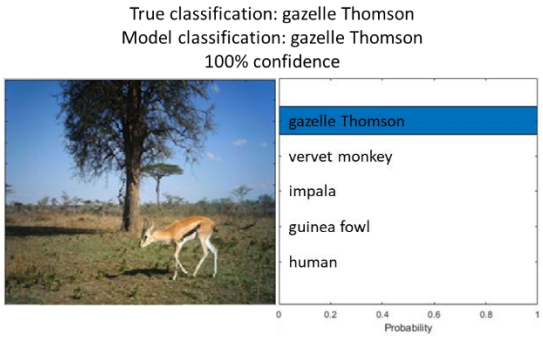
(i)



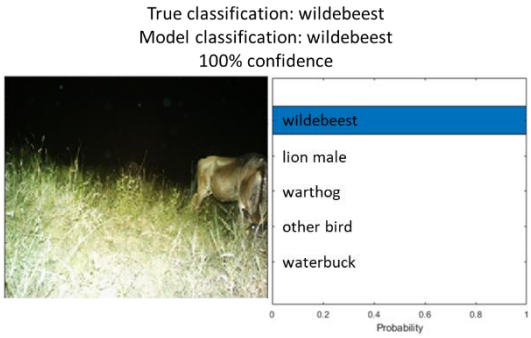
(j)

Figure 161: Model predictions with confidence level (Experiment 3 - GoogLeNet)

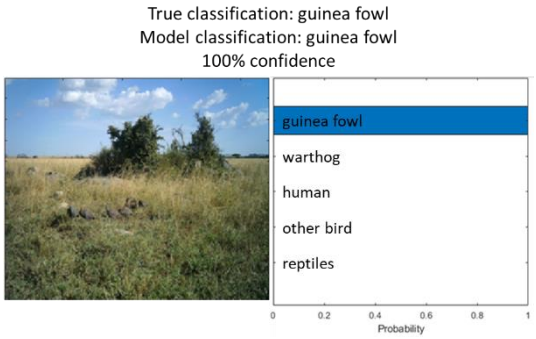
ResNet-101



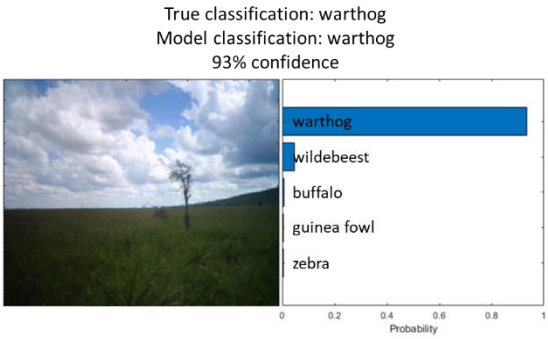
(a)



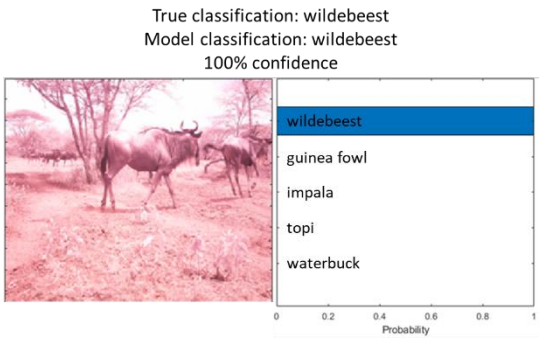
(b)



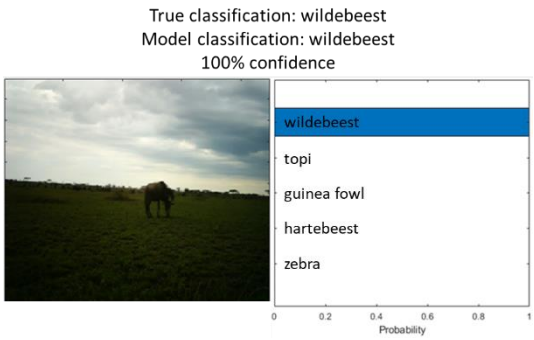
(c)



(d)



(e)



(f)



Figure 162: Model predictions with confidence level (Experiment 3 - ResNet-101)

VGG-19

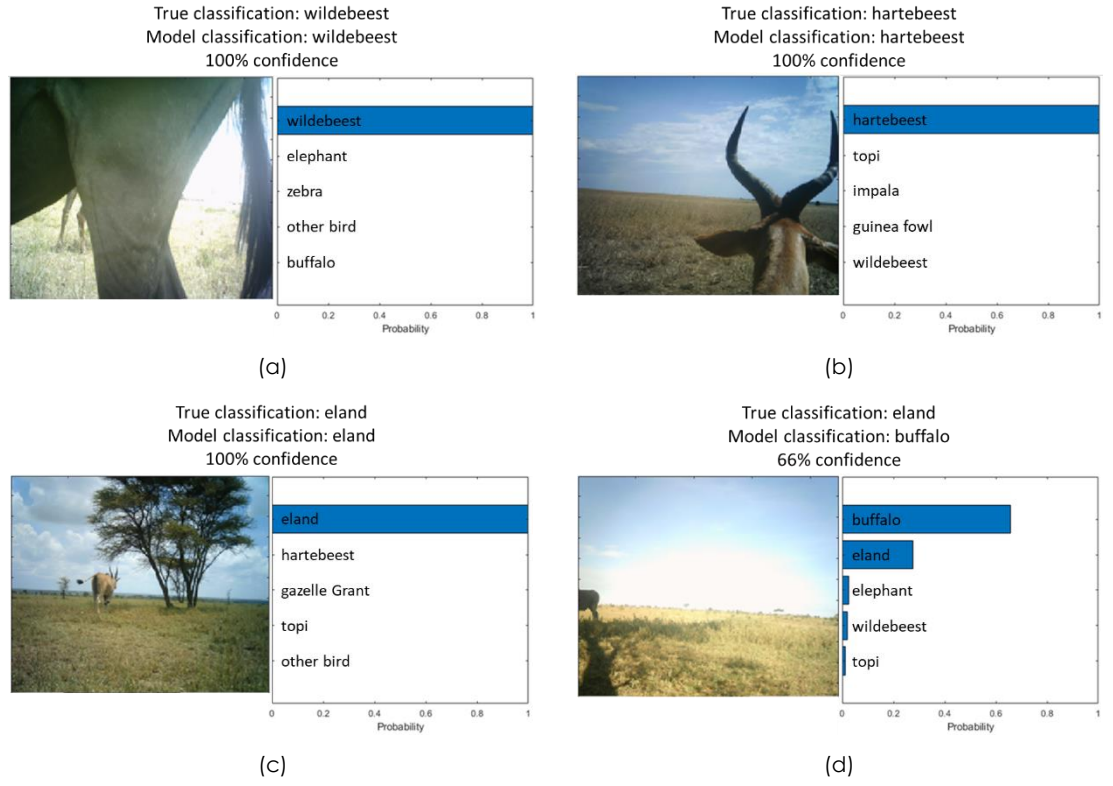




Figure 163: Model predictions with confidence level (Experiment 3 - VGG-19)

Appendix B

This section displays the confusion matrices for each experiment.

