

UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

**DRIVING COMPUTER VISION INNOVATION
THROUGH THE CONSTRAINTS OF
CAMERA-TRAP DATA**

FRANCISCO FAGNER DO REGO CUNHA

DRIVING COMPUTER VISION INNOVATION
THROUGH THE CONSTRAINTS OF
CAMERA-TRAP DATA

Thesis presented to the Graduate Program
in Informática of the Universidade Federal
do Amazonas in partial fulfillment of the re-
quirements for the degree of Doctor in In-
formática.

ADVISOR: EULANDA MIRANDA DOS SANTOS

CO-ADVISOR: JUAN GABRIEL COLONNA

Manaus - AM

August 2026

Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

C972d Cunha, Francisco Fagner do Rego
 Driving Computer Vision Innovation Through the Constraints of
 Camera-Trap Data / Francisco Fagner do Rego Cunha. - 2026.
 138 f. : il., color. ; 31 cm.

 Orientador(a): Eulanda Miranda dos Santos.
 Coorientador(a): Juan Gabriel Colonna.
 Tese (doutorado) - Universidade Federal do Amazonas,
 Programa de Pós-Graduação em Informática, Manaus, 2026.

 1. Camera Trap Image Analysis. 2. Wildlife Monitoring. 3. Deep
 Learning. 4. Long-Tailed Learning. 5. Multi-Object Tracking. I.
 Santos, Eulanda Miranda dos. II. Colonna, Juan Gabriel. III.
 Universidade Federal do Amazonas. Programa de Pós-Graduação
 em Informática. IV. Título



Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"DRIVING COMPUTER VISION INNOVATION THROUGH THE CONSTRAINTS OF CAMERA-TRAP DATA"

FRANCISCO FAGNER DO REGO CUNHA

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos professores:

Prof.^a Dr.^a Eulanda Miranda dos Santos - **Presidente**

Prof. Dr. Eduardo James Pereira Souto - **Membro Interno**

Prof. Dr. José Reginaldo Hughes Carvalho - **Membro Interno**

Prof. Dr. Marco Antônio Pinheiro de Cristo - **Membro Externo**

Prof. Dr. Luiz Eduardo Soares de Oliveira - **Membro Externo**

Manaus, 17 de Agosto de 2026.



Documento assinado eletronicamente por **Eulanda Miranda dos Santos, Professor do Magistério Superior**, em 21/08/2026, às 14:43, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Eduardo James Pereira Souto, Professor do Magistério Superior**, em 22/08/2026, às 09:47, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Marco Antônio Pinheiro de Cristo, Professor do Magistério Superior**, em 24/08/2026, às 20:36, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **José Reginaldo Hughes Carvalho, Professor do Magistério Superior**, em 25/08/2026, às 09:13, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Luiz Eduardo Soares de Oliveira, Usuário Externo**, em 25/08/2026, às 11:56, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3255807** e o código CRC **AABD3113**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroadó I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Referência: Processo nº 23105.027937/2026-05

SEI nº 3255807

To my parents, Zuleide and B   .

Acknowledgments

First of all, I would like to thank God for guiding my path in life and allowing me to see Him through other people.

I am deeply grateful to my family for their support throughout my academic journey: my mother Zuleide, my father Béb , and my brother Francisco.

I would like to thank my advisor, Professor Eulanda Miranda, for her continuous support, guidance, and encouragement. I am especially grateful for her belief in my research, her patience, and her valuable advice during important moments in my life.

I also thank my co-advisor, Professor Juan Colonna, for his responsiveness and support whenever I needed it. Additionally, I am grateful to Professor David Rolnick, my supervisor during my internship at Mila/eButterfly, for helping to expand my research connections.

Finally, I would like to thank my friends who supported me throughout my doctoral journey, especially my fellow PPGI students, Leandro Okimoto and Ludimila. I also wish to thank my friends from the Focolare Movement for their unity: Jos , Matheus, Vitor, Guilherme, and Roberto.

This work was carried out with the support of the Government of the State of Amazonas, through the Amazonas State Research Support Foundation (FAPEAM), by means of a scholarship grant. This study was financed in part by the Coordena  o de Aperfei oamento de Pessoal de N vel Superior - Brasil (CAPES) - Finance Code 001.

*“‘The Answer to the Great Question... of Life, the Universe and Everything... is...
Forty-two’, said Deep Thought, with infinite majesty and calm.”*
(Douglas Adams, *The Hitchhiker’s Guide to the Galaxy*)

Resumo

Armadilhas fotográficas são uma ferramenta não invasiva e de baixo custo para o monitoramento da fauna e a pesquisa ecológica. No entanto, elas produzem um grande volume de dados, cujo processamento manual é caro e demorado. Nos últimos anos, o aprendizado profundo tem possibilitado a extração automatizada de informações desses sensores, acelerando o processamento de dados e o monitoramento da biodiversidade em larga escala. Apesar disso, diversos desafios continuam sendo insuficientemente abordados, incluindo o extremo desbalanceamento entre classes, a disponibilidade limitada de dados rotulados, os recursos computacionais restritos e a dificuldade de contar indivíduos em sequências de imagens. Consequentemente, o objetivo principal deste trabalho é investigar como as restrições inerentes aos dados de armadilhas fotográficas podem impulsionar novos métodos de visão computacional em condições reais, com foco em três tarefas principais: filtragem de imagens vazias, classificação de espécies em cenários de cauda longa e contagem de animais em sequências de imagens. Para a filtragem de imagens vazias, analisamos a relação de compromisso entre modelos de classificação e de detecção sob restrições de implantação em dispositivos embarcados, mostrando que abordagens de classificação oferecem uma alternativa prática de baixa latência quando os recursos computacionais e as anotações de detecção são limitados. Para lidar com o desbalanceamento extremo na tarefa de reconhecimento de espécies, na qual espécies raras e ecologicamente importantes são frequentemente negligenciadas, é proposto o framework Square-Root Sampling Branch (SSB), que combina cabeças de classificação treinadas sob diferentes distribuições de amostragem para melhorar o desempenho nas classes da cauda, enquanto minimiza a degradação da acurácia nas classes majoritárias. Também foi realizada uma avaliação sistemática de estratégias para tratar o problema de cauda longa em quatro bases de dados de armadilhas fotográficas e quatro famílias de arquiteturas de visão computacional. Para a contagem de animais, motivados pelos grandes deslocamentos dos objetos e pela escassez de anotações de rastreamento para sequências de imagens de armadilhas fotográficas obtidas com baixa taxa de frames, introduz-se o PuTR-CouT, um framework de contagem por rastreamento baseado em Transformers, treinado em dados fracamente supervisionados gerados via conhecimento

prévio de armadilhas fotográficas e que permite contagem de múltiplas espécies. O método proposto atinge uma pontuação de 0,0248642 no benchmark iWildCam 2021, superando métodos tradicionais de rastreamento de múltiplos objetos e apresentando uma melhoria de aproximadamente 15,2% em relação à solução vencedora do desafio. Em suma, as contribuições desta tese incluem análises comparativas e novos métodos para filtragem eficiente de imagens vazias, classificação de espécies em cauda longa e contagem de animais por rastreamento, mostrando como as restrições inerentes aos dados de armadilhas fotográficas podem guiar inovações metodológicas em visão computacional para o monitoramento da vida selvagem.

Palavras-chave: análise de imagens de armadilhas fotográficas, monitoramento da vida selvagem, aprendizado profundo, aprendizado de cauda longa, rastreamento de múltiplos objetos

Abstract

Camera traps are a non-invasive and cost-effective tool for wildlife monitoring and ecological research. However, they produce large volumes of image data for which manual processing is expensive and time-consuming. In recent years, deep learning has enabled the automated extraction of ecological information from camera-trap images, accelerating data processing and supporting large-scale biodiversity monitoring. Nevertheless, several challenges remain insufficiently addressed, including extreme class imbalance, limited labeled datasets, constrained computational resources, and the difficulty of counting individuals across image sequences. Accordingly, the main objective of this work is to investigate how the inherent constraints of camera-trap data can drive the development of novel computer vision methods under real-world conditions, focusing on three main tasks: empty-frame filtering, long-tailed species classification, and animal counting across image sequences. For empty-frame filtering, we analyze the trade-off between classification and detection models under edge-deployment constraints, showing that classification approaches can provide a practical low-latency alternative when computational resources and detection annotations are limited. To address the extreme imbalance in species recognition, where rare yet ecologically important species are often neglected, we propose the Square-Root Sampling Branch (SSB), a framework that combines classification heads trained under different sampling distributions to improve tail-class performance while minimizing the degradation of head-class accuracy. We further provide a systematic evaluation of long-tail learning strategies across four camera-trap datasets and four model architectures. For animal counting, motivated by the large object displacements and scarcity of tracking annotations in low-frame-rate camera-trap sequences, we introduce PuTR-CouT, a transformer-based counting-by-tracking framework trained using a weakly labeled dataset generated from camera-trap priors that enables multi-species counting. The proposed method achieves a score of 0.0248642 on the iWildCam 2021 benchmark, outperforming the evaluated traditional multi-object trackers and improving by approximately 15.2% over the winning solution of the iWildCam 2021 challenge. Overall, the thesis contributes comparative analyses and new methods for efficient empty-frame filtering, long-tailed classification, and

counting-by-tracking, showing how constraints inherent to camera-trap data can guide methodological innovation in computer vision for wildlife monitoring.

Keywords: Camera Trap Image Analysis, Wildlife Monitoring, Deep Learning, Long-Tailed Learning, Multi-Object Tracking

List of Figures

2.1	Residual building block.	13
2.2	Comparison between standard convolutional filters and depthwise convolutional filters.	14
2.3	Comparison of EfficientNet models in terms of model size and ImageNet accuracy	15
2.4	Vision Transformer (ViT) model overview.	17
2.5	Comparison of feature maps produced by Swin Transformer and ViT.	17
2.6	Overview of the Faster R-CNN architecture.	18
2.7	Overview of SSD architecture using a VGG16 as backbone.	19
2.8	Overview of the EfficientDet architecture, highlighting the BiFPN layer.	20
2.9	PuTR functioning illustration	29
4.1	Precision-recall curve for the nonempty class	55
4.2	Sample classification results reached by the investigated models highlighting their confidence for the nonempty class.	56
4.3	Precision-recall curve for the nonempty class.	57
4.4	Latency vs. area under the precision-recall curve (PR AUC)	59
5.1	The number of instances per category observed in the camera-trap datasets used in this work follows a long-tail distribution	63
5.2	Schematic diagram of model training process.	68
5.3	Framework of our Square-root Sampling Branch (SSB)	70
5.4	Comparison in terms of macro-averaged F1-score for all combinations of datasets and models	74
5.5	Breakdown of F1-score improvement per class in the WCS dataset for the long-tail visual recognition methods over the baseline training procedure	75
5.6	Sample classification results for long-tail methods.	77
6.1	Failure case of Simple bounding box counting	84
6.2	The MaxBoxCount heuristic for counting	87
6.3	Fusion strategy for the classifier ensemble used in MaxBoxCount	88

6.4	Heuristic baseline MaxBoxCount for counting animals	89
6.5	Overview of the proposed Pure Transformer for Counting-by-Tracking (PuTR-CouT) framework	93
6.6	Example of synthetic sample for tracking	94
6.7	Example case of wrong frame ordering in iWildCam 2021	96
6.8	Sample sequences of animals appearing only in a single frame	99
6.9	Sample counting predictions	104
6.10	Selected failure cases of PuTR-CouT	105

List of Tables

2.1	List of transformations used in the RandAugment policy.	21
2.2	Confusion matrix for binary classification	30
3.1	Summary of limitations in the literature and how the proposed approaches address them across different camera-trap tasks.	47
4.1	Number of images used from the Caltech dataset	51
4.2	Number of instances used from the Snapshot Serengeti dataset.	52
4.3	Batch size and initial learning rate used for each classifier.	53
4.4	Training hyperparameters for detectors.	53
4.5	Comparison of precision for the nonempty class and the true negative rate (TNR)	55
4.6	Comparison of precision for the nonempty class and the true negative rate (TNR)	57
4.7	Performance comparison of MobileNetV2 models for various widths on the SS dataset	58
5.1	Number of images and locations per partition used from each dataset. . . .	65
5.2	Input scaling procedure for each architecture.	67
5.3	Comparison between SSB accuracy with different methods for long-tail classification of different models and datasets when using the entire image as the input	72
5.4	Results attained by different methods and models on different datasets when using the bounding box with the highest confidence as input.	76
5.5	Comparing the results of one-stage training using square-root re-sampling and CBFocal loss with the baseline training procedure and our proposed SSB.	78
6.1	Timestamp consistency analysis between EXIF metadata and JSON metadata in the iWildCam 2021 dataset.	96
6.2	Main results of the improved MaxBoxCount and PuTR-Count on iWildCam 2021	102

6.3	Effect of reordering (train/test) sequences and pre-filtering empty bounding boxes on counting performance	106
6.4	Ablation of different components of the affinity matrix in PuTR	106
A.1	Training hyperparameters for each stage of the classifiers training.	137
A.2	The iWildCam 2021 final results on the private test set.	138
A.3	An ablation study of our iWildCam2021 solution	138

Contents

Acknowledgments	vii
Resumo	ix
Abstract	xi
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Context and motivation	1
1.2 Challenges in automated analysis of camera-trap data	3
1.3 Computer vision tasks in automated analysis of camera-trap data . . .	4
1.4 Research Hypotheses	5
1.5 Objectives	7
1.6 Contributions	8
1.6.1 Publications	9
1.7 Document Outline	10
2 Fundamentals	12
2.1 Computer vision architectures	12
2.1.1 ResNet	13
2.1.2 MobileNet	13
2.1.3 EfficientNet	15
2.1.4 Vision transformers	16
2.2 Object detectors	18
2.2.1 Faster R-CNN	18
2.2.2 YOLO	19
2.2.3 SSD	19
2.2.4 EfficientDet	20

2.3	Image preprocessing techniques	20
2.4	Training techniques	21
2.5	Model optimization techniques	23
2.6	Training approaches for long-tail visual recognition	24
2.6.1	Re-sampling	24
2.6.2	Re-weighting	25
2.6.3	Two-stage training	26
2.6.4	Multi-branch networks	26
2.7	Multi-object Tracking Methods	27
2.7.1	SORT	27
2.7.2	ByteTrack	27
2.7.3	BoT-SORT	28
2.7.4	PuTR	28
2.8	Evaluation Metrics	29
2.8.1	Classification Metrics	29
2.8.2	Precision-Recall Curve	31
2.8.3	Counting Metrics	31
2.9	Final Remarks	32
3	Related Work	33
3.1	Filtering empty camera-trap images	33
3.2	Species identification	36
3.3	Classifying capture events	41
3.4	Counting animals	43
3.5	Edge AI for camera traps	44
3.6	Final remarks	46
4	Filtering Empty Camera-Trap Images in Embedded Systems	49
4.1	Introduction	49
4.2	Materials and Methods	50
4.2.1	Datasets	50
4.2.2	Architectures	52
4.2.3	Implementation Details	52
4.2.4	Evaluation Procedure	53
4.3	Experimental Results	54
4.3.1	Classifiers vs. Detectors	54
4.3.2	Training with more Images	56

4.3.3	Model Optimization	58
4.3.4	Threats to Validity	59
4.4	Final Remarks	60
5	SSB: Square-Root Sampling Branch	62
5.1	Introduction	62
5.2	Materials and Methods	63
5.2.1	Datasets	64
5.2.2	Architectures	66
5.2.3	Baseline training settings	66
5.2.4	Training approaches for long-tail visual recognition	67
5.2.5	SSB: Square-root Sampling Branch	68
5.3	Experiments and Results	69
5.3.1	Comparison between SSB and two-stage methods for long-tail visual recognition	71
5.3.2	Comparison between methods when using cropped images	73
5.3.3	Comparing SSB to one-stage training strategies	78
5.4	Discussion	78
5.5	Final Remarks	82
6	PuTR-CouT: Counting-by-Tracking in Camera-Trap Images	83
6.1	Introduction	83
6.2	Task definition and dataset	85
6.2.1	Counting task in camera trap image sequences	85
6.2.2	iWildCam 2021 dataset	86
6.3	Heuristic counting baseline	86
6.3.1	Maximum bounding box count heuristic	87
6.3.2	Improving the baseline heuristic	88
6.4	Pure Transformer for Counting-by-Tracking in Camera Trap Image Se- quences	91
6.4.1	Generating a synthetic training set for tracking using camera trap sequence priors	92
6.4.2	Modeling the distribution of sequence-level animal counts for synthetic data generation	96
6.4.3	Adapting PuTR for tracking under camera trap constraints . . .	97
6.4.4	Tuning PuTR inference for the counting task	98
6.5	Experiments and results	100

6.5.1	Improved MaxBoxCount heuristic results	100
6.5.2	PuTR-CouT results	100
6.5.3	Ablation studies on PuTR-CouT	103
6.6	Discussion	106
6.7	Final remarks	110
7	Conclusions	111
7.1	Limitations	113
7.2	Future Directions	113
	Bibliography	115
	Appendix A Heuristic Counting Solution for the iWildCam 2021 Chal-	
	lenge	135
A.1	Species classifier	135
A.2	Results	137

Introduction

In this Chapter, we present the context, challenges, and motivation for this research. We also outline the research hypotheses, objectives, and structure of the document.

1.1 Context and motivation

Wildlife monitoring is an essential data source used by ecologists and biologists to analyze and answer diverse scientific research questions about wildlife ecology. For example, it helps in understanding the complex dynamics of wildlife systems, assessing the impact of human actions and climate change on ecosystems, and establishing effective species conservation strategies [3, 60, 155]. This monitoring can be conducted using a variety of sensors [155], which allow the collection of data on animal populations from different perspectives, such as remote sensing (e.g., images obtained through satellites or drones, GPS sensors implanted on animals), static sensors (e.g., camera traps and bioacoustic sensors), or even through citizen science platforms, where ordinary people can report their wildlife encounters, like iNaturalist [64] and eButterfly [113].

Among these types of sensors, camera traps are a well-established, non-invasive, and cost-effective method that has been used for decades to collect data on animals in their daily lives. The collected data can support ecological studies on the terrestrial vertebrate community, including species occupancy, diversity, abundance, density, community structure, and animal behavior [3, 11]. These cameras are usually activated by motion sensors or at programmed time intervals to record short sequences of images or videos [60]. They are able to produce a large number of pictures; for instance, the Snapshot Serengeti project captured 3.2 million images between June 2010 and May 2013 over six seasons [143], with this number growing to 7.1 million images over eleven seasons in recent years [89]. However, processing this amount of data to extract

information is expensive and time-consuming if done manually by experts or citizen scientists. Using the Snapshot Serengeti project as an example again, its volunteers donated more than 17,000 hours of effort to manually label the 3.2 million images in the original dataset [107]. Nevertheless, recruiting such a large community of volunteers may be unfeasible for small-scale projects or studies focused on less charismatic fauna.

In the last decade, deep learning has become the dominant approach in image analysis and has been investigated for automatically processing and extracting information from camera-trap data in various tasks, such as filtering empty images [4, 5, 107, 161, 167, 173, 178, 179, 180], classifying animal species [14, 83, 105, 107, 130, 149, 163, 173, 190], counting individuals [8, 106, 107, 127, 136, 166], and identifying behavior [107, 128]. Meanwhile, several platforms have been developed to assist researchers in extracting information and speeding up this process using deep-learning models [162]. Some of these platforms allow a wider audience of camera-trap users to train their own models with user-friendly interfaces. For example, MegaDetector [13] is a generalist model trained to localize animals, people, and vehicles, helping users to identify animals but not classifying them at the species level, and is effective in filtering empty images. While MegaDetector is based on the language Python, the Machine Learning for Wildlife Image Classification (MLWIC2) package was developed in R to facilitate the use of deep-learning models by biologists, since they are usually more familiar with this language [148]. MLWIC2 provides trained models for animal species classification and for filtering empty images, and also allows users to train new models. Other platforms have also developed graphical user interfaces, such as Wildlife Insights [2], Conservation AI [26], IFLA [176], and Animal Scanner [184], to aid researchers in extracting information.

These efforts in developing AI-assisted tools have significantly accelerated camera-trap data processing and labeling workflows, enabling their practical use at scale [16, 18, 19, 24, 28, 30, 44, 48, 98, 119]. Some studies have reported that species labels generated by computer vision models for calculating ecological metrics – such as species richness, occupancy, and activity patterns – can approximate those produced from expert-labeled data [109, 169]. However, many challenges, such as extreme class imbalance and domain generalization, are not yet well addressed [10]. Moreover, a considerable portion of this data is still processed and analyzed manually, which limits the scalability of biodiversity monitoring due to the impractical amount of human labor required [16].

1.2 Challenges in automated analysis of camera-trap data

Extracting information from camera-trap images presents numerous challenges that make this process difficult even for specialists, such as animals being too far from the camera, camouflage, animals in complex poses, and poor illumination, among other issues, as shown in the literature [14, 52, 107, 163]. Due to such conditions, capturing more than one image per sensor activation can increase the chance of correctly identifying the photographed animal, as its pose can be crucial for viewing distinguishing characteristics of a species [14].

Another challenge is the extreme imbalance among categories, where some species have many images while others have just a few, following a long-tail distribution [59]. As reinforced by Schneider et al. [130], models generally achieve poor recall scores for species with fewer training images, although these rare species are often the ones of greatest interest to ecologists.

The capacity for domain generalization of the models is also a relevant issue since camera traps are static, causing the models to become biased toward the backgrounds of the sites used during training [14, 105, 130, 149].

Furthermore, most camera-trap practitioners typically have access only to limited computational resources, making it impractical to run large models [130]. This highlights the importance of designing computationally efficient algorithms, which can also be useful for the development of sensors capable of processing data directly on-device. This could bring many advantages; for instance, in projects where the equipment is connected to a network, prior filtering avoids the unnecessary transmission of empty images, saving network bandwidth and energy [42]. However, few studies address the trade-off between model performance and computational resource utilization directly on devices with limited computational power [156, 193]. There is also a growing concern about the environmental impact of utilizing larger models that demand substantial power consumption [119].

Camera-trap data also presents many other challenges, such as noisy labels, non-independent occurrences, and limited-size datasets [107, 130]. These challenges provide a unique opportunity to develop innovative solutions.

1.3 Computer vision tasks in automated analysis of camera-trap data

Before ecologists can use the camera-trap data in their ecological models, the raw data must be processed to extract the required information, such as identifying which species are present, counting the individuals of each species in a capture event, and identifying the presence of males, females, or young animals, as well as their activities (e.g., eating, resting, fighting, etc.) [107]. The specific inputs required vary depending on the study or project. Such processing steps can be framed as computer vision tasks and serve as a research catalyst, allowing for the development of new methods.

A well-studied task is species identification as a visual classification task, and this is usually done by considering each picture as an independent sample. Even though classification has been a widely studied problem in computer vision, camera-trap data presents multiple challenges to this task, such as extreme class imbalance [130], few-shot classification [5], and fine-grained classification [157], all occurring within the same dataset. Additionally, when it is not possible to identify an animal at the species level, it is still valuable to ecologists to classify it at higher taxonomic ranks [129]. This hierarchical classification adds complexity to the problem. Other tasks can also be mapped as classification tasks, such as age estimation, sex identification, and activity recognition [107].

As most projects capture more than one picture per capture event in hopes of getting a good shot to identify the photographed animal, one task should be to identify the species based on multiple images [14]. Usually, each picture is considered an independent sample during both training and inference. Using the sequence of images can be helpful to improve species classification, as identifiable marks may only be visible in certain pictures within the sequence. The task, therefore, is to identify the species of the animal based on all images from the capture event, rather than using a single picture.

Since the majority of camera-trap images are empty frames due to accidental sensor activation, one relevant task that can be automated is filtering out those empty frames. MegaDetector has played an efficient role in solving this task [48]; however, this step has typically been performed after data collection. It may be beneficial to execute certain tasks directly on the edge device, which requires developing efficient models to run on limited computational power devices. This can help reduce the amount of data transmitted over a limited-bandwidth network and decrease the need for storage, thereby increasing the time required to retrieve data from the camera. Detector

models have already proven effective in locating animals [13, 48], as demonstrated by MegaDetector, but they require higher computational power. More computationally efficient alternatives, such as binary classifiers and background filtering with Siamese networks [4], can also be explored.

An even more challenging task is counting the number of individuals of each species in a given sequence of images. This task requires re-identifying individuals appearing in one image across the sequence. This can be achieved using methods that localize the animals in images, using the spatiotemporal coherence, by a tracking method, or by a re-identification method [11, 127].

Many other tasks can be cited, such as re-identifying individuals in species with distinct markings [21, 37, 70, 132], such as zebras and jaguars, pose estimation [141], and tracking animals for estimating their trajectories to infer direction, speed, or migration patterns [121].

1.4 Research Hypotheses

The automated analysis of camera-trap data presents a variety of challenges that not only make the problem complex but also create opportunities for developing new learning strategies tailored to real-world constraints. This work adopts an application-driven perspective, treating these constraints as catalysts for methodological innovation rather than mere obstacles to be overcome.

Traditionally, machine learning research has followed a methods-driven paradigm, in which models are developed in isolation from specific domain characteristics and subsequently evaluated on standardized benchmark tasks. However, when applied to real-world applications, these methods often fall short in many use cases [120].

In contrast, an application-driven paradigm emphasizes the role of real-world problems as the primary drivers of research directions. From this perspective, the inherent characteristics of camera-trap data are not only challenges to be addressed, but also valuable sources of insight that motivate the design of new strategies. As argued by Rolnick et al. [120], such application-driven settings can lead to innovations that extend beyond their original domain, contributing to the development of more robust, efficient, and generalizable machine learning methodologies. Consequently, this research treats camera-trap data not simply as an application domain for downstream tasks, but as a primary source of research questions that drive innovation.

This thesis adopts an application-driven perspective in which the constraints inherent to camera-trap data are used to identify research questions and motivate

methodological choices. This perspective is examined through three concrete and independently testable hypotheses concerning empty-frame filtering, long-tailed classification, and counting in low-frame-rate image sequences, as described in the following paragraphs.

As stated in the previous section, filtering empty images directly on-device can offer several advantages, such as reducing storage requirements and minimizing the amount of data transmitted over limited-bandwidth networks. Although detection models can achieve superior performance due to their inherent capabilities, their deployment on edge devices may be constrained by limited computational resources. Additionally, bounding-box annotations are often more costly to obtain, particularly for small-scale projects, whereas image-level classification labels are generally more feasible to acquire at scale.

Therefore, we investigate whether the trade-off between annotation cost, filtering performance, and inference latency can make classification models a practical alternative to detection models in resource-constrained settings. Based on this premise, we hypothesize:

Hypothesis 1: When the cost and availability of annotations are taken into account, classification models trained with a larger set of image-level labels can achieve empty-frame filtering performance comparable to detection models trained with fewer bounding-box annotations, while requiring lower inference latency on edge devices.

While species identification in camera-trap images is the most relevant and widely studied task, several challenges remain, particularly the long-tail issue, in which tail classes often exhibit low performance [130]. Although previous studies acknowledge this problem, their approaches vary. Some apply imbalance-handling methods to improve performance on tail classes, often at the cost of reduced performance on head classes [12, 18, 102, 107, 130]. Others simply ignore rare species by removing or grouping them [14, 149, 163, 173]. However, these rare species are, in many cases, of primary ecological interest, highlighting the need for more effective strategies to handle extreme imbalance. Given previous results, as well as the development of methods such as Balanced Group Softmax [87], we hypothesize that:

Hypothesis 2: Aggregating predictions from independent classification heads trained under different data distributions, e.g. balancing and original global distribution, improves performance on tail classes while maintaining comparable performance on head classes compared to single-branch training strategies.

Finally, the task of counting individuals of each species across multiple images in a sequence is particularly challenging, mainly due to the lack of labeled data in most camera-trap datasets. Some efforts have been made to address this problem, notably through the introduction of the iWildCam 2021 dataset and challenge [11]. The top solutions and subsequent literature [136, 166] have largely relied on heuristic approaches based on counting the maximum number of bounding boxes across a sequence. However, this strategy has key limitations, as it often assumes a single species per sequence and is prone to over- and under-counting.

Tracking provides a natural alternative by associating detections of the same individual across image sequences. Recently, tracking and individual re-identification have been explored for wildlife monitoring using videos, such as in GorillaWatch [127]. However, camera traps typically capture burst sequences at low frame rates (e.g., one frame per second). Due to large spatial displacements and significant visual variation between consecutive frames in these low-frame-rate bursts, traditional trackers that rely primarily on geometric and visual features tend to perform poorly. A promising alternative is the use of trackers with learned association modules, such as PuTR [94]. Based on this, we hypothesize that:

Hypothesis 3: A tracking-based approach with learned association mechanisms can accurately estimate the number of individuals in camera-trap image sequences, achieving comparable performance to heuristic methods based on independent frame-level detections while also enabling multi-species estimation.

1.5 Objectives

The main objective of this work is to investigate how the inherent constraints of camera-trap data can drive novel computer vision methods, treating challenges like long-tailed distributions, small datasets, and limited hardware as catalysts for developing robust and efficient wildlife monitoring solutions. To this end, we focus on three fundamental tasks in camera-trap data analysis: empty-frame filtering, long-tailed species classification, and counting individuals across image sequences.

The specific objectives are as follows:

- Quantify and characterize the trade-off between filtering effectiveness, annotation requirements, and inference latency for classification and detection models across different training data regimes, under limited computational resources.
- Enhance classification performance for extreme long-tailed data distributions, achieving a better trade-off between improving tail-class accuracy and maintaining head-class performance.
- Propose and validate a tracking-based approach for multi-species counting in image sequences that is robust to the large animal displacements caused by low-frame-rate sampling.

The specific objectives above are addressed through the contributions summarized in the following section.

1.6 Contributions

This thesis investigates how the inherent constraints of camera-trap data can drive the development of robust and efficient computer vision methods. The main contributions of this work address three challenges arising from camera-trap data: limited computational resources, long-tailed species distributions, and low frame rates in image sequences.

The main contributions are:

- A systematic and comparative analysis of detection and classification models for filtering empty camera-trap images under computational and data constraints. The study characterizes the trade-off between filtering performance, training data availability, and inference latency, providing insights into the use of these models on resource-constrained devices. The results are presented in Chapter 4.
- The proposal of the Square-Root Sampling Branch (SSB), a framework for long-tailed species classification that combines predictions from independent classification heads trained under different data distributions. The proposed approach improves performance on tail classes while maintaining comparable performance on head classes. The method and its evaluation are presented in Chapter 5.
- A systematic evaluation of long-tail learning strategies across multiple datasets and architectures, providing empirical baselines and insights into the effectiveness

of different approaches for long-tailed animal species classification. This analysis is presented in Chapter 5.

- The development of PuTR-CouT, a transformer-based counting-by-tracking framework for estimating the number of individuals of multiple species across camera-trap image sequences. The approach uses learned association mechanisms to address the large displacements caused by low frame rates and produces track-level predictions that facilitate the verification of the estimated counts. The proposed framework is presented in Chapter 6.
- A strategy for generating weakly labeled tracking datasets from camera-trap data priors, enabling the training of tracking methods with learned association mechanisms in the absence of manually annotated tracking data. The strategy is described in Chapter 6.

1.6.1 Publications

The research conducted in this thesis has resulted in several publications, including works developed in collaboration with other authors. The following publications are related to the research presented in this thesis:

- Cunha, F., dos Santos, E. M., Barreto, R., and Colonna, J. G. (2021). Filtering empty camera trap images in embedded systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2438–2446.

Summary: This paper presents a comparative study of animal recognition models, analyzing the trade-off between accuracy and inference latency on edge devices. Its core contributions are presented in Chapter 4.

- Cunha, F., dos Santos, E. M., and Colonna, J. G. (2023). Bag of tricks for long-tail visual recognition of animal species in camera-trap images. *Ecological Informatics*, 76:102060.

Summary: This work introduces the Square-Root Sampling Branch (SSB) framework to improve long-tailed visual recognition by leveraging camera trap context. The proposed method is systematically evaluated against state-of-the-art approaches. Its content is mostly covered in Chapter 5.

- Santos, E. M. d., Cunha, F., Colonna, J. G., and Carvalho, J. R. H. (2023). Monitoramento ambiental não invasivo utilizando dados de sensores e técnicas de aprendizagem de máquina. *Computação Brasil*, 1(50):24–28

Summary: This paper outlines initiatives developed at the Institute of Computing (UFAM) for wildlife monitoring using machine learning, based on data from non-invasive sensors such as camera traps and acoustic sensors.

- Alencar, L., Cunha, F., and dos Santos, E. M. (2023). A context-aware approach for filtering empty images in camera trap data using siamese network. In *2023 36th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 85--90. IEEE

Summary: This work presents an approach based on a Siamese convolutional neural network for filtering empty camera trap images by leveraging contextual information. It compares newly captured images with reference images obtained regularly. The paper also proposes a protocol for selecting image pairs to properly train the Siamese network.

- Alencar, L., Cunha, F., and dos Santos, E. M. (2024). Zero and few-shot learning with modern mllms to filter empty images in camera trap data. In *2024 37th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE

Summary: This paper investigates the use of large-scale multimodal language models in zero- and few-shot settings for filtering empty camera trap images. It evaluates models such as CLIP, BLIP, and Gemini, comparing them against a specialized Siamese ResNet50 baseline across multiple datasets, and analyzes the impact of multimodal integration on animal presence detection.

1.7 Document Outline

The remainder of this text is organized as follows:

- Chapter 2 introduces the main concepts related to this work, including machine learning principles, computer vision architectures, and long-tail classification methods. We also describe the metrics used to evaluate our proposed methods.
- We present the related works in Chapter 3. We organize them according to each task approached: filtering empty camera-trap images, long-tail classification, multi-image classification, and animal counting.

- In Chapter 4, we present a comparative study on empty frame detection in camera-trap images to analyze the trade-off between precision and inference latency on edge devices. We investigate classifiers and object detectors of various input resolutions, optimizing them using quantization and by reducing the number of model filters.
- In Chapter 5, we propose a simple and effective framework called Square-Root Sampling Branch (SSB) to improve long-tail visual recognition of animal species in camera-trap images. We systematically evaluate it against state-of-the-art methods for handling this task: square-root sampling, class-balanced focal loss, and balanced group softmax.
- In Chapter 6, we propose a counting-by-tracking approach for estimating the number of animals per species in camera-trap image sequences. The method leverages a transformer-based tracker with a learned association model, trained on a synthetic dataset derived from camera-trap data priors. We compare our approach against state-of-the-art trackers and a strong heuristic baseline based on maximum bounding-box counting.
- In Chapter 7, we summarize the main findings of the thesis, discuss its limitations, and suggest directions for future research.

Fundamentals

This chapter presents the key concepts underlying this work. We begin by providing an overview of machine learning topics relevant to our study, including computer vision architectures, image preprocessing and data augmentation techniques, training and model optimization methods, approaches for addressing long-tail classification, and methods for multiple-object tracking. Finally, we describe the evaluation metrics used to assess the performance of our proposed methods.