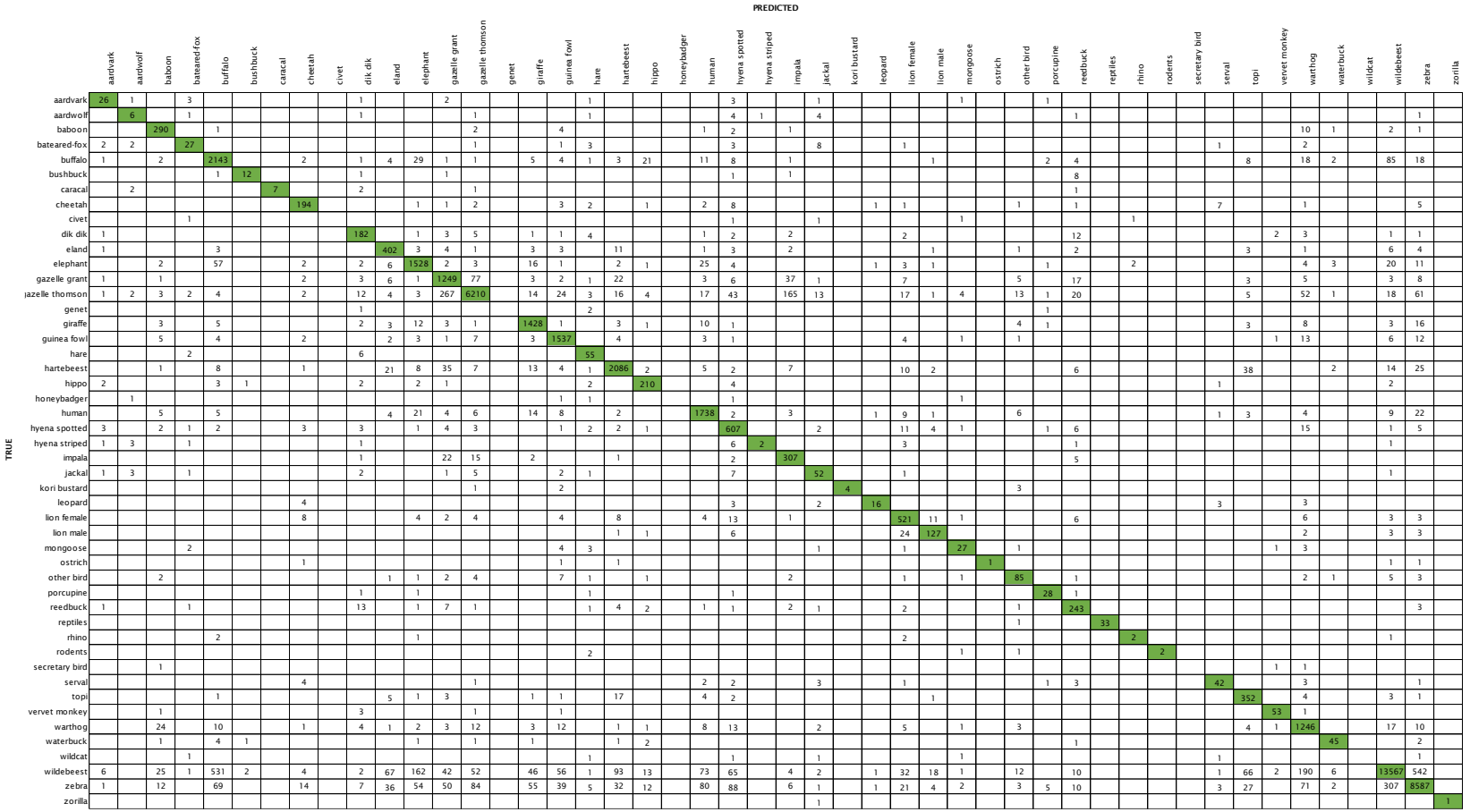


Figure 164: Confusion matrix for Experiment 1A - AlexNet

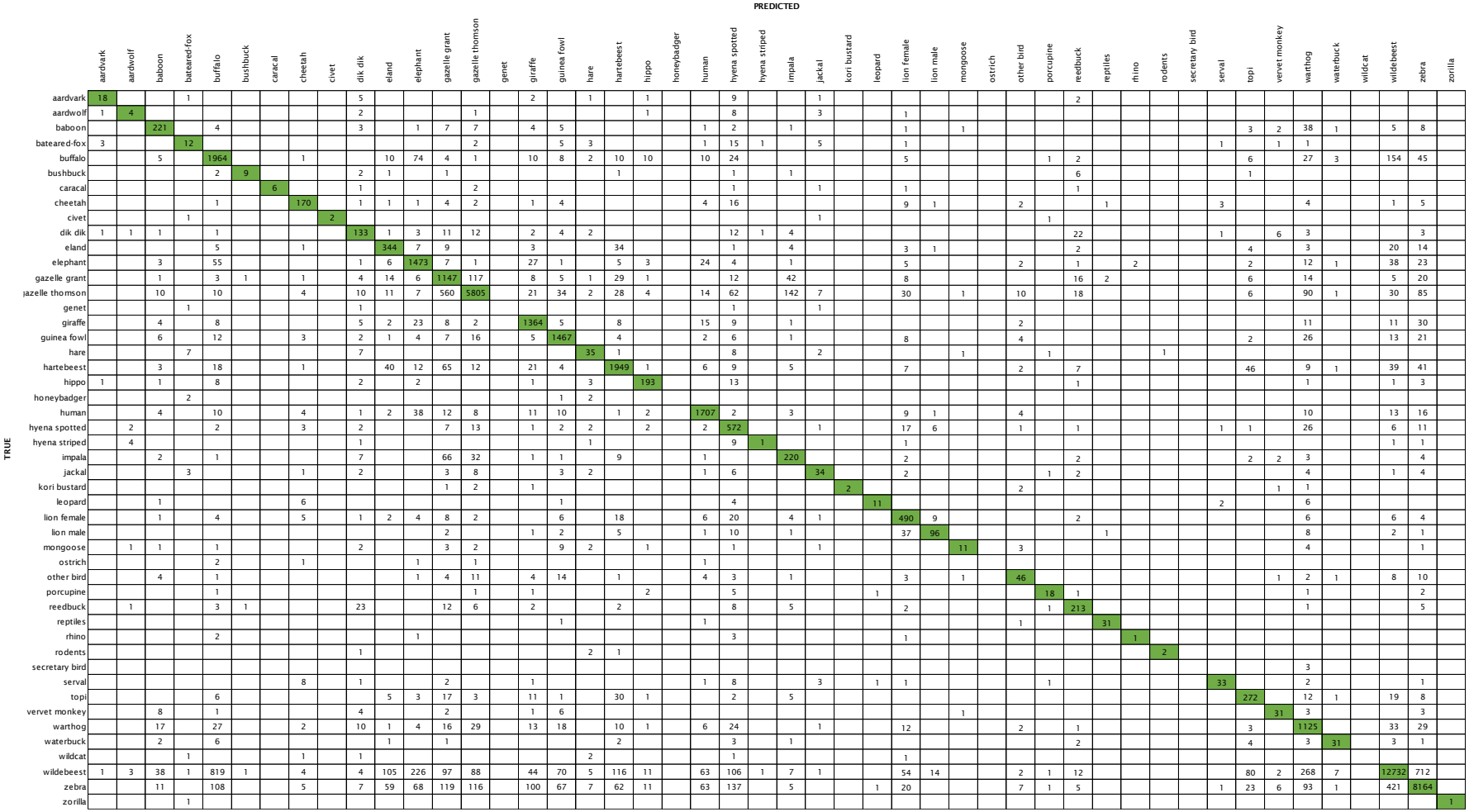


Figure 165: Confusion matrix for Experiment 1A - GoogLeNet

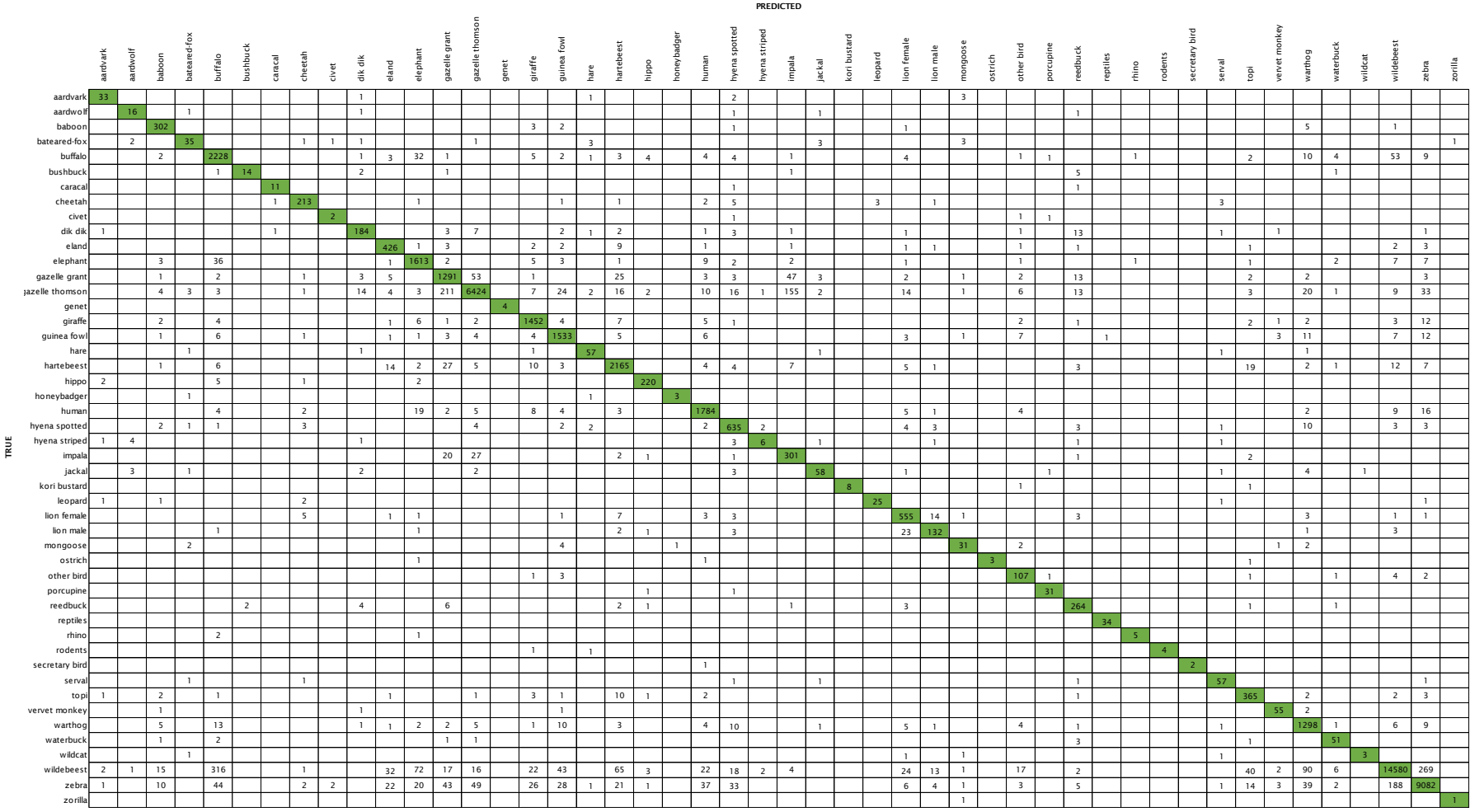


Figure 166: Confusion matrix for Experiment 1A - ResNet-101

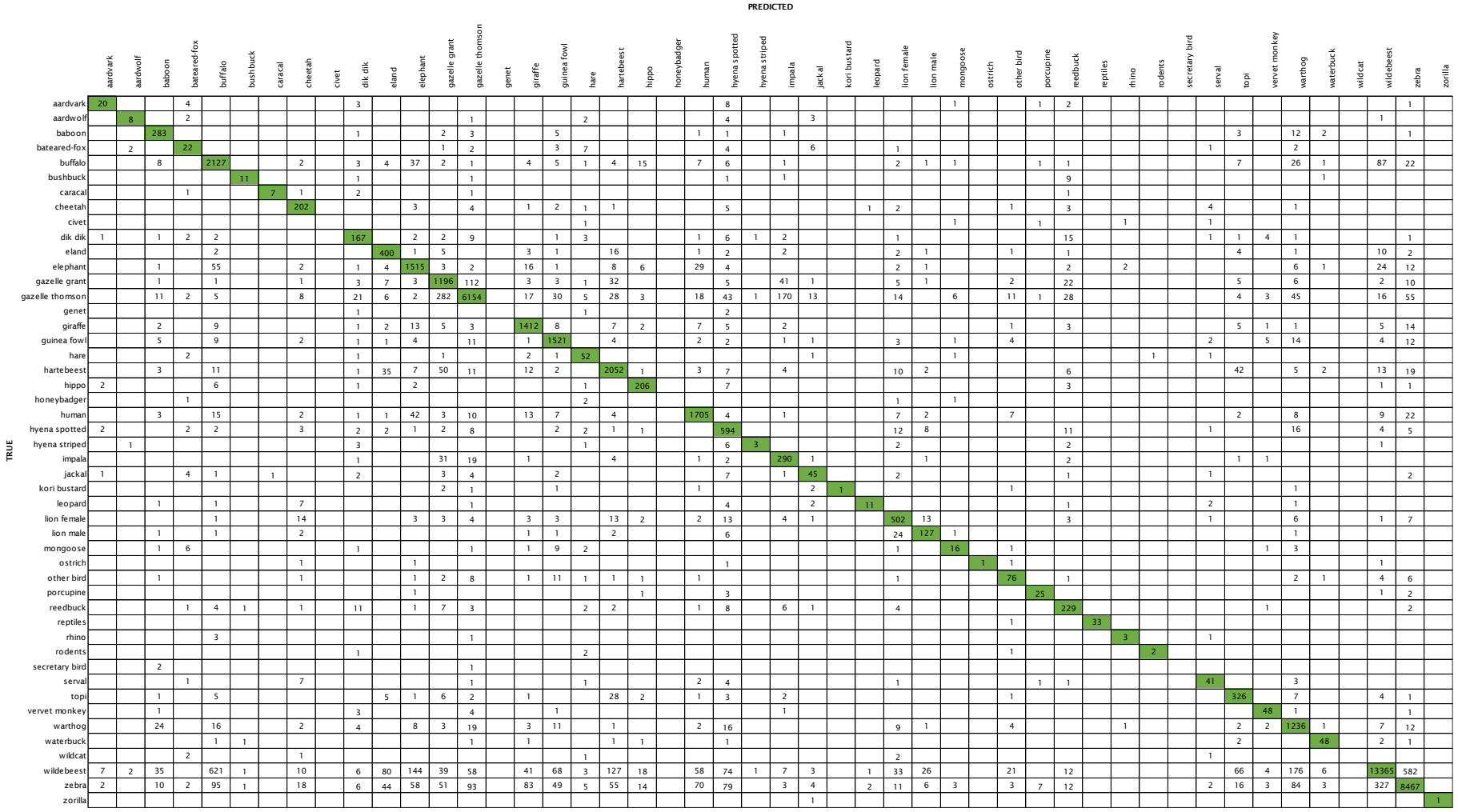


Figure 167: Confusion matrix for Experiment 1B - AlexNet

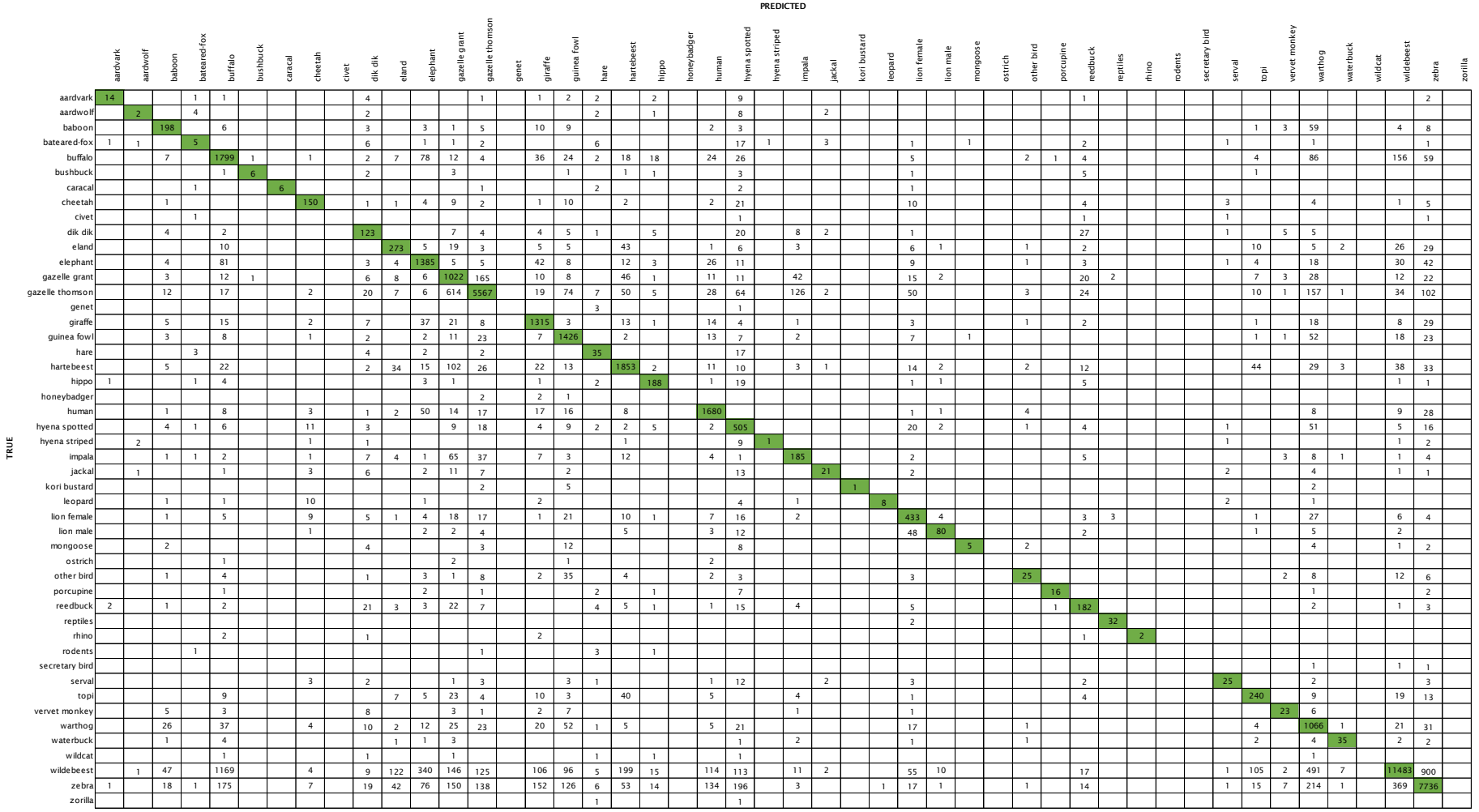


Figure 168: Confusion matrix for Experiment 1B - GoogLeNet

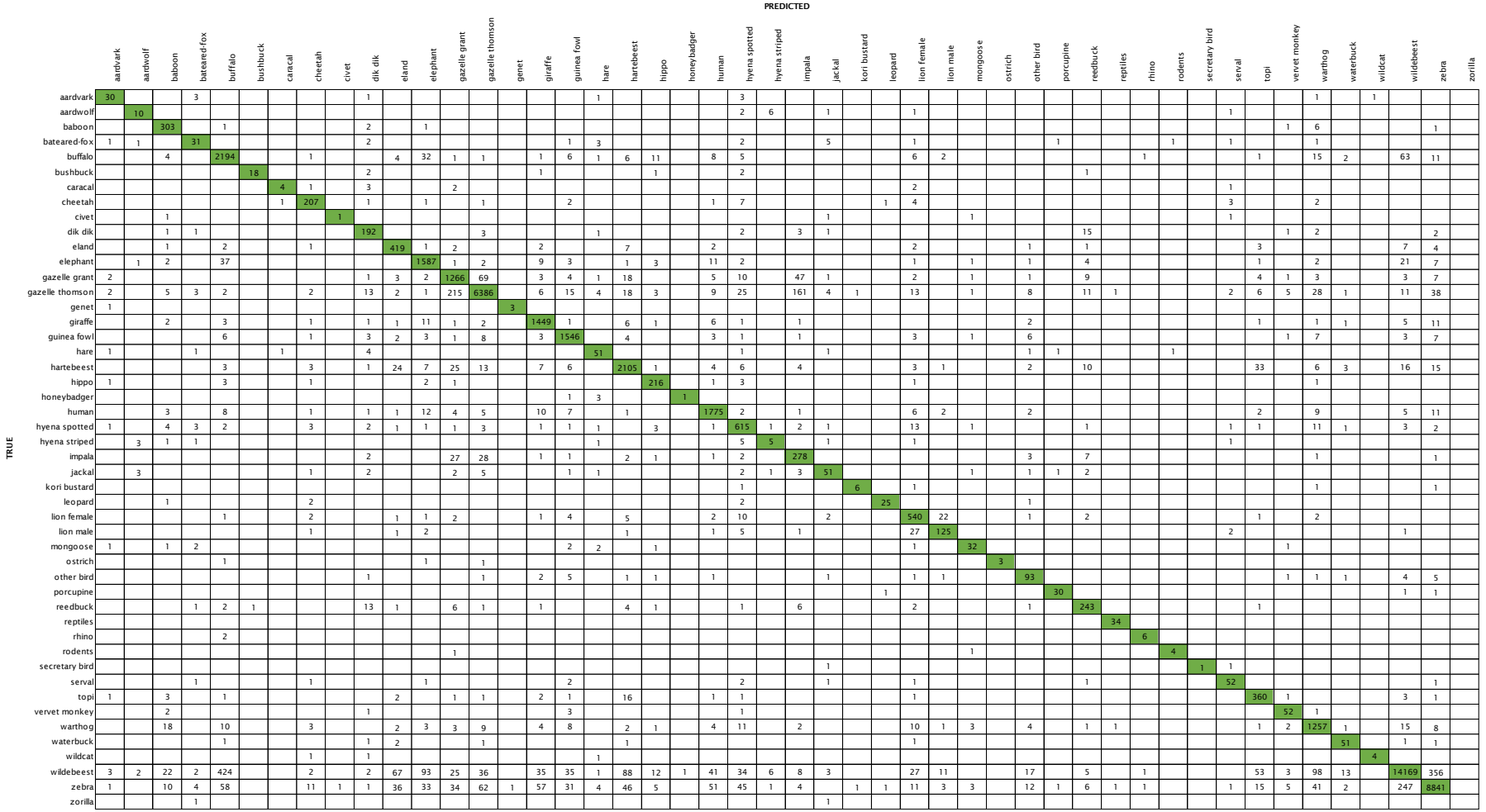


Figure 169: Confusion matrix for Experiment 1B - ResNet-101

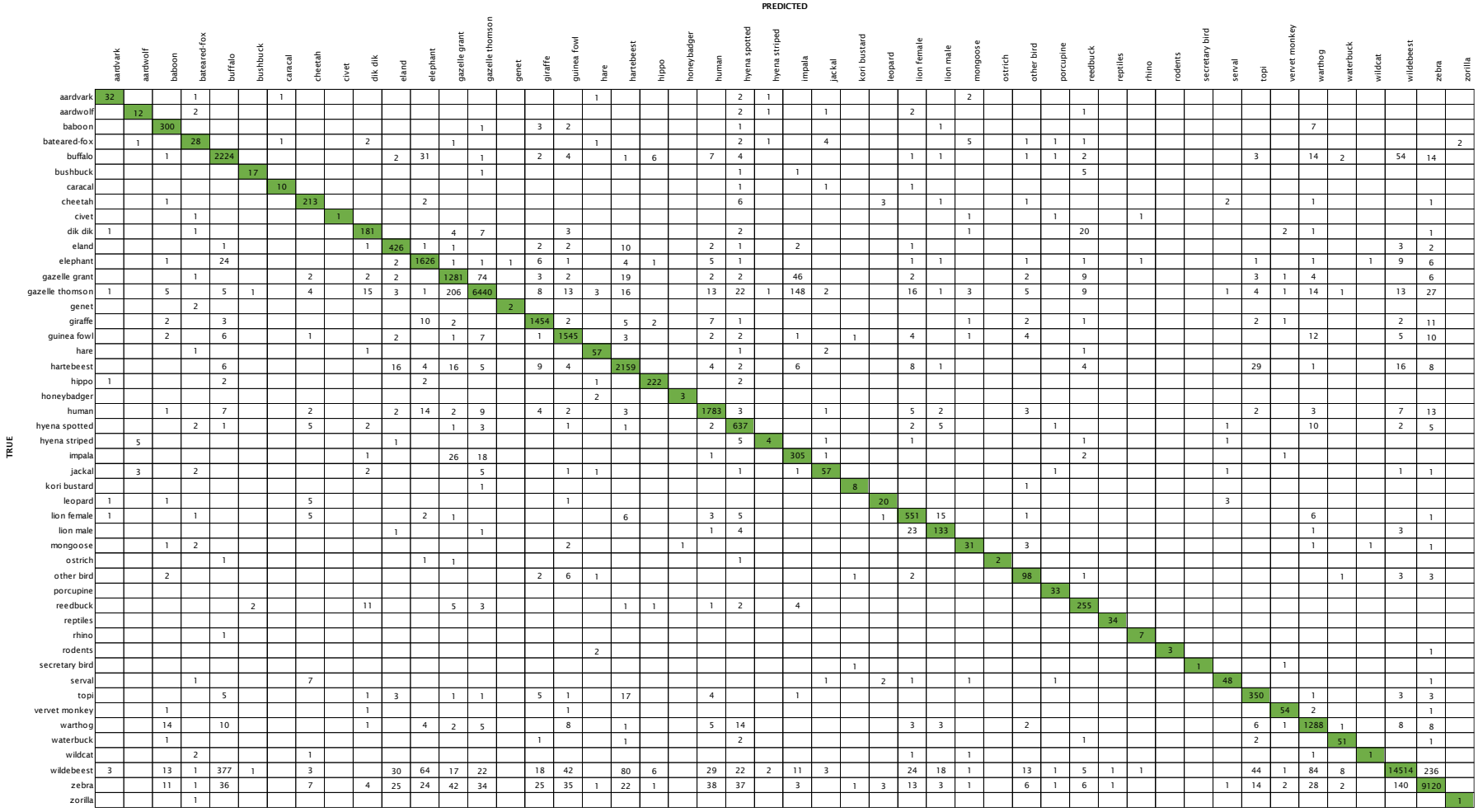


Figure 170: Confusion matrix for Experiment 1B - VGG-19

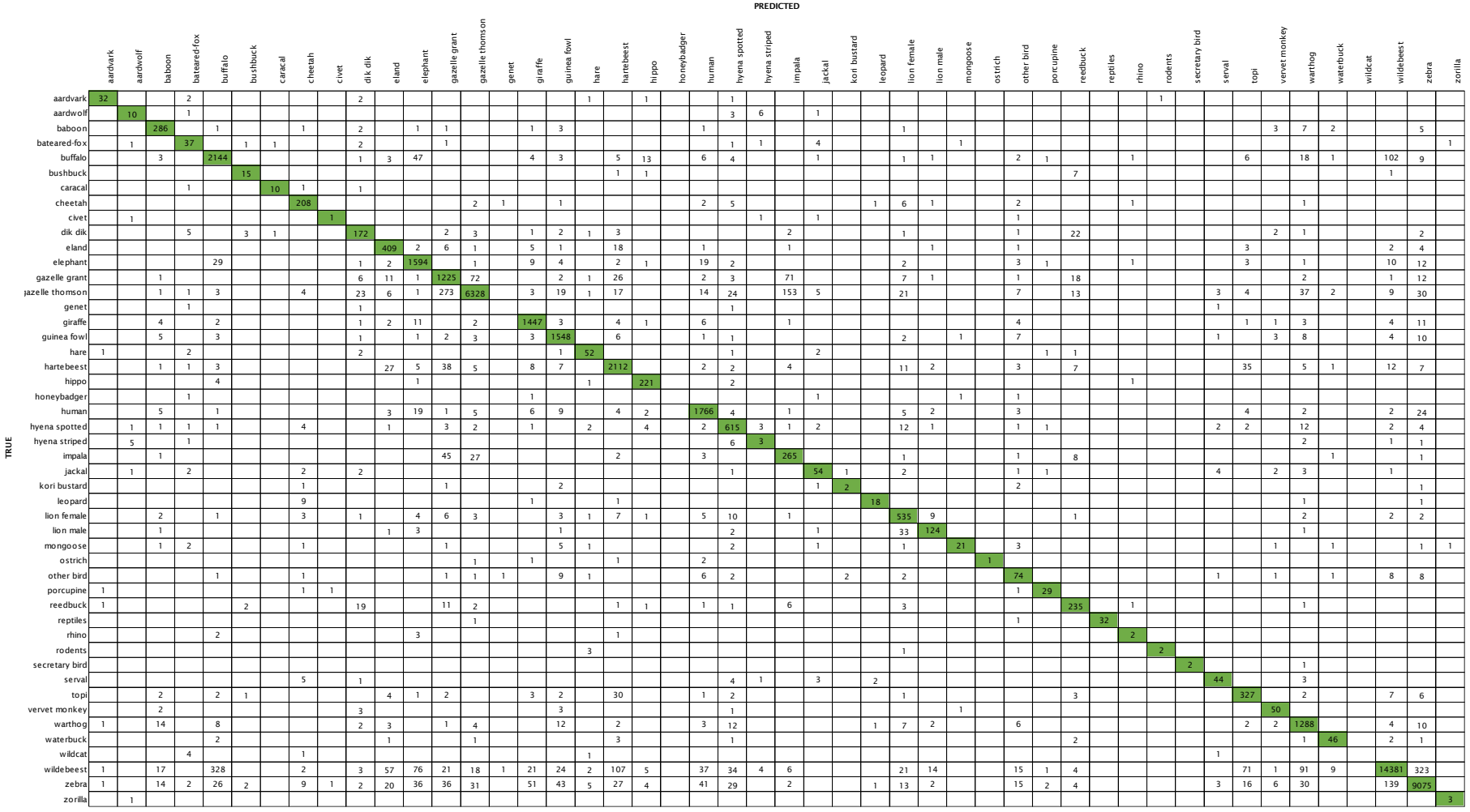


Figure 171: Confusion matrix for Experiment 2A - GoogLeNet

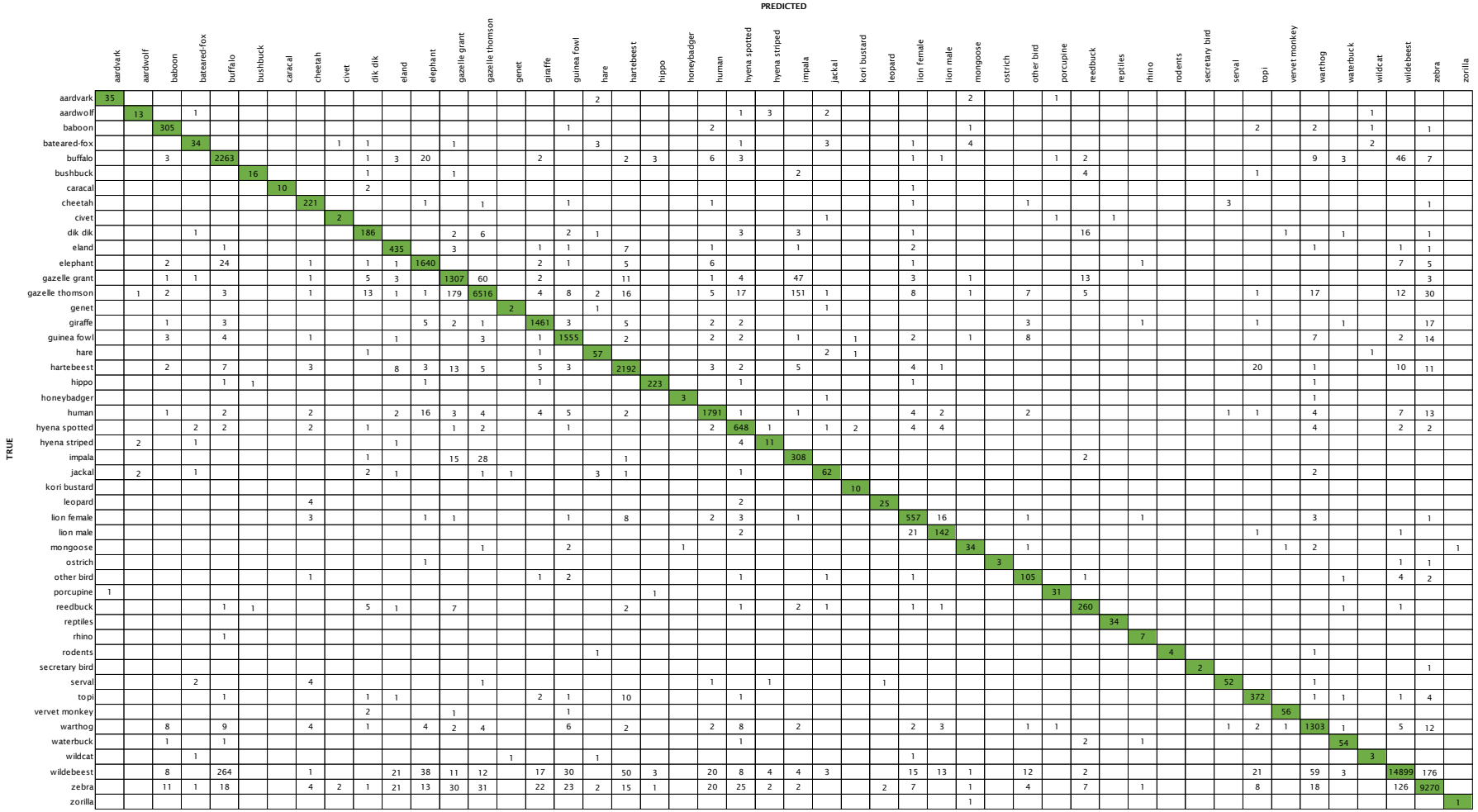


Figure 172: Confusion matrix for Experiment 2A - ResNet-101

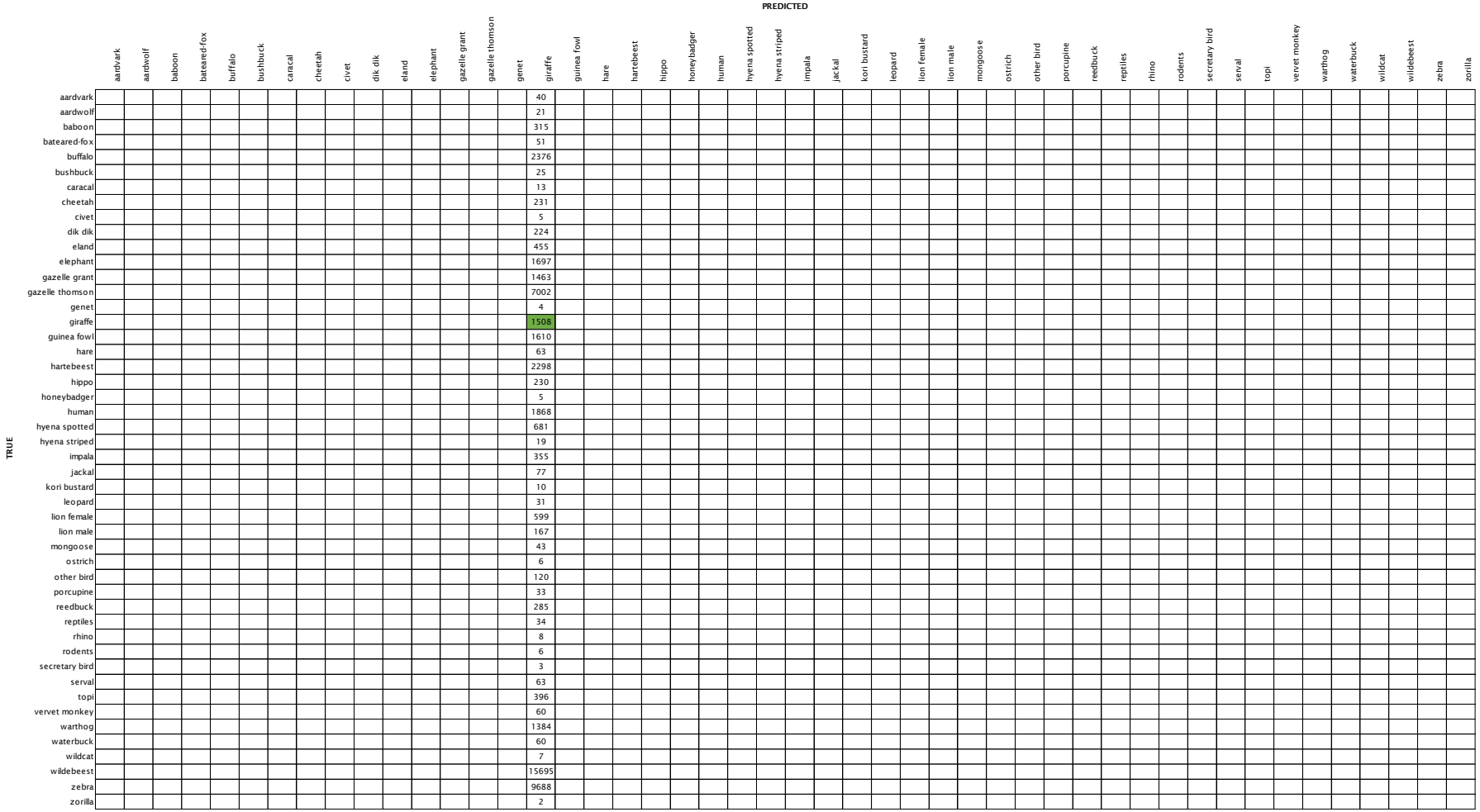


Figure 173: Confusion matrix for Experiment 2A - VGG-19

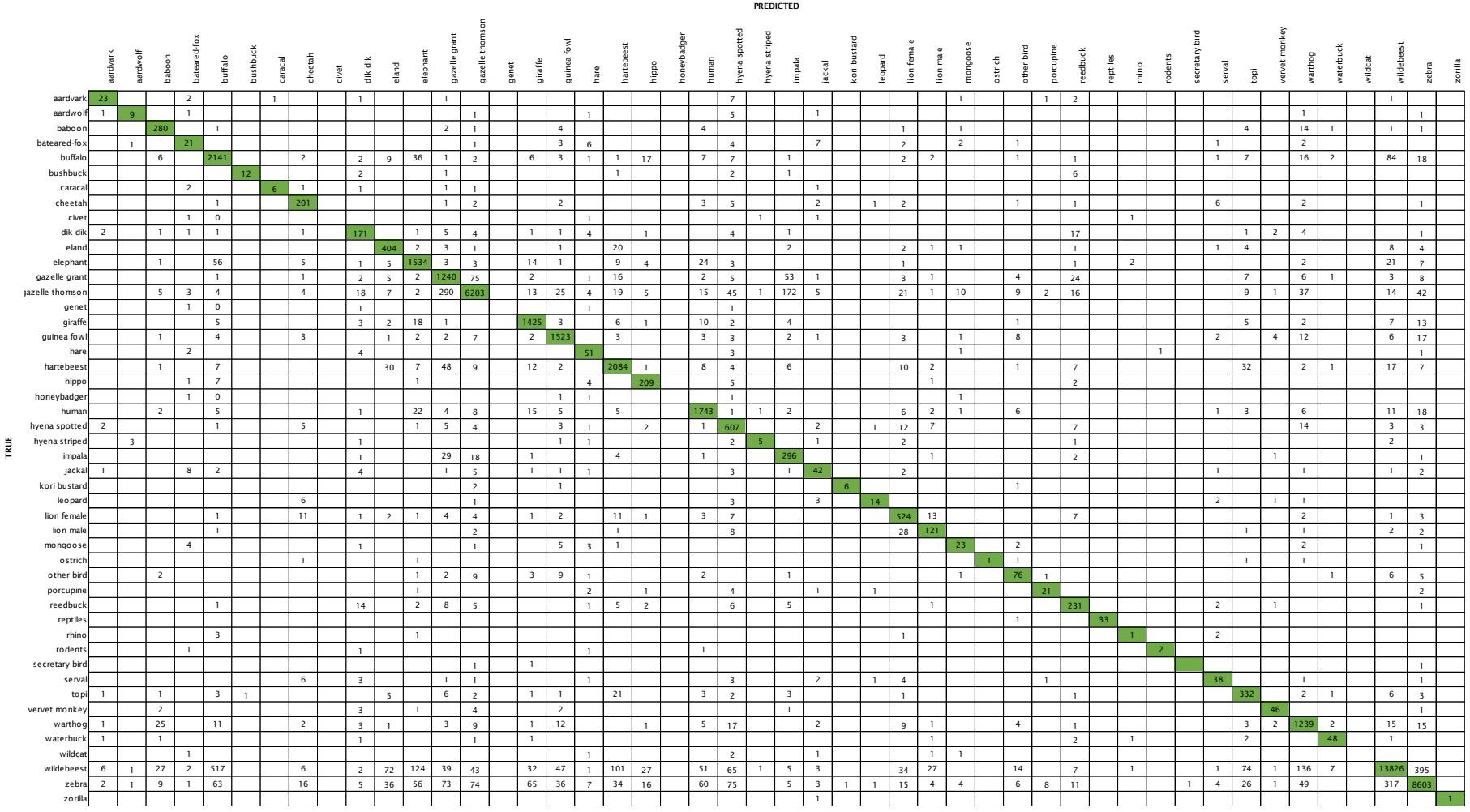


Figure 174: Confusion matrix for Experiment 2B - AlexNet

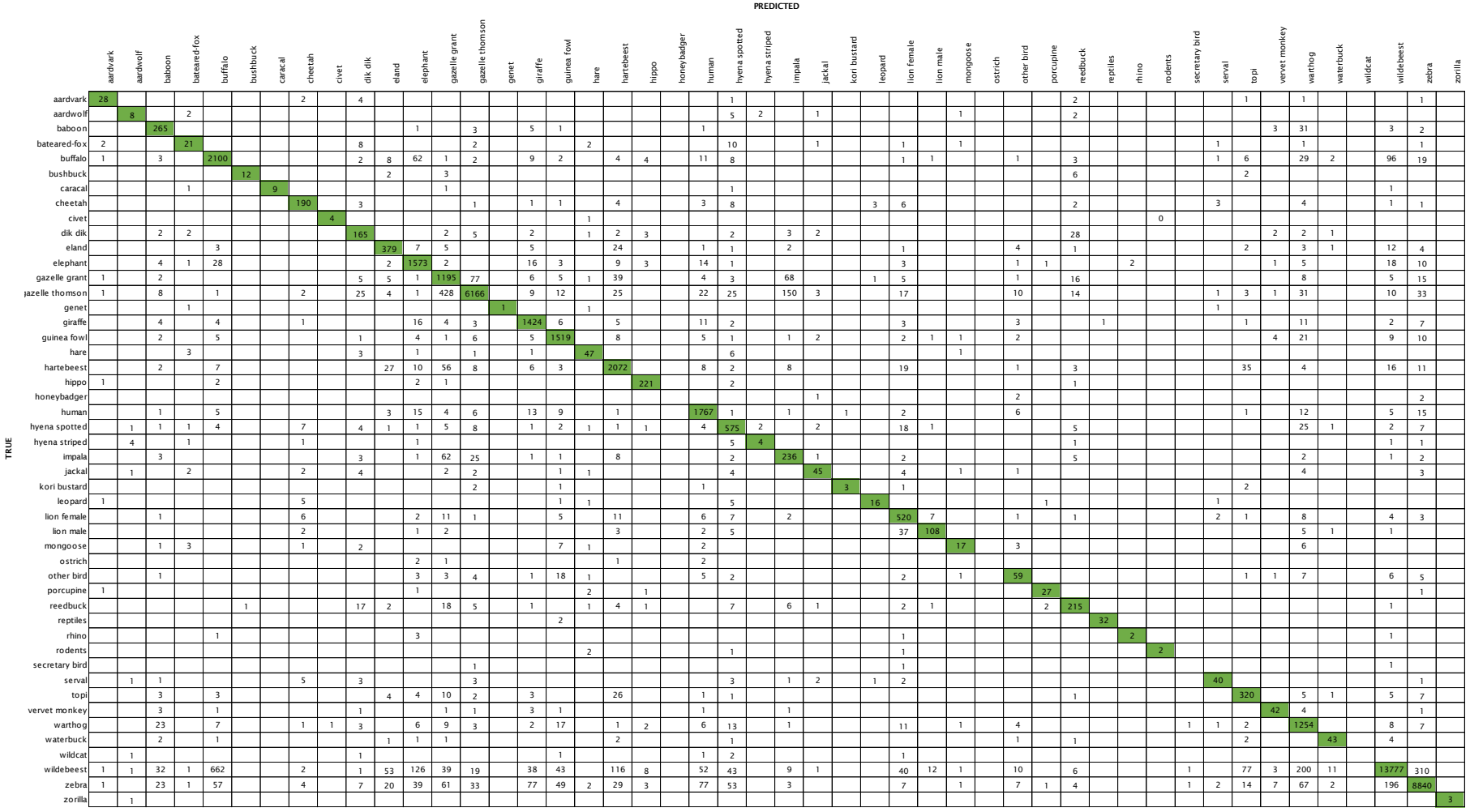


Figure 175: Confusion matrix for Experiment 2B - GoogLeNet

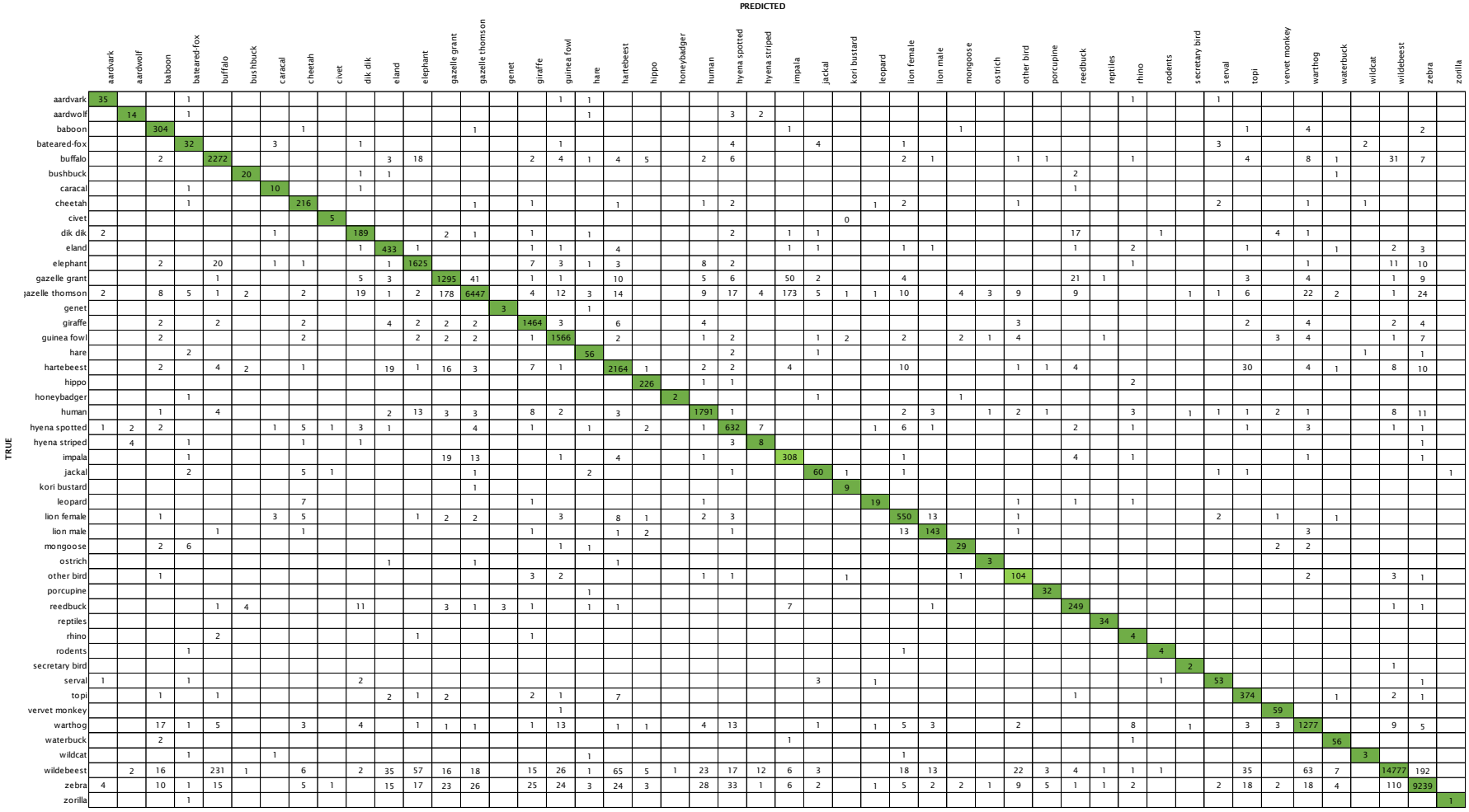


Figure 176: Confusion matrix for Experiment 2B - ResNet-101