2.1 Computer vision architectures

Convolutional neural networks (CNNs) have become the dominant approach in computer vision since AlexNet [82] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 by a substantial margin over the second-place entry (15.3% vs. 26.2% top-5 error). Since then, numerous innovations have been introduced to develop new architectures, such as increasing depth with very small (3×3) convolutional filters in VGG [137], the use of lateral branches, and more complex building blocks like the inception module employed by the Inception family [145, 146]. More recently, Vision Transformers [39] have emerged as a competitive modeling paradigm in computer vision. In this work, depending on the task, we evaluate several architecture families to assess methods' behavior across different models. Specifically, we use ResNet as a widely adopted baseline, MobileNets for resource-constrained environments, EfficientNets for parameter-efficient designs, and Vision Transformers to explore architectures beyond CNNs.

2.1.1 ResNet

ResNet (Residual Network) [57] is one of the most widely used convolutional neural network architectures. Its main innovation over previous deep architectures was the introduction of residual blocks (see Figure 2.1), which include shortcut (skip) connections that bypass one or more layers by adding the input x of a block $\mathcal{F}$ directly to its output: $y = \mathcal{F}(x) + x$. These residual connections were developed to address the degradation problem, where increasing network depth caused the accuracy to saturate and then degrade, primarily due to vanishing gradients or optimization difficulties. To explore the benefits of deeper networks made feasible by residual blocks, the authors introduced several ResNet versions with different numbers of layers: 18, 34, 50, 101, and 152. Among these, ResNet-50 has become the most popular backbone used as baseline for almost any computer vision task. ResNet achieved a top-5 error rate of 3.57%, winning first place in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015.

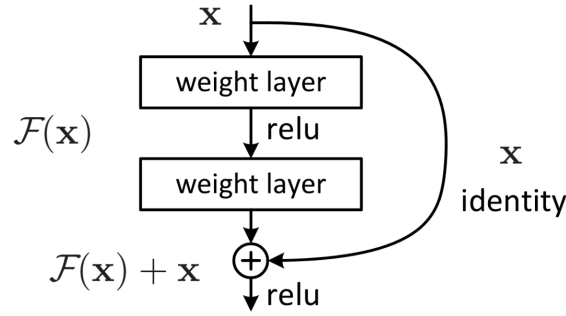


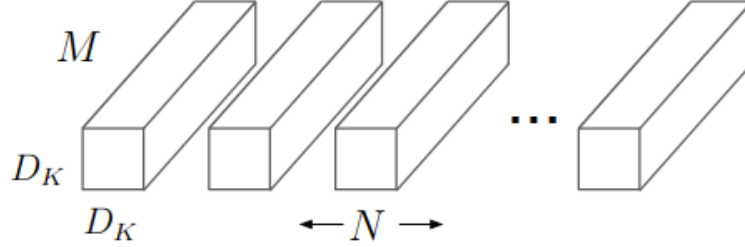
Figure 2.1. Residual building block.

Source: He et al. [57].

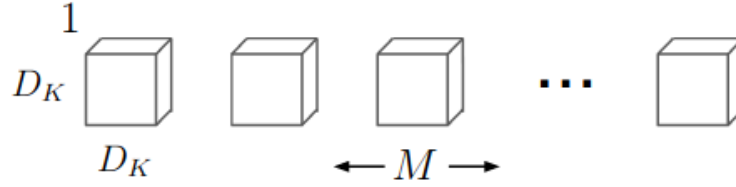
2.1.2 MobileNet

MobileNet [66] is a family of models developed with the objective of being efficient for mobile and embedded vision applications. To reduce computation and model size, the architecture relies primarily on depthwise separable convolutions, which factorize standard convolution into a depthwise convolution followed by a 1×1 pointwise convolution (see Figure 2.2). To enable smaller and faster models, MobileNet also introduces two shrinking hyperparameters: the width multiplier, which uniformly thins the network by reducing the number of input channels M to αM and the number of output channels N to αN , with $\alpha \in \{1, 0.75, 0.5, 0.25\}$; and the resolution multiplier, which scales the input image resolution, typically taking values in $\{224, 192, 160, 128\}$. Compared to

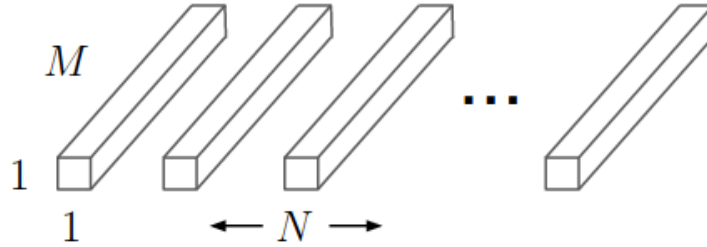
GoogLeNet [145] on ImageNet, MobileNet is more accurate while being approximately 2.7 times less computationally intensive. When compared to VGG16 [137], MobileNet is nearly as accurate but requires 27 times less computation.



(a) Standard convolutional filters.



(b) Depthwise convolutional filters.



(c) 1×1 convolutional filters (pointwise convolutional filters).

Figure 2.2. MobileNet replaces standard convolutional filters (a) with depthwise convolutional filters (b) followed by pointwise convolutional filters (c).

Source: Adapted from Howard et al. [66].

MobileNetV2 [124] introduces a new building block called inverted residual (MB-Conv) that takes a low-dimensional compressed representation as input, first expands it to a higher-dimensional space, applies a depthwise convolution, and then projects it back to a low-dimensional representation using a linear convolution. It also includes a residual connection. This module enables memory-efficient inference and achieves 1.4% higher accuracy on ImageNet than MobileNetV1, with an inference time that is 1.5 times faster.

In its next iteration, MobileNetV3 [65] explores neural architecture search to optimize the models for mobile phone CPUs, resulting in two new versions: MobileNetV3-Large and MobileNetV3-Small. MobileNetV3-Large achieved 3.2% higher accuracy on ImageNet compared to MobileNetV2, while reducing latency by 20%. MobileNetV3-Small, on the other hand, achieved 6.6% higher accuracy than a MobileNetV2 model with similar latency.

2.1.3 EfficientNet

When more resources are available, it is common to improve accuracy by scaling up convolutional networks along different dimensions—for example, depth (increasing the number of layers), width (increasing the number of channels), and resolution (using higher input image sizes). Tan and Le [150] showed that scaling all dimensions using a compound coefficient is more effective than scaling only one, as was the case with most previous methods. Motivated by this, they used neural architecture search to design a new baseline network based on the MBConv block [124], optimizing both accuracy and FLOPs. Applying compound scaling to this baseline resulted in a new family of models called EfficientNet, which achieved better accuracy and efficiency than previous architectures (see Figure 2.3).

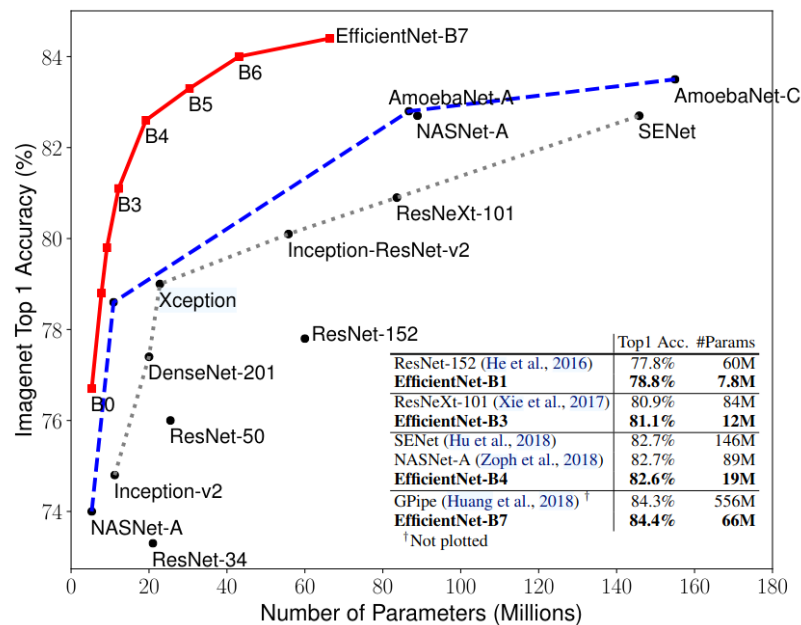


Figure 2.3. Comparison of EfficientNet models and other architecture families in terms of model size and ImageNet accuracy.

Source: Tan and Le [150].

Subsequently, the EfficientNetV2 [151] family was developed using neural architecture search to generate a baseline that optimizes not only parameter efficiency but also training speed. In this version, alongside the MBConv block, the fused-MBConv block was added to the search space. In this block, the depthwise separable convolution is replaced by a standard convolutional layer to better utilize current hardware accelerators. EfficientNetV2 achieved 87.3% accuracy on ImageNet, which is 2% higher than ViT [39] using the same computing resources, while training 5 to 11 times faster.

2.1.4 Vision transformers

While CNNs have dominated modeling in computer vision, self-attention architectures have become prevalent in natural language processing (NLP) – especially the Transformer [160], which has emerged as the dominant architecture due to its efficiency and scalability, enabling unprecedented advancements in the field. Even though multiple works attempted to combine CNNs with self-attention, ViT [39] was the first architecture to apply the Transformer directly to images. The input image is split into a sequence of patches, with a positional embedding added to each one to retain positional information. These patches are then processed by a standard Transformer encoder, treated the same way as tokens in NLP (see Figure 2.4). When this strategy is combined with pre-training on large datasets, using image classification as a supervised learning task, it performs well, with the best model achieving an accuracy of 88.55% on ImageNet.

One of the challenges in transferring the high performance of Transformers from NLP to vision applications lies in scalability, as the computational complexity of self-attention is quadratic with respect to image size. To address this, the Swin Transformer [96] computes self-attention locally within non-overlapping windows, partitioning the image into fixed-size patches. While the standard Vision Transformer produces a single low-resolution feature map, the Swin Transformer builds hierarchical feature maps by gradually merging these patches in deeper layers (see Figure 2.5). This design not only reduces computational complexity to linear with image size, but also produces hierarchical features that can effectively replace convolutional neural networks as backbones in advanced computer vision frameworks. To overcome the lack of connections across windows, the Swin Transformer introduces shifted window partitioning, which alternates the window configuration across consecutive blocks. These advancements resulted in a transformer-based backbone compatible with a wide range of tasks such as classification, object detection, and semantic segmentation, achieving, for example, 86.4% top-1 accuracy on ImageNet.

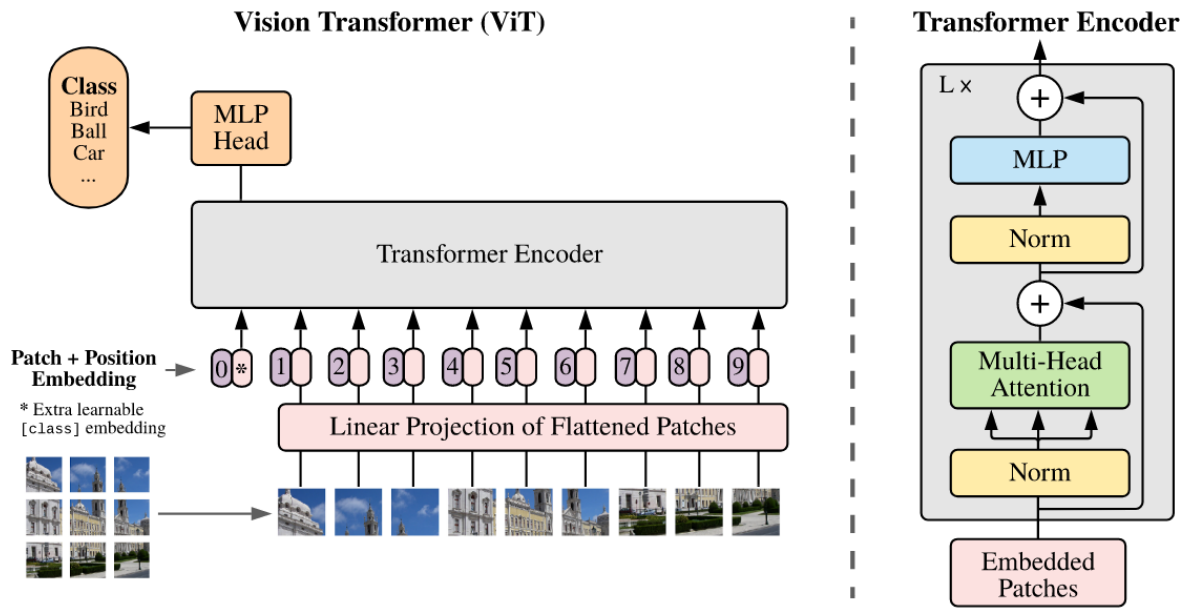


Figure 2.4. Vision Transformer (ViT) model overview.

Source: Dosovitskiy et al. [39].

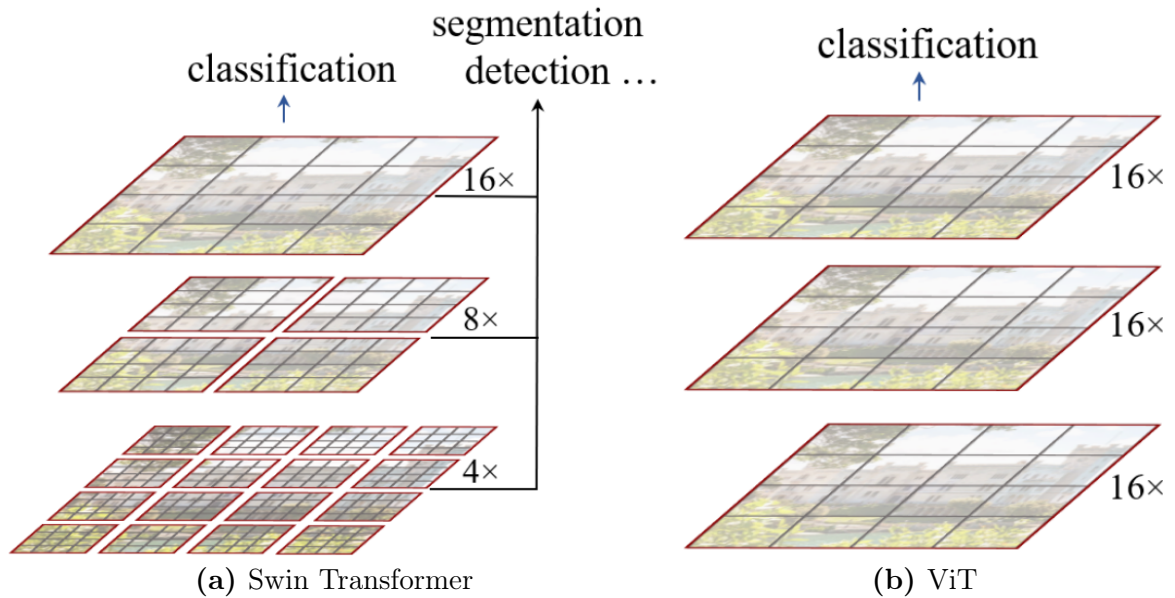


Figure 2.5. Comparison of feature maps produced by Swin Transformer and ViT.

Source: Liu et al. [96].

2.2 Object detectors

For the task of object detection – localizing, defining a bounding box around objects of interest and classifying them – several frameworks have been proposed, which can generally be divided into one-stage and two-stage approaches. In the two-stage approach, with Faster R-CNN [117] being the main representative, the first stage generates a set of candidate regions, and the second stage classifies each region. However, this approach is computationally expensive. In contrast, one-stage methods, such as the YOLO family [114], SSD [95], and EfficientDet [152], perform detection in a single pass over the image.

2.2.1 Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Network) [117] is a two-stage object detection framework in which both stages share the feature map generated by a computer vision backbone (see Figure 2.6). In the first stage, a Region Proposal Network (RPN) with a fixed input size of $n \times n$ is applied over the feature map as a sliding window to generate multiple candidate regions. These regions are then classified in the second stage. The output of the RPN includes both a regression component and a binary classifier that indicates the confidence of the presence of an object.

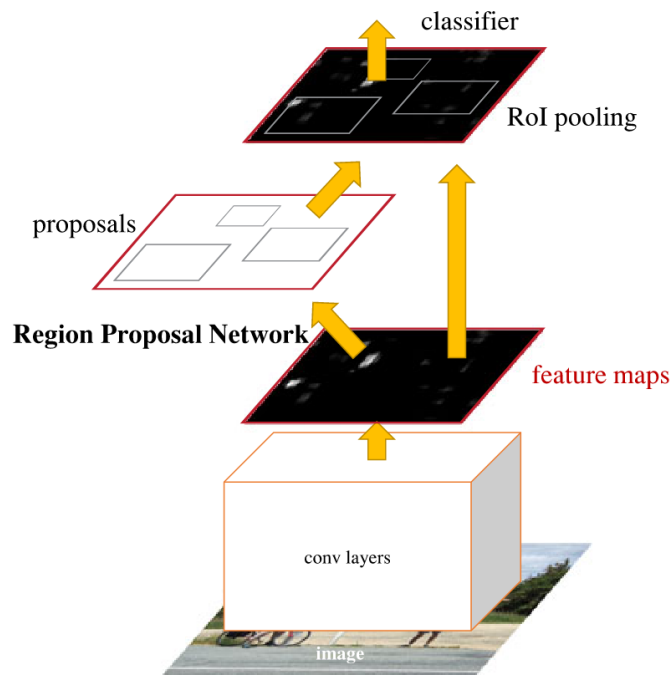


Figure 2.6. Overview of the Faster R-CNN architecture.

Source: Ren et al. [117].

In the second stage, the proposed regions of interest (RoI) are projected onto the feature map and used as input to a network responsible for both classifying the object and refining the region regression to better fit the object. As region proposals may have arbitrary sizes, a RoI pooling layer is used to adjust the feature map to a fixed input size. Therefore, the first stage acts as a guide to direct the attention of the second stage.

2.2.2 YOLO

To reduce the computational cost of object detection, YOLO [114] formulates the task as a regression problem to spatially separated bounding boxes and associated class probabilities. The neural network predicts both bounding boxes and class probabilities in a single pass over the image, allowing the model to be optimized end-to-end using a loss function that directly reflects detection performance. Subsequently, multiple versions of YOLO have been developed to improve accuracy and reduce latency [74, 75, 115, 116, 159].

2.2.3 SSD

Single Shot Detector (SSD) [95] uses a single deep neural network to detect objects in an image. It applies a set of default boxes with multiple aspect ratios at each location in a discretized grid over the feature map, generating scores for the presence of each object category in each default box and producing adjustments to better fit the objects. One innovation of SSD is the addition of several feature layers on top of the backbone, allowing predictions of detections at multiple scales (see Figure 2.7).

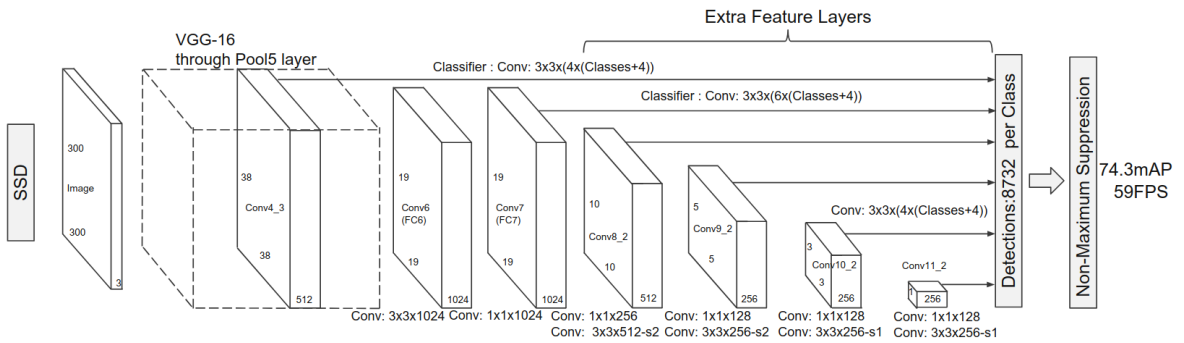


Figure 2.7. Overview of SSD architecture using a VGG16 as backbone.

Source: Adapted from Liu et al. [95].

Following the overall MobileNetV2 design, Sandler et al. [124] introduced a variant of SSD that replaces all standard convolutional layers with depthwise separable

convolutions, calling it SSDLite. While this model is $20\times$ more efficient and $10\times$ smaller, it still outperforms YOLOv2 on COCO.

2.2.4 EfficientDet

EfficientDet [152] is a one-stage family of object detection models that follows the efficiency principles of EfficientNet by jointly scaling resolution, depth, and width using a compound scaling method. A key innovation is the Bi-directional Feature Pyramid Network (BiFPN), which fuses multi-scale features through top-down and bottom-up pathways. EfficientDet uses ImageNet-pretrained EfficientNets as backbones, BiFPN as the feature network, and a shared class/box prediction network (see Figure 2.8), all of which are jointly scaled using the compound scaling method.

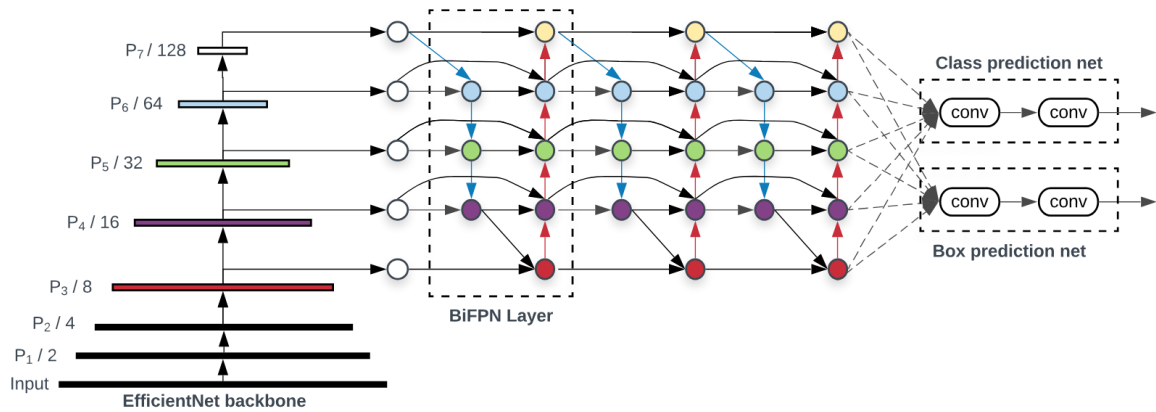


Figure 2.8. Overview of the EfficientDet architecture, highlighting the BiFPN layer.

Source: Tan et al. [152].

2.3 Image preprocessing techniques

Before images can be used in computer vision models, they must first be prepared in an appropriate format for each architecture through a preprocessing step. These steps typically include resizing and normalization. Another essential part of preprocessing is data augmentation, in which images are slightly modified to increase data diversity during training, improving model generalization.

One important preprocessing step involves standardizing image pixel values to a common range, with the most typical being in $[0, 1]$ or $[-1, 1]$. It is also common to normalize data using the training set's or ImageNet's mean and standard deviation. This can help stabilize and accelerate the training process [53].

Images also usually need to be resized to match the model’s input size, which varies depending on the architecture, although some models can handle inputs of any size [53].

Data augmentation in computer vision can be done using a wide variety of transformations. One common technique is random cropping, where a rectangular region is randomly sampled from the input image and then resized to the model’s input size. This rectangular region may vary significantly; for example, He et al. [58] use a pipeline with a random crop with aspect ratio in $[3/4, 4/3]$ and area in $[8\%, 100\%]$, which may heavily distort the image – yet this approach has proven effective in the literature. For camera-trap images, where animals may occupy only a small portion of the frame, it is important to use a narrower cropping range to reduce the risk of generating crops without the animal. Another common technique is flipping the image with a 50% probability [58]. In most cases – such as with camera-trap images – only horizontal flipping is suitable.

One challenge in designing data augmentation for different applications is determining which transformations and parameters are most effective for each case. RandAugment [31] proposes a simple parameterization to reduce the search space when applying a wide range of transformations (see Table 2.1) and selecting the probability for applying each transformation. The approach consists of choosing two parameters: N , the number of sequential transformations applied, and M , the magnitude for all transformations. This method has proven effective, increasing the accuracy of EfficientNet-B7 by 1%.

Table 2.1. List of transformations used in the RandAugment policy.

RandAugment transformations	
identity • autoContrast • equalize • rotate • solarize •	
color • posterize • contrast • brightness • sharpness •	
shear-x • shear-y • translate-x • translate-y	

2.4 Training techniques

Training a deep learning model involves choosing multiple interdependent components that guide the learning process. Given a training set and a cost function, the training process consists of searching for a set of parameters θ of a model that minimizes the cost function, thereby reducing the error on the training data. While this can be seen as an optimization problem, the true objective is to learn a model that performs well

on unseen data [53]. Key components in this process include the loss function, weight initialization, optimizer, and learning rate scheduler, among others.

The objective function is the function to be minimized or maximized during the training process. It is commonly referred to as the cost function, loss function, or error function when it is being minimized [53]. For classification problems, a common loss function is the softmax cross-entropy, given by:

$$CE_{\text{softmax}}(\mathbf{z}, y) = -\log \left(\frac{\exp(z_y)}{\sum_{j=1}^C \exp(z_j)} \right), \quad (2.1)$$

where $\mathbf{z} = [z_1, z_2, \dots, z_C]^T$ is the predicted output vector from the model for all C classes, and y is the ground-truth label of the sample. In this work, unless otherwise stated, softmax cross-entropy is the loss function used. When addressing specific problems, a common approach is to modify or design a new loss function tailored to the task. In Section 2.6.2, we describe loss functions developed to handle long-tail classification.

The initialization of model weights (also referred to as parameters) can be crucial in determining whether training will converge [53]. Various initialization strategies exist, such as random sampling from a normal or uniform distribution, initialization with zeros, or more advanced methods like Xavier initialization [51] and He initialization [56].

Another common approach is to use the weights from a previously trained model, also known as a pretrained model – for example, using weights from a model pretrained on ImageNet. This technique, known as transfer learning, is often used to accelerate training by requiring fewer steps to converge. Transfer learning can be applied in two ways: using frozen weights, where the pretrained weights are not updated during training, or fine-tuning, where the pretrained weights are further updated to adapt to the new task [182].

Most optimizers used in deep learning are based on the gradient descent method, where model weights are updated in small steps by iteratively minimizing the loss function [53]. When the entire training set is used for each update step, the optimizer is called deterministic. When only a single sample is used, it is referred to as stochastic. A middle-ground approach is to use minibatches, which contain more than one sample but fewer than the entire training set. The most important hyperparameter in Stochastic Gradient Descent is the learning rate, which controls the step size used for updating the weights. Another relevant hyperparameter is the batch size, which often depends

on the available computational resources. However, various issues can arise during optimization, such as getting stuck in local minima or saddle points. To address these problems, several techniques have been developed, leading to alternative optimizers such as RMSProp [61], Adam [80], and AdamW [100].

While determining the initial learning rate for a task is important for convergence speed and stability, adjusting it during training is also crucial [58]. A common strategy is to reduce the learning rate over time, often using exponential decay. For example, He et al. [57] apply a step decay schedule, reducing the learning rate by a factor of 0.1 every 30 epochs. Similarly, Szegedy et al. [146] adopt an exponential decay approach, decreasing the learning rate by a factor of 0.94 every two epochs. Loshchilov and Hutter [99] propose a cosine annealing strategy, with more recent works adopting a simplified version in which the initial learning rate is gradually reduced to zero according to a cosine function [58]. In this schedule, usually called cosine decay or cosine schedule, given a total of batches T , the learning rate η_t at the batch t is given by:

$$\eta_t = \frac{1}{2} \left(1 + \cos \left(\frac{t\pi}{T} \right) \right) \eta, \quad (2.2)$$

where η is the initial learning rate. Another complementary strategy commonly used to schedule the learning rate is to “warm it up” at the beginning of training to avoid numerical instability. This strategy, proposed by Goyal et al. [54], consists of linearly increasing the learning rate from 0 to the initial value over the first m batches – for example, during the first 2 epochs.

2.5 Model optimization techniques

With AI becoming ubiquitous in our daily lives, optimizing models for both efficiency and accuracy on mobile and edge devices has become increasingly important [71]. One category of approaches focuses on designing more efficient architectures, such as MobileNets and EfficientNets. Another approach involves adapting models to specific hardware constraints or leveraging the availability of specialized hardware accelerators. This area encompasses a variety of techniques, including pruning [55], tensor decomposition [84], distillation [62], and quantization [71] (used in this work), among others.

Quantization is a technique in which the weights and/or activations of a model are converted to a lower bit-depth representation – for example, from 32-bit floating point to 8-bit integer. Jacob et al. [71] propose a quantization scheme that relies solely

on integer arithmetic to approximate floating-point operations. Quantization can be applied after training, in a process known as post-training quantization, which may result in a reduction in model accuracy. An alternative is quantization-aware training, which simulates quantization effects during training and can help mitigate the accuracy loss introduced by the quantization process.

2.6 Training approaches for long-tail visual recognition

This section describes state-of-the-art methods for handling the long-tail visual recognition problem. The methods are grouped into four main families according to their focus: re-sampling the data, re-weighting the loss, using two stages to focus on representation learning, and creating multiple branches in the network.

2.6.1 Re-sampling

Re-sampling methods attempt to sample the dataset in order to obtain a more balanced data distribution by over- or under-sampling instances according to the number of instances per class. On the one hand, random over-sampling consists of replicating instances from the minority classes; however, this method can lead to overfitting these instances [22, 29]. On the other hand, random under-sampling discards instances from the majority classes, which can degrade the representation learning, especially for extremely imbalanced datasets [29].

In some re-sampling strategies, the probability p_j of sampling an instance from class j is given by:

$$p_j = \frac{n_j^q}{\sum_{i=1}^C n_i^q}, \quad (2.3)$$

where C is the number of training classes, n_i is the number of training instances from class i , and $q \in [0, 1]$. When $q = 1$, each class has a probability p_j directly proportional to its number of instances, i.e., each instance has the same probability of being sampled, for this reason, this method is known as instance-balanced sampling, and, in practice, it uses the original data distribution [76]. When $q = 0$, each class has an equal probability ($p_j = 1/C$) of being selected, this is also called class-balanced sampling [76, 187]. In recent works [76, 187], an alternative between those two extremes was chosen by setting $q = 1/2$, called square-root sampling, with this approach returning a less imbalanced dataset.

2.6.2 Re-weighting

The re-weighting methods work by modifying the loss function in order to redefine the importance given to each class or instance during the training. These methods guide the network on where to pay more attention by assigning different weights to different categories or instances [76, 165, 187]. One naïve approach is to weight the dataset based on the inverse class frequency or the inverse square root of class frequency[29]. More sophisticated approaches, such as the Focal loss and the Class-Balanced loss, have been developed and take into consideration factors such as how easy one sample is, or the effective number of samples.

Focal loss [92] was proposed in the context of the object detection problem, in which there is an extreme foreground-background class imbalance. This loss automatically down-weights well-classified samples to make the training focus on hard training instances. To achieve this objective, given a class probability prediction p_i for the instance x_i , Focal loss L_{focal} is defined as

$$L_{focal} = -(1 - p_i)^\gamma \log(p_i), \quad (2.4)$$

where $\gamma \geq 0$ is a parameter that controls the rate at which easy examples are down-weighted.

The Class-Balanced loss (CB loss) [32] weights each class based on the effective number of samples E_n , which is given by:

$$E_n = \frac{1 - \beta^n}{1 - \beta}, \quad (2.5)$$

where n is the number of samples and $\beta \in [0, 1)$ is a hyperparameter. Given a class j with n_j training samples, the loss L with respect to this class is weighted by a factor using the effective number of samples:

$$CB(\mathbf{p}, j) = \frac{1}{E_{n_j}} L(\mathbf{p}, j) = \frac{1 - \beta}{1 - \beta^{n_j}} L(\mathbf{p}, j), \quad (2.6)$$

where $\mathbf{p}$ denotes the class probabilities estimated by the model.

The Class-Balanced Focal (CB-Focal) loss is the combined version of Focal loss and Class-Balanced loss. In this case, $L(\mathbf{p}, j)$ in Equation 2.6 is replaced by L_{focal} from Equation 2.4. This loss weights the importance of each instance on the loss from the perspective of both class distribution and sample easiness.

2.6.3 Two-stage training

The extensive experiments conducted by Kang et al. [76] on common long-tail benchmarks showed that data imbalance is not an issue for learning high-quality representations. Their results evidenced that it is possible to attain a good long-tail recognition score by training only the classifier on top of those learned representations. Taking this into account, Kang et al. [76] propose a two-stage training method for long-tail recognition. In the first stage, the model is trained using the original unbalanced training distribution. In the second, only the classifier is trained using a balancing method, such as re-weighting or re-sampling.

2.6.4 Multi-branch networks

Methods using multi-branch networks handle the long-tail problem by creating expert branches to treat relatively balanced sub-groups of classes separately. The Bilateral Branch Network (BBN) [189] uses two backbone branches, one is trained with a uniform sampler and the second one uses a sampler with a reversed distribution w.r.t. the original long-tail distribution; a cumulative learning strategy is then used to combine their output feature vectors. Ally Complementary Experts (ACE) [22] works by using multiple expert branches on top of a shared backbone, in which each expert has individual learning blocks and a prediction layer for a diverse but overlapping set of categories. Then, the experts' predictions are aggregated by averaging the logits that are re-scaled according to the data splits. The main idea is that the dominating categories (head classes) for each expert are different, and the overlapping categories support each other during the aggregation, especially for categories with few samples. Balanced Group Softmax (BAGS) [87] also uses a shared backbone for representation learning, but the categories are divided into disjoint groups according to the number of instances.

BAGS is trained in a two-stage fashion. First, the model is trained using a single softmax classification head containing all categories. Then, the classification head is replaced by the new groups of classification heads randomly initialized. Finally, only these heads are trained, with all other weights frozen. Each category j is assigned to a single group G_k based on its number of training instances n_j if $s_k^l \leq n_j < s_k^h$, with $k > 0$, where s_k^l and s_k^h are the minimum and maximum number of instances for the group G_k and $s_{k+1}^l = s_k^h$. Li et al. [87] split the categories into four groups, with s^l and s^h set to $(0, 10)$, $(10, 10^2)$, $(10^2, 10^3)$, and $(10^3, +\infty)$, and the special group G_0 for foreground/background classification. Each group includes an extra category *others* for the samples from classes belonging to the other groups. To balance training,

the *others* category within each batch was undersampled by a factor of $\beta \times n_k$, where n_k is the number of instances from categories belonging to the group in that batch ($\beta = 8$ was used). During the inference, a softmax is applied to each group and the probabilities are remapped back to the original category order and rescaled by the foreground probability. By this design, classes with a significantly different number of instances are isolated from each other during the training, thus avoiding the tail classes being substantially suppressed by the head classes.

2.7 Multi-object Tracking Methods

Multi-object tracking is a task in computer vision that aims to detect all objects in a video, associate them across frames, and assign a unique identity to each object in the scene. The most effective paradigm in the current state-of-the-art is the tracking-by-detection approach, in which an object detector generates bounding boxes for each frame of a video sequence, and a tracking algorithm is then responsible for associating the boxes that belong to the same object [186]. These methods must handle several challenges, such as occlusions and objects leaving and re-entering the scene. SORT [174] is one of the most representative methods of the tracking-by-detection paradigm, with more recent approaches building upon it, such as ByteTrack [186] and BoT-SORT [1]. PuTR [94] also follows the tracking-by-detection paradigm but employs a transformer to learn the association of objects across frames.

2.7.1 SORT

Simple Online and Realtime Tracking (SORT) [174] focuses on efficiently associating objects in online and real-time applications. It operates by presenting to the tracker only detections from consecutive frames. At each time step, the tracker receives bounding boxes from the current frame and matches them to existing tracks based on predictions from prior frames. SORT relies on computationally efficient classical methods, namely the Kalman filter for motion prediction and the Hungarian algorithm for data association. This combination enables fast and lightweight tracking while maintaining competitive performance, making SORT particularly suitable for real-time scenarios.

2.7.2 ByteTrack

Most trackers associate detections by considering only those with confidence scores above a predefined threshold, discarding low-confidence detections (e.g., from occluded

objects). This often results in a non-negligible number of missed objects and fragmented tracks. ByteTrack [186] addresses this limitation by introducing a simple and effective association method, called BYTE, which leverages nearly all detections. In this approach, high-confidence detections are associated in a first pass, while low-confidence detections are used in a second pass to recover tracks that were not matched initially.

2.7.3 BoT-SORT

While SORT relies solely on motion features for data association, DeepSORT [174] extends this approach by incorporating visual (ReID) features extracted from an appearance embedding model, improving robustness in challenging scenarios. Building on these ideas, BoT-SORT [1] adopts ByteTrack’s two-pass association framework and enhances it by integrating appearance (ReID) features to improve matching accuracy. Additionally, it introduces further refinements, including camera motion compensation and a more accurate Kalman filter state representation, resulting in more robust tracking in dynamic scenes.

2.7.4 PuTR

Pure Transformer (PuTR) [94] is a multi-object tracker that models trajectories as directed acyclic graphs (DAGs), enabling the representation of object sequences across frames through a binary adjacency matrix. This formulation allows trajectories to be learned using a pure Transformer architecture, where each detection is represented as a token. The attention mechanism employs a causal mask, restricting each token to attend only to detections from previous frames. Additionally, PuTR assumes that detections within the same frame cannot belong to the same trajectory. While tokens are temporally ordered across frames, they remain spatially unordered within each frame. To account for this structure, the positional encoding is adapted to incorporate both temporal and spatial information. Figure 2.9 provides a schematic illustration of the PuTR framework.

To train PuTR, the affinity matrix is generated through the inner product between the representations of object tokens in the current image and those in previous images, followed by the application of a softmax function to obtain a probability distribution that maps each object in the current frame to those in previous frames. At inference time, PuTR employs a runtime manager and the Hungarian algorithm to resolve assignment conflicts produced by the learned association module, which gener-

ates the adjacency matrix. Additionally, this affinity matrix is enhanced with spatial and geometric compensations.

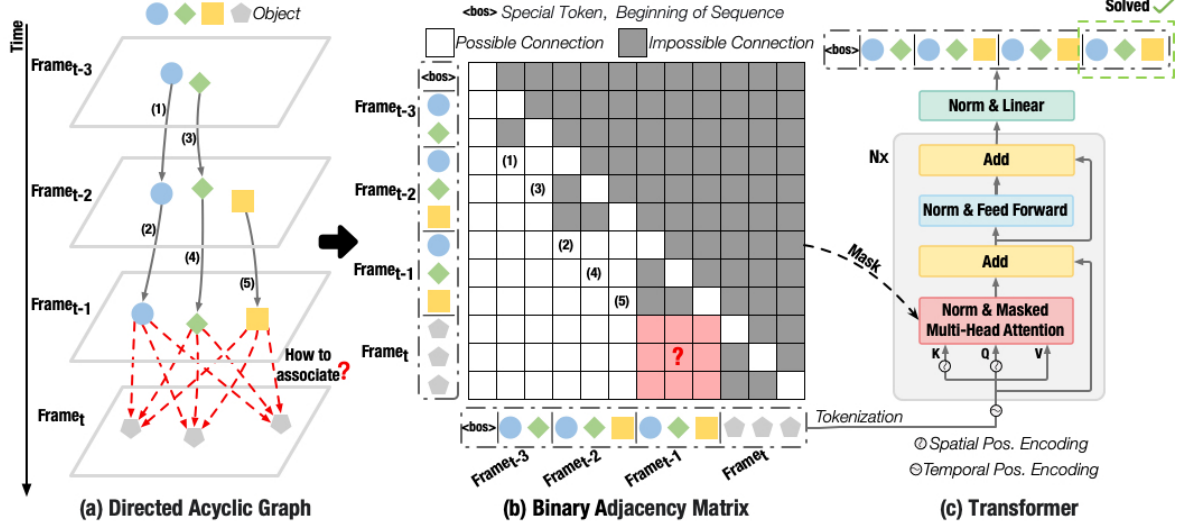


Figure 2.9. The main idea of PuTR is to model detections as a directed acyclic graph (a), where each detection is connected to detections in previous frames using a binary adjacency matrix (b). This matrix is produced by a transformer that leverages both spatial and temporal positional encodings with a causal mask, ensuring that each bounding box can only be associated with boxes from past frames.

Source: Liu et al. [94].

2.8 Evaluation Metrics

This section describes the metrics used to evaluate the methods proposed for the animal recognition tasks in camera-trap images addressed in this work. These include metrics for classification problems, with a focus on specific contexts such as long-tail visual classification and empty frame filtering. Additionally, we describe metrics for animal counting.

2.8.1 Classification Metrics

There are several measures to assess the performance of classification models. In this work, we used metrics obtained from the confusion matrix, which indicates the number of correctly and incorrectly classified instances for each class [139], as shown in Table 2.2. True positive (tp_i) refers to the number of correct predictions for class i . False positive (fp_i) refers to the number of instances incorrectly classified as belonging to class i . False negative (fn_i) refers to the number of instances from class i that are

wrongly classified as not belonging to class i . True negative (tn_i) indicates the number of instances correctly classified as not belonging to class i .

Table 2.2. Confusion matrix for binary classification of a given class i

		Predicted	
		Positive	Negative
Actual	Positive	tp_i	fn_i
	Negative	fp_i	tn_i

Accuracy: Accuracy is the most commonly used metric to evaluate classification performance, defined as the number of correct predictions divided by the total number of instances:

$$Acc = \frac{tp + tn}{tp + fp + fn + tn}. \quad (2.7)$$

This metric does not distinguish between the number of correct predictions among different classes, nor does it favor any particular application [139]. Top-k accuracy is a widely used variation, where a prediction for an instance is considered correct if the true label is among the top k highest predictions given by the model [122].

Binned accuracy: Another variation is binned accuracy, where the categories are grouped into bins according to specific criteria, and accuracy is reported for each group. For example, Li et al. [87], when evaluating long-tailed datasets, group the categories into four bins based on the number of training instances, where Bin_i contains the categories composed of 10^{i-1} to 10^i instances.

Recall: Recall measures the capacity of a model to correctly classify the instances of a given class i , calculated as the ratio of correctly classified instances (tp_i) to the total number of instances (tp_i and fn_i) belonging to class i :

$$Recall_i = \frac{tp_i}{tp_i + fn_i}. \quad (2.8)$$

Precision: Precision measures the confidence of a model when classifying an instance as belonging to class i , and it is calculated by dividing the number of correctly classified instances (tp_i) belonging to class i by the number of all instances classified (tp_i and fp_i) as belonging to class i :

$$Precision_i = \frac{tp_i}{tp_i + fp_i}. \quad (2.9)$$

True Negative Rate: The True Negative Rate is the Recall of the negative class. Instead of focusing on the positive class, it measures the proportion of negative instances correctly classified:

$$TNR_i = \frac{tn_i}{tn_i + fp_i}. \quad (2.10)$$

It can be used, for example, in the problem of filtering empty camera-trap frames, where images with animals are the positive class, but we want to measure the model’s effectiveness in correctly removing frames without animals.

F1-Score: As accuracy tends to bias the results in favor of majority classes, when evaluating problems that follow the long-tail distributions observed in nature, a more suitable metric is the macro-averaged F1-score [112]. The F1-score F_{1_i} of a class i represents a trade-off between precision p_i and recall r_i , and is calculated as their harmonic mean:

$$F_{1_i} = 2 \times \frac{p_i \times r_i}{p_i + r_i}. \quad (2.11)$$

The macro-averaged F1-score F_1^m across C classes is the mean of the class-wise F1-score:

$$F_1^m = \frac{1}{C} \sum_{i=1}^C F_{1_i} = \frac{1}{C} \sum_{i=1}^C \left(2 \times \frac{p_i \times r_i}{p_i + r_i} \right). \quad (2.12)$$

2.8.2 Precision-Recall Curve

The Precision-Recall curve is a graphical representation of a binary classifier’s performance across multiple thresholds, with recall values plotted on the x-axis and corresponding precision values plotted on the y-axis [123]. The area under the Precision-Recall curve (PR AUC) is an effective metric for comparing the performance of multiple classifiers [36].

2.8.3 Counting Metrics

To assess the task of counting animals in a sequence of camera-trap images, we follow the approach proposed by the iWildCam 2021 Competition [11], where the prediction is given by the count of individuals for each species, and the model is evaluated using the *mean columnwise root-mean-squared error* (MCRMSE). Let $\mathbf{Y} \in \{0, 1, 2, \dots\}^{n \times m}$ be the matrix of ground truth counts, where each entry y_{ij} corresponds to the count for species $j \in \{1, \dots, m\}$ in sequence $i \in \{1, \dots, n\}$. Similarly, let $\hat{\mathbf{Y}} \in \{0, 1, 2, \dots\}^{n \times m}$

represent the matrix of predicted counts. The MCRMSE is given by:

$$MCRMSE = \frac{1}{m} \sum_{j=1}^m \sqrt{\frac{1}{n} \sum_{i=1}^n (y_{ij} - \hat{y}_{ij})^2}. \quad (2.13)$$

This metric accounts for both the species identification and counting errors, ensuring that false predictions on empty sequences also contribute to the overall error.

2.9 Final Remarks

This chapter presented the fundamental concepts related to the methods and approaches applied and developed throughout this work. In particular, we described key computer vision architectures, training strategies, and data processing techniques, which are adapted to better meet the specific needs of tasks in automated camera-trap image processing. We also presented state-of-the-art methods relevant to the specific tasks addressed in this work, including model optimization strategies aimed at improving performance on edge devices, as discussed in Chapter 4; techniques for handling long-tail distributions – particularly pertinent to Chapter 5; and multi-object tracking methods used in our proposed counting-by-tracking approach in Chapter 6.

These concepts are essential not only for understanding the methods used but also for contextualizing the related work. In the next chapter, we review existing literature on the selected tasks in automated camera-trap image processing covered in this work and highlight the research gaps our contributions aim to address.

Related Work

This chapter reviews related work on automated camera trap processing tasks addressed in this research, identifying limitations in the literature that motivate the contributions of this work. We begin by discussing methods for filtering empty frames. Next, we examine the automated species identification task, considering multiple aspects such as class imbalance, generalization, and strategies for improving performance. We then explore approaches that leverage all images within a sequence for species identification. Subsequently, we review the literature on automated counting of individuals in camera trap images using computer vision techniques. Finally, we discuss methods for automated preprocessing on edge devices, independent of the specific task being performed.

3.1 Filtering empty camera-trap images

Empty or blank frames represent a significant portion of camera trap data – for instance, they comprise approximately 76% of the Snapshot Serengeti dataset [89] – and are often useless. Filtering them out can reduce human workload [178], minimize data transmission over limited-bandwidth networks [42, 193], and save storage, extending the duration a camera can remain in the field without SD card retrieval [161].

This task has been studied both as an additional category in classifiers, alongside species classes [14, 130, 149, 163], and as a two-stage approach [106, 107, 148, 173] where the first stage is responsible for recognizing the presence of animals and the second for identifying the species. Several tools have been developed for filtering empty images [13, 72, 167, 184], and the challenge has been explored from multiple perspectives [4, 5, 9, 108, 178, 179, 180].

Classification approaches. Norouzzadeh et al. [107] trained nine models to detect images containing animals in the Snapshot Serengeti dataset, with all models achieving an accuracy greater than 95.8%, and VGG achieving the highest accuracy of 96.8%. Tabak et al. [149] developed the R package *Machine Learning for Wildlife Image Classification* (MLWIC), which includes an empty class among the species categories and correctly classified 85.1% of the empty images in the Snapshot Serengeti. In its later version, MLWIC2 included a specific model for filtering empty images, achieving an accuracy of 94.1% on Snapshot Serengeti. Vargas-Felipe et al. [158] used an SVM on top of embeddings generated by a pretrained ResNet50, achieving an accuracy of 99.95% for the binary classification of empty frames and desert bighorn sheep.

Background-based and hybrid methods. Zilong [167] is a tool based on a non-machine learning algorithm developed to identify empty camera trap images based on color and edge changes among images in a sequence. It correctly identified 85% of empty images and retained 87% of animal images in the Snapshot Serengeti dataset, compared to 69% and 65%, respectively, achieved by MLWIC. However, the authors recognize that a major limitation of the method is that it cannot identify empty images when there is vegetation swinging. EventFinder [72] is a similar tool based on background subtraction for identifying empty camera trap images. The authors report an accuracy of 96.1% on their private dataset and a 90% reduction in human workload. Other studies also rely on background subtraction and traditional computer vision techniques using hand-crafted features to detect animals [50, 91, 103, 118, 183, 188].

Animal Scanner [184], in contrast, combines background subtraction with a deep learning image classifier to classify a frame into three classes: human, animal, or empty. In their study, Yousif et al. [184] also analyze the effects of gradually reducing the model’s input size and the number of filters on the trade-off between model performance and inference time.

Detection-based approaches. A widely used model is MegaDetector [13], a generalist object detector trained to localize animals, people, and vehicles. It helps users identify animals but does not classify them at the species level, being effective for filtering empty images [85]. Fennell et al. [44] demonstrated the effectiveness of integrating MegaDetector into existing camera-trap workflows, reducing manual processing time by 8.4×. The early versions of MegaDetector were based on Faster-RCNN, which was computationally intensive. However, to develop more lightweight models, the architecture was changed from version 5 onwards to be based on the YOLO architecture family.

Örn [108] compared a DenseNet201 classifier, YOLOv3, and MegaDetector on camera-trap images obtained from the Cairngorms National Park, United Kingdom,

categorizing them as empty, animal, or human, with MegaDetector performing best.

Learning under data constraints. While detection-based and classification models achieve strong performance, they often rely on large-scale annotated datasets. To address this limitation, several works focus on improving performance under limited or imbalanced data conditions.

Yang et al. [180] used an ensemble based on AlexNet, InceptionV3, and ResNet18 to automatically identify empty images from cameras in Lhasa Mountains, China. They trained each architecture on two versions of the dataset: one using the original distribution and one with a balanced distribution. They also used thresholding to reduce classification errors. In a complementary study, Yang et al. [178] studied the impact of the class imbalance problem on model performance for identifying empty camera trap images using AlexNet architecture on Snapshot Serengeti dataset. They found that when the empty image rate was between 10% and 70%, there was no significant impact on the results. However, for rates higher than 70%, they found that oversampling helped reduce the error associated with the non-empty images being classified as empty.

Auer et al. [9] used an active learning approach to train location-specific models for daytime and nighttime to detect empty camera trap images. These models outperform MLWIC by over 10%. While MegaDetector achieves 5% higher accuracy, the active learning models use only 150 to 160 annotated training images, whereas MegaDetector is trained on millions.

Alencar et al. [4] employed a Siamese network to filter empty camera-trap images based on a reference image in a scenario where such a reference is obtained regularly. This approach achieves accuracy levels that are 3% to 8% higher than its counterpart using a non-siamese binary classifier with the same backbone. In a subsequent work, Alencar et al. [5] investigated large-scale multimodal language models in zero-shot and few-shot learning scenarios, achieving promising results. Few-shot learning using BLIP [86] outperformed a binary classifier based on ResNet-50, but its performance was still lower than that of the Siamese version.

Our approach. Similar to the work of Yousif et al. [184], in Chapter 4, the problem of filtering empty frames is approached from the perspective of the trade-off between model performance and inference time. However, while they analyze this trade-off for desktop computers, our approach focuses on embedding the models on edge devices and includes model optimization analysis. As in the work of Örn [108], we also compare the performance of object detectors and classifiers.

As detailed in Section 3.5, some studies explore empty camera-trap image filtering directly on edge devices. However, their focus is on validating applications on the

edge, whereas our study aims to assess the trade-off between inference latency and performance. In Chapter 5, when studying the long-tail distribution, and in Chapter 6, within our counting strategy, we consider the empty class as an additional category on the same classification head, alongside the animal classes.

3.2 Species identification

Species identification in camera trap images is the most extensively studied task in the automated analysis of camera-trap data. Early approaches used classical computer vision techniques, such as hand-crafted features and shallow machine learning classifiers [46, 49, 144, 172, 185].

With the advent of deep learning, various aspects of this problem have been explored in the literature, including the effectiveness of transfer learning for model training, the impact of training data size, and the extreme class imbalance typical of biodiversity datasets.

Given the static nature of camera traps, studies have also examined the ability of models to generalize to new locations beyond those seen during training. Specific approaches have been investigated to improve classification accuracy, such as localizing and cropping animals before classification, as well as techniques like confidence thresholding to improve prediction reliability. Additionally, efforts have been made to develop models that can run efficiently on devices with limited computational resources and to create practical applications that allow ecologists to identify animal species without requiring expertise in model training or computer programming.

Transfer learning approaches. Transfer learning has been explored to train models with limited labeled data. In one of the first works applying modern deep architectures for animal species classification in camera-trap images, Villa et al. [163] used transfer learning by freezing the backbone of ImageNet-pretrained models using them as feature extractor and training only the classifier. They experimented with AlexNet, VGG, GoogLeNet and ResNet to recognize species in the Snapshot Serengeti dataset. They also explored fine-tuning, allowing the pretrained backbone to be further trained with camera-trap data, achieving around a 5% improvement over training only the classifier head.

In their experiments, Norouzzadeh et al. [107] found that transfer learning with ImageNet-pretrained models had only a small positive impact compared to training from scratch on Snapshot Serengeti. However, they recognized the benefits of using transfer learning for smaller datasets.

Instead of using ImageNet-pretrained models, Willi et al. [173] pretrained their models on Snapshot Serengeti and applied transfer learning to other camera-trap projects – Camera CATalogue, Elephant Expedition, and Snapshot Wisconsin – achieving better results than training from scratch. Transfer learning across camera-trap datasets is also explored by Norouzzadeh et al. [106]. In recent years, transfer learning with pretrained weights, either from the ImageNet or COCO datasets, has become a common approach to reducing training time and improving performance [14, 40, 45, 48, 102, 105, 109, 110, 111, 130, 134, 147, 169, 192, 193]. We also adopt this strategy in this work, initializing our models with ImageNet weights.

Impact of training data size. Schneider et al. [130] point out that previous researchers trained models using a massive number of labeled images, such as those available in the Snapshot Serengeti dataset, but this approach would be impractical for the vast majority of small-scale projects. Their work evaluates whether deep learning models trained on modestly sized training datasets can succeed and analyzes their performance relative to the amount of training data.

Willi et al. [173] also assess the impact of training dataset size on model performance, finding that transfer learning can benefit smaller datasets. While using the full Camera CATalogue dataset resulted almost the same accuracy when training from scratch or with transfer learning, when using only 12.5% of the data, transfer learning resulted in an accuracy 10% higher. Shahinfar et al. [134] evaluated camera-trap training sets with 10, 20, 50, 150, 500, 1000 images per class and found that precision and recall metrics scale logarithmically with the number of training images. Pantazis et al. [109] assess the impact of training set size on ecological metrics, finding that while smaller training sets can significantly affect results, some metrics can still be accurately estimated with less data.

Data-efficient learning. Some studies have explored active learning approaches to minimize the number of images needed to train models – reducing the manual labeling effort – while maintaining high model performance in camera-trap image analysis [20, 102, 106]. Pantazis et al. [110] studied self-supervised learning to reduce the amount of labeled data required for training models on camera-trap datasets. In order to consider the scenario of smaller-scale projects, in Chapter 5, we evaluate the methods using multiple datasets with varying numbers of training images.

Class imbalance. The extreme class imbalance resulting from a long-tail distribution is another challenge in camera-trap data, as it is both prevalent and severe, as shown by He et al. [59], and explored in the context of species identification. Villa et al. [163] compared model performance using the original data distribution and a fully balanced dataset, with the latter achieving accuracies 10% higher. However, in

their experiments, Villa et al. [163] discarded species with fewer than 1,000 images, which is problematic since rare species may be of particular ecological interest.

Norouzzadeh et al. [107] found that their models obtained extremely low accuracy on rare classes. To mitigate this issue, they tried three approaches: weighted loss, oversampling, and emphasis sampling. They found that all three methods improved accuracy for some rare classes but not for all, highlighting the need for further research on the topic.

To handle data imbalance in their experiments, Schneider et al. [130] added artificially augmented images to the training set for rare species to ensure at least 100 images per class. They also applied a heuristic sampling method to select training images, which, combined with the augmented images, can be seen as an oversampling technique. Despite this approach, they found that species with fewer than 500 training samples often exhibited poor recall.

Beery et al. [12] generated synthetic training examples using the game engine Unity and the simulation platform Microsoft AirSim to improve the models' performance on rare classes, achieving a reduction of 10% in error rate for the class deer by adding only 5 synthetic examples, at the cost of a $<0.5\%$ increase in average class-wise error. Synthetic training examples generated using Unity were also used by Straka et al. [141] to improve pose estimation performance for *Lynx lynx* in camera trap images.

Böhner et al. [18] included images from other datasets to increase the number of training images of rare species; however, the training set was still unbalanced. To handle the long-tail distribution in their data, Miao et al. [102] used the Open Long-Tailed Recognition [97] method to balance learning between head and tail categories. Other studies have not applied any specific techniques to address class imbalance [14, 40, 48, 105, 149, 169, 173, 192, 193].

In Chapter 5, similar to Norouzzadeh et al. [107], we directly address the long-tail issue using state-of-the-art methods to improve performance on tail classes. However, we propose a new method that better balances the trade-off between improving tail class performance and preserving head class accuracy.

Generalization across locations. Another challenge arises from the fact that camera traps are static, retaining the same background for images taken by each camera, which may bias models toward trained locations. However, a key capability desired in models for the automated analysis of camera-trap data is the ability to generalize to new locations beyond those seen during training [130]. In fact, as demonstrated in the literature [14, 105, 130, 149], performance is poor when applied to untrained locations, especially for species identification models. Beery et al. [14] presented the Caltech Camera Traps dataset, designed to measure generalization to new environ-

ments through a location-based train/test split, where images from the same location were kept in either the train or test set. This differs from other works, where the train/test splits were done randomly [149, 163] or by capture event [102, 107, 173].

Splitting the dataset randomly or by capture event can lead to data leakage in the test set, as images taken within short time intervals may be near-duplicates. To avoid this, Beery et al. [14] ensured that, when testing on images from locations used during training, the training set included images from even days, while the test set contained only images from odd days. In recent years, the location-based or temporal-based train/test splits have become the standard for evaluating classification models on camera-trap data [12, 15, 18, 40, 41, 48, 101, 105, 109, 110, 130, 162, 169]. In this work, we adopt location-based splitting to validate our models.

Large multimodal language models for generalization. Recent works have explored the use of large multimodal language models to address generalization in camera trap images, particularly under domain shifts caused by unseen locations or species. These approaches are often evaluated in open-world scenarios, including few-shot and zero-shot learning settings. WildCLIP [47] builds upon CLIP by fine-tuning it to retrieve scenic, behavioral, and species-level attributes from camera trap images. Similarly, Eco-VLM [181] adapts CLIP using an ecologically curated subset of ImageNet-1K, improving performance in zero- and few-shot classification tasks. BioCLIP [140] is a vision foundation model for biology trained using taxonomic information, enabling fine-grained classification and zero-shot recognition of unseen taxa.

Santamaria et al. [125] introduced the Camera Trap Language-guided Contrastive Learning (CATALOG) model, which integrates multiple foundation models through multimodal fusion, including GPT-3.5, CLIP, LLaVA, and BERT. By jointly leveraging visual and semantic representations, CATALOG learns more robust and transferable features, improving performance under domain shifts in camera trap image recognition.

Pre-cropping animals. A common issue in camera trap images is that animals appear too small when they are far from the camera. Some studies have addressed this by cropping bounding boxes around the animals to highlight them. Villa et al. [163] analyze this issue using a manually checked subset containing only animals in the foreground, achieving accuracies 10% to 15% higher compared to using all images.

Beery et al. [14] compare classifiers trained on full images and cropped bounding boxes, with the latter achieving accuracies around 10% to 20% higher than the full-image model. Norman et al. [105] also found that cropping images to bounding boxes greatly improves model performance, with an F1-score of 0.77 versus 0.47. These results motivated others to train their models directly on animal crops generated by object detectors rather than full images, minimizing background influence and reducing bias

toward training locations [20, 41, 106, 109, 110, 125].

Gadot et al. [48] showed that incorporating an object detector into a classification pipeline improves performance compared to a whole-image classifier, using a dataset of 42 million camera-trap images as well as smaller datasets such as iWildCam.

In this work, as presented in Chapter 5, we also conduct a study comparing the effectiveness of using MegaDetector as a pre-cropping strategy versus using the full images.

Confidence thresholding and calibration. Another investigated approach to improving classification performance is confidence thresholding, which selects high-confidence predictions while reducing coverage – i.e., the number of images for which the model provides predictions. For example, this can reduce human effort by sending only low-confidence predictions for manual review. Norouzzadeh et al. [107] find that using a 43% confidence threshold allows them to automatically process 97.2% of images from the species identification task at human-level accuracy. Willi et al. [173] report accuracies between 96.7% and 98.9% with a 99% confidence threshold, while coverage remained between 76% and 86.5%. Whytock et al. [169] evaluate several thresholds, finding that for their models, a threshold of $>70\%$ confidence was required to achieve an accuracy of 90% or more, with $>25\%$ of data being discarded.

In chapter 4, we evaluate the model’s performance using a precision-recall curve to assess its accuracy across various confidence thresholds, in line with other works in the literature [45, 162]. To improve the robustness of ecological analysis, some studies use a fixed confidence threshold based on their data to filter out uncertain predictions [18, 52, 98, 101, 109]. Dussert et al. [41] studied the calibration of models through temperature scaling for species classification in camera trap images, in order to allow the interpretation of confidence scores as probabilities.

Model efficiency. Considering practical use cases, some studies have explored more efficient models in their research, for example, recognizing that end users may not have access to powerful computers to run computationally intensive models or that some applications require models to run on edge devices for in-field inference. Norouzzadeh et al. [107] evaluate several models, including ResNet18 – a smaller version of the well-known ResNet family – which achieved only 0.5% lower accuracy than the best-performing model in their experiments. This result motivated other authors to adopt ResNet18 in their studies to balance accuracy and computational costs [110, 148, 149, 173]. With the same purpose, Schneider et al. [130] included in their experiments architectures specifically designed for constrained computational resources: NASNetMobile, MobileNetV2, and Xception.

On the other hand, when high-performance computing resources are available,

larger models or ensembles can provide improved classification results [48, 107, 130]. Pantazis et al. [109] found that the choice of model architecture has minimal impact on the ecological metrics used in their study, despite differences in machine learning performance metrics. Other works [73, 192, 193] have also explored species identification on edge devices by using smaller models and optimizing them for embedded systems, as discussed in more detail in [section 3.5](#).

In this work, along with state-of-the-art models, we also include architectures that are computationally efficient for both edge devices and low computational power, considering scenarios where high computational power is not available.

User-oriented tools. Efforts have been made to make computer vision models accessible to users with limited experience in Python or deep learning model training. Tabak et al. [149] developed MLWIC (Machine Learning for Wildlife Image Classification), an R package designed to be accessible to ecologists, allowing them to apply trained models to their datasets and train their own models. They later released an improved version, MLWIC2 [148]. Their next iteration, CameraTrapDetectoR [147], was based on the Faster R-CNN object detector and enabled not only species classification but also direct bounding-box detection within R. Similarly, Böhner et al. [18] proposed an AI-assisted semi-automatic workflow in R for processing camera-trap images, making advanced computer vision techniques more accessible to ecologists. Concerned on respecting human privacy, Fennell et al. [44] developed a tool to blur human-classified bounding boxes based on MegaDetector outputs.

3.3 Classifying capture events

Most camera trap projects capture more than one picture per sensor trigger, hoping to get a good shot to identify the passing animal. Usually, most works in the literature treat each picture in the sequence as an independent sample during both training and testing, as this approach allows the use of standard computer vision architectures and evaluation metrics without additional post-processing. An alternative to improve species identification is to consider all images in the capture event to identify the animal present.

To classify capture events, Norouzzadeh et al. [107] used two approaches: aggregating predictions from each image within a sequence, and training a classifier to directly predict capture events. In the aggregation approach, the vector predictions of images within a sequence were averaged, resulting in an accuracy score on average 1% higher than when using individual images. In the second approach, they trained a

model based on ResNet50 using a 9-channel input image, where three images from a capture event were stacked on top of each other. For this case, only capture events with exactly three images were considered (around 55,000), achieving an accuracy slightly lower than that of the individual image model, likely due to the reduced training set size.

Beery et al. [14] used two approaches for sequence classification: most confident, where the prediction for a capture event is taken from the most confident prediction among the images in that event; and oracle, where a capture event is considered correctly classified if any of its images are correctly classified—an optimistic evaluation. In all experiments, using sequence information outperformed individual image predictions, reducing the error rate by between 1% and 12.35%. The most confident approach was also adopted by Willi et al. [173]; however, their work reports results only at the capture event level, without comparing them to individual image predictions.

Dussert et al. [41] also explored the most confident and average prediction approaches. A key distinction in their work is that they evaluated both methods using softmax scores and logits, applying the softmax function after aggregation in the latter case. Sequence-based predictions consistently outperformed single-image predictions, with the average approach achieving the best results – improving accuracy by an average of 3.71% with softmax scores and 3.56% with logits.

In their turn, Shashidhara et al. [135] explored training models to incorporate temporal information from capture events by stacking channels, similar to Norouzzadeh et al. [107], but with additional channels containing sequence information generated by computer vision algorithms such as MOG2 [191] and Optical Flow [43]. Their results indicate that using concatenated channels with temporal information led to significant improvements – ranging from 10% to 20% – over single-image models. However, their work lacks comparison with simpler methods, such as aggregating scores.

Beery et al. [15] proposed Context R-CNN, an approach based on a “memory bank” to integrate short- and long-term sequence information for improving animal detection in the current frame. Even though this method uses temporal information across sequences, it is not directly applied to species identification at the capture-event level.

In our iWildCam 2021 solution (as described in Appendix A), we evaluated two approaches: majority voting over image-level predictions within the same sequence and averaging predictions across the sequence. As averaging achieved better results, we adopt this strategy in Chapter 6, where it is used to improve species classification by leveraging all images within a sequence in the context of counting.

3.4 Counting animals

In recent years, some efforts have been made to address the task of counting animals in camera trap images. Norouzzadeh et al. [107] treated animal counting as a classification problem, where each image was assigned to one of 12 possible bins representing the range of individuals present. The authors recognized that object detection provides a natural mechanism for counting but justified their design choice of using classification due to the lack of bounding box labels in the Snapshot Serengeti dataset, which contains only count labels at the image level.

In a subsequent work, Norouzzadeh et al. [106] used a pre-trained object detector to generate bounding boxes on camera trap images and counted the number of animals in an image based on the number of bounding boxes with a confidence higher than 90%. With this approach, they were able to achieve a slight improvement compared to the previous classification approach, but without using any labeled image from the target dataset. Kilima et al. [78] used YOLOv4 to count animals in single camera-trap images using a small subset of Snapshot Serengeti. However, they did not compare their results when using more images for training.

In 2021, the iWildCam competition proposed a new challenge to encourage the development of approaches for counting individuals across camera trap image sequences, providing a dataset for this problem [11]. The challenge explored the scenarios where count labels are not available, which is the case for most publicly available camera trap datasets, promoting alternative methods for counting animals, with labels being collected only for the test set. All the top solutions used heuristics to count animals based on the maximum number of bounding boxes detected across any image in the sequence, including our solution described in Appendix A, which won the challenge. This approach is also applied by Simões et al. [136].

In the 2022 edition, the iWildCam challenge re-focused on handling only the task of counting the number of animals present in sequences of images, as it is considered a particularly challenging problem. For this new version of the challenge, Wang et al. [166] proposed two methods for counting animals: FilterDetector and DLEDetector, which achieved first and third place, respectively, in iWildCam 2022. Both methods focused on improving the detection of animals in images and, consequently, enhancing the overall counting. However, these methods still rely on the heuristic of using the maximum number of animals detected across images in the sequence as the counting prediction.

Multi-object tracking provides a natural mechanism for counting animals by associating detections across images in a sequence, as seen in other domains such as

pig counting using top-down view cameras [27, 68, 79], fish counting with underwater cameras in fishways [175], and fish counting in sonar videos [77].

However, the low frame rate of camera trap image sequences – typically just one frame per second – poses challenges for applying state-of-the-art tracking methods. In this context, Arshad et al. [8] developed an edge device for monitoring animals, focusing on deer, including a tracking algorithm for counting. Nevertheless, their approach uses videos rather than image sequences and lacks evaluation on standard datasets, making it difficult to assess its effectiveness.

More recently, Schall et al. [127] introduced GorillaWatch, an end-to-end system integrating detection, multi-object tracking, and individual re-identification for in-the-wild gorilla monitoring. Their work demonstrates the potential of exploiting temporal correspondence and individual-level representations for wildlife monitoring and counting. While their work focuses primarily on video sequences of a single species and relies on re-ID for counting, our approach is complementary: we investigate sequence-level counting in low-frame-rate camera traps, leveraging tracking as an explicit intermediate representation for count estimation.

In Chapter 6, we present a strategy built upon a transformer-based tracking model for counting animals. Specifically, we use a transformer to learn associations between detections, with the goal of handling challenges inherent to this type of data, such as low frame rates and large displacements across images. To mitigate the lack of training labels, we also propose a method for generating a synthetic dataset.

3.5 Edge AI for camera traps

Various studies have investigated the use of intelligent camera traps that embed deep learning models on board to identify animals directly on the edge [6, 8, 35, 38, 42, 73, 104, 156, 161, 164, 168, 192, 193]. This approach can provide significant advantages. For example, identifying species on the edge can be useful for issuing real-time alerts to mitigate human-wildlife conflicts [38] or enable rapid responses to events, such as poaching or feral cat control programs [138, 161], especially for those cameras connected to networks with limited bandwidth.

Similarly, embedded models can perform prior image filtering to prevent the transmission of unnecessary images, thereby saving network bandwidth and energy [42, 193]. Filtering can also help save data storage, extending the duration a camera can remain in the field collecting images without requiring SD card retrieval [161].