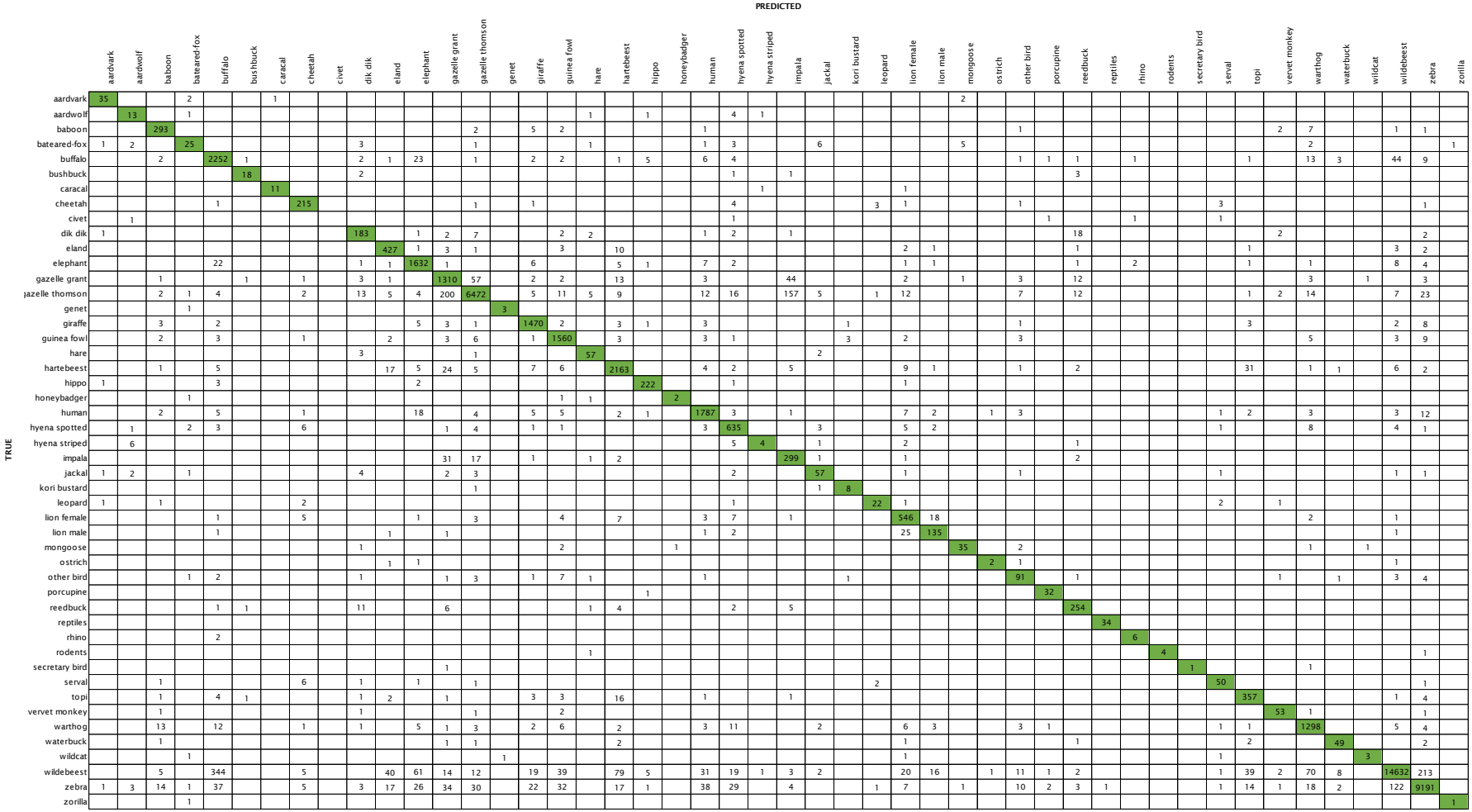


Figure 177: Confusion matrix for Experiment 2B - VGG-19

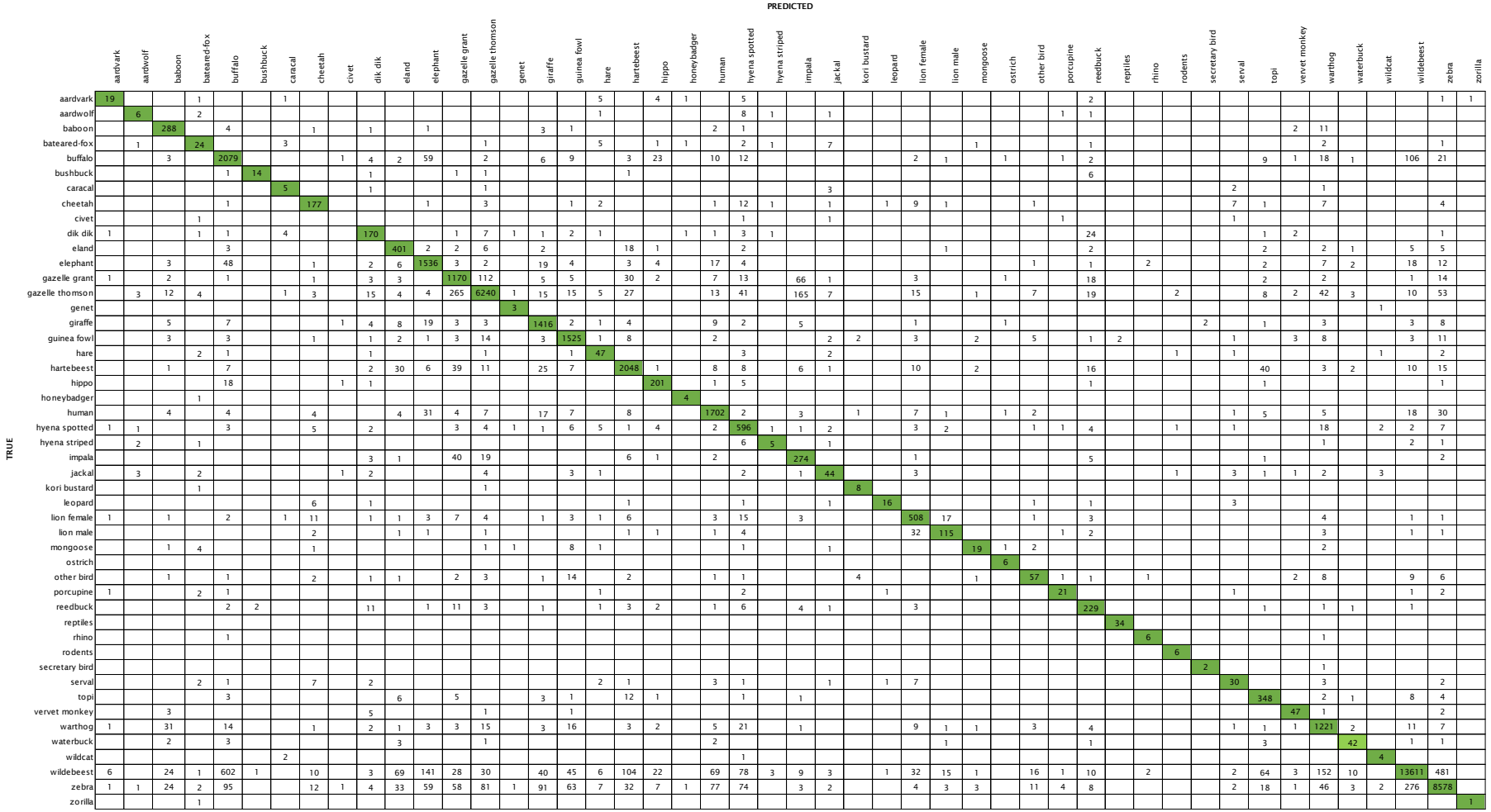


Figure 178: Confusion matrix for Experiment 3 - AlexNet

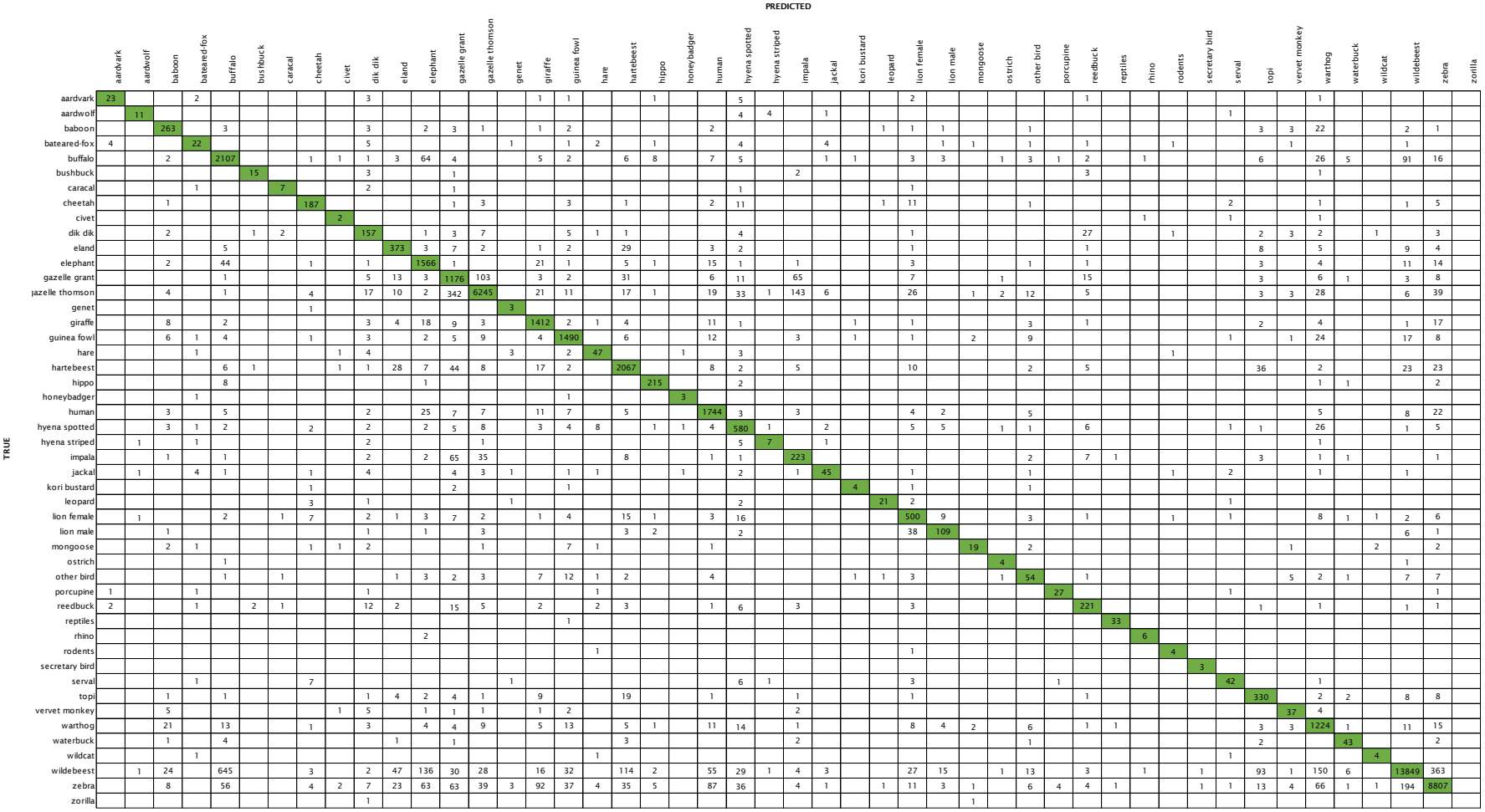


Figure 179: Confusion matrix for Experiment 3 - GoogLeNet

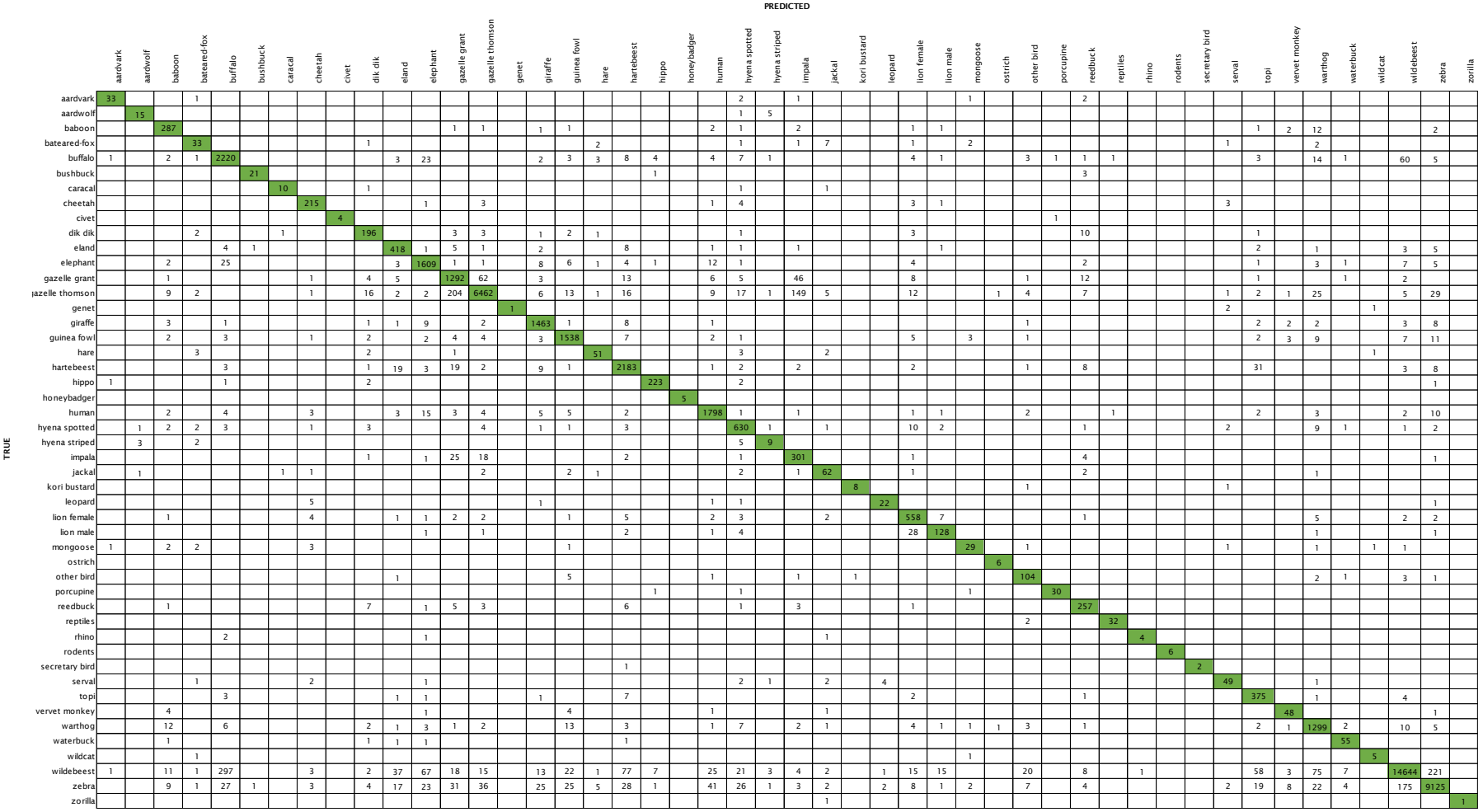


Figure 180: Confusion matrix for Experiment 3 - ResNet-101

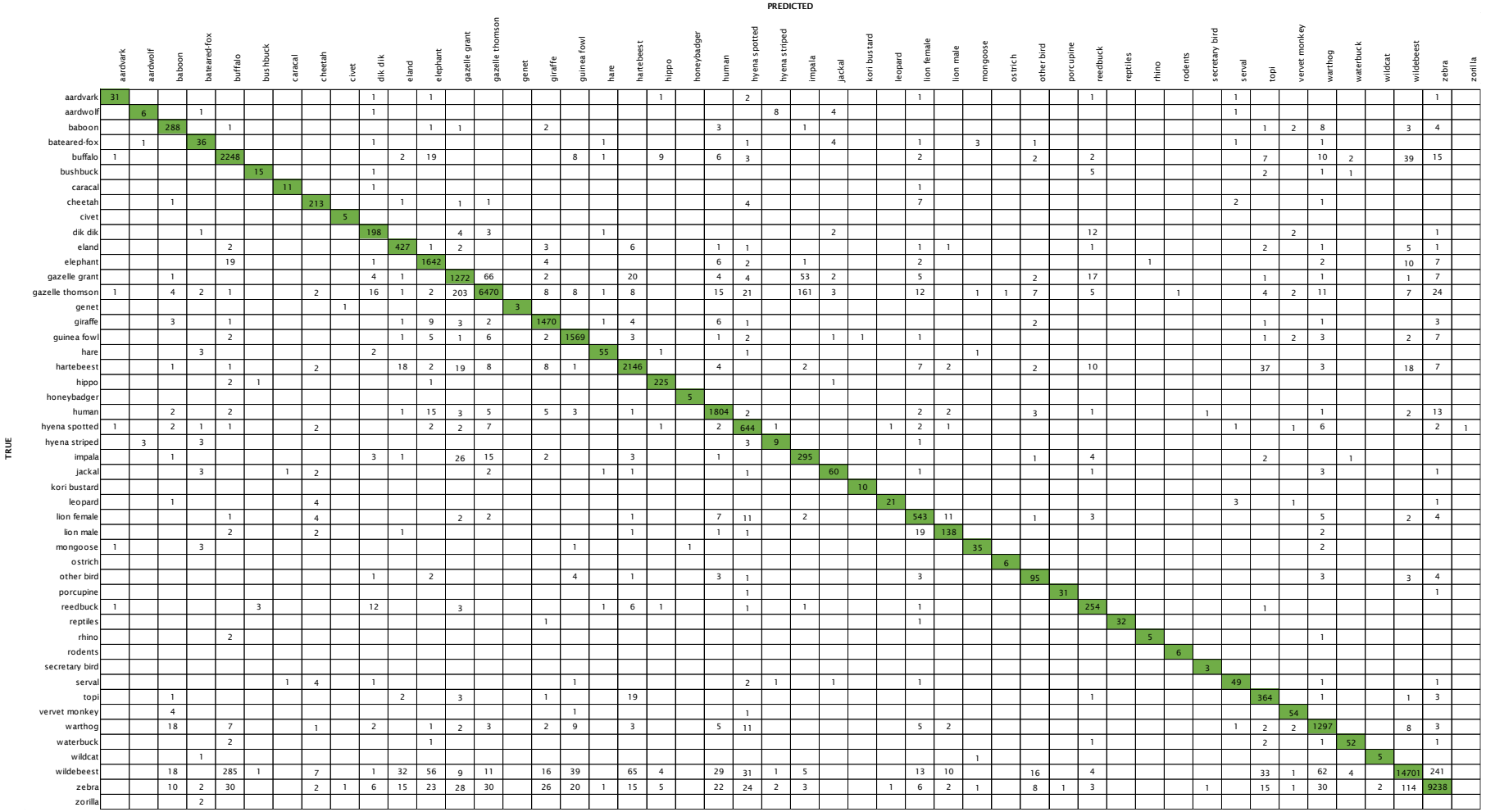


Figure 181: Confusion matrix for Experiment 3 - VGG-19

Appendix C

This section displays the precision and recall values by animal class for all experiments.