Elias et al. [42] developed an IoT system composed of camera traps connected

to a private network for performing classification and filtering at the edge in order to reduce network transfer over slow networks by around 70%. However, the model does not run on the camera, but on a compute node located near the deployment sites in the private network to which the cameras are connected. Dertien et al. [38] report the deployment of TrailGuard AI – an AI-enabled onboard camera trap – to monitor tigers and review use cases of this technology for enabling real-time alerts to mitigate human-wildlife conflict, filtering the species of interest on-board. Curtin and Matthews [35] developed a prototype of a low-cost \$50.00 camera for wildlife monitoring based on the Raspberry Pi 3B+, featuring a custom on-board convolutional neural network to identify snow leopards, humans, or empty images.

AlHammadi et al. [6] use the Forward-Forward (FF) [63] algorithm to train camera-specific models directly on the camera itself to distinguish between empty and nonempty images. While FF requires less memory during training compared to back-propagation, its training time was more than twice as long, and the performance was 4% lower.

Velasco-Montero et al. [161] developed a smart camera trap using a Raspberry Pi 4B to filter empty images and identify feral cats. They trained a model based on SqueezeNet [69] and used OpenCV and TensorFlow Lite for inference. To improve the model performance on-site, they retrained the model directly on-board using background images captured at the deployment site and synthetic training data generated from those backgrounds. They report that their approach achieved an F1-score of 10% higher than MegaDetector.

In a scenario where only predictions and timestamps are transmitted over the network, Zualkernan et al. [193] evaluate several computer vision architectures for classifying animal species from camera trap images directly on a Raspberry Pi 4B connected to a commercial camera trap via a Wi-fi enabled SD card. They also assess the prediction latency of InceptionV3, their best-performing model, after converting it to TensorFlow Lite. In a subsequent work, Zualkernan et al. [192] expanded their analysis by including more architectures and, in addition to Raspberry Pi 4B, embedded the models on Google Coral and Nvidia Jetson boards, assessing prediction latency, current consumption, and CPU utilization.

Also using a Wi-Fi-enabled SD card, Whytock et al. [168] connect a commercial camera trap to a Raspberry Pi that uses an embedded model to classify animals and transmit alerts over the Iridium satellite network. Nazir and Kaleem [104] investigated the classification of animal species on a low-power customized camera trap based on the Raspberry Pi Zero. They used MobileNetV2, EfficientNetV2B0, and MobileViT, along with explainable AI techniques, to identify relevant areas for classification. Tydén and

Olsson [156] evaluate object detection models based on SSD [95] on a Raspberry Pi to recognize animals, focusing on the trade-off between inference speed and accuracy. However, the image dataset used consists of 4,000 instances spanning only 8 species, which limits the scope of the conclusions. Jia et al. [73] used Neural Architecture Search to design a classification model for identifying animal species in camera trap images on edge devices, achieving accuracies of 95.35% and 81.86% on the NACTI-b and D3-a datasets, respectively, when tested on the NVIDIA Jetson TX2 board.

In Chapter 4, we consider the scenario where the camera uses an embedded model to filter certain images – specifically, empty frames – before transmitting them over a limited network or helping to reduce storage usage.

While most of the works focus on validating an application on an edge device, our main focus is the trade-off between inference latency and performance, as explored in the work of Tydén and Olsson [156]. However, we validate our models on datasets that are 100 times larger. Also, we present a comparative study between classifiers and detectors of varying complexities and apply model optimization methods.

3.6 Final remarks

In this chapter, we review the literature on multiple aspects of the automated analysis of camera-trap data. Table 3.1 summarizes the main limitations identified in the literature and how the proposed approaches address them.

Throughout the literature review on automated camera-trap image processing, we identified that counting individuals across sequences of images remains challenging. Existing camera-trap counting solutions have largely relied on heuristics [166] and classification [107], while recent work has also explored tracking and individual re-identification [127]. However, existing approaches do not directly address the combination of multi-species counting, low-frame-rate image sequences, and limited tracking supervision considered in this thesis. We address this task in Chapter 6.

On the other hand, the species identification task is widely explored in the literature from various perspectives. However, a major challenge – not only in camera-trap data but also in real-world problems in general – is the long-tail distribution and how to improve performance for categories in the tail. We explored state-of-the-art solutions applied to camera-trap images and proposed a new approach in Chapter 5. We also identified that, although species identification can benefit from using multiple images in a sequence to improve predictions, a more systematic study exploring such approaches is still lacking.

Table 3.1. Summary of limitations in the literature and how the proposed approaches address them across different camera-trap tasks.

Task	Challenge	Key Works	Prior capability	Key limitation	Proposed approach
Empty Frame Filtering	Performance vs. latency trade-off	[107, 149, 184]	High accuracy for empty vs. animal classification	No systematic evaluation under resource constraints	Systematic analysis of the accuracy–latency trade-off
	Edge deployment	[6, 35, 42, 161, 192, 193]	Demonstrate the feasibility of edge inference	Focus on system validation rather than model-level optimization	Model-level comparison and optimization for edge deployment
	Detector vs. classifier behavior	[13, 44, 108]	Strong performance from both paradigms	Lack of controlled comparison	Direct comparison under equivalent constraints
Species Classification	Long-tail distribution	[12, 18, 102, 107, 130]	Improves tail-class performance through re-weighting and re-sampling strategies	Degrades head-class performance or provides limited gains	SSB framework balancing tail improvement with minimal head-class loss
	Long-tail handling in practice	[14, 149, 163, 173]	Simplify the problem by removing or grouping rare classes	Ignores ecologically important rare species	Explicit modeling of the long-tail without discarding rare classes
	Architecture effectiveness	[107, 130, 149, 163, 173]	CNNs achieve strong performance on camera-trap datasets	Limited exploration of transformer-based architectures	Demonstrate strong performance of Swin Transformer without specialized imbalance handling
Individual Counting	Sequence-level counting	[11, 136, 166]	Practical counting using bounding-box heuristics	Often assumes a single species per sequence and is prone to over- or under-counting	Counting-by-tracking framework enabling multi-species predictions across sequences
	Lack of tracking supervision	[127]	Tracking based on geometric and visual feature distances	Large displacements and visual variation can lead to incorrect tracks	Synthetic data generation for training trackers with learned association modules

Although the task of filtering empty or blank frames is less complex than the tasks mentioned above, and solutions based on object detectors such as MegaDetector [13] have shown satisfactory results, we identified an opportunity to run this task directly on edge devices. These devices have limited computational resources, and running large object detectors may be unfeasible. With this in mind, in the next chapter, we investigate approaches to improve the performance of empty image filtering directly on edge devices.

Taken together, the literature review reveals three broader challenges that motivate this thesis: (i) the need for systematic evaluation of camera-trap analysis under realistic computational and annotation constraints; (ii) the challenge of improving recognition of rare species while preserving performance on frequent classes; and (iii) the challenge of counting individuals across low-frame-rate image sequences when suitable tracking supervision is limited. The thesis addresses these challenges through systematic evaluation and model-level analysis for resource-constrained deployment, the Square-Root Sampling Branch (SSB) for long-tailed species classification, and a counting-by-tracking framework based on learned association and synthetic supervision for multi-species sequence-level counting.

Filtering Empty Camera-Trap Images in Embedded Systems

In this chapter, we present a comparative study on empty frame detection in camera-trap images to analyze the trade-off between precision and inference latency on edge devices. To accomplish this objective, we investigate classifiers and object detectors of various input resolutions and optimize them using quantization and reducing the number of model filters.

4.1 Introduction

Monitoring wildlife through camera traps produces a large number of images, a significant proportion of which does not contain animals, being later discarded. For instance, about 75% of the images from the Snapshot Serengeti dataset do not contain animals [142].

Several studies have investigated the use of deep learning models to recognize animals in camera trap images [14, 107, 130, 149, 163, 173]. Due to the development of new technologies for this type of monitoring, embedding models to identify animals directly in the devices may provide advantages. For instance, in projects whose equipment is connected to a network, prior filtering avoids the unnecessary transmission of empty images, saving network bandwidth and energy [42]. Another example is traditional approaches, which usually involve going to the capture points to retrieve cameras and/or memory cards [153]. In this case, discarding empty images can save data storage, extending the time a camera can stay in the field collecting images without needing maintenance.

Although there are studies comparing the performance of several modern deep learning architectures on recognizing animals in camera trap images [107, 130, 163], there are few studies regarding the trade-off between model performance and inference latency directly on edge devices [130, 156, 193]. Thus, this chapter presents a comparative study between classifiers and detectors of different complexities running on edge devices to recognize nonempty images.

Our main findings are as follows:

- Detection models generally outperform classifiers with comparable latency if they are trained on the same camera trap image set.
- Classifiers can obtain high performance on empty image identification, given the difficulty of generating labels to train detectors. The performance, however, depends on the amount of images available for training, as well as on specific factors that can vary from dataset to dataset.
- Models with a higher input resolution perform better. Their inference latency can be kept low by reducing the number of model filters at an acceptable performance cost for the problem.

4.2 Materials and Methods

Two datasets are investigated in this chapter: 1) Caltech Camera Traps; and 2) Snapshot Serengeti (SS). These datasets are described below.

4.2.1 Datasets

Caltech Camera Traps [14]: It contains 243,100 images taken from 140 capture locations in the Southwestern United States. The instances were labeled in 22 categories, and about 66,000 bounding boxes localizing animals were also provided. For our experiments, we selected a subset of images from the empty and nonempty classes, grouping into the nonempty class images from all other categories that have bounding boxes. Then, the dataset was partitioned into training and validation set according to the locations recommended in [88]. Moreover, a subset of the training partition was split, consisting of 20 locations chosen at random, to be used to adjust the training hyper-parameters (called here validation dev). Due to the fact that some locations concentrate a large number of images of the empty class, which may lead the models to be biased in certain backgrounds, we decided to limit to 1,000 instances per location

the number of empty class instances in both training and validation dev partitions. Table 4.1 summarizes the number of instances obtained.

Table 4.1. Number of images used from the Caltech dataset. The empty class of training and validation dev partitions was limited to 1000 instances per location. The nonempty class is composed of images with bounding boxes from all other categories.

Class	Training	Val-dev	Validation
Empty	8,574	2,824	19,892
Nonempty	32,032	3,877	23,410

Snapshot Serengeti [142]: This dataset currently contains more than 7 million camera trap images collected over 11 seasons in the Serengeti National Park, Tanzania. In this work, we use images from the first six seasons in order to keep consistency with previous works using the Snapshot Serengeti dataset [107, 163, 173]. These images were divided into training and validation sets according to the locations (SS-Site), as recommended in [89]. As was done for the Caltech dataset, we have also created a validation dev split for hyper-parameter adjustment. This partition was obtained by randomly selecting 23 locations from the training set. In addition to the SS-Site configuration, we have also performed a partitioning by time (SS-Time) taking into account that the models could be used in images from new seasons. To generate SS-Time, the first four seasons were grouped into a training set, while the fifth season was used as validation dev, and the sixth season as the validation set. Finally, the dataset was adapted to represent a bi-class problem, where instances from the blank category were labeled to the empty class and the others were used to compose the nonempty class. We also balanced the classes for the training partition.

In addition, since there are approximately only 78,000 images from the nonempty class annotated with bounding boxes, we also perform experiments using subsets of instances (called small), which are balanced for the training and validation dev partitions. The objective here is to perform a fair comparison between classifiers and detectors when both are trained using the same number of images. Unless otherwise specified, the experiments conducted in this work use these subsets with fewer images. Details about all data partitions of the Snapshot Serengeti dataset investigated in this work are shown in Table 4.2.

Table 4.2. Number of instances used from the Snapshot Serengeti dataset.

	Class	Training	Val(dev)	Validation
SS-Site	Empty	524,804	278,578	535,817
	Nonempty	523,891	84,531	209,183
SS-Site (small)	Empty	51,281	6,041	535,817
	Nonempty	52,081	6,041	209,183
SS-Time	Empty	516,635	588,406	383,981
	Nonempty	516,630	225,647	75,328
SS-Time (small)	Empty	43,612	18,957	383,981
	Nonempty	44,579	18,957	75,328

4.2.2 Architectures

The MobileNetV2 [124] architecture is a natural choice to be used as a classifier baseline, since it was designed specifically for computationally limited devices. Besides the original MobileNetV2 model (input resolution 224×224), a version with input 320×320 was also used in our experiments due to the fact that animals can appear very far from the camera and visually small as a consequence [107]. From the family of models EfficientNet [150], we chose two versions with input resolutions comparable to MobileNetV2’s: EfficientNet-B0 (224×224) and EfficientNet-B3 (300×300).

For the detectors, we investigate the SSDLite [124] with a MobileNetV2 (320×320 input) as backbone. For the family of detectors using EfficientNets as backbones [152], we included in our experiments the EfficientDet-D0, which works with an input resolution of 512×512 , the smallest of this group of models.

4.2.3 Implementation Details

Image preprocessing for classification: We chose a standard procedure for image preprocessing. Initially, a random rectangular crop of the image is applied with aspect ratio and area sampled in $[3/4, 4/3]$ and $[65\%, 100\%]$, respectively. Then, each image is scaled to the input size of each architecture and a horizontal flip is applied with 50% probability. Next, we apply data augmentation using RandAugment [31] with parameters $N = 2$ and $M = 2$. Finally, the pixel values of the image are scaled in $[-1, 1]$ for MobileNetV2 and in $[0, 1]$ for EfficientNets. During validation, the preprocessing consists of only image resizing and pixel scaling depending on the architecture.

Classifiers training procedure: Each model was initialized with ImageNet pre-trained weights and then trained during 10 epochs using the Stochastic gradient

descent (SGD) on an NVIDIA GeForce GTX 1080 Ti graphics card. The initial learning rate was 0.01 for a batch size of 256 and scaled linearly according to the batch size effectively used, which varied according to the model due to the graphics card limited memory, as shown in Table 4.3. The learning rate was linearly increased from 0 to the initial rate during 30% of the steps of the first epoch and then reduced using the cosine decay scheduling, as suggested by He et al. [58].

Table 4.3. Batch size and initial learning rate used for each classifier. The initial learning rate is scaled linearly according to the batch size b , defined by $0.01 \times b/256$.

Architecture	Batch size	Learning rate
MobileNetV2 (224)	128	0.005
MobileNetV2 (320)	64	0.0025
EfficientNet-B0	32	0.00125
EfficientNet-B3	16	0.000625

Detectors training procedure: Both detectors were initialized with weights pre-trained on COCO [93] and then trained on each dataset using the Tensorflow Object Detection API [67]. We have used the standard training procedure for each detector, besides 32 and 8 as batch size for SSDLite+MobileNetV2 and EfficientDet-D0 respectively. The learning rate and the number of training epochs were adjusted according to the performance measured on the validation dev partition. These values are shown in Table 4.4.

Table 4.4. Training hyperparameters for detectors. The same hyperparameters were used for both time and site partitioning of Snapshot Serengeti (SS).

Model	Dataset	Learning rate	Training steps
SSDLite+MobileNetV2	Caltech	0.008	12,000
	SS	0.004	36,000
EfficientDet-D0	Caltech	0.008	50,000
	SS	0.001	150,000

4.2.4 Evaluation Procedure

In order to compare detectors and classifiers, predictions related to the bounding box coordinates were ignored. Thus, only the detection confidence value that represents the class with the highest score was used as the detected label.

Taking into account that the confidence threshold can be adjusted to avoid nonempty images discarding, we decided to use the precision-recall curve as a graphical tool to compare the general performance of the models considering all possible thresholds. To compare the models effectiveness when discarding empty images, i.e., the true negative rate (TNR), the confidence threshold of each model was adjusted so that the recall for nonempty images was set to 96% – a value reached by models investigated in [107].

We used a Raspberry Pi 3 model B running Raspbian GNU/Linux 10 as a reference edge device to assess the models' latency. The models were converted to the TensorFlow Lite format without quantization and their latency was calculated using the native benchmark tool provided by TensorFlow¹. The reported latency values were calculated as the average over 50 runs for each model. Additionally, we also evaluate the models version obtained as a result of post-training quantization [71]. In this scenario, a subset of 500 training instances was employed to calibrate the operations to work with the integer type, while maintaining model inputs and outputs as floating point to keep the original model interface.

4.3 Experimental Results

4.3.1 Classifiers vs. Detectors

In order to compare the performance of classifiers and detectors as fairly as possible, the models were trained using subsets composed by instances of the nonempty class annotated with bounding boxes. Despite of the fact that these subsets were generated using a much smaller amount of the total images available for classification, leading to possible sub-optimal models, this number of instances can provide a more realistic perspective of the problem. Indeed, as pointed out by Schneider et al. [130], the vast majority of small-scale research projects focused on camera trap do not have a large amount of labeled images.

Results: Figure 4.1 shows the precision-recall curves of the investigated models for each dataset. As expected, detectors outperformed classifiers in all datasets, especially Caltech. Considering a confidence threshold producing 96% of recall, EfficientDet-D0 was able to eliminate more than twice as many empty images when compared to the classifiers using Caltech dataset. In terms of SSDLite+MobileNetV2, it also obtained a significantly higher true negative rate (at least 19% in absolute values), as reported in Table 4.5. The scenario is quite similar for the Snapshot Serengeti

¹<https://www.tensorflow.org/lite/performance/measurement>

dataset, since detectors also outperformed classifiers by a significant margin (at least 8% of precision). Figure 4.2 shows some images to illustrate the results attained in our experiments.

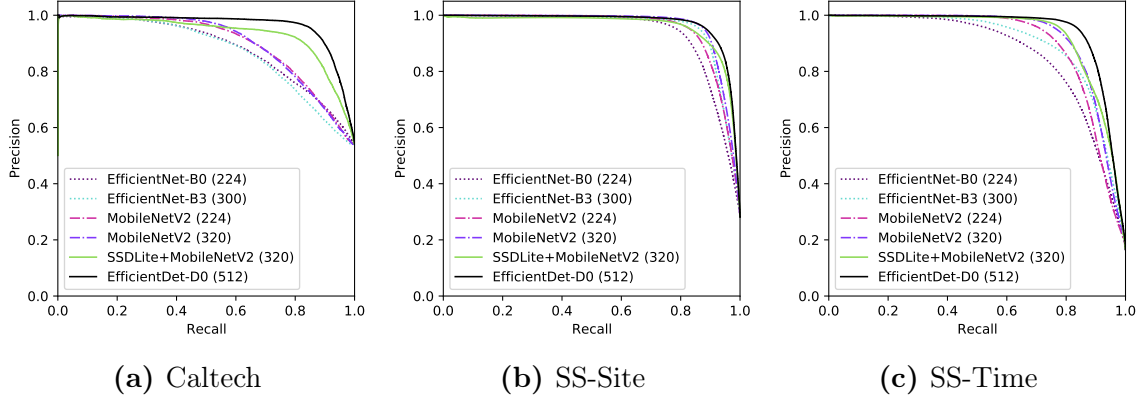


Figure 4.1. Precision-recall curve for the nonempty class. Detectors reached higher performances, with wider advantage on the Caltech dataset.

Table 4.5. Comparison of precision for the nonempty class and the true negative rate (TNR) where the confidence threshold of each model was adjusted to achieve a recall of 96% on the nonempty class. The reported CPU latency corresponds to the Caltech dataset, but it is similar to the others, being calculated from the average of 50 runs. The true negative rate indicates the percentage of images without animals that would no longer be stored unnecessarily.

Model	CPU	Caltech			SS-Site			SS-Time		
		Precision	TNR	Thresh.	Precision	TNR	Thresh.	Precision	TNR	Thresh.
Efficientnet-B0	801ms	60.26%	25.50%	0.355	50.32%	63.00%	0.153	33.08%	61.90%	0.159
Efficientnet-B3	3205ms	56.42%	12.75%	0.305	57.21%	71.97%	0.155	39.27%	70.87%	0.170
MobileNetV2-224	324ms	58.60%	20.18%	0.228	57.72%	72.55%	0.188	30.51%	57.11%	0.126
MobileNetV2-320	638ms	58.58%	20.13%	0.239	62.84%	77.84%	0.191	35.74%	66.13%	0.147
SSDLite+MNetV2	838ms	67.03%	44.42%	0.166	75.32%	87.72%	0.167	47.14%	78.89%	0.147
Efficientdet-D0	4686ms	73.31%	58.86%	0.148	79.14%	90.12%	0.150	47.35%	79.06%	0.143

The poor classifier performance on the Caltech dataset can be due to several factors, such as the low variability of backgrounds, the size of animals, camouflage, quality of images, among others. However, our experiments were not designed to identify these nuances intrinsic to this dataset. On the other hand, this may be an interesting topic to be investigated in future work.

Latency-precision trade-off: Table 4.5 shows the latency for the inference of each model. Although EfficientDet-D0 offers a strong baseline for the problem, its latency is more than four seconds, which is prohibitive because camera trap images are usually obtained within one second between them. In this context, SSDLite+MobileNetV2 may be deemed to attain superior performance since its latency

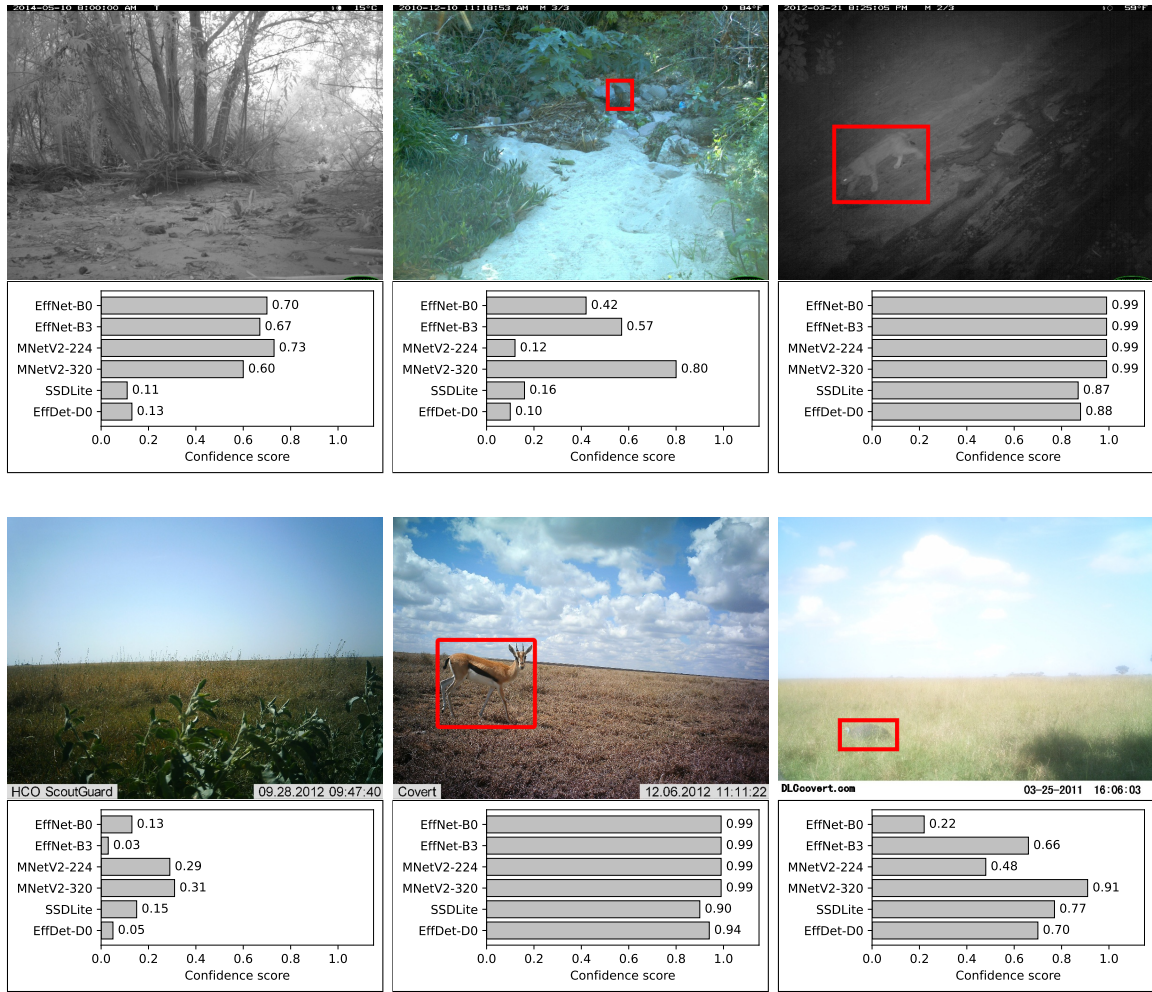


Figure 4.2. Sample classification results reached by the investigated models highlighting their confidence for the nonempty class. The top row depicts images from the Caltech dataset and the bottom row from the Snapshot Serengeti dataset. The first column depicts instances of the empty class while the remainder are nonempty images with the animals highlighted.

is below one second and it eliminated at least 10% more images than the classifiers with comparable latencies.

4.3.2 Training with more Images

A common way to improve model performance is to use more training instances [81]. Following this strategy, an experiment was carried out to train the classifiers using the subsets of the SS dataset. This dataset contains ten times more images than the one used in the previous experiment. Figure 4.3 and Table 4.6 summarize the results attained in this scenario. The results indicate that classifiers outperformed detectors. We can highlight that MobileNetV2 (224) reached performance similar to SSDLite+MobileNetV2, with less than half of inference latency though. It is important

to note, however, that all detectors were trained using data subsets from the previous experiment. Therefore, in order to provide a fair comparison, a similar amount of training instances used to train the classifiers should be used to train the detectors. On the other hand, obtaining more annotated instances for the detection problem is expensive. This is the reason we do not show results using the same dataset for training classifiers and detectors. However, we can conclude based on the results obtained in this experiment that, depending on the number of labeled instances available, classifiers can be a viable option for the problem of identifying empty images, also showing better efficiency in terms of computational resources.

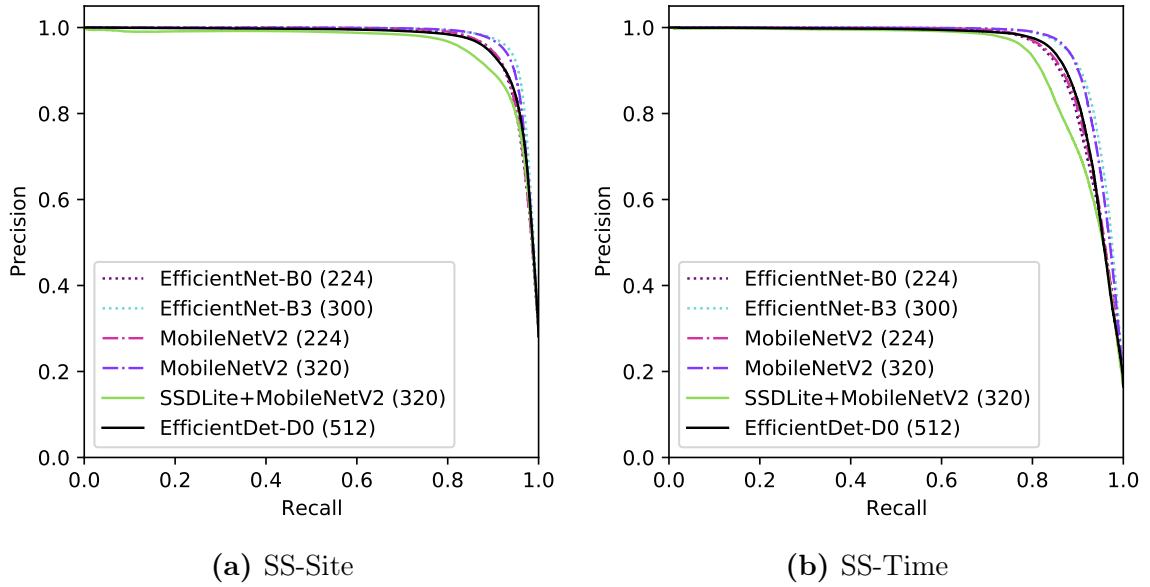


Figure 4.3. Precision-recall curve for the nonempty class of classifiers trained in sets with ten times more images from the SS dataset. The curves of the detectors refer to the models trained on the original smaller set. Best viewed in color.

Table 4.6. Comparison of precision for the nonempty class and the true negative rate (TNR), with a recall at 96%, for classifiers trained in sets with ten times more images from the SS dataset. Note the significant improvement in the performance of the classifiers, exceeding by a large margin the detectors trained in the original set.

Model	SS-Site		SS-Time	
	Precision	TNR	Precision	TNR
Efficientnet-B0	73.92%	86.78%	48.81%	80.25%
Efficientnet-B3	87.67%	94.73%	64.28%	89.54%
MobileNetV2-224	75.18%	87.63%	49.12%	80.50%
MobileNetV2-320	82.89%	92.26%	58.61%	86.70%

4.3.3 Model Optimization

It is possible to observe from the results shown in previous sections that models dealing with higher image input resolutions achieved higher performance but at the expense of higher inference latency. In order to reduce latency of these models, we performed an experiment using the SS dataset and MobileNetV2 (320). In this experiment, the number of filters was reduced by adjusting the alpha parameter. We also evaluated the post-training quantization of the models. Since the swish activation function used by EfficientNets is not very well supported for quantization [177], models of this family were not used here. In addition, models evaluation was carried out on a sample of 5,000 instances obtained from the validation set, because the TensorFlow Lite quantized models are not optimized for inference on x86 architectures and graphics cards used for training.

The results shown in Figure 4.4 reinforce the role of input resolution, since a MobileNetV2 with half the filters (MobileNetV2-0.50) but higher resolution (320) obtained overall performance superior to the performance of the standard model (MobileNetV2 (224)). In Table 4.7, we may observe that it is possible to reduce about 23% of model latency due to both quantization (SS-site) and number of filters reduction (SS-time), maintaining compatible performance. It is worth mentioning that the quantized version of the SSDLite + MobileNetV2 detector did not obtain good results at the 96% recall level used. These results indicate the importance of assessing the performance of the models after quantization, especially when the confidence threshold is adjusted at certain levels. One way to avoid this problem would be quantization-aware training.

Table 4.7. Performance comparison of MobileNetV2 models for various widths on the SS dataset. The models were converted to TensorFlow Lite and evaluated using a sample of 5,000 instances randomly chosen from the validation set.

Model		CPU	Memory	SS-Site		SS-Time	
				Precision	TNR	Precision	TNR
MobileNetV2-224	Float	322ms	22.7MB	58.20%	73.07%	25.67%	45.33%
	Int8	237ms	9.3MB	55.52%	69.99%	26.18%	46.74%
	Int8 (quant aware)	239ms	9.8MB	57.31%	72.10%	-	-
MobileNetV2-0.50-320	Float	259ms	19.7MB	54.86%	69.15%	27.76%	50.86%
	Int8	237ms	9.6MB	51.50%	64.70%	26.67%	48.09%
MobileNetV2-0.75-320	Float	484ms	19.8MB	64.89%	79.72%	31.37%	58.71%
	Int8	408ms	13.3MB	59.50%	74.49%	30.44%	56.87%
	Int8 (quant aware)	413ms	13.4MB	61.38%	76.41%	-	-
MobileNetV2-320	Float	635ms	33.5MB	66.65%	81.25%	30.34%	56.65%
	Int8	485ms	13.9MB	66.70%	81.28%	30.98%	57.95%
SSDLite+MobileNetV2	Float	840ms	31.7MB	73.68%	86.62%	39.50%	71.09%
	Int8	575ms	13.6MB	36.83%	35.72%	19.27%	20.90%

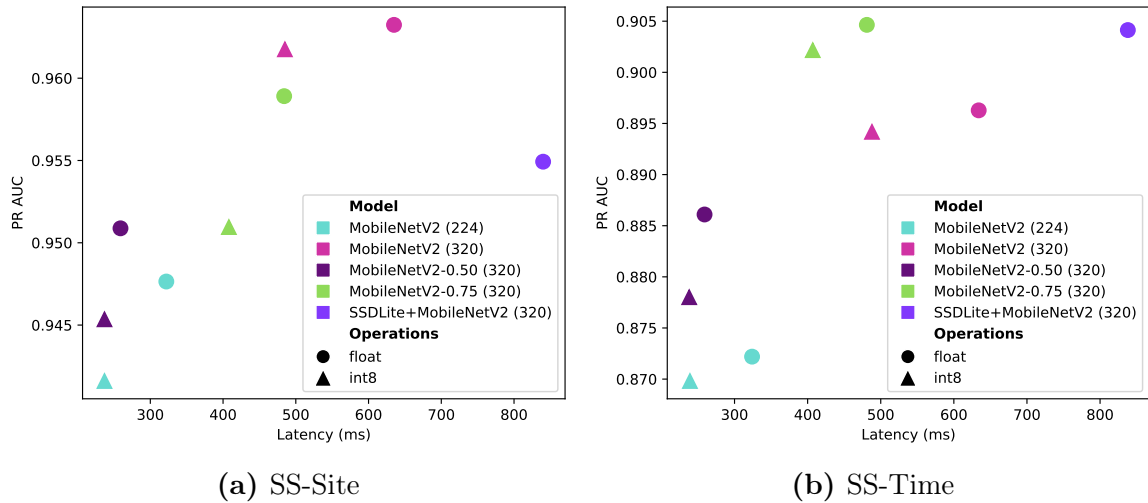


Figure 4.4. Latency vs. area under the precision-recall curve (PR AUC) for various versions of MobileNetV2 in the SS. The results refer to the models converted to TensorFlow Lite (floating point and integer) and evaluated in a subset of 5,000 test instances. Best viewed in color.

To assess this, we fine tuned the models MobileNetV2-224 and MobileNetV2-0.75-320 trained on SS-Site using quantization-aware training for 2 epochs with the initial learning rate divided by 10. In this case, we were able to improve the results of the quantized versions, as shown in Table 4.7. Unfortunately, we were not able to assess SSDLite in the same way due to the fact that Tensorflow Object Detection API did not support quantization-aware training for TF2 at the experiments’ time.

Although the focus of this work is the latency-precision trade-off, we also measured the memory usage of each model, as shown in Table 4.7. Quantized models have a significant reduction in memory usage compared to the floating point models. However, when observed the amount of RAM available on Raspberry Pi (1GB), the memory required by the most complex model is very low (33.5MB), therefore, not affecting the performance of other processes that may be running on the device.

4.3.4 Threats to Validity

Several factors should be considered when interpreting the results presented in this chapter. Regarding internal validity, the experiments were designed to control potential sources of bias, including limiting the number of empty images per location in the Caltech dataset and using the same training subsets when directly comparing classifiers and detectors. However, the availability of training data differs across some experiments, particularly because obtaining bounding-box annotations for detectors is more costly. Thus, differences in training data may contribute to the observed per-

formance differences. Regarding external validity, the experiments were conducted on the Caltech Camera Traps and Snapshot Serengeti datasets, with spatial and temporal splits used to assess generalization to unseen locations and seasons. Nevertheless, the results may not generalize to other geographic regions, wildlife species, camera configurations, or environmental conditions. Moreover, latency measurements were obtained on a Raspberry Pi 3 Model B and may differ on other edge devices or hardware configurations.

4.4 Final Remarks

In this chapter, we presented a comparative study between detection and classification models in the context of identification of nonempty images of animals on edge devices. Our results showed that detection models outperform classifiers when both are trained using the same training set, but their higher inference latency can limit their use on edge devices. This work resulted in the publication of “Filtering Empty Camera Trap Images in Embedded Systems” [33] in the Mobile AI Workshop and Challenges (MAI) of CVPR 2021.

Moreover, depending on the dataset, it is possible to train classifiers to obtain satisfactory results, especially when there is massive number of images available, as in the Snapshot Serengeti dataset. When detector is essential, e.g. Caltech dataset, but the model’s latency is not within the design requirements, it may be necessary to use hardware accelerators, such as EdgeTPUs or DSPs. Another limitation to detectors is the difficulty on obtaining new instances annotated with bounding boxes, which is time-consuming and expensive. In this situation, one possibility is to use techniques tackling few or no labels, such as semi-supervised learning and self-supervised learning. Other alternative is training an agnostic detection model designed to run on edge devices, inspired by MegaDetector. Regarding the optimization strategies evaluated, the post-training quantization proved to be effective, but it requires a careful evaluation of the resulting model, which may decrease its performance due to the quantization process. However, quantization-aware training may solve this issue. Finally, using models with fewer filters but with higher resolution was also effective. Therefore, techniques such as knowledge distillation and model pruning are interesting directions for future work.

In this chapter, we handled the filtering of empty images considering a two-stage approach, where in the first stage a binary classifier distinguishes empty from nonempty images, leaving the species identification to the second stage. In the next chapter, the empty class is incorporated directly into the species classifier as just another category.

As previously highlighted, empty images usually represent the majority category in camera-trap datasets, with the remaining species categories exhibiting a highly imbalanced, long-tail distribution. While we balanced the Snapshot Serengeti dataset in this chapter to better handle the filtering task, in the next chapter, we keep the original dataset distributions. In fact, the next chapter treats this extreme imbalance as an inherent characteristic of camera-trap data, focusing on improving performance under this constraint.

SSB: Square-Root Sampling Branch

In this Chapter, we propose a simple and effective framework called Square-Root Sampling Branch (SSB) to improve long-tail visual recognition of animal species in camera-trap images. We systematically evaluate it against state-of-the-art methods for handling this task: square-root sampling, class-balanced focal loss, and balanced group softmax.

5.1 Introduction

One major problem with camera-trap data is the extreme imbalance among categories as shown in [Figure 5.1](#), in which some species have many images while others have just a few, following a long-tail distribution. As reinforced by Schneider et al. [130], models generally achieve poor recall scores for species with fewer training images, although these rare species are often the ones of greatest interest to ecologists.

Many works in the literature on machine learning have proposed techniques for addressing the long-tailed issue, such as re-sampling the data [76], re-weighting the loss [32], or creating multiple branches in the networks [22, 87, 189]. Despite this, most of the works for classifying animal species in camera-trap images do not apply any method for handling long-tail classification. Some works remove the tail classes [163, 173] or group them into generic categories [193].

Based on the empirical observation that square-root sampling [76] was the method that most improved the performance of the tail classes in our preliminary experiments, though at the cost of strongly degrading the head classes, in this chapter a framework called Square-root Sampling Branch (SSB) is proposed, which combines the predictions of two classification branches: one trained using square-root sampling and the other one trained without any additional method for long-tail classification.

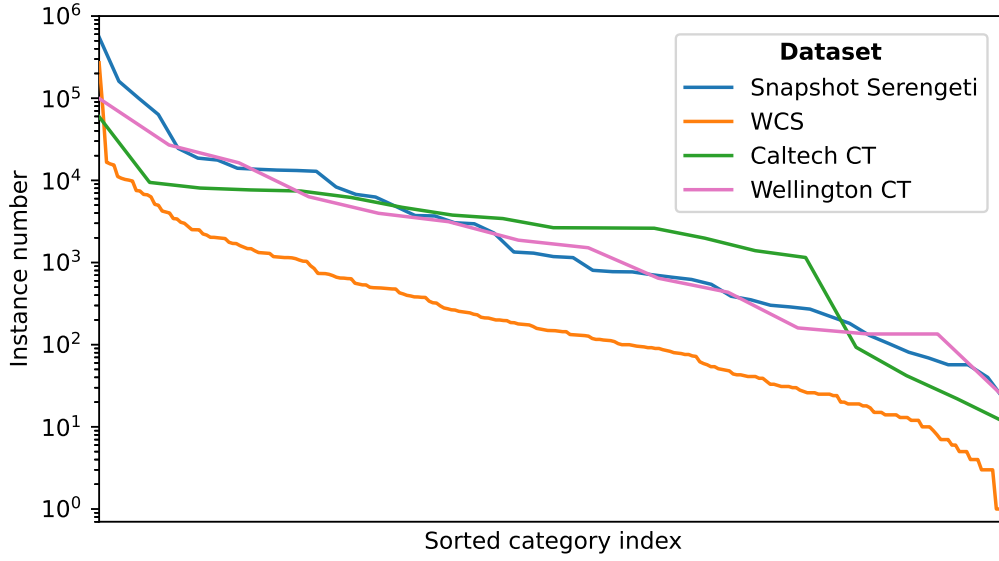


Figure 5.1. The number of instances per category observed in the camera-trap datasets used in this work follows a long-tail distribution. Categories are sorted by the number of instances in the training set and aligned among datasets for better visualization.

As the contributions, we can highlight:

- The proposal of a simple and effective framework called Square-root Sampling Branch (SSB) for long-tail visual recognition, which achieves a better trade-off as it improves performance in tail classes at a low cost of head classes' accuracy when compared to the state-of-the-art methods.
- New baselines in camera-trap datasets are established using training/test partitioning that is based on locations by combining the most recent training tricks with both traditional architectures, such as ResNet-50, and newer ones such as Swin Transformer.
- Our experiments show that Swin Transformer can achieve excellent results in camera-trap datasets even without using any additional method for handling the long-tail issue.

5.2 Materials and Methods

Our methodology was built to assess the effectiveness of our proposed SSB framework in improving the recognition performance in the tail classes and comparing it with methods available in the literature. First, a strong baseline was trained with several

recent training tricks available in the literature. Then, each long-tail method was applied and evaluated using appropriate metrics to assess long-tail performance. To reach a more general conclusion, the performance of the methods on four long-tail camera-trap datasets with different characteristics and four families of computer vision models were evaluated.

In this section, the datasets, the chosen computer vision architectures, and the baseline training procedure are described. Then, the families of methods for long-tail recognition and the selected methods used in this work are presented. Finally, a description of our proposed SSB framework is given.

5.2.1 Datasets

This subsection describes the four camera-trap datasets used in this study: Snapshot Serengeti, Caltech Camera Traps, WCS Camera Traps, and Wellington Camera Traps. These datasets were obtained from different regions around the globe and have different characteristics, such as the number of training instances, imbalance factor, and available species. While annotations for these datasets are provided for bursts (i.e., a sequence of images), we assign each image the label inherited from its burst annotations during the training process. However, this approach may result in mislabeling, especially when the animal appears in only a few images within a burst. Since manual verification was not feasible due to the large amount of images in the datasets, the data was used as provided, as has been done in prior studies [107, 163].

Nonetheless, it is important to note that the recommended approach is to have annotations at an image-level, as this provides more accurate training data. Following the recent recommendations, the datasets were split into training/validation/test partitions based on camera locations to better assess the models' generalization. Additionally, images were removed from all classes which, due to the location-based split, do not have instances in the training or test partitions. Images containing more than one species as well as corrupted pictures were also removed. Table 5.1 summarizes the number of images used from each dataset in this work. All of these datasets follow a long-tail distribution, as shown in Figure 5.1, in which a few classes (head) concentrate most of the instances and the remaining classes have just a few samples in comparison.

Snapshot Serengeti [142] This dataset is composed of more than 7 million images obtained from 225 locations spread across the Serengeti National Park in Tanzania, and is divided into 11 seasons. To maintain consistency with the literature, only the first six seasons are used in this work. The recommended training/validation partitions

Table 5.1. Number of images and locations per partition used from each dataset.

Dataset	Classes	Training		Validation		Test	
		# loc	# instances	# loc	# instances	# loc	# instances
Snapshot Serengeti	47	156	1,059,793	23*	360,317	46*	738,404
Caltech CT	19	80	122,693	20*	57,162	40*	61,665
WCS	247	2,506	599,040	313	79,906	314	62,734
Wellington CT	14	111	160,937	36	49,127	35	55,424

* As long as there are no official test sets for Snapshot Serengeti and Caltech CT datasets, their recommended validation sets were used as the test sets in this work. For these datasets, validation sets (which were called minival) were held out from their respective training sets following the locations selected in Chapter 4.

based on locations were used [89], with the validation set being used to evaluate the models. To adjust the hyperparameters, a mini-validation (minival) partition was held out from the training set using the locations selected in Chapter 4. Specifically for this dataset, given that images from empty bursts (background or blank) are near identical, only one picture was picked at random from each empty burst. This also helps to keep a tractable number of images to train the models. In total, approximately 2 million pictures distributed in 47 categories were used.

Caltech Camera Traps [14] This dataset is composed of approximately of 240,000 images from 140 locations in the southwestern United States. The recommended location-based training/validation partitions were used [88], and, as for the Snapshot Serengeti, the validation set was used for testing, and a mini-validation partition was created from the training set for hyperparameter adjustment based on locations selected in Chapter 4.

WCS Camera Traps [90] This dataset is provided by the Wildlife Conservation Society, and it is composed of 1.4 million images of around 675 species from 12 countries, thus making it one of the most diverse datasets publicly available. In our experiments, the recommended training/validation/test partitions based on locations were used. However, since some locations were not allocated to any partition, images from them were not used. The generic unknown categories and those not related to animals, such as misfire, start, and end were also manually removed. As images from classes without instances in training or testing sets were also removed, the resulting selected subset is composed of images from 247 categories.

Wellington Camera Traps [7] This dataset is originally composed of around 270,000 images collected from 187 camera locations in the Wellington region of New Zealand. The training/validation/test proposed by Shashidhara et al. [135] was used. Images

marked as unclassifiable were removed, in addition to images from the categories not present in the training or test partitions, thus resulting in a dataset of around 265,000 images from 14 categories. Unlike the other datasets, where the empty class is the majority category, in this dataset the empty class has the second-largest number of instances – with the generic bird category appearing three times more frequently.

5.2.2 Architectures

Four popular computer vision architectures with different purposes and characteristics were chosen to assess whether long-tail classification methods work well on different backbones: ResNet-50 [57], MobileNetV3-Large [65], EfficientNetV2-B2 [151], and Swin-S [96]. First, ResNet-50 was selected because it is the most common baseline for almost any method in computer vision. As using computer vision models directly on edge devices for processing camera-trap data has gained attention in the literature [33, 42, 192, 193], MobileNetV3-Large was chosen as a benchmark for this situation. For the state-of-the-art families of EfficientNet [150] and Swin Transformer [96] models, lightweight versions – EfficientNetV2-B2 and Swin-S, respectively – were chosen, since most of the camera-trap practitioners usually do not have access to powerful computers to run big models [130].

5.2.3 Baseline training settings

As shown in several works [17, 58, 171], improved training strategies are responsible for significant gains in the accuracy of classical architectures such as ResNet-50. Following these ideas, several tricks were included in our baseline training procedure, and these are described below.

Image preprocessing Initially, all images were resized so that the largest dimension was at most 768 px, and the smallest one was at least 450 px. During training, first, a rectangular region of aspect ratio sampled in $[3/4, 4/3]$ and area in $[65\%, 100\%]$ was randomly cropped and resized to the input size of the architecture. Next, the image was flipped horizontally with a 50% probability. Then, RandAugment [31] was used with parameters $N = 2$ and $M = 9$. Finally, the image was rescaled to match the input scale of each architecture, as shown in Table 5.2. During the evaluation, the preprocessing consists only of resizing the image to the input architecture size and rescaling.

Table 5.2. Input scaling procedure for each architecture.

Architecture	Scaling Procedure
ResNet-50	Convert from RGB to BGR, zero-center w.r.t. the ImageNet dataset, without scaling
MobileNetV3-Large	Pixels between 0 and 255 (model includes preprocessing layer)
EfficientNetV2-B2	Scale pixels between -1 and 1
Swin-S	Scale pixels between 0 and 1, normalize w.r.t. the ImageNet dataset

Training procedure The models were initialized with ImageNet pre-trained weights, replacing the classifier layer, and, then, were trained with a batch size of 64 for 30 epochs when fine-tuning the whole model or for 12 epochs when training only the classifier layer. The AdamW optimizer [100] was employed for training with an initial learning rate of 10^{-5} and weight decay of 10^{-7} . The learning rate was linearly warmed up from 0 to the initial value for two epochs when fine-tuning – one epoch when training only the classifier –, and then decayed to 0 using the cosine schedule [58].

5.2.4 Training approaches for long-tail visual recognition

To achieve more general conclusions, we performed a systematic study using state-of-the-art methods for handling the long-tail visual recognition problem. As described in Section 2.6, we grouped them into four main families according to their focus: re-sampling the data, re-weighting the loss, using two stages to focus on representation learning, and creating multiple branches in the network, selecting one method from each family to run our experiments. Figure 5.2 illustrates a functional summary of the model training process, highlighting the stages where each method studied in this work operates.

For representing the re-weighting methods, we selected the square-root resampling [76, 187] by setting $q = 1/2$ in Equation 2.3, as this approach returns a less imbalanced dataset. For the re-weighting methods, we used the Class-Balanced Focal (CB-Focal) loss, which is a combined version of Focal loss and Class-Balanced loss. In our experiments, we set the hyperparameters β to 0.9 and γ to 2.0. In our experiments, we also compare one-stage and two-stage training strategies. In the one-stage approach, the balancing method is applied end-to-end, while in the two-stage method [76], the model is initially trained using the original unbalanced training distribution. Then, only the classifier is trained using a balancing method, such as re-weighting or re-

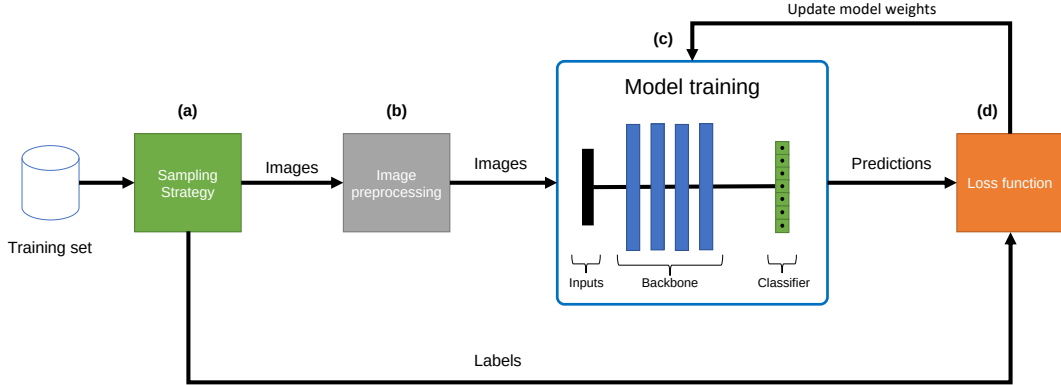


Figure 5.2. Schematic diagram of model training process. Each method studied in this work operates at one or more stages: re-sampling methods change the sampling strategy (a), re-weighting methods modify the loss function (d), methods using multi-branch networks modify the model structure (c), and two-stage training methods operate by selecting which model weights (c) are trained at each stage. Our proposed SSB method operates by changing both the sampling strategy (a) and the model structure (c). Additionally, cropping images before classification is performed at the image preprocessing stage (b).

sampling. Among the various multi-branch network approaches available in the literature, we included BAGS [87] in our experiments.

The method of using an object detector to crop regions of interest from camera-trap images before classifying them was also included in our experiments. Despite not dealing directly with the imbalance problem, it has been shown in the literature [13, 131] that this trick improves the recognition score in camera-trap images. In our experiments, we used MegaDetectorV4 [13] to generate bounding box predictions, considering only those with a confidence score higher than 0.6¹; otherwise, the entire image was used. In this operation, we applied a squared crop around the bounding box with the highest confidence, executed before the random crop operation in the preprocessing pipeline (subsection 5.2.3).

5.2.5 SSB: Square-root Sampling Branch

Our proposed framework is a multi-branch network approach that was inspired by an empirical observation in our preliminary experiments: the square-root sampling strategy was the method that most improved the accuracy of the tail classes over the baseline; however, at the cost of significantly decreasing the performance of the head

¹The threshold of 0.6 was selected based on the top-performing solutions from the iWildCam 2020 and iWildCam 2021 challenges, which used MegaDetector to generate cropped regions and found this threshold to be effective.

classes. The question to answer is: is it possible to combine the **S**quare-root **S**ampling **B**ranch (SSB) predictions with the baseline to improve the tail classes with minimal performance degradation for the head classes?

As illustrated in [Figure 5.3](#), the proposed framework is trained in a two-stage fashion. First, the classifier (f_i) and the backbone are trained jointly using the original data distribution (instance-based sampling). For the second stage, the network backbone is frozen and the classifier is replaced by a new randomly initialized one. Then, this new classifier (f_{sqr}) is trained using data sampled by the square-root sampling strategy. Both classifiers are connected on top of the backbone during the inference. Their predictions are aggregated by masking the softmax outputs to keep only the head classes probabilities from the f_i head and the remaining classes probabilities from the f_{sqr} head.

The C classes were divided into four groups based on the number of training instances, in which a class j with n_j instances belongs to group G_k if $s_k^l \leq n_j < s_k^h$. The limits of each group are set to $(0, 10)$, $(10, 10^2)$, $(10^2, 10^3)$, and $(10^3, +\infty)$ following the approach adopted by BAGS [87]. However, SSB does not have a special group for the background class, with this class belonging to the group G_4 . Given a feature vector h generated by the backbone and the predictions $p_i = f_i(h)$ and $p_{sqr} = f_{sqr}(h)$ generated by each classification head with softmax, the final remapped prediction vector p_r is given by:

$$p_r = f_r(p_i, p_{sqr}) = Qp_i + (\mathbf{1} - Q)p_{sqr}, \quad (5.1)$$

where Q is a matrix $C \times C$ that masks the softmax outputs based on which group a class belongs to:

$$Q = \left\{ q_{a,b} \middle| q_{a,b} = \begin{cases} 1 & \text{if } a = b \text{ and the class } a \in G_4, \\ 0 & \text{otherwise.} \end{cases} \right\}. \quad (5.2)$$

As with BAGS post-processing, the final prediction vector produced by SSB is not a real probability vector because its sum is not equal to 1.

5.3 Experiments and Results

Our experiments were broken down into three main series. In the first, our proposed method SSB is compared to three two-stage methods for long-tail problems, as well as the baseline. In the second series, the same methods are compared, but using cropped images as inputs instead of the entire images. Finally, in the third series, SSB is

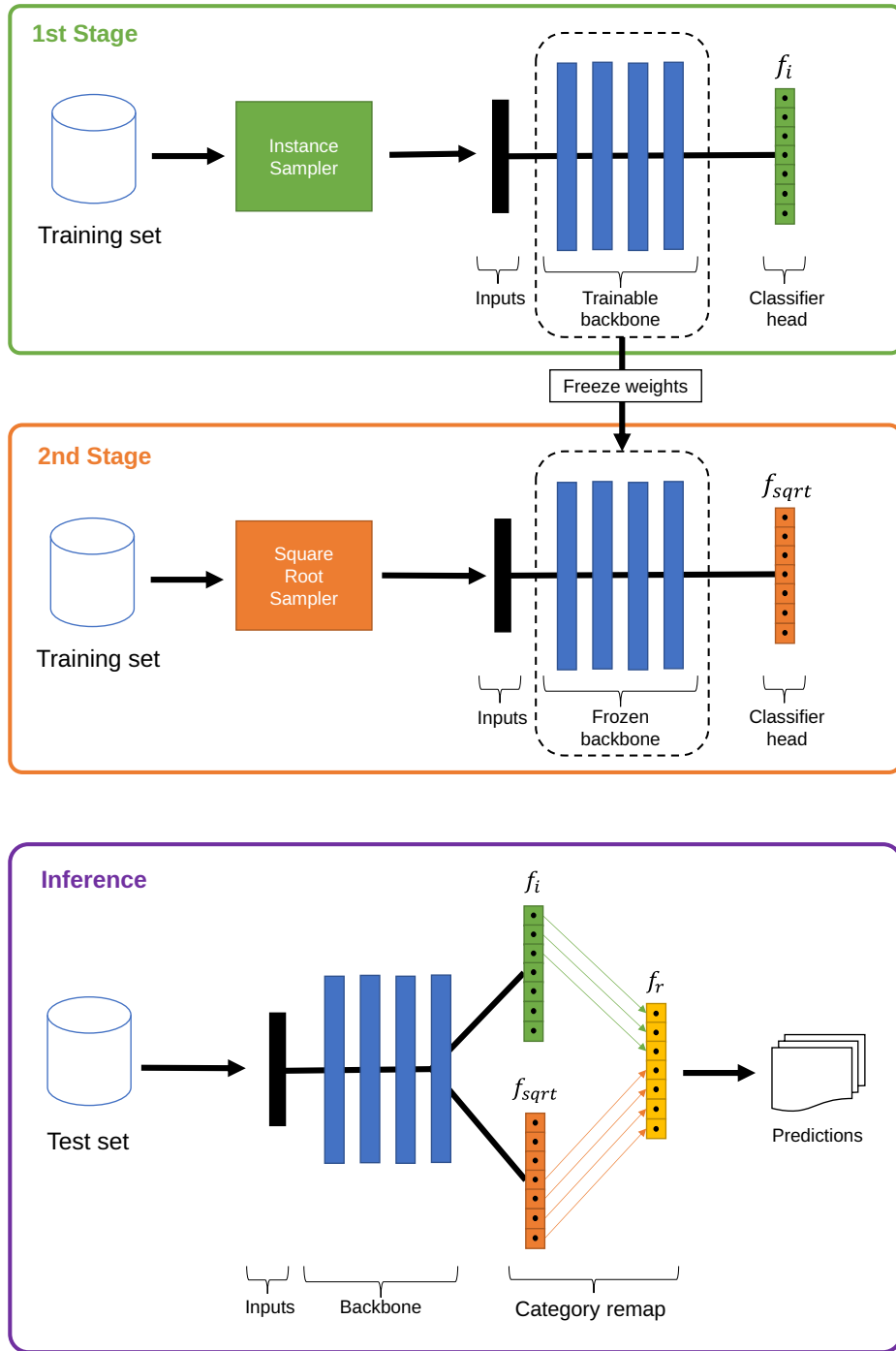


Figure 5.3. Framework of our Square-root Sampling Branch (SSB). It is trained in two stages. In the first stage, the classifier (f_i) and the backbone are jointly trained using the original data distribution. In the second stage, a new classifier (f_{sqr}) is trained using square-root sampling and the backbone trained in the first stage is kept frozen. During the inference, predictions from both classifier heads are aggregated by masking the classes to keep the head classes probabilities of f_i and the remaining classes probabilities of f_{sqr} .

compared to single-stage training methods. The results presented in this section were obtained using the test sets specified for each dataset in [Table 5.1](#).

5.3.1 Comparison between SSB and two-stage methods for long-tail visual recognition

In the first experiment, the effectiveness of our proposed SSB method when using the full image as input for the training and testing stages was evaluated. As our proposed method is trained using a two-stage strategy, in this experiment, only the results with two-stage training methods were compared. Initially, each architecture was trained using our baseline training settings, and then each long-tail method was applied to the classifier layer, though keeping the backbone frozen during the second-stage training. This allows a fair comparison between the methods in terms of the representation learned by the backbone.

The accuracy results are shown in [Table 5.3](#). The proposed SSB consistently improved accuracy for Bin_1 , Bin_2 , and Bin_3 when compared to the baseline for almost all cases, with a minimal loss in the head class accuracy (Bin_4), and achieved the best or the second-best accuracy per bin for most of the cases. The best results for Bin_1 , Bin_2 , and Bin_3 were achieved by the square-root sampling, with an improvement ranging from 3% to more than 15% over the baseline. However, such an improvement came at the cost of reducing the accuracy of the majority classes (Bin_4) by at least 3%. This aspect may be observed in the overall accuracy, since the square-root sampling achieved the worst results among all the methods. On the other hand, BAGS achieved the best overall accuracy results for some cases; while, in others, it even reduced the accuracy for the tail classes compared to the baseline. All methods, however, failed to improve the performance on tail classes (Bin_2) for the smaller datasets (Caltech and Wellington). In this case, accuracies were close to zero.

Table 5.3. Comparison between SSB accuracy with different methods for long-tail classification of different models and datasets when using the entire image as the input. Bold and underlined values, respectively, indicate the best and second-best result for each model and dataset combination. Acc_i is the top-1 accuracy for the Bin_i . As some category bins do not occur in some of the datasets, in this case, the corresponding column was removed.

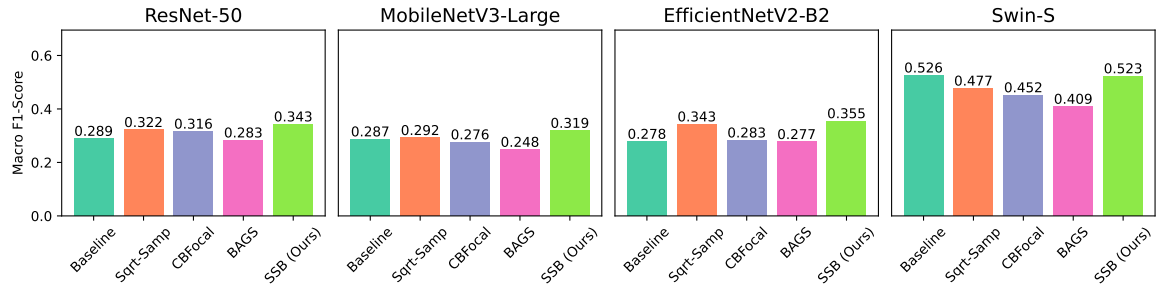
Model	Training	WCS					Snapshot Serengeti				Caltech CT			Wellington CT			
		Acc_1	Acc_2	Acc_3	Acc_4	Acc_{all}	Acc_2	Acc_3	Acc_4	Acc_{all}	Acc_2	Acc_4	Acc_{all}	Acc_2	Acc_3	Acc_4	Acc_{all}
ResNet-50	Baseline	0.00	13.05	32.06	85.02	<u>81.53</u>	2.90	44.82	92.96	<u>92.79</u>	0.00	44.95	42.19	0.0	4.58	75.66	72.23
	Sqrt-Samp	16.25	28.94	48.98	75.38	73.47	<u>6.52</u>	51.58	87.29	87.16	0.00	39.59	37.16	0.0	<u>10.65</u>	70.91	68.01
	CBFocal	5.00	14.42	40.61	83.41	80.43	0.72	38.66	92.88	92.69	0.00	44.47	41.74	0.0	4.84	75.12	71.73
	BAGS	16.25	14.19	34.84	84.43	81.14	9.42	41.27	93.12	92.94	1.56	48.04	45.19	0.0	25.92	75.00	72.64
	SSB (Ours)	<u>15.00</u>	<u>21.79</u>	<u>40.97</u>	<u>84.85</u>	81.91	5.07	48.16	92.91	92.75	0.03	46.09	43.26	0.0	5.51	75.62	72.24
MBNetV3	Baseline	0.00	11.92	36.52	84.24	<u>80.98</u>	2.17	29.78	92.73	<u>92.52</u>	0.00	46.14	43.31	0.0	5.70	75.65	<u>72.28</u>
	Sqrt-Samp	<u>7.50</u>	28.26	46.85	76.71	74.61	<u>4.35</u>	43.95	87.62	87.47	0.16	37.76	35.45	0.0	<u>21.68</u>	70.45	68.10
	CBFocal	0.00	9.19	36.98	79.41	76.42	0.72	29.78	92.44	92.23	0.00	41.27	38.74	0.0	4.58	74.94	71.55
	BAGS	10.00	10.44	28.68	<u>84.09</u>	80.48	5.07	28.78	92.19	91.97	0.00	43.96	41.26	0.0	23.41	74.02	71.58
	SSB (Ours)	1.25	20.32	<u>40.43</u>	<u>83.99</u>	81.04	3.62	36.97	92.72	92.53	<u>0.11</u>	45.93	43.12	0.0	13.54	<u>75.34</u>	72.36
EffV2-B2	Baseline	0.00	10.22	32.31	86.35	82.75	2.90	53.71	93.82	93.68	<u>0.08</u>	56.36	52.91	0.0	5.70	<u>77.11</u>	73.67
	Sqrt-Samp	10.00	22.36	50.34	78.24	76.12	<u>6.52</u>	59.25	90.46	90.35	0.00	51.15	48.01	0.0	<u>8.06</u>	74.14	70.95
	CBFocal	0.00	4.20	35.55	87.20	83.62	0.00	46.51	<u>94.51</u>	<u>94.34</u>	0.00	57.36	53.83	0.0	4.09	77.31	<u>73.78</u>
	BAGS	1.25	7.15	32.99	<u>87.01</u>	83.37	7.25	51.45	94.77	94.62	0.26	58.88	55.28	0.0	30.05	76.00	73.79
	SSB (Ours)	<u>7.50</u>	<u>18.39</u>	<u>42.39</u>	<u>86.18</u>	<u>83.17</u>	5.80	<u>55.96</u>	93.81	93.67	<u>0.08</u>	59.82	56.15	0.0	6.75	77.01	73.62
Swin-S	Baseline	8.75	37.12	62.99	90.86	88.76	10.87	60.42	95.09	94.97	4.17	68.94	64.96	0.0	7.61	78.94	<u>75.51</u>
	Sqrt-Samp	16.25	44.84	67.22	83.49	82.13	13.04	64.85	89.08	88.99	1.08	64.95	61.02	0.0	<u>13.09</u>	76.54	<u>73.48</u>
	CBFocal	0.00	25.09	56.57	90.32	87.78	4.35	53.71	94.97	94.82	3.78	69.61	65.57	0.0	5.44	78.49	74.98
	BAGS	6.25	21.68	52.33	90.08	87.32	15.22	52.32	94.81	94.67	3.67	66.01	62.18	0.0	19.65	78.62	75.78
	SSB (Ours)	16.25	<u>40.18</u>	<u>64.91</u>	<u>90.65</u>	<u>88.69</u>	<u>11.59</u>	<u>61.94</u>	<u>95.05</u>	<u>94.93</u>	1.08	67.57	63.49	0.0	7.16	<u>78.86</u>	75.41

In terms of the macro-averaged F1-score, SSB achieved a score that was higher than the other methods for most datasets and architectures, as shown in [Figure 5.4](#). For all the datasets, the Swin-S trained with the baseline settings attained the best macro F1-score among all architectures, with Swin-S trained with SSB having the second-best result. [Figure 5.5\(a\)](#) shows the increment/decrement of the F1-score class-wise over the baseline for ResNet-50 in the WCS dataset after applying each long-tail method. SSB is the method with the most consistent increment in terms of F1-score across classes for ResNet-50. [Figure 5.5\(b\)](#) shows that the decrease in F1-score for the Swin-S is generalized over all classes, with SSB being the method that lowers the F1-score the least.

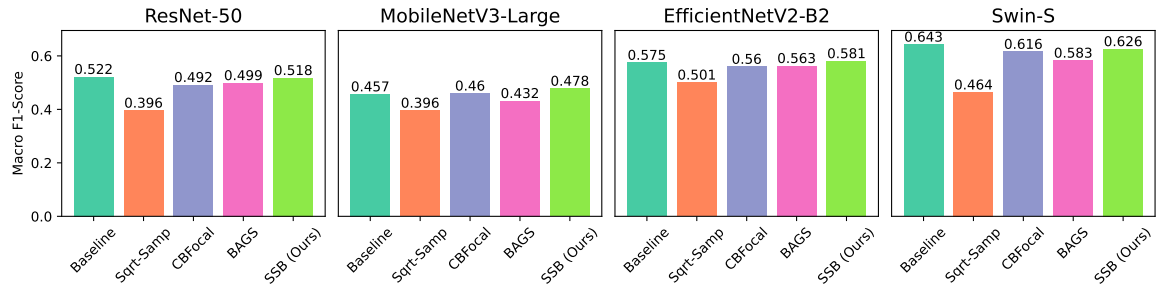
5.3.2 Comparison between methods when using cropped images

In this experiment, an assessment was made to see whether using an object detector before classifying an animal can improve the recognition score for the tail classes. The same settings from the previous experiment were used, except for using MegaDetectorV4 to generate cropped regions around animals as inputs, though reverting back to the whole image when the confidence score was less than 0.6. For the pre-cropping experiments, the Snapshot Serengeti dataset was not used due to the time required to generate bounding boxes with MegaDetectorV4 and run the experiments with the large quantity of images available.

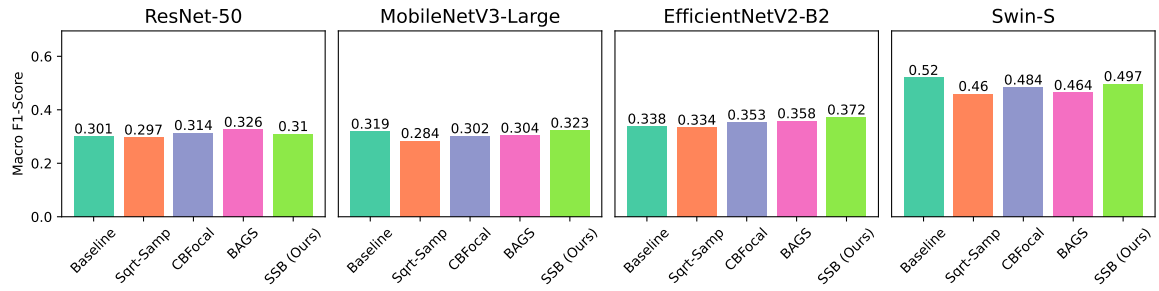
As shown in [Table 5.4](#), there was a general accuracy and macro F1-score increase when the models were trained using cropped images. However, these results follow the same pattern observed when using the entire image as the input. Specifically, square-root sampling attained the highest improvements for the tail classes at the cost of reducing performance for the head classes. Our proposed SSB was able to improve the tail classes' accuracy with minimal accuracy loss for the head classes. In addition, it attained the best macro F1-score in most of the cases. Lastly, [Figure 5.6](#) shows some sample predictions for pictures randomly picked from the WCS dataset with bounding boxes generated by MegaDetectorV4.



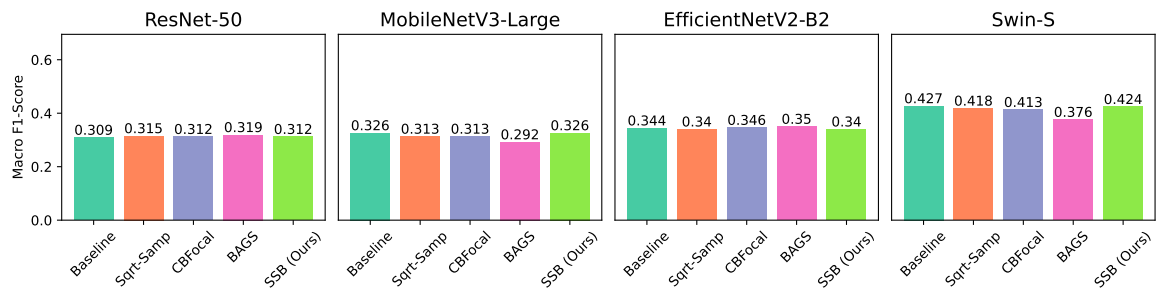
(a) WCS Camera Traps



(b) Snapshot Serengeti



(c) Caltech Camera Traps



(d) Wellington Camera Traps

Figure 5.4. Comparison in terms of macro-averaged F1-score for all combinations of datasets and models. SSB achieved a score that was higher than the other long-tail visual recognition methods. It is worth mentioning that for the Swin-S model, the baseline obtained the best macro F1-score for all the datasets. SSB is the second best.