Table 32: Precision values per animal class for all experiments[1]

Class	Experiment 1A				Experiment 1B				Experiment 2A				Experiment 2B				Experiment 3			
	AlexNet	GoogLeNet	ResNet-101	VGG-19	AlexNet	GoogLeNet	ResNet-101	VGG-19	AlexNet	GoogLeNet	ResNet-101	VGG-19	AlexNet	GoogLeNet	ResNet-101	VGG-19	AlexNet	GoogLeNet	ResNet-101	VGG-19
aardvark	65%	45%	83%	n/a	50%	35%	75%	80%	n/a	80%	88%	0%	58%	70%	88%	88%	48%	58%	83%	78%
aardwolf	29%	19%	76%	n/a	38%	10%	48%	57%	n/a	48%	62%	0%	43%	38%	67%	62%	29%	52%	71%	29%
baboon	92%	70%	96%	n/a	90%	63%	96%	95%	n/a	91%	97%	0%	89%	84%	97%	93%	91%	83%	91%	91%
bat-eared fox	53%	24%	69%	n/a	43%	10%	61%	55%	n/a	73%	67%	0%	41%	41%	63%	49%	47%	43%	65%	71%
buffalo	90%	83%	94%	n/a	90%	76%	92%	94%	n/a	90%	95%	0%	90%	88%	96%	95%	88%	89%	93%	95%
bushbuck	48%	36%	56%	n/a	44%	24%	72%	68%	n/a	60%	64%	0%	48%	48%	80%	72%	56%	60%	84%	60%
caracal	54%	46%	85%	n/a	54%	46%	31%	77%	n/a	77%	77%	0%	46%	69%	77%	85%	38%	54%	77%	85%
cheetah	84%	74%	92%	n/a	87%	65%	90%	92%	n/a	90%	96%	0%	87%	82%	94%	93%	77%	81%	93%	92%
civet	0%	40%	40%	n/a	0%	0%	20%	20%	n/a	20%	40%	0%	0%	80%	100%	0%	0%	40%	80%	100%
dik-dik	81%	59%	82%	n/a	75%	55%	86%	81%	n/a	77%	83%	0%	76%	74%	84%	82%	76%	70%	88%	88%
eland	88%	76%	94%	n/a	88%	60%	92%	94%	n/a	90%	96%	0%	89%	83%	95%	94%	88%	82%	92%	94%
elephant	90%	87%	95%	n/a	89%	82%	94%	96%	n/a	94%	97%	0%	90%	93%	96%	96%	91%	92%	95%	97%
gazelle Grant	85%	78%	88%	n/a	82%	70%	87%	88%	n/a	84%	89%	0%	85%	82%	89%	90%	80%	80%	88%	87%
gazelle Thomson	89%	83%	92%	n/a	88%	80%	91%	92%	n/a	90%	93%	0%	89%	88%	92%	92%	89%	89%	92%	92%
genet	0%	0%	100%	n/a	0%	0%	75%	50%	n/a	0%	50%	0%	0%	25%	75%	75%	75%	75%	25%	75%
giraffe	95%	90%	96%	n/a	94%	87%	96%	96%	n/a	96%	97%	100%	94%	94%	97%	97%	94%	94%	97%	97%
guinea fowl	95%	91%	95%	n/a	94%	89%	96%	96%	n/a	96%	97%	0%	95%	94%	97%	97%	95%	93%	96%	97%
hare	87%	56%	90%	n/a	83%	56%	81%	90%	n/a	83%	90%	0%	81%	75%	89%	90%	75%	75%	81%	87%
hartebeest	91%	85%	94%	n/a	89%	81%	92%	94%	n/a	92%	95%	0%	91%	90%	94%	94%	89%	90%	95%	93%

hippo	91%	84%	96%	n/a	90%	82%	94%	97%	n/a	96%	97%	0%	91%	96%	98%	97%	87%	93%	97%	98%
honey badger	0%	0%	60%	n/a	0%	0%	20%	60%	n/a	0%	60%	0%	0%	0%	40%	40%	80%	60%	100%	100%
human	93%	91%	96%	n/a	91%	90%	95%	95%	n/a	95%	96%	0%	93%	95%	96%	96%	91%	93%	96%	97%
hyena spotted	89%	84%	93%	n/a	87%	74%	90%	94%	n/a	90%	95%	0%	89%	84%	93%	93%	88%	85%	93%	95%
hyena striped	11%	5%	32%	n/a	16%	5%	26%	21%	n/a	16%	58%	0%	26%	21%	42%	21%	26%	37%	47%	47%
impala	86%	62%	85%	n/a	82%	52%	78%	86%	n/a	75%	87%	0%	83%	66%	87%	84%	77%	63%	85%	83%
jackal	68%	44%	75%	n/a	58%	27%	66%	74%	n/a	70%	81%	0%	55%	58%	78%	74%	57%	58%	81%	78%
kori bustard	40%	20%	80%	n/a	10%	10%	60%	80%	n/a	20%	100%	0%	60%	30%	90%	80%	80%	40%	80%	100%
leopard	52%	35%	81%	n/a	35%	26%	81%	65%	n/a	58%	81%	0%	45%	52%	61%	71%	52%	68%	71%	68%
lion female	87%	82%	93%	n/a	84%	72%	90%	92%	n/a	89%	93%	0%	87%	87%	92%	91%	85%	83%	93%	91%
lion male	76%	57%	79%	n/a	76%	48%	75%	80%	n/a	74%	85%	0%	72%	65%	86%	81%	69%	65%	77%	83%
mongoose	63%	26%	72%	n/a	37%	12%	74%	72%	n/a	49%	79%	0%	53%	40%	67%	81%	44%	44%	67%	81%
ostrich	17%	0%	50%	n/a	17%	0%	50%	33%	n/a	17%	50%	0%	17%	0%	50%	33%	100%	67%	100%	100%
other bird	71%	38%	89%	n/a	63%	21%	78%	82%	n/a	62%	88%	0%	63%	49%	87%	76%	48%	45%	87%	79%
porcupine	85%	55%	94%	n/a	76%	48%	91%	100%	n/a	88%	94%	0%	64%	82%	97%	97%	64%	82%	91%	94%
reedbuck	85%	75%	93%	n/a	80%	64%	85%	89%	n/a	82%	91%	0%	81%	75%	87%	89%	80%	78%	90%	89%
reptiles	97%	91%	100%	n/a	97%	94%	100%	100%	n/a	94%	100%	0%	97%	94%	100%	100%	100%	97%	94%	94%
rhino	25%	13%	63%	n/a	38%	25%	75%	88%	n/a	25%	88%	0%	13%	25%	50%	75%	75%	75%	50%	63%
rodents	33%	33%	67%	n/a	33%	0%	67%	50%	n/a	33%	67%	0%	33%	33%	67%	67%	100%	67%	100%	100%
secretary bird	0%	0%	67%	n/a	0%	0%	33%	33%	n/a	67%	67%	0%	0%	0%	67%	33%	67%	100%	67%	100%
serval	67%	52%	90%	n/a	65%	40%	83%	76%	n/a	70%	83%	0%	60%	63%	84%	79%	48%	67%	78%	78%
topi	89%	69%	92%	n/a	82%	61%	91%	88%	n/a	83%	94%	0%	84%	81%	94%	90%	88%	83%	95%	92%
vervet monkey	88%	52%	92%	n/a	80%	38%	87%	90%	n/a	83%	93%	0%	77%	70%	98%	88%	78%	62%	80%	90%
warthog	90%	81%	94%	n/a	89%	77%	91%	93%	n/a	93%	94%	0%	90%	91%	92%	94%	88%	88%	94%	94%
waterbuck	75%	52%	85%	n/a	80%	58%	85%	85%	n/a	77%	90%	0%	80%	72%	93%	82%	70%	72%	92%	87%
wildcat	0%	0%	43%	n/a	0%	0%	57%	14%	n/a	0%	43%	0%	0%	0%	43%	43%	57%	57%	71%	71%
wildebeest	86%	81%	93%	n/a	85%	73%	90%	92%	n/a	92%	95%	0%	88%	88%	94%	93%	87%	88%	93%	94%
zebra	89%	84%	94%	n/a	87%	80%	91%	94%	n/a	94%	96%	0%	89%	91%	95%	95%	89%	91%	94%	95%
zorilla	50%	50%	50%	n/a	50%	0%	0%	50%	n/a	75%	50%	0%	50%	75%	50%	50%	50%	0%	50%	0%

[1]The numbers denote the precision values of classified images using a scale of 0% to 100%, denoted by shades of red to green, respectively.

Table 33: Recall values per animal class for all experiments[2]

	Experiment 1A				Experiment 1B				Experiment 2A				Experiment 2B				Experiment 3			
Class	AlexNet	GoogLeNet	ResNet-101	VGG-19	AlexNet	GoogLeNet	ResNet-101	VGG-19	AlexNet	GoogLeNet	ResNet-101	VGG-19	AlexNet	GoogLeNet	ResNet-101	VGG-19	AlexNet	GoogLeNet	ResNet-101	VGG-19
aardvark	54%	72%	79%	n/a	57%	74%	67%	80%	n/a	84%	97%	0%	58%	74%	78%	85%	59%	77%	89%	86%
aardwolf	30%	25%	62%	n/a	62%	29%	50%	57%	n/a	50%	72%	0%	60%	44%	64%	46%	35%	73%	75%	60%
baboon	76%	64%	86%	n/a	72%	56%	79%	84%	n/a	79%	88%	0%	77%	68%	81%	85%	71%	73%	82%	81%
bat-eared fox	61%	40%	74%	n/a	41%	25%	57%	56%	n/a	58%	76%	0%	40%	53%	53%	66%	46%	56%	63%	62%
buffalo	75%	64%	83%	n/a	71%	53%	79%	82%	n/a	84%	87%	0%	75%	73%	89%	83%	72%	72%	85%	86%
bushbuck	75%	75%	88%	n/a	73%	75%	95%	81%	n/a	63%	89%	0%	92%	92%	69%	82%	82%	79%	91%	75%
caracal	100%	100%	85%	n/a	88%	100%	67%	83%	n/a	83%	100%	0%	86%	100%	50%	92%	29%	58%	83%	85%
cheetah	80%	77%	91%	n/a	70%	70%	84%	84%	n/a	82%	87%	0%	74%	82%	82%	86%	72%	83%	88%	87%
civet	0%	100%	40%	n/a	0%	0%	50%	100%	n/a	33%	40%	0%	0%	80%	63%	0%	0%	22%	100%	71%
dik-dik	72%	54%	85%	n/a	67%	43%	76%	80%	n/a	69%	83%	0%	69%	63%	79%	78%	70%	61%	80%	79%
eland	72%	57%	83%	n/a	68%	53%	74%	83%	n/a	74%	87%	0%	70%	74%	83%	83%	70%	73%	81%	85%
elephant	83%	75%	91%	n/a	82%	68%	88%	91%	n/a	88%	94%	0%	84%	83%	93%	91%	82%	82%	91%	92%
gazelle Grant	73%	52%	79%	n/a	70%	44%	78%	80%	n/a	73%	83%	0%	70%	62%	83%	80%	71%	65%	80%	80%
gazelle Thomson	95%	92%	97%	n/a	94%	89%	96%	97%	n/a	97%	98%	0%	95%	97%	98%	97%	95%	96%	97%	98%
genet	0%	0%	100%	n/a	0%	0%	75%	67%	n/a	0%	50%	0%	0%	100%	50%	75%	38%	23%	100%	100%
giraffe	89%	82%	94%	n/a	87%	73%	91%	94%	n/a	92%	96%	3%	89%	87%	95%	95%	86%	86%	95%	95%
guinea fowl	89%	84%	92%	n/a	87%	72%	92%	92%	n/a	91%	94%	0%	90%	89%	94%	92%	88%	90%	94%	94%
hare	58%	47%	81%	n/a	54%	40%	67%	81%	n/a	71%	78%	0%	53%	72%	74%	80%	51%	66%	77%	87%
hartebeest	90%	84%	92%	n/a	86%	78%	90%	92%	n/a	89%	94%	0%	89%	87%	93%	93%	88%	87%	92%	93%
hippo	77%	79%	94%	n/a	75%	71%	83%	93%	n/a	87%	97%	0%	73%	89%	92%	93%	73%	90%	94%	91%
honey badger	0%	0%	75%	n/a	0%	0%	50%	75%	n/a	0%	75%	0%	0%	0%	67%	67%	50%	50%	100%	83%

human	87%	88%	94%	n/a	89%	80%	92%	93%	n/a	92%	96%	0%	90%	88%	95%	94%	88%	87%	94%	94%
hyena spotted	66%	50%	84%	n/a	63%	42%	76%	81%	n/a	79%	87%	0%	67%	71%	84%	84%	64%	73%	83%	83%
hyena striped	67%	25%	55%	n/a	50%	50%	25%	40%	n/a	16%	50%	0%	56%	50%	24%	57%	38%	47%	41%	41%
impala	57%	48%	58%	n/a	54%	46%	53%	58%	n/a	52%	58%	0%	53%	48%	55%	57%	51%	48%	58%	56%
jackal	54%	54%	82%	n/a	53%	60%	67%	77%	n/a	70%	78%	0%	53%	73%	71%	71%	54%	70%	69%	77%
kori bustard	100%	100%	100%	n/a	100%	100%	75%	67%	n/a	40%	71%	0%	86%	75%	64%	62%	53%	50%	89%	91%
leopard	76%	79%	89%	n/a	73%	89%	89%	69%	n/a	78%	89%	0%	74%	76%	76%	76%	80%	84%	76%	91%
lion female	77%	67%	84%	n/a	77%	59%	79%	83%	n/a	78%	87%	0%	76%	73%	87%	83%	78%	74%	83%	85%
lion male	74%	75%	77%	n/a	67%	77%	74%	72%	n/a	78%	78%	0%	65%	82%	79%	75%	73%	72%	81%	82%
mongoose	61%	69%	69%	n/a	50%	71%	70%	63%	n/a	84%	72%	0%	48%	65%	73%	80%	61%	70%	73%	83%
ostrich	100%	0%	100%	n/a	100%	0%	100%	100%	n/a	100%	100%	0%	100%	0%	33%	50%	55%	36%	75%	86%
other bird	60%	51%	67%	n/a	55%	56%	59%	68%	n/a	47%	72%	0%	55%	50%	65%	65%	53%	42%	69%	68%
porcupine	67%	69%	89%	n/a	68%	89%	88%	80%	n/a	78%	89%	0%	62%	84%	74%	84%	66%	82%	94%	97%
reedbuck	68%	67%	79%	n/a	62%	53%	76%	78%	n/a	72%	83%	0%	66%	68%	79%	81%	63%	72%	79%	78%
reptiles	100%	89%	97%	n/a	100%	86%	92%	94%	n/a	100%	97%	0%	100%	97%	89%	97%	94%	92%	94%	100%
rhino	40%	33%	71%	n/a	43%	100%	67%	70%	n/a	29%	58%	0%	17%	50%	14%	60%	55%	67%	80%	83%
rodents	100%	67%	100%	n/a	67%	0%	67%	100%	n/a	67%	100%	0%	67%	100%	57%	100%	55%	44%	100%	86%
secretary bird	0%	0%	100%	n/a	0%	0%	100%	100%	n/a	100%	100%	0%	0%	0%	40%	100%	50%	60%	100%	60%
serval	70%	79%	83%	n/a	68%	63%	78%	83%	n/a	73%	91%	0%	61%	75%	80%	79%	54%	76%	79%	83%
topi	69%	59%	80%	n/a	67%	53%	75%	76%	n/a	69%	87%	0%	65%	68%	78%	79%	68%	64%	75%	77%
vervet monkey	87%	60%	83%	n/a	66%	46%	70%	83%	n/a	69%	95%	0%	77%	66%	78%	83%	72%	60%	71%	79%
warthog	75%	62%	86%	n/a	74%	45%	84%	87%	n/a	85%	91%	0%	80%	72%	90%	90%	77%	76%	87%	89%
waterbuck	71%	65%	72%	n/a	73%	69%	67%	77%	n/a	72%	82%	0%	75%	68%	75%	77%	62%	68%	75%	87%
wildcat	0%	0%	75%	n/a	0%	0%	80%	33%	n/a	0%	38%	0%	0%	0%	43%	60%	31%	44%	63%	71%
wildebeest	96%	94%	98%	n/a	96%	94%	97%	98%	n/a	98%	99%	0%	96%	97%	99%	99%	97%	97%	98%	99%
zebra	92%	88%	96%	n/a	91%	85%	95%	96%	n/a	95%	97%	0%	94%	95%	97%	97%	92%	94%	97%	96%
zorilla	100%	100%	50%	n/a	100%	0%	0%	33%	n/a	60%	50%	0%	100%	100%	50%	50%	50%	0%	100%	0%

[2]The numbers denote the precision values of classified images using a scale of 0% to 100%, denoted by shades of red to green, respectively.