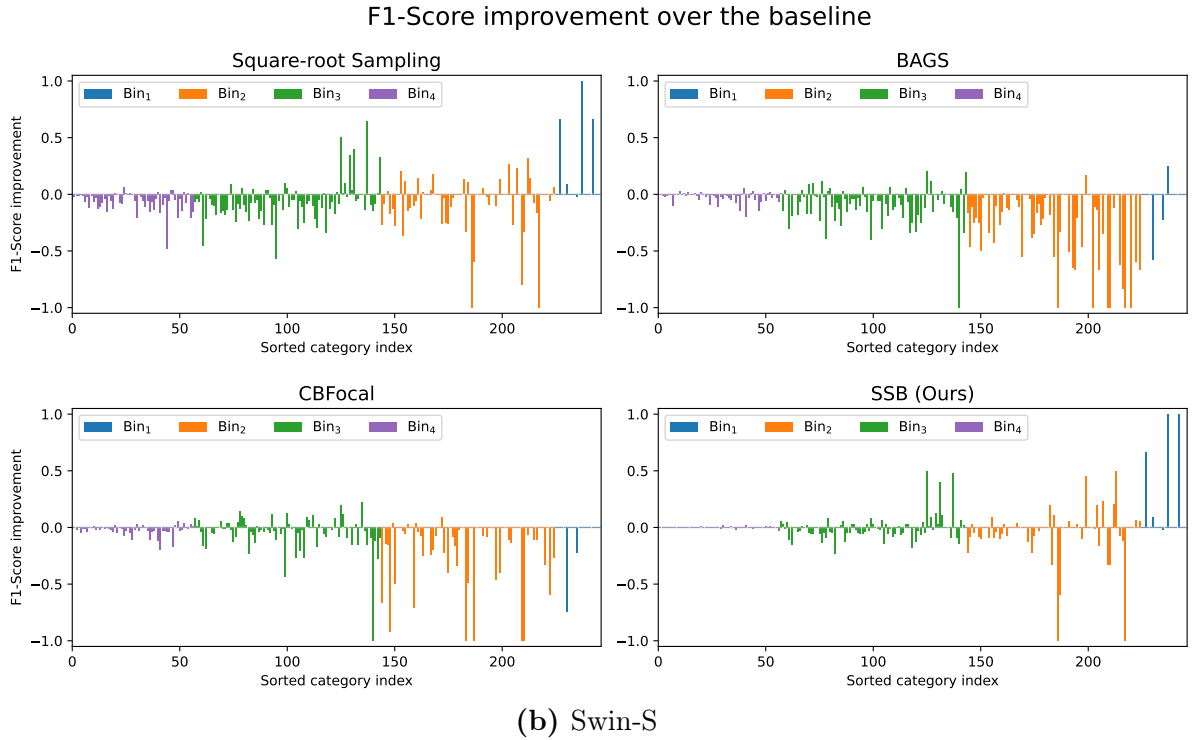
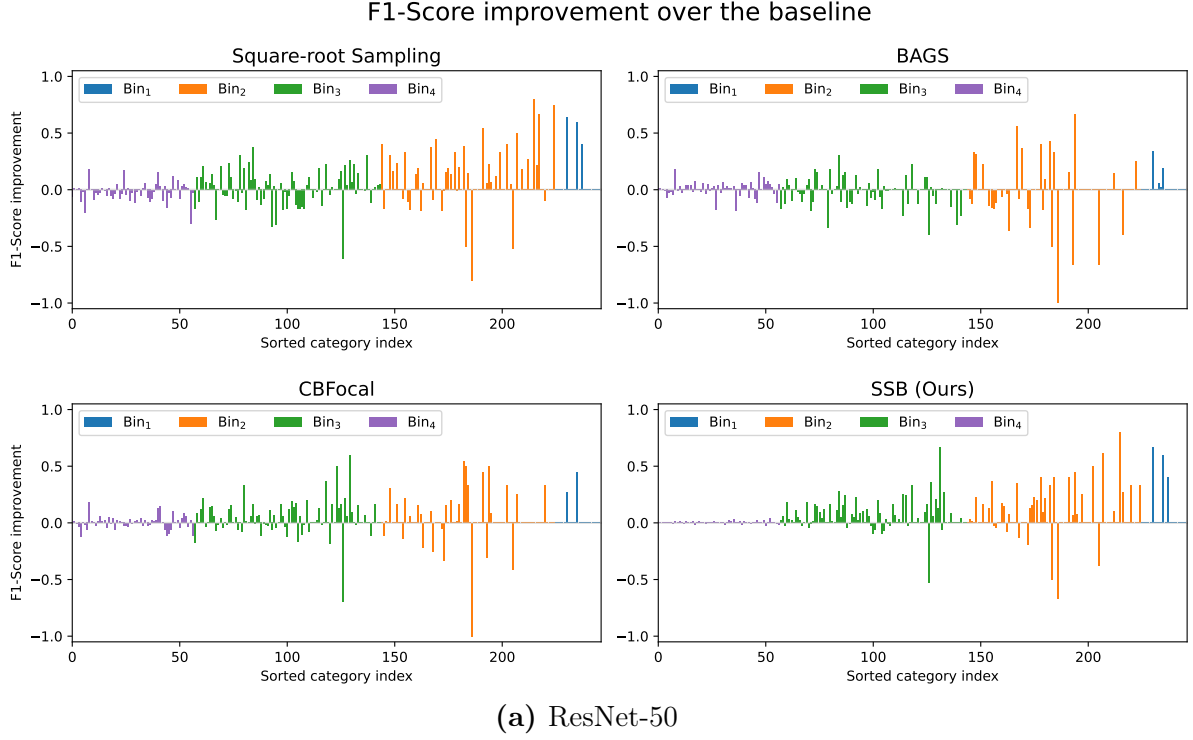


Figure 5.5. Breakdown of F1-score improvement per class in the WCS dataset for the long-tail visual recognition methods over the baseline training procedure. For ResNet-50, our proposed SSB is the most consistent method to increase F1-score throughout the classes. Similar behavior occurs for MobileNetV3-Large and EfficientNetV2-B2 models (not included in the graph). However, for the Swin-S architecture, all the evaluated methods degraded the F1-score, but SSB is the one that lowers the F1-score the least.

Table 5.4. Results attained by different methods and models on different datasets when using the bounding box with the highest confidence as input. Bounding boxes were generated by MegaDetectorV4. Bold and underlined values, respectively, indicate the best and second-best result for each model and dataset combination.

Model	Training	WCS						Caltech CT				Wellington CT				
		Acc_1	Acc_2	Acc_3	Acc_4	Acc_{all}	F_1^m	Acc_2	Acc_4	Acc_{all}	F_1^m	Acc_2	Acc_3	Acc_4	Acc_{all}	F_1^m
ResNet-50	Baseline	0.00	21.68	52.94	90.61	87.85	0.429	0.00	60.59	56.87	0.423	0.0	8.18	<u>78.52</u>	75.14	0.408
	Sqrt-Samp	<u>20.00</u>	43.47	67.19	85.79	84.28	0.473	0.24	62.62	58.79	0.448	0.0	<u>22.21</u>	75.32	72.76	0.431
	CBFocal	3.75	26.33	61.67	<u>90.98</u>	88.65	0.468	0.00	64.32	60.37	0.458	0.0	10.77	78.47	75.21	<u>0.420</u>
	BAGS	26.25	23.16	53.33	91.00	<u>88.28</u>	0.414	<u>0.13</u>	67.01	62.91	<u>0.464</u>	0.0	36.98	77.92	75.95	0.409
	SSB (Ours)	17.50	<u>35.41</u>	<u>61.95</u>	<u>90.39</u>	<u>88.25</u>	0.501	0.11	<u>65.79</u>	<u>61.76</u>	0.467	0.0	12.23	78.73	<u>75.53</u>	0.416
MBNetV3	Baseline	0.00	26.11	55.93	91.34	<u>88.72</u>	0.434	0.00	64.80	60.82	<u>0.471</u>	0.0	11.89	<u>80.02</u>	76.74	0.434
	Sqrt-Samp	26.25	42.22	64.41	87.30	<u>85.57</u>	0.458	<u>0.29</u>	57.41	53.91	0.436	0.0	<u>23.89</u>	<u>75.67</u>	73.17	0.441
	CBFocal	1.25	19.64	56.86	91.04	88.39	<u>0.418</u>	0.13	61.29	57.53	0.468	0.0	11.44	79.30	76.03	0.422
	BAGS	13.75	22.47	45.07	90.13	87.07	0.373	0.42	<u>65.10</u>	<u>61.12</u>	0.461	0.0	38.60	79.31	77.34	0.403
	SSB (Ours)	<u>21.25</u>	<u>35.19</u>	<u>60.35</u>	<u>91.13</u>	88.88	0.498	0.11	66.59	62.50	0.489	0.0	15.98	80.30	<u>77.20</u>	<u>0.438</u>
EffV2-B2	Baseline	0.00	19.64	48.13	90.46	87.46	0.383	<u>0.11</u>	61.34	57.58	0.424	0.0	8.40	<u>79.61</u>	76.18	0.427
	Sqrt-Samp	17.50	35.19	65.09	88.37	86.49	<u>0.469</u>	0.08	60.33	56.63	0.437	0.0	<u>23.33</u>	<u>76.86</u>	74.28	0.452
	CBFocal	0.00	10.33	53.58	91.34	88.40	0.392	0.00	64.72	60.74	<u>0.468</u>	0.0	12.53	79.39	76.17	0.430
	BAGS	0.00	13.17	51.87	<u>91.24</u>	<u>88.26</u>	0.384	0.61	68.03	63.89	0.478	0.0	44.22	79.15	77.46	0.429
	SSB (Ours)	<u>16.25</u>	<u>30.31</u>	<u>60.28</u>	<u>90.38</u>	<u>88.10</u>	0.474	0.08	<u>67.10</u>	<u>62.98</u>	0.465	0.0	12.27	79.80	<u>76.55</u>	<u>0.439</u>
Swin-S	Baseline	16.25	53.35	78.20	<u>93.63</u>	92.28	0.637	<u>10.32</u>	<u>70.49</u>	<u>66.80</u>	0.597	0.0	11.25	<u>80.14</u>	<u>76.82</u>	0.503
	Sqrt-Samp	25.00	62.09	79.37	90.18	89.22	0.586	2.90	67.84	63.85	0.524	0.0	19.05	78.42	<u>75.56</u>	<u>0.504</u>
	CBFocal	8.75	39.95	75.24	93.66	91.97	<u>0.599</u>	11.40	69.41	65.84	<u>0.596</u>	0.0	9.41	79.25	75.88	0.476
	BAGS	8.75	33.83	69.93	93.52	91.52	0.511	8.29	73.86	69.84	<u>0.560</u>	0.0	<u>18.64</u>	79.36	76.44	0.447
	SSB (Ours)	<u>23.75</u>	<u>58.34</u>	<u>78.55</u>	<u>93.53</u>	92.28	0.637	3.27	69.52	65.45	0.576	0.0	12.34	80.75	77.46	0.514

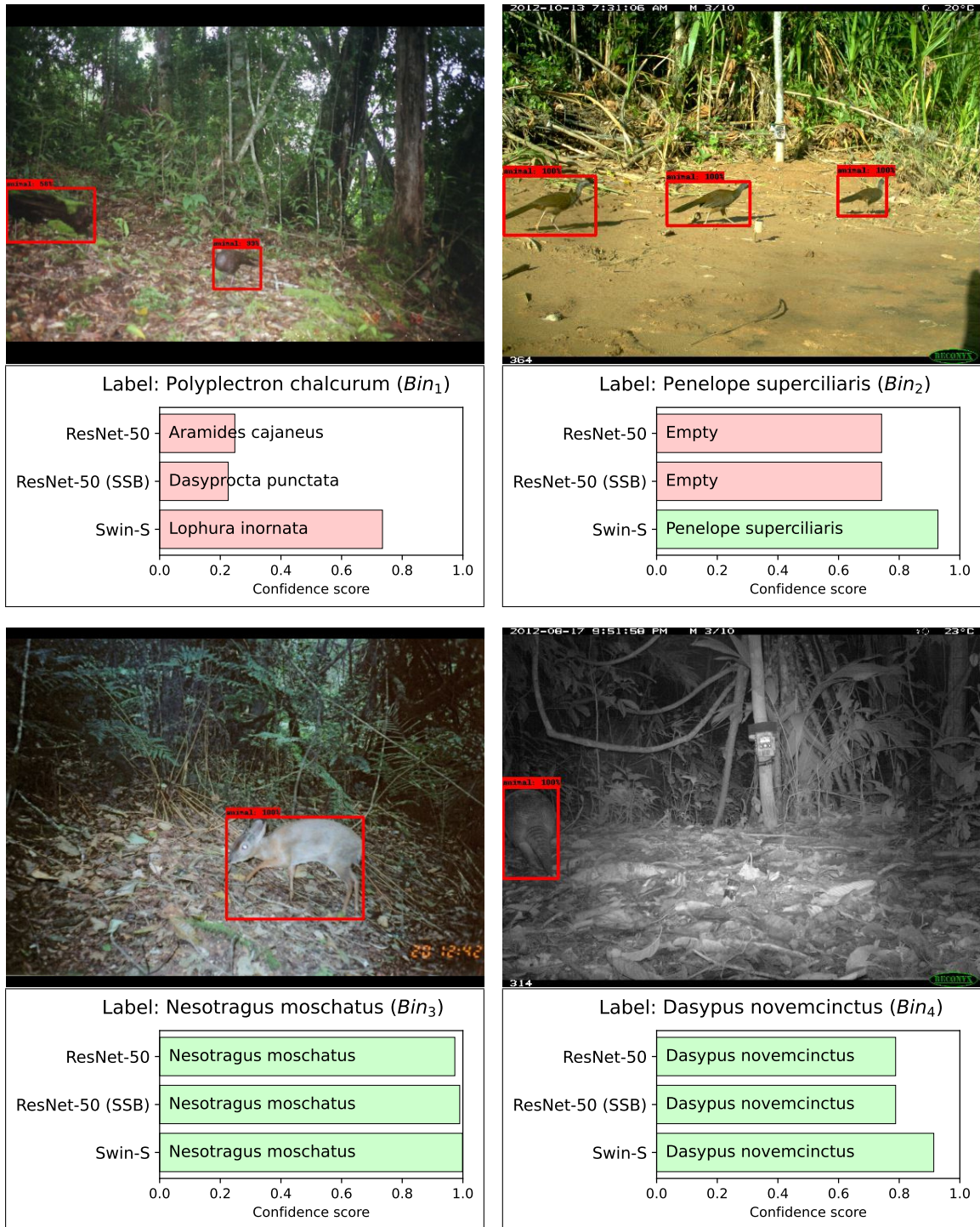


Figure 5.6. Sample classification results for the baseline ResNet-50, ResNet-50 with the proposed SSB, and Swin-S. One picture was randomly picked from each bin of categories from the WCS dataset. Bounding boxes were generated using MegaDetectorV4.

5.3.3 Comparing SSB to one-stage training strategies

In this experiment, the performance of single-stage training methods was evaluated by applying re-weighting (CBFocal) and re-sampling (square-root sampling). For both methods, cropping was also used before classification, with bounding boxes being generated by MegaDetectorV4. Table 5.5 shows the results and compares them to the baseline training procedure and to our proposed two-stage SSB. The results show that square-root sampling presents similar behavior as observed in the two-stage training, i.e., improving the performance of the tail classes, but reducing the performance of the head classes. On the other hand, CBFocal obtained a great improvement over its results in two-stage training, and achieved the best overall accuracy in most cases. However, this improvement occurred mostly in the head classes. One-stage training methods also fail to improve the classification performance of the tail classes from the Caltech and Wellington datasets. Considering the objective of improving accuracy of the tail classes, our proposed SSB still represents the best trade-off between improving accuracy of the tail classes and reducing performance in the head classes.

Table 5.5. Comparing the results of one-stage training using square-root re-sampling and CBFocal loss with the baseline training procedure and our proposed SSB. Bold and underlined values, respectively, indicate the best and second-best result for each model and dataset combination.

Model	Training	WCS						Caltech CT				Wellington CT				
		Acc_1	Acc_2	Acc_3	Acc_4	Acc_{all}	F_1^m	Acc_2	Acc_4	Acc_{all}	F_1^m	Acc_2	Acc_3	Acc_4	Acc_{all}	F_1^m
ResNet-50	Baseline	0.00	21.68	52.94	90.61	87.85	0.429	0.00	60.59	56.87	0.423	0.0	8.18	78.52	75.14	0.408
	Sqrt-Samp	20.00	45.86	71.61	84.07	82.90	<u>0.530</u>	0.90	64.87	60.94	<u>0.529</u>	0.0	<u>13.58</u>	77.29	74.22	<u>0.458</u>
	CBFocal	15.00	35.98	<u>67.58</u>	92.13	90.15	0.538	0.05	71.67	67.27	0.541	0.0	15.00	79.07	75.99	0.460
	SSB (Ours)	<u>17.50</u>	35.41	61.95	90.39	88.25	0.501	0.11	65.79	61.76	0.467	0.0	12.23	78.73	<u>75.53</u>	0.416
MBNetV3	Baseline	0.00	26.11	55.93	91.34	88.72	0.434	0.00	64.80	60.82	0.471	0.0	11.89	80.02	76.74	0.434
	Sqrt-Samp	22.50	48.13	71.75	84.85	83.67	0.504	1.08	58.20	54.69	<u>0.487</u>	0.0	14.55	76.35	73.37	0.459
	CBFocal	3.75	24.86	61.99	91.69	89.31	0.463	0.08	64.23	60.29	0.485	0.0	15.64	80.11	77.00	0.452
	SSB (Ours)	<u>21.25</u>	<u>35.19</u>	<u>60.35</u>	91.13	88.88	0.498	0.11	66.59	62.50	0.489	0.0	15.98	80.30	77.20	0.438
EffV2-B2	Baseline	0.00	19.64	48.13	90.46	87.46	0.383	0.11	61.34	57.58	0.424	0.0	8.40	79.61	76.18	0.427
	Sqrt-Samp	26.25	56.64	78.09	85.66	84.84	0.565	2.61	61.37	57.76	0.500	0.0	16.54	77.07	74.15	0.460
	CBFocal	0.00	13.85	60.24	92.37	89.72	0.410	0.00	71.39	67.00	0.529	0.0	18.60	80.75	77.76	0.471
	SSB (Ours)	<u>16.25</u>	<u>30.31</u>	<u>60.28</u>	90.38	88.10	0.474	0.08	67.10	62.98	0.465	0.0	12.27	79.80	76.55	0.439
Swin-S	Baseline	16.25	53.35	78.20	93.63	92.28	0.637	10.32	<u>70.49</u>	<u>66.80</u>	<u>0.597</u>	0.0	11.25	80.14	76.82	0.503
	Sqrt-Samp	23.75	51.87	78.62	92.03	90.78	0.634	2.16	67.71	63.68	0.574	0.0	11.29	79.36	76.08	0.479
	CBFocal	<u>21.25</u>	<u>54.03</u>	78.41	93.97	92.62	0.642	<u>6.76</u>	71.62	67.63	0.606	0.0	13.43	<u>80.56</u>	<u>77.33</u>	0.524
	SSB (Ours)	23.75	58.34	<u>78.55</u>	93.53	92.28	0.637	3.27	69.52	65.45	0.576	0.0	<u>12.34</u>	80.75	77.46	<u>0.514</u>

5.4 Discussion

The long-tail distribution is a characteristic of real-world tasks, such as visual recognition of species in camera-trap images, in which some classes concentrate the majority of available instances and a large number of classes have just a small quantity of samples. However, as indicated by Schneider et al. [130], computer vision models generally

often perform poorly on recognizing species in camera-trap images with fewer training instances (tail classes). In this Chapter, a simple and effective framework called Square-root Sampling Branch (SSB) is proposed, which can improve the performance in tail classes with a minimal percentage of accuracy loss in the head classes. The proposed framework was systematically evaluated against state-of-the-art methods for handling the long-tail distribution issue in four different camera-trap datasets and four families of computer vision models. Our experiments show that our proposed framework presents competitive results with the methods in the literature, achieving the best trade-off between increasing the performance of the tail classes and reducing the head classes' accuracy in most cases.

Comparing our improved baseline training procedure with the literature, ResNet-50 achieved an overall accuracy of 92.79% for the Snapshot Serengeti dataset using the full image, while the same architecture achieved an accuracy of 93.6% in the work conducted by Norouzzadeh et al. [107] and approximately 85% in that of Villa et al. [163]. However, while these works use burst or random split, in our methodology we split the training/test partitions based on locations, which tends to achieve a significantly lower accuracy, as demonstrated for other datasets by Beery et al. [14], Schneider et al. [130], and Tabak et al. [149], in which the performance of models on untrained locations drops from 15% to 25% in accuracy in terms of absolute values. Furthermore, the categories and test sets used are different among the works, thus making the results not directly comparable.

The same occurs when comparing our results for the Caltech CT dataset with the results from Beery et al. [14]. The lack of well-defined benchmarks for visual recognition of animal species in camera-trap images makes it difficult to compare the results of different methods across works. Although most of the applications have their specificities and requirements regarding the data preparation or even the target dataset, it would be important to apply them using a standardized benchmark as well. However, designing such a benchmark dataset is challenging due to the wide variety of factors that need to be considered, including taxonomic coverage, biogeography, habitat and climate conditions, as well as the tasks to be included, such as species classification, long-tail visual recognition, animal detection, counting, and behavior identification. Therefore, it is recommended that the research community work collaboratively towards developing standardized benchmarks for automatic information extraction from camera-trap images, which would facilitate comparisons between different methods and ultimately lead to advancements in this field.

In general, the method that most improved the performance for the tail classes over the baseline was the square-root sampling. However, this method strongly de-

grades the head classes’ accuracy. This behavior may be due to the high imbalance of the datasets, leading the square-root sampling to cause a high undersampling of the majority classes. Our proposed SSB can improve the accuracy for the tail classes more than CBFocal and BAGS for most of the cases, but, unlike square-root sampling, the degradation caused to the head classes is minimal, representing the best trade-off among all the evaluated methods. SSB also represents the best performance in terms of macro F1-score for most of the cases when compared with the two-stage methods. On the other hand, our experiments using one training stage show that CBFocal achieved the best overall accuracy for most of the cases, primarily because of its superior performance in the head classes.

This CBFocal performance improvement may indicate that pretraining using softmax cross-entropy does not generate a good representation for fine-tuning using Focal loss, or that the learning of representation and classification is not well decoupled when using this loss function. However, our experiments were not designed to assess the performance based on representation (feature extractor) training strategies. Furthermore, Kang et al. [76] do not evaluate the performance of re-weighting methods when proposing the two-stage training approach. Although this approach was not explored in our experiments, a potential alternative could be to combine the outcomes of square-root resampling and CBFocal, which respectively performed better on the tail and head classes. However, this approach would require running inference using two models, effectively doubling the computational cost. In contrast, our proposed SSB only needs to be run once and is still competitive.

Looking into the model’s performance, Swin Transformer obtained the best results considering all the datasets, which was expected since this model has more parameters than the other selected models. It is worth noting that the Swin-S that was trained using the baseline settings attained an excellent performance even for the tail classes, reaching the best mean F1-score in most of the datasets. This superior performance of vision transformers in long-tail classification corroborates with what has been stated in the literature, as in the classification of fungi [112], for instance.

The results also suggest a possible boundary for the applicability of SSB. The benefit of explicitly addressing the long-tail distribution appears to decrease as the underlying visual representation becomes stronger. In our experiments, SSB provided more consistent improvements for ResNet-50, MobileNetV3-Large, and EfficientNetV2-B2, whereas the Swin-S baseline already achieved the best macro F1-score across all four datasets, with SSB obtaining the second-best results. This observation is consistent with the hypothesis that stronger backbones may learn representations that are sufficiently robust to the long-tailed distribution, reducing the additional benefit

provided by SSB. However, the experiments in this thesis were not designed to isolate representation quality from architectural capacity, and therefore this interpretation should not be considered conclusive.

Regarding the question of whether using cropping before classification can improve model performance, when comparing the results of the first and the second series of experiments, it is possible to note a significant increase in the overall accuracy when applying cropping. This improvement is more relevant for the WCS and Caltech datasets, in which, for some cases, there was an improvement of 10% and 15%, respectively. This performance improvement is consistent with the results in the literature, for example, in [14], there was a 20% increase in the accuracy when cropping was used before classification. The use of an object detector as a preprocessing step before species identification is also part of the methodology of Schneider et al. [131] and Norouzzadeh et al. [106]; however, its impact on the performance is not evaluated. Looking at the binned accuracy results, a performance increase is observed for all bins of the classes, except for the tail classes (Bin_2) for the Caltech CT and Wellington CT datasets, in which all models and methods fail to learn them. The lowest performance increases due to the use of cropping before classification were observed for the Swin-S model, with an improvement varying from 1% to 4%. It is worth noting that the accuracies for Swin-S using the full images are already very high when compared to the other models. This can be explained by the vision transformer’s attention mechanism that can allow it to focus on the animal even when using the full image.

Moreover, our experiments did not explain why the models were unable to learn to recognize the tail classes in the smaller datasets. The Caltech and Wellington datasets, with only 19 and 14 classes respectively, had a limited number of tail classes (4 and 1, respectively). It is possible that these tail classes were ignored due to the relatively low number of training samples available. To investigate this further, future studies could use the WCS dataset and progressively remove classes to evaluate the impact on model performance. This would also provide insights into the minimum number of training images per class needed to see an improvement in performance when using the evaluated methods.

One limitation to our study is that only the WCS dataset has classes belonging to all bins of classes in regards to the number of training instances. Another limitation is that, during our experiments, there was an effort to keep our settings as similar as possible across the methods and models in order to make a fair comparison; therefore, our hyperparameters are suboptimal and better tuning would lead to improvements.

5.5 Final Remarks

Although rare or endangered species may be the classes of interest for projects using camera traps for monitoring wildlife, they are often neglected when developing deep-learning models for automatic species recognition, mostly because of the difficulty involved in learning classes with few training instances from long-tail datasets. In this chapter, a simple and effective framework called Square-root Sampling Branch (SSB) is proposed, which uses a multi-branch approach to improve performance recognition on tail classes of long-tail camera-trap datasets with minimal loss in the performance recognition of the head classes. Our proposed method was systematically evaluated against square-root sampling, CBFocal loss, and Balanced Group Softmax (BAGS) considering four camera-trap datasets: Snapshot Serengeti, WCS, Caltech CT, and Wellington CT. Our results showed that our proposed approach achieved the best trade-off between improving accuracy for the tail classes and decreasing the accuracy of the head classes, and is competitive with the state-of-the-art methods. It was also found that Swin-S performed better for most of the datasets even without applying any additional method for handling imbalance, and achieved an overall accuracy of 88.76% for the WCS dataset and 94.97% for the Snapshot Serengeti, when considering location-based splitting. This work resulted in the publication of “Bag of Tricks for Long-Tail Visual Recognition of Animal Species in Camera-Trap Images” [34].

Despite the fact that our proposed framework improves the recognition of tail classes, our experiments clearly show that it is necessary to develop more sophisticated methods to handle the long-tail distribution in the recognition of species in camera-trap images. Specifically for the tail classes (Bin_1 and Bin_2), for which the results were extremely low, and especially for the small datasets, one could evaluate few-shot learning methods. Another issue that was discussed is the lack of standardized benchmarks for evaluating the performance of long-tail visual recognition of animal species when comparing new methods. It is hoped that our well-defined methodology for dataset preprocessing and training procedure helps establish good baselines for other researchers. In future works, other methods can be explored such as progressive-balanced sampling, Ally Complementary Experts, and Rebalanced Mixup (Remix). Additional training tricks, such as label smoothing and mixup, can also be performed, but they must be used with caution because one trick may affect the other.

The next chapter explores counting the number of individuals of each species in a sequence of camera-trap images. For this task, a good model for species identification that can handle long-tail distributions is essential, since the task considers not only the total number of individuals but also whether the count is correct for each species.

PuTR-CouT: Counting-by-Tracking in Camera-Trap Images

In this chapter, we present a strategy built upon a transformer-based tracking model trained on a synthetic dataset to count the number of animals of each species in camera-trap image sequences. We evaluate it on the iWildCam 2021 benchmark against state-of-the-art multi-object tracking methods, as well as a strong baseline based on the maximum bounding box count. Although effective, this heuristic is limited to predicting a single species per sequence, highlighting the advantages of the counting-by-tracking formulation, which achieves competitive results.

6.1 Introduction

Camera traps have been widely used for monitoring wildlife, collecting large amounts of images worldwide. In the last decade, the computer vision community has investigated diverse approaches to improve automated information extraction from this data, speeding up data preprocessing and reducing or eliminating the need for manual data processing. This allows ecologists to focus on their ecological research questions instead of spending time on manually reviewing images. One of the widely studied data extraction tasks is species identification, which provides information that enables scientists to model species richness, species occurrence, and species distribution [11]. However, solely identifying the species is not sufficient for other types of modeling, such as those focused on species abundance or density. For these cases, counting the number of individuals captured across image sequences is required.

Counting the number of individuals in a single image can be done using an object detector such as MegaDetector. However, most camera trap projects capture more

than one picture per capture event in the hope of obtaining a good shot to identify the photographed animal. In this context, simply counting the number of bounding boxes in the pictures can overestimate or underestimate the actual number of individuals, as the same individual may appear in multiple images, or different individuals may appear in different images of the same sequence (see Figure 6.1). Thus, it is necessary to use more sophisticated approaches, such as tracking individuals or re-identifying them across the images in the sequence. However, this is a challenging task due to temporal discontinuities, as the images are usually taken at intervals of one second or more, and most public camera trap datasets do not include count labels, making supervised training unfeasible.



Figure 6.1. Example of an image sequence in which simple bounding box counting leads to over- or underestimation, as the same individual may appear in multiple frames and different individuals may appear at different times.

Source: iWildCam 2021 challenge [11].

Some efforts have been made to address the counting of individuals in camera trap images. For instance, Norouzzadeh et al. [107] used a classifier to count the number of animals in the Snapshot Serengeti dataset. In a subsequent work, Norouzzadeh et al. [106] employed a pre-trained object detection model to count animals at the image level based on the number of bounding boxes. An effort to incentivize the development of methods for counting the number of individuals across camera trap image sequences was the launch of iWildCam 2021 Challenge [11], which provided a dataset for this problem. During the challenge, the top solutions relied on heuristics that counted bounding boxes, but were restricted to predicting a single species per sequence.

Recent work has demonstrated the potential of combining detection, multi-object tracking, and individual re-identification for wildlife monitoring, such as GorillaWatch [127]. However, this work focuses on a single species using video, whereas this chapter considers multi-species counting in low-frame-rate camera-trap image sequences, where tracking annotations are typically unavailable.

In this chapter, we present a new counting-by-tracking strategy for counting animals in sequences of camera trap images using a transformer-based multi-object

tracking model. The main contributions of this chapter are:

- An improved maximum bounding box counting heuristic leveraging state-of-the-art detection and classification models, enhancing counting performance on the iWildCam 2021 Challenge benchmark.
- A novel strategy for generating synthetic labeled data for multi-object tracking that takes advantage of camera trap image sequence priors, allowing supervised training of trackers with learned affinity modules.
- The development of a transformer-based counting-by-tracking framework that estimates the number of individuals using tracks, enabling multi-species predictions across the sequence and species classification based on multiple views of each individual, while providing track-level outputs that facilitate inspection and verification of the estimated counts.

6.2 Task definition and dataset

In this section, we describe the problem setting for the counting task in camera-trap image sequences and present the iWildCam 2021 dataset, which is used to validate our counting approach.

6.2.1 Counting task in camera trap image sequences

Automatically extracting animal counts from camera-trap image sequences is challenging due to several factors. First, the low frame rate, usually 1 frame per second, may lead to large displacements between consecutive frames, resulting in low bounding box overlap across frames and reduced temporal consistency. Second, significant appearance changes due to variations in pose, illumination, or occlusion may make re-identification across frames difficult. Third, most camera-trap datasets do not contain counting or tracking labels, making supervised training difficult. Although recent datasets such as GorillaWatch [127] provide tracking and individual re-identification annotations for specific wildlife monitoring scenarios, such resources remain limited in species coverage and do not directly address the multi-species, low-frame-rate camera-trap setting considered in this work.

In this context, the counting task in camera-trap image sequences considered in this chapter assumes that count labels are not available for training and that the frame rate is low, usually 1 fps. To encourage the development of innovative methods, Beery

et al. [11] built the iWildCam 2021 dataset (described below), for which counting labels were collected only for the test set, which remains private.

6.2.2 iWildCam 2021 dataset

The iWildCam 2021 dataset comprises three components: camera trap images from the WCS dataset, citizen science images from iNaturalist, and multispectral imagery for the camera trap locations. The camera trap data, provided by the Wildlife Conservation Society (WCS) [90], contains 263,528 images of 206 species from 414 locations across 12 countries worldwide. The official training/test split is based on camera location, with the training set containing 36,547 sequences totaling 203,314 images from 323 locations, and the test set containing 11,057 sequences totaling 60,214 images from 91 locations. As most publicly available camera trap data does not include count labels, the count labels for image bursts were collected exclusively for the test set to encourage the development of methods that can learn to count without explicit labels [11].

The dataset also includes a subset of the iNaturalist 2017-2019 competition dataset [64], with 13,051 additional images from 75 species. Although these images typically present a different data distribution and are of higher quality, they can still be valuable for improving classification.

Finally, the dataset includes raw remote sensing data for each camera location, collected by the Landsat 8 satellite, allowing investigation into whether the model performance can be improved using this kind of multimodal data. GPS coordinates for each camera location are provided, with most of them obfuscated for privacy and security reasons. Each location has been randomly adjusted to lie within 1 km of the original location center.

Unless otherwise stated, all models, synthetic data generation procedures, ablation studies, and evaluations presented in this chapter are based exclusively on the iWildCam 2021 dataset. External datasets are used only as sources of auxiliary priors or for discussion purposes.

6.3 Heuristic counting baseline

In this section, we detail our heuristic strategy that serves as a strong baseline for the problem of counting animals in camera-trap images, as well as an improved version leveraging state-of-the-art detection and classification models.

6.3.1 Maximum bounding box count heuristic

Our proposed heuristic baseline consists of two components: an image burst classifier based on an ensemble of EfficientNet-B2 [150] models and a counting heuristic based on the bounding boxes generated by MegaDetectorV4 [13]. The main idea is to estimate the number of animals present in the sequence based on the image with the highest number of bounding boxes (MaxBoxCount), which could provide a lower bound on the number of animals across the sequence if the animal detector achieves perfect detection (see Figure 6.2). Additionally, since the vast majority of the images contain only a single species, the simplest approach for classification is to assume a single species prediction across the entire sequence and use the most likely species prediction to the total count of individuals.

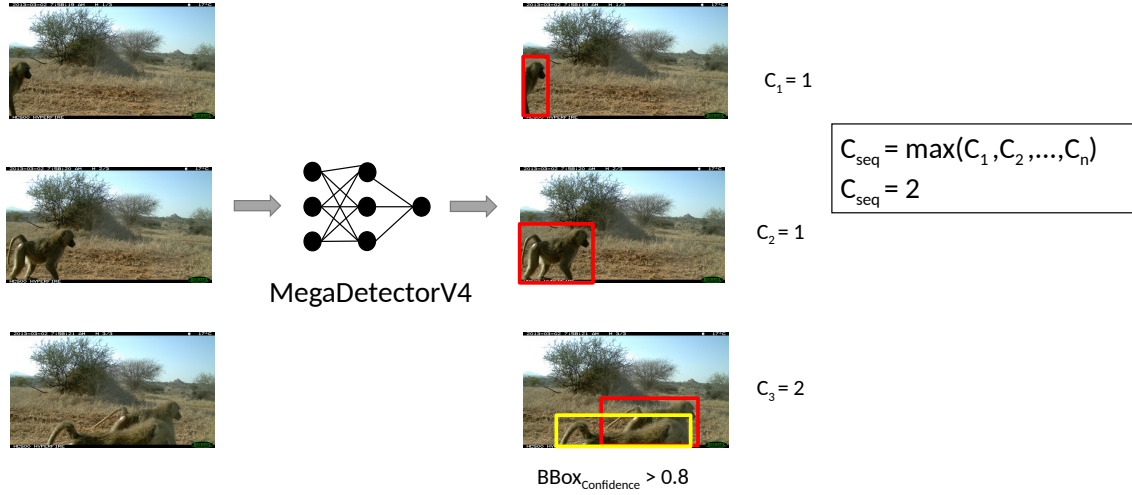


Figure 6.2. The MaxBoxCount heuristic estimates the count for a sequence as the number of bounding boxes in the image with the highest number of detections. Only bounding boxes with confidence higher than τ are considered (for our iWildCam 2021 challenge final submission, $\tau = 0.8$).

Initially, MegaDetectorV4 is run on all images to detect animals. Next, we run the image classifiers on both the full image and the bounding box with the highest confidence score. The image prediction is calculated as the weighted average of the predictions from both models, following the approach used by the winning solution of iWildCam 2020: 0.15 (full image) + 0.15 (mirrored full image) + 0.35 (bbox) + 0.35 (mirrored bbox). Figure 6.3 illustrates this fusion strategy. To predict the species in the sequence, we average the species predictions of the non-empty images in the burst.

Finally, we generate a prediction vector containing the number of individuals of each species – in our case, only a single species – based on the maximum number of bounding boxes with confidence score higher than a threshold τ across any image in

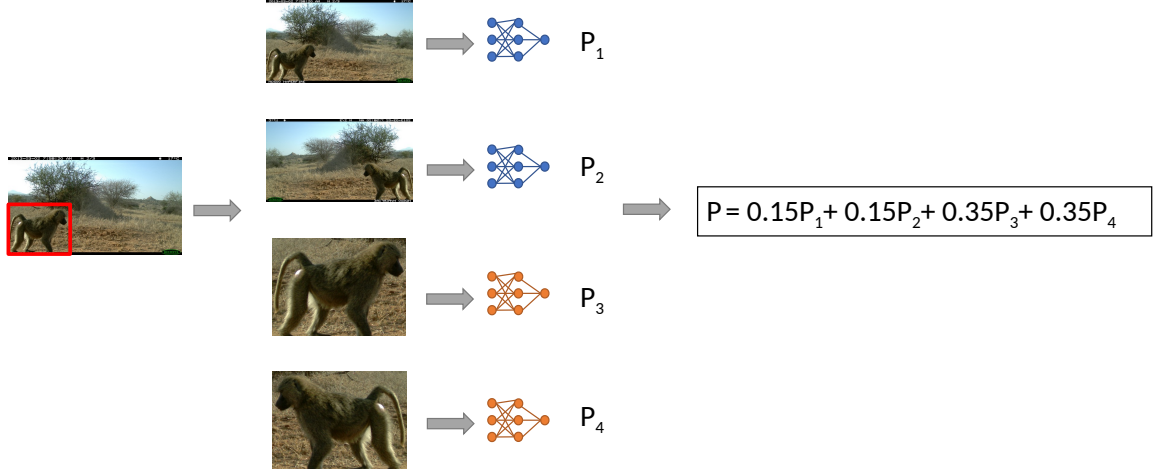


Figure 6.3. Fusion strategy for the classifier ensemble used in MaxBoxCount. The species prediction P for a given image is the weighted average of the predictions P_1 for the full image, P_3 for the bounding box with the highest score, and their horizontally mirrored versions (P_2 and P_4).

the sequence. If the classifiers identify only empty images, the vector will contain only zeros. Figure 6.4 presents a diagram summarizing the steps of our proposed heuristic baseline.

This classification strategy limits us to predicting only one species per sequence. However, it has proven to be a strong baseline, as it was the winning solution in iWildCam 2021 challenge (see Appendix A for a detailed description of our final submission). Next, we describe how we enhanced the MaxBoxCount heuristic with state-of-the-art models, further improving its performance after the competition.

6.3.2 Improving the baseline heuristic

In Chapter 4 and Chapter 5, we explored diverse training innovations that were responsible for substantial improvements in the performance of classical computer vision architectures. Since the conclusion of the iWildCam 2021 challenge, improved wildlife monitoring models were introduced, such as MegaDetectorV1000¹ for animal detection and MegaDescriptor [25] for animal re-identification. Meanwhile, Vision Transformers have consolidated as a new dominant paradigm in computer vision. In this section, we explore how to improve the proposed MaxBoxCount heuristic by applying these innovations, establishing a stronger baseline for the animal counting task in camera trap image sequences.

¹<https://github.com/agentmorris/MegaDetector/blob/main/docs/release-notes/mdv1000-release.md>

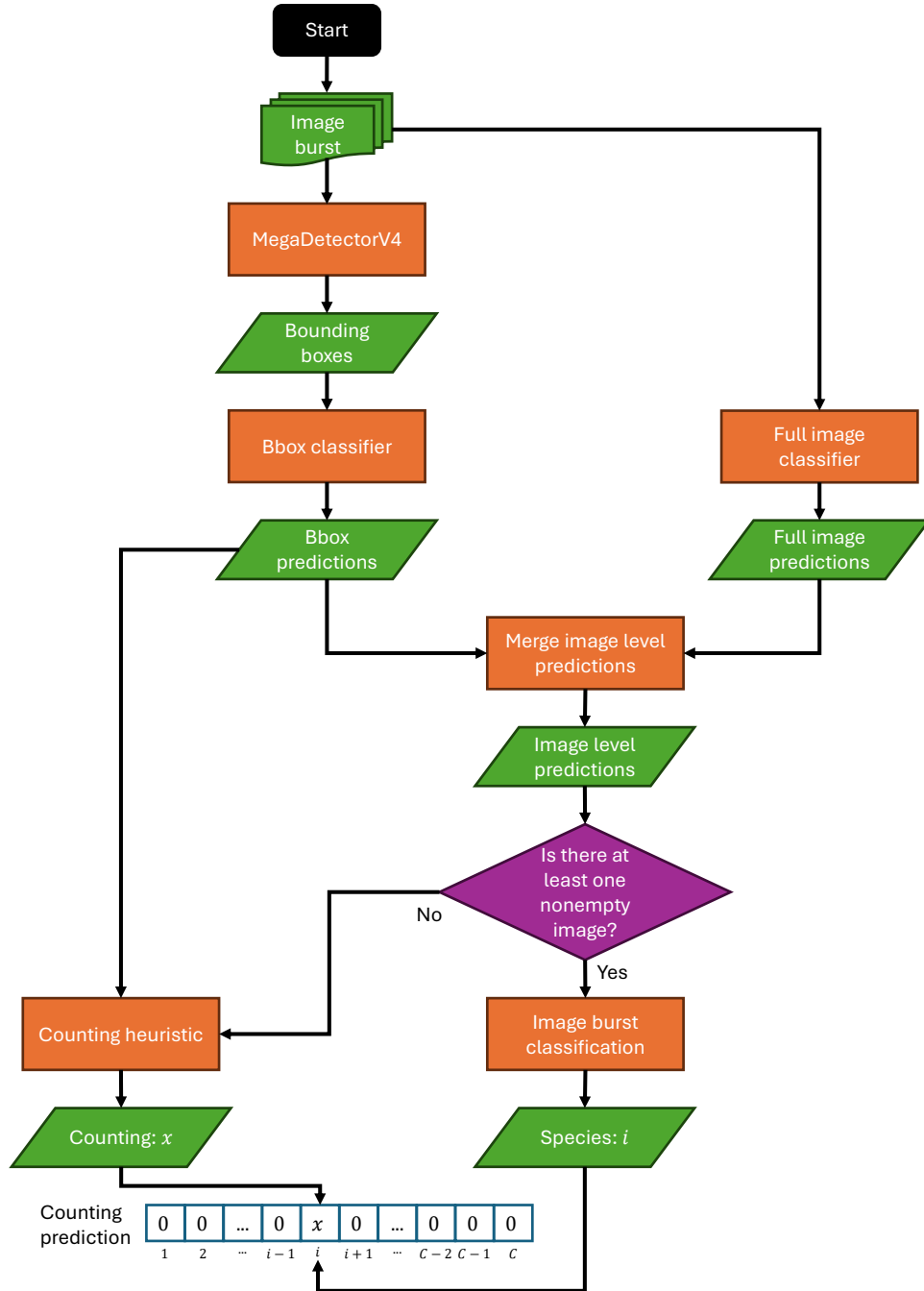


Figure 6.4. Schematic diagram of our heuristic baseline MaxBoxCount for counting animals. This heuristic relies on bounding boxes generated by MegaDetectorV4, which are used to count the detected individuals. Species identification is performed using two classifiers: one for the full image and another for the bounding boxes.

In general, we followed the same pipeline shown in Figure 6.4, updating only the animal detector and the classifiers. Initially, we replaced MegaDetectorV4 with the most recent release, MegaDetectorV1000-redwood (the largest model in the MegaDetectorV1000 family), using it both to generate bounding boxes for training the classifier and for the counting heuristic. The authors recommend a default threshold of 0.3 or 0.4 for this new family. We set the threshold to 0.3 when generating bounding boxes for classifier training, as the classifier is more robust to noisy detections, and to 0.4 for the counting heuristic as a more conservative choice. This follows our original approach, in which we used a threshold of 0.6 for training the classifier and 0.8 for counting bounding boxes with MegaDetectorV4. It is important to note that confidence thresholds do not directly translate across MegaDetector versions, as stated by the authors.

For the classifiers, we kept the same ensemble approach with two models: one trained on full images and another trained on cropped bounding boxes. We replaced EfficientNet-B2 with Swin-B from the Swin Transformer family. Swin-B has greater capacity than Swin-S, which had already outperformed EfficientNet-B2 in the experiments presented in Chapter 5.

Image preprocessing. For the crop classifier, we first applied a square crop centered on each bounding box. Then, for both the full-image and crop models, the images were resized so that the smallest side was 450 pixels. During training, we used a set of standard data augmentation operations: a random crop with aspect ratio sampled in $[3/4, 4/3]$ and area in $[65\%, 100\%]$, random horizontal flipping with 50% probability, and RandAugment [31] with parameters $N = 2$ and $M = 9$. Finally, images were normalized using the ImageNet mean and standard deviation and resized to 224×224 . During inference, the preprocessing consisted only of normalization and resizing the image to match the input resolution of the architecture.

Implementation details. As the Swin Transformer achieved excellent performance on camera trap data in Chapter 5, even without employing the evaluated methods to address the long-tail issue, we did not incorporate any additional long-tail handling techniques in order to maintain training simplicity. This also allowed us to adopt a single-stage training procedure – unlike the original heuristic, which used multiple stages – using a fixed input resolution of 224×224 for both training and inference. The models were initialized with ImageNet-1k pretrained weights from the timm library [170] and trained using the AdamW [100] optimizer with a batch size of 64 for 30 epochs. The initial learning rate was set to 10^{-5} , and the weight decay was set to 10^{-7} . The learning rate was linearly warmed up from 0 to the initial value for 2 epochs and then decayed to 0 using cosine schedule [58]. We also employed label smoothing [146] with $\epsilon = 0.1$.

Although these innovations have improved the results on the iWildCam 2021 benchmark, this heuristic, by construction, is unable to predict multiple species within a sequence. To address this limitation, the next section describes an alternative counting approach based on tracking methods.

6.4 Pure Transformer for Counting-by-Tracking in Camera Trap Image Sequences

We argue that a more natural mechanism for counting animals is to track them across images in a sequence. In this approach, detections in each image are associated with the corresponding detections in other images of the same sequence, framing the problem as a multi-object tracking task. The number of animals can then be obtained by counting the number of unique tracks, which we refer to as counting-by-tracking. Since the estimated count is directly derived from the resulting tracks, the individual tracks can be inspected to verify whether the estimated count is consistent with the animals observed across the sequence.

Another advantage of counting-by-tracking is that, unlike the MaxBoxCount heuristic, it naturally enables multi-species counting within a sequence, since each track corresponds to an individual animal that can be classified by species. Moreover, species identification may benefit from multiple views across the sequence, as the visual markers required for identification may only be visible in certain frames depending on the animal position.

However, camera-trap image sequences present several challenges for tracking, as mentioned previously, such as low frame rates and low bounding-box overlap across frames, which can result in large displacements between consecutive images. These factors may harm the performance of classical multi-object trackers such as SORT and ByteTrack, which rely mainly on motion cues and geometric overlap. We hypothesize, however, that a tracker with a learned association module can learn to associate bounding boxes across images. In particular, transformer-based trackers, such as the recently proposed Pure Transformer (PuTR), can learn such associations by leveraging visual features as well as temporal and spatial information.

In this section, we explore a counting-by-tracking approach built upon a Pure Transformer (PuTR) trained to track animals in camera trap image sequences. Each track is then classified individually using all associated bounding boxes, and the number of individuals per species is estimated by counting unique tracks. Figure 6.5 summarizes our approach, which we call Pure Transformer for Counting-by-Tracking (PuTR-

CouT).

Since suitable labeled datasets for multi-object tracking in camera trap image sequences are scarce, we propose a method for generating a synthetic training set with pseudo-labels, enabling weakly supervised training of the tracking model, as described in Sections 6.4.1 and 6.4.2. Section 6.4.3 presents the adaptations required to train PuTR given the characteristics of camera trap data. Finally, Section 6.4.4 describes the necessary modifications for predicting tracks in camera trap image sequences.

6.4.1 Generating a synthetic training set for tracking using camera trap sequence priors

While classical trackers such as SORT and ByteTrack rely on motion models and geometric overlap and therefore do not require a labeled tracking dataset, trackers that incorporate learned affinity modules, such as PuTR, require tracklet annotations for training. Unfortunately, obtaining such labels is infeasible in most camera trap projects. We overcome this limitation by exploiting structural priors inherent to camera trap data, such as static backgrounds and short temporal bursts, to heuristically generate pseudo-labels for tracking in a weakly supervised manner.

We begin by defining a base case for generating tracking labels as sequences in which an animal detector produces at most one bounding box per image. Under this condition, we assume that only a single individual appears throughout the entire sequence. Although this assumption does not always hold, it provides a reasonable starting point, and we accept the potential noise introduced by this simplification.

Next, to synthesize sequences containing more than one individual – for example, to simulate animal herds – we leverage the static nature of camera traps by merging base-case tracks from sequences obtained at the same location, as shown in Figure 6.6. Although the background may gradually change due to vegetation growth over time, illumination variations throughout the day, or different weather conditions, for simplicity we restrict the merging process to sequences captured at the same location and within the same coarse timeslot²: morning, afternoon, or night.

Importantly, under the PuTR formulation, each input token corresponds to a bounding box rather than to the full image. This allows the direct use of cropped boxes from the original sequences without explicitly generating synthetic composite images. However, this simplification also constitutes a limitation of the proposed synthetic dataset, as the appearance of each individual is preserved from its original image.

²We define the timeslots as follows: morning from 05:00 to 11:00, afternoon from 11:00 to 17:00, and night for the remaining hours.

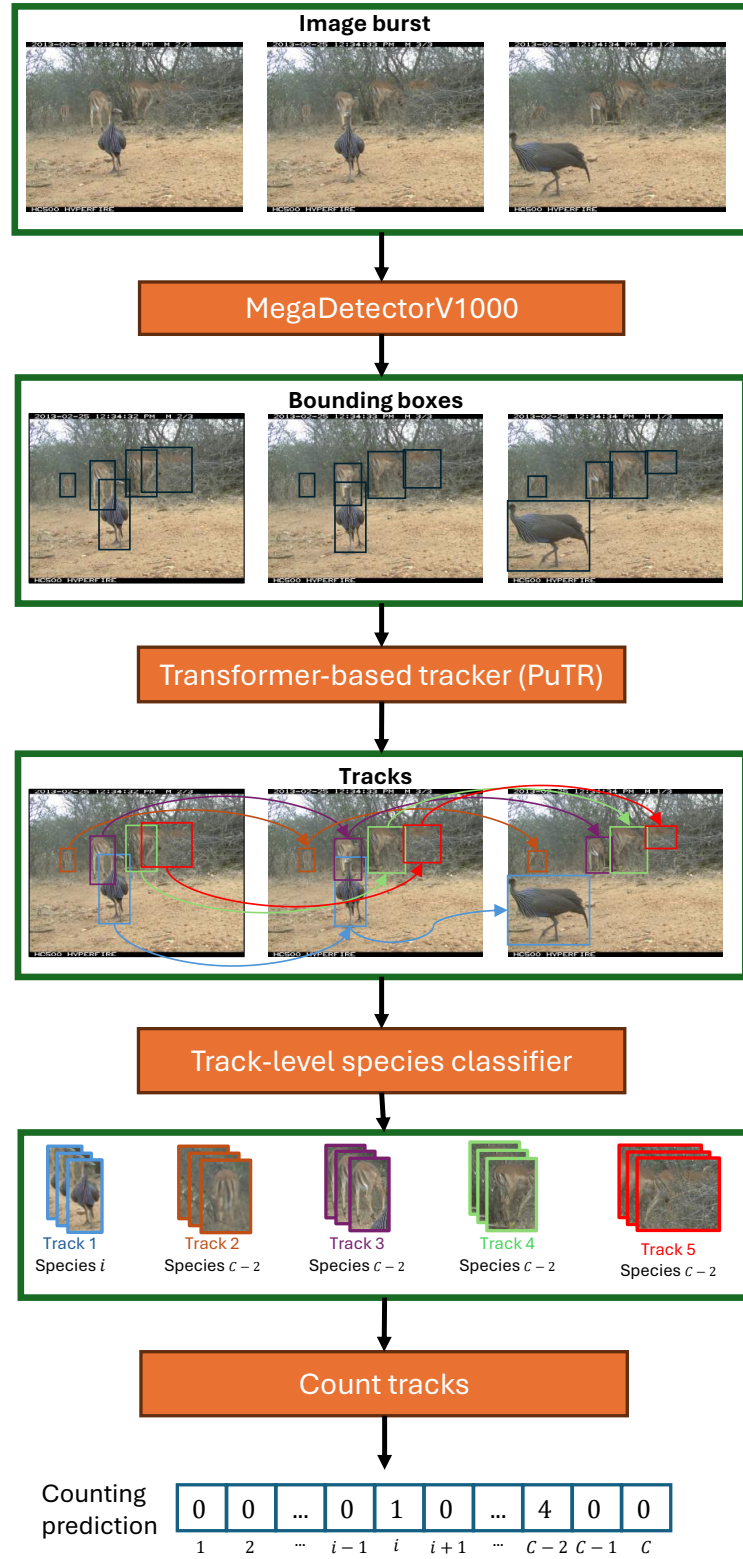


Figure 6.5. Overview of the proposed Pure Transformer for Counting-by-Tracking (PuTR-CouT) framework. Given a sequence of camera-trap images, the MegaDetectorV1000 model is first applied to obtain bounding boxes. These detections are associated across frames by a Transformer-based multi-object tracker trained on synthetic data, forming tracks that correspond to individual animals. Each track is classified independently, enabling multi-species counting within a sequence by estimating the number of unique tracks per species.

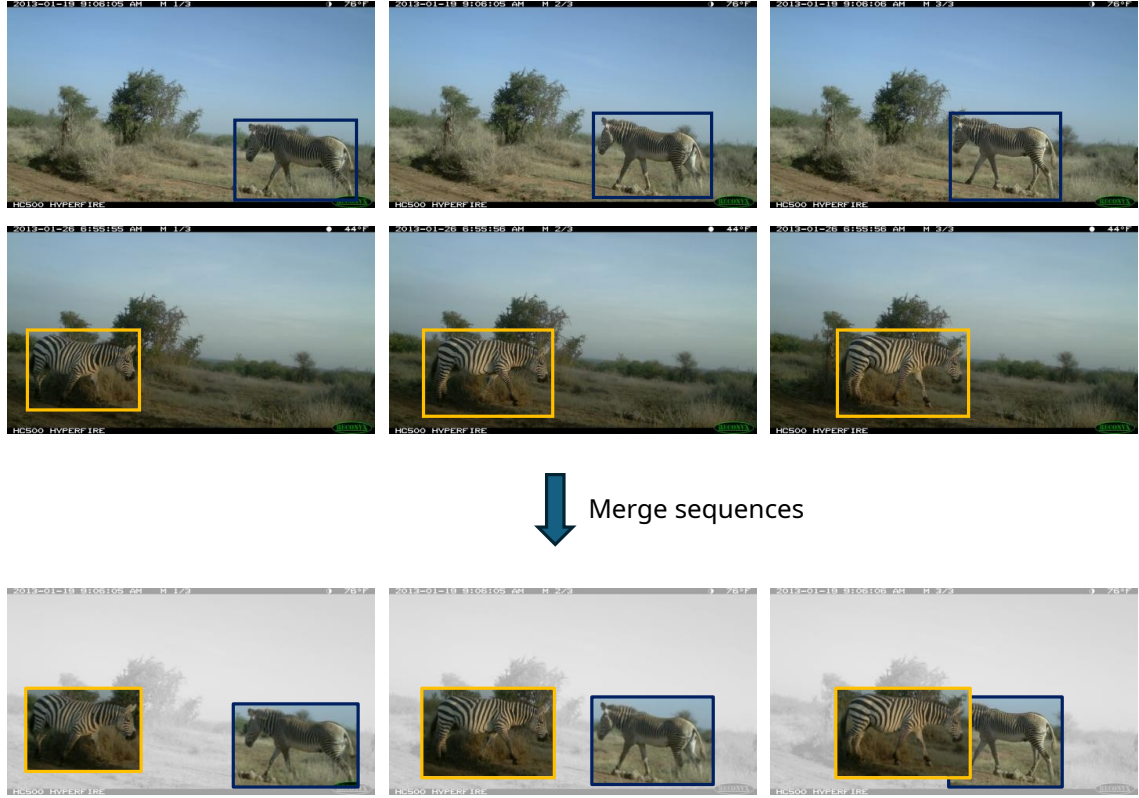


Figure 6.6. Illustration of the proposed synthetic sequence generation. Base-case tracks captured at the same location and within the same coarse timeslot can be merged to simulate a multi-individual sequence. Each cropped box is treated as a token by PuTR, removing the need to reconstruct composite images.

Consequently, visual interactions that would naturally arise in a true composite scene, such as shared illumination conditions, color consistency, partial occlusions, and background blending effects, are not modeled. We nevertheless adopt this approximation because PuTR operates directly on object tokens, making the generated samples suitable for learning association patterns while substantially simplifying the synthetic data generation process.

Finally, the number of individuals or different species present in a sequence may vary across species and locations. Since we use only single-individual sequences to generate the synthetic dataset, we approximate the distribution of individual counts per sequence by sampling, for each synthetic training example, the number of individuals according to the Negative Binomial distribution fitted to the counts produced by our MaxBoxCount heuristic, as described in Section 6.4.2. We sample the number of species present using statistics derived from the Snapshot Serengeti dataset [142] as a reference prior. Importantly, Snapshot Serengeti is not used for training or evaluation;

all experiments in this chapter are conducted exclusively on the iWildCam 2021 dataset. In Snapshot Serengeti, approximately 95% of the sequences contain only one species. Accordingly, we set the probability of two species to 4.9% and three species to 0.1%. Algorithm 1 provides a pseudo-code for generating the synthetic dataset for tracking.

Algorithm 1 Pseudo-code for synthetic sequence generation

Require: Training set $\mathcal{T}$, animal detector Det , target samples N

Ensure: Synthetic dataset $\mathcal{D}_{syn}$

```

1:  $\mathcal{O} \leftarrow Det(\mathcal{T})$  ▷ Extract detections per image
2:  $\mathcal{C} \leftarrow \text{MaxBoxCount}(\mathcal{O})$  ▷ Sequence counts via heuristic
3: Fit Negative Binomial parameters  $(\mu, \alpha)$  to  $\mathcal{C}$ 
4:  $\mathcal{S}_1 \leftarrow \{s \in \mathcal{T} \mid \mathcal{C}(s) = 1\}$  ▷ Single-individual sequences
5:  $\mathcal{L}_1 \leftarrow \text{listLocations}(\mathcal{S}_1)$ 
6:  $\mathcal{D}_{syn} \leftarrow \emptyset$ 
7: for  $j = 1$  to  $N$  do
8:   Sample count  $i \sim \text{NB}(\mu, \alpha)$  ▷ Number of individuals
9:   Sample species limit  $k \sim \text{Cat}(\{1 : 0.95, 2 : 0.049, 3 : 0.001\})$ 
10:  Sample location  $p \sim \mathcal{L}_1$  and timeslot  $t \sim \{\text{morning, afternoon, night}\}$ 
11:   $\mathcal{Q} \leftarrow \text{Select } i \text{ sequences from } \mathcal{S}_1 \text{ matching } (p, t)$ 
12:    such that total distinct species  $\leq k$ 
13:   $d_j \leftarrow \text{mergeSeqs}(\mathcal{Q})$ 
14:   $\mathcal{D}_{syn} \leftarrow \mathcal{D}_{syn} \cup \{d_j\}$ 
15: end for
16: return  $\mathcal{D}_{syn}$ 

```

In the iWildCam 2021 dataset, sequences contain between 1 and 10 images. To preserve temporal consistency, images are aligned from the first capture of each sequence, and synthetic sequences are generated with the same length as the longest base sequences. Initially, we used the frame sequence information provided in the JSON metadata. However, we noticed mismatches caused by incorrect frame ordering, which could harm temporal consistency, as exemplified in Figure 6.7. To address this issue, we reordered the frames in both the training and test sets using the EXIF [23] datetime information when available in the image metadata. Around 50% of the images present timestamp mismatches between the JSON metadata and the EXIF information, but only 3.90% of the sequences had a change in image order, as shown in Table 6.1.

Implementation details. To generate the bounding boxes, we used MegaDetectorV1000-redwood with a confidence threshold of 0.4, following the improved baseline for consistency (see Section 6.3.2). We also used the training labels to filter the data, discarding sequences labeled as empty. We set the generator to produce 75,000 synthetic sequences, resulting in 35,804 non-empty sequences – empty sequences



Figure 6.7. Example of a sequence whose frame ordering based on JSON metadata is incorrect, as verified by both the timestamps in the EXIF metadata and the timestamp shown in the top metadata overlay of the images.

Table 6.1. Timestamp consistency analysis between EXIF metadata and JSON metadata in the iWildCam 2021 dataset. Percentages are relative to the total number of images/sequences in each set.

	Training Set	Test Set
Total images	203,314	60,214
Total sequences	36,547	11,057
Images with valid EXIF	202,486 (99.6%)	60,168 (99.9%)
Images with mismatched timestamps	101,932 (50.1%)	28,883 (48.0%)
Sequences with timestamp mismatches	23,247 (63.6%)	6,474 (58.6%)
Images reordered	7,485 (3.7%)	1,579 (2.6%)
Sequences reordered	1,424 (3.9%)	325 (2.9%)

are ignored during PuTR training and therefore do not contribute to the loss. This is close to the number of sequences in the original iWildCam 2021 training set.

6.4.2 Modeling the distribution of sequence-level animal counts for synthetic data generation

Since the number of individuals per sequence is unknown in our problem setting, we cannot directly replicate real-world count distributions when generating our synthetic dataset by merging single-individual sequences. To address this limitation, we employ the MaxBoxCount heuristic as a proxy for the number of individuals per sequence in a weakly supervised formulation. By generating synthetic samples that follow this distribution, we aim to mitigate the impact of distribution shift in the number of tracks per sequence when evaluating the model on the real test set.

Running MaxBoxCount on the original training set, we found that the mean number of counts per sequence is 1.02 and the variance is 4.90, which characterizes

overdispersed count data. A common distribution used to model overdispersed counts is the Negative Binomial distribution. Another characteristic of camera trap data is the high number of empty sequences (zero counts), with the iWildCam 2021 training set presenting 46.67% zero counts according to the MaxBoxCount heuristic. In such cases, a zero-inflated model may be required to account for a possible excess of zeros.

Given the counts produced by MaxBoxCount, we used the `statsmodels` library [133] to fit both a Negative Binomial distribution and its zero-inflated variant. The Negative Binomial distribution was able to fully explain the observed distribution, with fitted parameters $\alpha = 1.319389376732665$ and $\mu = 1.0207130544230718$. For the Zero-Inflated Negative Binomial model, the parameter π , which represents the probability of additional structural zeros, was estimated as 0.000014. This indicates that there is no significant excess of structural zeros beyond what is already predicted by the Negative Binomial distribution. Given this result, we use the Negative Binomial distribution with the fitted parameters to sample the number of individuals present in each generated synthetic example.

6.4.3 Adapting PuTR for tracking under camera trap constraints

We build our counting-by-tracking framework on top of the original PuTR implementation, introducing adaptations to handle the characteristics of camera-trap image sequences. We retain the original model architecture, composed of 6 transformer blocks with multi-head self-attention (8 heads), including temporal and spatial positional encodings and a causal attention mask. The model is trained to associate each bounding box with one from the previous frame.

Detections. To maintain consistency with the MaxBoxCount baseline, we use MegaDetectorV1000-redwood to generate bounding boxes for both training and inference. This version is based on YOLOv5x6, while the original PuTR uses YOLOX during inference. We assume that MegaDetectorV1000 produces high-quality detections for this task, and all detections with confidence above 0.4 are treated as ground truth during training, as described in Section 6.4.1.

Tokenization. In PuTR, each bounding box is translated into a token. Although the authors suggest using Re-ID models or DETR-based features to obtain high-dimensional representations, the embeddings for the tokens are extracted directly from raw image patches. Given the high appearance variance across images in camera-trap data, in our framework we choose to adopt MegaDescriptor to generate the embeddings, a Re-ID model designed for wildlife data. Additionally, we evaluate embeddings

extracted from the species classifier trained on iWildCam 2021, the same model used to identify the species in the framework, allowing us to assess the impact of task-specific representations.

Sequence length. During training, the original PuTR implementation progressively increases the number of frames used per clip by adjusting the sampling rate, with the number of frames going from 4 to 128 according to the epoch. However, while the original implementation considers videos with a regular frame rate (for example, 30 FPS), this is not the case for camera-trap image sequences, which usually have 1 FPS and a small number of frames per sequence. In our dataset, bursts vary from 1 to 10 frames. Therefore, we fix the sequence length to 10 frames for both training and inference, padding sequences with fewer frames as needed.

Implementation details. To generate the tokens, the bounding boxes are cropped and resized to the MegaDescriptor input resolution (224×224), resulting in 1024-dimensional embeddings, which are precomputed during synthetic dataset generation for computational efficiency. These embeddings are then passed through a linear projection layer to obtain the token representation, matching the PuTR model dimension ($d_{\text{model}} = 512$).

To prevent the model from exploiting ordering biases introduced during synthetic data generation, we randomly permute the bounding boxes within each frame during training. Finally, no image-level data augmentation is applied.

We train PuTR from scratch on the synthetic dataset following the original training recipe. The model is trained for 7 epochs with a batch size of 4. We use the AdamW optimizer with an initial learning rate of 0.0002 and a weight decay of 0.0005. Gradients are accumulated over 32 steps, and gradient clipping is applied with a maximum norm of 1.0. We also employ a cosine learning rate decay schedule.

6.4.4 Tuning PuTR inference for the counting task

At inference time, we use tracking results as a proxy for counting animals, where each track corresponds to an individual. Accordingly, we prioritize inference design choices in PuTR that favor accurate track counts over other multi-object tracking metrics, such as identity switches. PuTR relies on a runtime manager to handle tracklet states and uses the Hungarian algorithm to associate detections across frames based on the Transformer affinity matrix. We retain the original runtime manager with minor adaptations for camera-trap image sequences.

First, we set the detection threshold to 0.4, so only these detections are visible to the tracker. Each detection is associated with a detection in the previous frame;

if unmatched, it is initialized as a new track only if its confidence exceeds 0.5. These settings maintain consistency with the thresholds used in the MaxBoxCount baseline and other multi-object trackers in our experiments.

In the PuTR runtime, a track must accumulate at least three hits to be confirmed, remaining in a “new track” state until then to ensure stability. While this is suitable for videos with regular frame rates, it is impractical for camera-trap sequences at 1 FPS, where animals may appear in only a single frame (as shown in Figure 6.8). Given this constraint, and considering the high quality of MegaDetector detections, we remove this stabilization requirement. Additionally, we remove the heuristic that keeps all detections in the first five frames in the “new track” state.



(a) Animal appearing only in the first frame



(b) Animal appearing in the last frame

Figure 6.8. Occurrences of animals appearing in only a single frame are relatively common in camera-trap image sequences. (a) illustrates a frequent scenario in which the animal appears only in the first frame due to its speed. (b) presents a case in which a new individual enters the scene in the final frame. In both cases, these individuals would be ignored by the stabilization step originally employed in the PuTR inference runtime.

To improve tracking, PuTR applies a series of geometric and positional compensations to the affinity matrix produced by the Transformer. However, the large displacements of animal detections between frames in camera-trap sequences can make these compensations ineffective or even harmful. Therefore, we remove them, relying solely on the Transformer predictions for association, and evaluate this design choice in an ablation study.

We also relax the association thresholds `ASSO_THRE1` and `ASSO_THRE2`, which control the association of detections to tracks based on bounding box IoU and visual

affinity, respectively. We set `ASSO_THRE1` to 0.0, effectively disabling the IoU constraint, and reduce `ASSO_THRE2` from 0.8 to 0.25 to account for large appearance variations across frames. This helps reduce track fragmentation, i.e., cases where a track is incorrectly split into multiple tracks, artificially increasing the counts.

We also relax `DETS_IOU_THRESH`, which suppresses highly overlapping detections. Since we assume MegaDetector produces high-quality bounding boxes, such aggressive filtering is unnecessary; therefore, we increase this threshold from 0.4 to 0.95.

6.5 Experiments and results

Our experiments were conducted on the iWildCam 2021 dataset using the official challenge metric, the mean column-wise root mean squared error (MCRMSE). Since the test set labels are not publicly available, we evaluated our method on the Kaggle competition platform via late submissions. We report both public and private scores, each corresponding to approximately 50% of the test set: the public score was visible during the competition, whereas the private score, revealed only after the competition ended, determined the final rankings. Additional details about our solution submitted to the competition and the final standings can be found in [Appendix A](#).

6.5.1 Improved MaxBoxCount heuristic results

To evaluate the extent to which the proposed improvements in [Section 6.3.2](#) impact the performance of the MaxBoxCount heuristic, we first replace only the classifier component. As shown in [Table 6.2](#), while keeping the bounding box counts from MegaDetectorV4 as in the winning solution, the error is reduced by 12.4%, from 0.0293278 to 0.0256999. Notably, the error gap between the first- and second-place solutions is less than 0.2%, while the gap to the third place is approximately 4.9%. When using MegaDetectorV1000 to generate bounding boxes for counting, the error decreases further to 0.0238785, corresponding to an improvement of approximately 18.5% relative to the winning solution. These results demonstrate the effect of the updated models on the heuristic. It is worth noting that all top-3 solutions employed heuristics similar to our improved MaxBoxCount.

6.5.2 PuTR-CouT results

Trackers experimental settings. The improved MaxBoxCount heuristic, despite its strong performance, is inherently limited to predicting a single species per sequence

(sequence-level classification). In contrast, PuTR-CouT overcomes this limitation by adopting a count-by-tracking strategy, where each track is classified independently (track-level classification). Therefore, we also include classical multi-object trackers in our experiments to compare PuTR-CouT with approaches that support track-level classification, enabling a fairer evaluation. In particular, we evaluate ByteTrack, a tracker derived from SORT that relies only on geometric features, and BoT-SORT, which builds upon ByteTrack and incorporates ReID features. The latter allows for a more direct comparison with PuTR-CouT, as both approaches can utilize the same embedding model to encode detections. Additionally, for a fair comparison, all trackers use the same MegaDetectorV1000 bounding boxes as MaxBoxCount, as well as the same bounding box classifier based on Swin-B to classify tracks.

For ByteTrack, we set the high track threshold (used as the primary criterion for selecting bounding boxes for tracking) to 0.4 to maintain consistency with MaxBoxCount, while keeping the default low track threshold of 0.1. For the new track threshold, we follow the original ByteTrack implementation, setting it to be 0.1 higher than the high track threshold (i.e., 0.5 in this case). For BoT-SORT, we use the same threshold values. For the additional thresholds in BoT-SORT, we relax both the proximity and appearance thresholds (to 0.95 and 0.5, respectively) to account for the characteristics of camera trap image sequences, which often exhibit large displacements and higher appearance variability across images. Furthermore, since camera traps are static, we disable the Camera Motion Compensation module. The PuTR-CouT thresholds are described in Section 6.4.4.

To compare the impact of sequence-level and track-level classification on counting performance, we evaluate both approaches using ByteTrack while keeping the tracking component fixed. Specifically, in the sequence-level setting, the number of tracks produced by ByteTrack replaces the maximum bounding-box count used in the MaxBoxCount heuristic. For the remaining tracker experiments, we adopt only track-level classification, following the counting-by-tracking strategy.

For BoT-SORT, we conduct experiments both with and without ReID module to evaluate their impact. For the ReID features, we use MegaDescriptor as well as features produced by the classifier, in order to assess the impact of task-specific representations, using the same models employed during the tokenization process for PuTR.

Trackers results. The results shown in Table 6.2 highlight several important trends. When comparing ByteTrack configurations, track-level classification performs slightly worse than sequence-level classification on the private score; however, it shows a larger advantage on the public score. Although ByteTrack yields the weakest performance among all evaluated trackers, it still surpasses the winning solution.

Table 6.2. Main results of the improved MaxBoxCount and PuTR-Count for counting (MCRMSE), compared with the top-3 iWildCam submissions and classical multi-object trackers on iWildCam 2021. In the classifier column, all entries are ensembles of the specified architecture. Bold and underlined values indicate the best and second-best scores, respectively. Lower is better.

Strategy	BBoxes Source	Classifier	ReID Features	Counting	Classification	Public ↓	Private ↓
2021 Winning solution (ours)	MDv4	EfficientNet-B2	–	Boxes count	Sequence classif.	0.0300970	0.0293278
2nd place solution	MDv3	EfficientNet-B2	–	Boxes count	Sequence classif.	0.0297267	0.0293786
		ResNet152					
3rd place solution	MDv4	+ EfficientNet-B3	–	Boxes count	Sequence classif.	0.0315964	0.0308570
		+ EfficientNet-B7					
MaxBoxCount (ours)	MDv4	Swin-B	–	Boxes count	Sequence classif.	0.0269689	0.0256999
MaxBoxCount (ours)	MDv1000	Swin-B	–	Boxes count	Sequence classif.	0.0254706	0.0238785
ByteTrack	MDv1000	Swin-B	–	Tracks count	Sequence classif.	0.0297142	0.0282785
ByteTrack	MDv1000	Swin-B	–	Tracks count	Tracks classif.	0.0285004	0.0284988
BoT-SORT	MDv1000	Swin-B	–	Tracks count	Tracks classif.	0.0279936	0.0283493
BoT-SORT	MDv1000	Swin-B	Swin-B classif.	Tracks count	Tracks classif.	0.0272002	0.0273195
BoT-SORT	MDv1000	Swin-B	MegaDescriptor	Tracks count	Tracks classif.	0.0270137	0.0272918
PuTR-CouT (ours)	MDv1000	Swin-B	Swin-B classif.	Tracks count	Tracks classif.	<u>0.0254377</u>	<u>0.0248642</u>
PuTR-CouT (ours)	MDv1000	Swin-B	MegaDescriptor	Tracks count	Tracks classif.	0.0250704	0.0250820

For BoT-SORT, the use of ReID module leads to improved performance compared to configuration without ReID. Finally, PuTR-CouT outperforms all other trackers, achieving an error of 0.0248642, corresponding to an improvement of nearly 9% over BoT-SORT when using the same ReID model. In particular, PuTR-CouT with Swin-B features reduces the error by approximately 15.2% compared to the winning solution on the private score. Moreover, PuTR-CouT with MegaDescriptor achieves the best result on the public score, surpassing even the improved MaxBoxCount heuristic (0.0250704 vs. 0.0254706).

Figures 6.9 and 6.10 illustrate representative examples of model performance. Figure 6.9 presents sample predictions comparing PuTR-CouT and MaxBoxCount, highlighting scenarios in which PuTR-CouT effectively handles multi-species counting. In contrast, Figure 6.10 shows selected failure cases of PuTR-CouT, emphasizing common challenges such as missed detections, identity switches, and the lack of contextual reasoning across individuals.

6.5.3 Ablation studies on PuTR-CouT

Improving tracks. When constructing the synthetic training set for tracking, we observed that some image sequences were incorrectly ordered. In this ablation study, we evaluate the impact of reordering sequences based on timestamp information extracted from EXIF metadata. For PuTR, we assess all combinations of applying or not applying reordering during both training and inference. In contrast, since ByteTrack and BoT-SORT do not involve a training phase, we evaluate reordering only at inference time. As shown in Table 6.3, reordering slightly degrades performance for both ByteTrack and BoT-SORT, increasing the private error by less than 1%. However, for PuTR-CouT, reordering reduces the error by approximately 3% compared to the original dataset ordering. This improvement is consistent regardless of whether reordering is applied during training, highlighting the effectiveness of sequence reordering for PuTR-CouT. We also investigate filtering out detections classified as empty before passing them to the tracker. This strategy, however, leads to an increase in error.



Figure 6.9. Illustrative examples of counting predictions. Bounding box colors denote the tracks generated by PuTR-CouT; only the first three images of each sequence are displayed. The first row demonstrates a key advantage of PuTR-CouT over MaxBoxCount in handling multi-species counting scenarios. Best viewed in color.



Figure 6.10. Selected failure cases of PuTR-CouT. In the first row, four individuals are present; however, the detector captures only three. As the third individual is difficult to identify, PuTR-CouT classifies each track independently, resulting in an implausible prediction that mixes *Panthera onca* and *Panthera leo*, whereas contextual information could suggest that all individuals belong to the same species. In the second row, an identity switch occurs, and the presence of juveniles further degrades performance, leading to incorrect predictions. Incorporating context could help classify all individuals as belonging to the same adult species. In the third row, individual predictions are again incorrect, where contextual reasoning could support more accurate classification. Best viewed in color.

Table 6.3. Effect of reordering camera-trap sequences based on EXIF information for both the training and test sets, and pre-filtering empty bounding boxes when using a counting-by-tracking strategy. Lower is better.

Tracker	Reordering		Pre-filter bboxes	Public ↓	Private ↓
	Train	Test			
Bytetrack				0.0282960	0.0282683
Bytetrack		✓		0.0285004	0.0284988
BoT-SORT (MegaDesc.)				0.0270303	0.0270292
BoT-SORT (MegaDesc.)		✓		0.0270137	0.0272918
PuTR (MegaDesc.)				0.0252463	0.0256394
PuTR (MegaDesc.)		✓		0.0250080	0.0248314
PuTR (MegaDesc.)	✓			0.0252018	0.0258654
PuTR (MegaDesc.)	✓	✓		0.0250704	0.0250820
PuTR (MegaDesc.)	✓	✓	✓	0.0250210	0.0253143

Improving the affinity matrix. In this ablation study, we evaluate the impact of disabling the additional components of the affinity matrix introduced in the original PuTR implementation. As shown in Table 6.4, none of these components improve performance in our setting. This finding supports our hypothesis that incorporating geometric features is challenging for this task, and suggests that predictions based solely on transformer representations are already competitive.

Table 6.4. Ablation study on the effects of different components of the affinity matrix in PuTR when applied to a camera-trap data context. Lower is better.

Tracker	IOU comp.	HMIOU	HIOU	Traj. conf.	Detec. conf.	Public ↓	Private ↓
PuTR						0.0250704	0.0250820
PuTR	✓					0.0251859	0.0252154
PuTR		✓				0.0252193	0.0253234
PuTR			✓			0.0252193	0.0253234
PuTR				✓		0.0251123	0.0254082
PuTR					✓	0.0250931	0.0251428
PuTR	✓	✓	✓	✓	✓	0.0253234	0.0253416

6.6 Discussion

Counting animals in camera trap image sequences is an essential task for ecological studies, particularly those focused on modeling the abundance and density of a

species [11]. However, automating this task using computer vision is challenging due to the inherent characteristics of camera trap data, such as low frame rates and the absence of counting annotations in most publicly available datasets. In this chapter, we presented a counting-by-tracking strategy built upon a transformer-based tracking model, which achieved competitive results compared to the heuristic-based MaxBoxCount method. Furthermore, we improved MaxBoxCount to achieve state-of-the-art performance in this setting.

While counting animals in single images could be performed using object detectors trained for camera trap data, as demonstrated by Norouzzadeh et al. [106], extending this approach to image sequences is not straightforward due to the challenges mentioned above. The iWildCam 2021 challenge [11] highlighted this difficulty in a setting where counting labels are not available for training, with top-performing solutions relying on heuristics based on the maximum number of bounding boxes across images. Building on this idea, we improved the MaxBoxCount heuristic by incorporating more recent versions of MegaDetector and an improved classifier, reducing the error rate by 18.5% relative to the winning solution. This result highlighted the dependence of this heuristic on strong detection and classification models. However, by design, this approach is limited to predicting a single species per sequence and can only provide a lower bound on the total number of individuals, which limits further error reduction. Additionally, animals can disappear and reappear across frames, requiring some form of individual re-identification to avoid under- or overestimation of counts.

In contrast, PuTR-CouT addresses these limitations by explicitly tracking individuals across image sequences, assigning a species label to each track, and counting them based on the number of unique tracks. While using the same animal detector and species classifier as MaxBoxCount, PuTR-CouT achieved competitive results, although there remained significant room for improvement. For instance, tracking performance could potentially be enhanced through improved synthetic dataset generation.

One limitation of the method used to build the synthetic training set is that it does not account for species-specific social behavior, as some species tend to appear in groups while others are typically solitary. This has several implications. For example, species that appear in groups may yield fewer single-individual sequences, which can introduce a distribution shift in species representation. As a result, the set of single-individual sequences may be dominated by solitary species, leading to an overrepresentation of these species compared to the original distribution.

Another related limitation is that the approach does not account for differences in count distributions across species when generating the random number of individuals per sequence, being species-agnostic. We also do not account for species co-occurrence

when randomly selecting which species may appear together in a synthetic tracking sample, although this effect may be partially mitigated by restricting the mixing process to sequences from the same camera site. Although we expect the tracking model to be robust to these types of noise, our method does not explicitly address these cases, and our experiments were not designed to assess the impact of these simplifications.

A further limitation of the synthetic data generation procedure is that synthetic multi-individual sequences are created at the token level rather than as fully composited images. Because each bounding box is extracted directly from an original image, the generated samples do not reproduce visual interactions that would naturally occur when multiple individuals share the same scene, such as consistent illumination, color adaptation, occlusions, and other contextual effects. Although this approximation is compatible with the token-based formulation adopted by PuTR, it introduces a discrepancy between the synthetic training data and real multi-individual camera-trap sequences.

Despite these limitations, our method for generating a synthetic training set for tracking enabled PuTR-CouT to be trained effectively and thus to count individuals through tracking. Under the counting-by-tracking paradigm, PuTR-CouT substantially outperformed BoT-SORT, the best-performing traditional multi-object tracking method in our experiments, when both used the same object detector, feature embeddings, and species classifier. Our results indicate that the learned association mechanism used to link bounding boxes across images is better suited to handle the large temporal gaps and appearance variations present in camera trap image sequences. This is further supported by the ablation study, which showed that PuTR-CouT performed better when relying solely on the transformer-based learned association mechanism than when using an affinity matrix incorporating geometric cues, as originally proposed in PuTR.

Regarding high-dimensional representations, we replaced the raw pixel inputs used in PuTR with feature embeddings generated by MegaDescriptor or the species classifier. Our results showed no significant difference between using a re-identification (ReID) model and task-specific representations produced by the classifier. However, none of the evaluated models were explicitly trained to generate embeddings that capture consistent representations of animals across images in the iWildCam 2021 dataset.

Moreover, our experiments do not allow us to determine whether these representations are superior to using raw pixel inputs. This leaves open the question of whether a model specifically trained for feature representation could further improve PuTR-CouT. Such a model could be trained using the same synthetic dataset, for example by encouraging embeddings of bounding boxes belonging to the same track to

be similar.

Another observation is that the iWildCam 2021 dataset contains issues that may affect temporal coherence, such as sequences formed by mixed bursts, out-of-order frames, and missing images. Our ablation study showed that these issues negatively impact the inference of PuTR-CouT, possibly because the method primarily relies on associating bounding boxes between consecutive frames, which is sensitive to temporal ordering. This highlights the importance of high-quality data with correct frame ordering and complete sequences for our proposed approach.

At the same time, this finding exposes a limitation of the method, namely its reliance on frame-to-frame associations. It would be interesting to explore alternative approaches that allow the tracker to associate bounding boxes across the entire sequence, since all frames are available at inference time. This would also make the causal mask unnecessary.

Another limitation of our proposed PuTR-CouT lies in its classification strategy, which relies solely on the bounding boxes of individual tracks. In contrast, MaxBoxCount leverages full-image classification, allowing it to incorporate contextual information and improve species recognition. As a result, individuals that are difficult to classify based on local appearance alone may be more accurately identified by leveraging full-image context, especially in cases where animals appear in groups, an advantage that PuTR-CouT currently lacks. Incorporating a similar mechanism into PuTR-CouT, such as integrating full-image contextual features, could improve classification performance, particularly in low-confidence scenarios. This could also be beneficial in cases involving occlusions, barely visible individuals, or juveniles.

As a potential improvement to training PuTR-CouT, more advanced data augmentation strategies could be applied to the training sequences. While the current approach enables efficient training using precomputed embeddings, online augmentation could increase data variability and improve robustness. However, this would require computing embeddings and possibly generating bounding boxes during training, increasing computational cost.

Overall, our findings suggest a promising direction for counting animals in camera trap image sequences using a counting-by-tracking formulation. Our synthetic dataset generation enabled PuTR-CouT to learn association mechanisms, achieving competitive results compared to the current state-of-the-art heuristic, MaxBoxCount, while still leaving substantial room for improvement. Future work could focus on improving synthetic training data generation, including the creation of realistic composite scenes that better model interactions among multiple individuals; enhancing representation learning; and incorporating contextual information into the classification model, among

other directions.

Additionally, our tracker could be used to assist in the generation of track annotations. Instead of relying solely on expensive manual labeling, it could provide precomputed tracks that human annotators would only need to correct, thereby reducing annotation effort.

6.7 Final remarks

Counting animals in camera trap image sequences is a challenging task, particularly when count labels are unavailable, as is the case for most public datasets. In this chapter, we introduced PuTR-CouT, a counting-by-tracking approach built on a transformer-based multi-object tracker with learned association. The method was trained using a synthetic dataset generation strategy that exploits structural priors in camera trap sequences, within a weakly supervised setting.

We evaluated our approach against traditional multi-object trackers and the state-of-the-art heuristic MaxBoxCount. The results show that PuTR-CouT achieves competitive performance while offering key advantages: unlike MaxBoxCount, it inherently supports counting multiple species within the same sequence and provides track-level outputs that make the estimated counts easier to inspect and verify.

In our experiments, we also improved MaxBoxCount using recent model innovations, achieving an 18.5% relative improvement over the winning solution of the iWild-Cam 2021 challenge, on which we base our evaluation. Under the same base models, PuTR-CouT achieved competitive results, though with substantial room for improvement. In contrast, further gains in MaxBoxCount largely depend on improvements in detection and classification.

Future work on PuTR-CouT can follow several directions: enhancing the synthetic data generation process to better reflect species and count distributions; improving embedding learning to produce more consistent representations for bounding boxes belonging to the same sequence; and incorporating contextual information into the classifier, rather than relying solely on cropped regions.

In the next chapter, we highlight the main contributions of this thesis to research on the automated analysis of camera-trap data and outline future directions and opportunities for innovation in machine learning and computer vision driven by the challenges inherent to this domain.

Conclusions

The automated analysis of camera-trap data presents a variety of challenges that not only complicate the task but also create unique opportunities for innovation in computer vision. This thesis adopted the perspective that these challenges should be viewed not as obstacles to be overcome, but as drivers for the development of new methods.

Consequently, the main objective of this work was to investigate how the inherent constraints of camera-trap data can serve as catalysts for robust and efficient wildlife monitoring solutions. Following an application-driven approach, this research used these constraints to identify research questions and motivate methodological choices. Therefore, this thesis focused on three main tasks: filtering empty frames, long-tailed species classification, and animal counting across image sequences. The following sections summarize the main findings, limitations, and directions for future research.

Filtering empty frames. For the task of filtering empty frames, we investigate deployment on edge devices, where computational resources are limited and acquiring bounding-box annotations can be costly, particularly for small-scale projects. Within this setting, we conducted a systematic comparison of detection and classification models under different computational and data regimes. The results show that detection models achieve higher filtering performance when trained under the same data regime. However, when the cost and availability of annotations are taken into account, classification models trained with a larger set of image-level labels can achieve comparable filtering performance to detection models trained with fewer bounding-box annotations, while requiring lower inference latency on edge devices.

These findings provide empirical support for Hypothesis 1 and highlight the importance of considering both annotation cost and computational efficiency when selecting models for resource-constrained wildlife monitoring applications.

Long-tail classification. Although species classification in camera-trap data has been widely studied, handling long-tailed distributions remains challenging, particularly because rare species are often of greater ecological importance, while strong performance on frequent classes must still be preserved.

To address this challenge, we proposed the Square-Root Sampling Branch (SSB), a framework that aggregates predictions from independent classification heads trained under different data distributions. The results support Hypothesis 2, showing that this design improves performance on tail classes while maintaining comparable accuracy on head classes relative to single-branch approaches. In addition, the systematic evaluation of long-tail learning strategies across multiple datasets and architectures provides empirical evidence on the effectiveness of different approaches for long-tailed species classification.

Importantly, although the design of SSB is not specific to camera-trap data, its effectiveness in other imbalanced classification problems has yet to be evaluated.

Counting individuals. The counting task is the most challenging among the three, as existing solutions are largely based on maximum bounding-box counts and are typically limited to single-species prediction, while recent work has also explored tracking and individual re-identification. However, large object displacements caused by low frame rates make classical tracking methods unreliable. In addition, tracking and counting annotations are often absent in camera-trap data, requiring approaches that can learn without explicit labels.

To address this challenge, we developed a counting-by-tracking approach to test the hypothesis that a tracking method with a learned association mechanism can improve robustness under low-frame-rate conditions. Our solution, PuTR-CouT, is built on a transformer architecture and trained using a weakly labeled dataset constructed from camera-trap priors. It achieves competitive results while enabling multi-species estimation. These findings support the use of learned association mechanisms for tracking in low-frame-rate scenarios, where object motion is sparse and discontinuous.

Taken together, the results across the three tasks illustrate how the constraints of camera-trap data can inform the formulation of research questions and the design of computer vision methods. Rather than simply restricting methodological options, characteristics such as limited deployment hardware, long-tailed distributions, and low frame rates motivated specific methodological choices throughout this thesis. The resulting solutions have proven effective for the wildlife monitoring settings investigated in this work, while their applicability to other real-world scenarios remains to be established.

7.1 Limitations

Despite the contributions presented in this thesis, some limitations must be acknowledged. These limitations are primarily related to experimental design choices and inherent challenges of camera-trap data, and they highlight important directions for future research:

- Although the Raspberry Pi has been the most commonly used platform in the literature due to its low cost, evaluating edge-device performance solely on this hardware for filtering empty frames overlooks devices with specialized accelerators (e.g., GPUs, EdgeTPUs, DSPs), thereby limiting generalization.
- In our edge evaluation, computational cost was measured indirectly via inference latency and memory optimization. However, metrics such as power consumption and processor utilization may be more relevant depending on the application.
- This work lacks an evaluation of how the proposed methods can be integrated into existing pipelines for the automated analysis of camera-trap data.
- Hyperparameter choices were not fully optimized due to the need for controlled comparisons, potentially underestimating model performance.
- The proposed counting-by-tracking approach relies on synthetic data that does not fully capture species behavior, co-occurrence, and realistic count distributions. It is also highly dependent on the static nature of camera-trap data and the specific animal context.
- The counting framework depends on strong detection and classification models, and its performance is therefore limited by their accuracy.
- The counting approach was evaluated in a multi-species camera-trap setting without ground-truth tracking annotations. Although datasets such as GorillaWatch provide tracking and re-identification annotations, they focus on specific species and scenarios. Evaluating the proposed approach on independently annotated tracking datasets would help assess its generalization beyond the synthetic training setting.

7.2 Future Directions

Given the findings of this thesis and considering its limitations, several directions can be explored to further improve the proposed methods and their applicability.

In the context of edge deployment, future work may investigate stronger model optimization techniques such as knowledge distillation, model pruning, quantization-aware training, and architecture search for edge environments, as well as evaluation on specialized hardware such as Edge GPUs and DSPs.

Generative image models, particularly diffusion models, could be explored to generate images of rare species under diverse environmental and contextual conditions, helping to mitigate data scarcity and improve long-tail classification.

Vision-language models (VLMs) represent another promising direction, particularly in few-shot and zero-shot settings, where visual recognition can benefit from semantic knowledge encoded in pre-trained multimodal representations. However, this thesis prioritized specialized and computationally efficient methods, and evaluating VLMs was outside its scope. Future work could therefore investigate their ability to improve rare species recognition while preserving performance on head classes, as well as assess the trade-off between accuracy and computational cost in practical wildlife monitoring scenarios.

The proposed counting-by-tracking framework could benefit from improved representation learning, including models explicitly trained for re-identification. Future work should also evaluate the framework on datasets with reliable ground-truth tracking and counting annotations, such as GorillaWatch, while reproducing the low frame-rate sampling conditions of camera-trap data. By progressively subsampling annotated sequences, it would be possible to quantify the impact of sampling frequency on data association and compare PuTR-CouT with conventional tracking methods under controlled conditions. Using ground-truth detections initially could further isolate the data association problem from detection and classification errors. Such an evaluation would help determine whether the robustness of learned association under low frame rates extends beyond the camera-trap setting.

An important direction is the practical integration of the proposed methods into existing wildlife monitoring pipelines, such as those based on MegaDetector. Although the proposed methods were developed and evaluated in camera-trap settings, future work should investigate their applicability to other real-world computer vision applications with similar characteristics, such as precision agriculture and insect monitoring.

Bibliography

- [1] Aharon, N., Orfaig, R., and Bobrovsky, B.-Z. (2022). Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*.
- [2] Ahumada, J. A., Fegraus, E., Birch, T., Flores, N., Kays, R., O'Brien, T. G., Palmer, J., Schuttler, S., Zhao, J. Y., Jetz, W., et al. (2020). Wildlife insights: A platform to maximize the potential of camera trap and other passive sensor wildlife data for the planet. *Environmental Conservation*, 47(1):1--6.
- [3] Ahumada, J. A., Hurtado, J., and Lizcano, D. (2013). Monitoring the status and trends of tropical forest terrestrial vertebrate communities from camera trap data: a tool for conservation. *PloS one*, 8(9):e73707.
- [4] Alencar, L., Cunha, F., and dos Santos, E. M. (2023). A context-aware approach for filtering empty images in camera trap data using siamese network. In *2023 36th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 85--90. IEEE.
- [5] Alencar, L., Cunha, F., and dos Santos, E. M. (2024). Zero and few-shot learning with modern mllms to filter empty images in camera trap data. In *2024 37th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE.
- [6] AlHammadi, H., Mkadmi, M., Mitra, R., and Zualkernan, I. (2024). Exploring the forward-forward algorithm to train neural networks for camera trap images on the edge. In *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)*, pages 741--746.
- [7] Anton, V., Hartley, S., Geldenhuys, A., and Wittmer, H. U. (2018). Monitoring the mammalian fauna of urban areas using remote cameras and citizen science. *Journal of Urban Ecology*, 4(1).
- [8] Arshad, B., Barthelemy, J., Pilton, E., and Perez, P. (2020). Where is my deer?-wildlife tracking and counting via edge computing and deep learning. In *2020 IEEE SENSORS*, pages 1--4. IEEE.

- [9] Auer, D., Bodesheim, P., Fiderer, C., Heurich, M., and Denzler, J. (2021). Minimizing the annotation effort for detecting wildlife in camera trap images with active learning. In *INFORMATIK 2021*, pages 547--564. Gesellschaft für Informatik, Bonn.
- [10] Beery, S. (2021). Scaling biodiversity monitoring for the data age. *XRDS: Crossroads, The ACM Magazine for Students*, 27(4):14--18.
- [11] Beery, S., Agarwal, A., Cole, E., and Birodkar, V. (2021). The iwildcam 2021 competition dataset. *arXiv preprint arXiv:2105.03494*.
- [12] Beery, S., Liu, Y., Morris, D., Piavis, J., Kapoor, A., Joshi, N., Meister, M., and Perona, P. (2020a). Synthetic examples improve generalization for rare classes. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 863--873.
- [13] Beery, S., Morris, D., and Yang, S. (2019). Efficient pipeline for camera trap image review. *arXiv preprint arXiv:1907.06772*.
- [14] Beery, S., Van Horn, G., and Perona, P. (2018). Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision*, pages 456--473.
- [15] Beery, S., Wu, G., Rathod, V., Votel, R., and Huang, J. (2020b). Context r-cnn: Long term temporal context for per-camera object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13075--13085.
- [16] Beery, S. M. (2023). *Where the Wild Things Are: Computer Vision for Global-Scale Biodiversity Monitoring*. Dissertation (ph.d.), California Institute of Technology.
- [17] Bello, I., Fedus, W., Du, X., Cubuk, E. D., Srinivas, A., Lin, T.-Y., Shlens, J., and Zoph, B. (2021). Revisiting resnets: Improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 34.
- [18] Böhner, H., Kleiven, E. F., Ims, R. A., and Soininen, E. M. (2023). A semi-automatic workflow to process images from small mammal camera traps. *Ecological Informatics*, 76:102150.
- [19] Boscoe, B., Johnson, S., Osborn, A., Campbell, C., and Mager, K. (2025). Green-crossingai: A camera trap/computer vision pipeline for environmental science research groups. In *Practice and Experience in Advanced Research Computing 2025*:

- The Power of Collaboration*, PEARC '25, New York, NY, USA. Association for Computing Machinery.
- [20] Bothmann, L., Wimmer, L., Charrakh, O., Weber, T., Edelhoff, H., Peters, W., Nguyen, H., Benjamin, C., and Menzel, A. (2023). Automated wildlife image classification: an active learning tool for ecological applications. *Ecological Informatics*, 77:102231.
- [21] Buehler, P., Carroll, B., Bhatia, A., Gupta, V., and Lee, D. E. (2019). An automated program to find animals and crop photographs for individual recognition. *Ecological informatics*, 50:191--196.
- [22] Cai, J., Wang, Y., and Hwang, J.-N. (2021). Ace: Ally complementary experts for solving long-tailed recognition in one-shot. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 112--121.
- [23] Camera & Imaging Products Association (CIPA) (2023). Exchangeable image file format for digital still cameras: Exif version 3.0. <https://www.cipa.jp/std/documents/e/DC-008-2023-E.pdf>. CIPA DC-008-2023.
- [24] Celis, G., Ungar, P., Sokolov, A., Sokolova, N., Böhner, H., Liu, D., Gilg, O., Fufachev, I., Pokrovskaya, O., Ims, R. A., et al. (2024). A versatile, semi-automated image analysis workflow for time-lapse camera trap image classification. *Ecological Informatics*, 81:102578.
- [25] Čermák, V., Pícek, L., Adam, L., and Papafitsoros, K. (2024). Wildlifedatasets: An open-source toolkit for animal re-identification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5953--5963.
- [26] Chalmers, C., Fergus, P., Wich, S., and Montanez, A. C. (2019). Conservation ai: Live stream analysis for the detection of endangered species using convolutional neural networks and drone technology. *arXiv preprint arXiv:1910.07360*.
- [27] Chen, G., Shen, S., Wen, L., Luo, S., and Bo, L. (2020). Efficient pig counting in crowds with keypoints tracking and spatial-aware temporal response filtering. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10052--10058. IEEE.
- [28] Chen, H. and Reibman, A. R. (2024). Minimizing human labor for in-the-wild camera trap processing pipeline. In *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1--6. IEEE.

- [29] Chou, H.-P., Chang, S.-C., Pan, J.-Y., Wei, W., and Juan, D.-C. (2020). Remix: rebalanced mixup. In *European Conference on Computer Vision*, pages 95–110. Springer.
- [30] Clarfeld, L. A., Sirén, A. P., Mulhall, B. M., Wilson, T. L., Bernier, E., Farrell, J., Lunde, G., Hardy, N., Gieder, K. D., Abrams, R., et al. (2023). Evaluating a tandem human-machine approach to labelling of wildlife in remote camera monitoring. *Ecological informatics*, 77:102257.
- [31] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703.
- [32] Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. (2019). Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277.
- [33] Cunha, F., dos Santos, E. M., Barreto, R., and Colonna, J. G. (2021). Filtering empty camera trap images in embedded systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2438–2446.
- [34] Cunha, F., dos Santos, E. M., and Colonna, J. G. (2023). Bag of tricks for long-tail visual recognition of animal species in camera-trap images. *Ecological Informatics*, 76:102060.
- [35] Curtin, B. H. and Matthews, S. J. (2019). Deep learning for inexpensive image classification of wildlife on the raspberry pi. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0082–0087. IEEE.
- [36] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240.
- [37] de Lorm, T. A., Horswill, C., Rabaiotti, D., Ewers, R. M., Groom, R. J., Watermeyer, J., and Woodroffe, R. (2023). Optimizing the automated recognition of individual animals to support population monitoring. *Ecology and Evolution*, 13(7):e10260.

- [38] Dertien, J. S., Negi, H., Dinerstein, E., Krishnamurthy, R., Negi, H. S., Gopal, R., Gulick, S., Pathak, S. K., Kapoor, M., Yadav, P., et al. (2023). Mitigating human–wildlife conflict and monitoring endangered tigers using a real-time camera-based alert system. *BioScience*, 73(10):748–757.
- [39] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.
- [40] Duggan, M. T., Groleau, M. F., Shealy, E. P., Self, L. S., Utter, T. E., Waller, M. M., Hall, B. C., Stone, C. G., Anderson, L. L., and Mousseau, T. A. (2021). An approach to rapid processing of camera trap images with minimal human input. *Ecology and Evolution*, 11(17):12051–12063.
- [41] Dussert, G., Chamaillé-Jammes, S., Dray, S., and Miele, V. (2025). Being confident in confidence scores: Calibration in deep learning models for camera trap image sequences. *Remote Sensing in Ecology and Conservation*, 11(1):88–99.
- [42] Elias, A. R., Golubovic, N., Krintz, C., and Wolski, R. (2017). Where’s the bear?-automating wildlife image processing using iot and edge cloud systems. In *2017 IEEE/ACM Second Inter. Conf. on Internet-of-Things Design and Implementation (IoTDI)*, pages 247–258.
- [43] Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*, pages 363–370. Springer.
- [44] Fennell, M., Beirne, C., and Burton, A. C. (2022). Use of object detection in camera trap image identification: Assessing a method to rapidly and accurately classify human and animal detections for research and application in recreation ecology. *Global Ecology and Conservation*, 35:e02104.
- [45] Fergus, P., Chalmers, C., Matthews, N., Nixon, S., Burger, A., Hartley, O., Sutherland, C., Lambin, X., Longmore, S., and Wich, S. (2024). Towards context-rich automated biodiversity assessments: deriving ai-powered insights from camera trap data. *Sensors*, 24(24):8122.
- [46] Figueroa, K., Camarena-Ibarrola, A., García, J., and Villela, H. T. (2014). Fast automatic detection of wildlife in images from trap cameras. In *Progress in Pattern*

- Recognition, Image Analysis, Computer Vision, and Applications: 19th Iberoamerican Congress, CIARP 2014, Puerto Vallarta, Mexico, November 2-5, 2014. Proceedings 19*, pages 940–947. Springer.
- [47] Gabeff, V., Rußwurm, M., Tuia, D., and Mathis, A. (2024). Wildclip: Scene and animal attribute retrieval from camera trap data with domain-adapted vision-language models. *International Journal of Computer Vision*, 132(9):3770–3786.
- [48] Gadot, T., Istrate, S., Kim, H., Morris, D., Beery, S., Birch, T., and Ahumada, J. (2024). To crop or not to crop: Comparing whole-image and cropped classification on a large dataset of camera trap images. *IET Computer Vision*.
- [49] Giraldo-Zuluaga, J.-H., Salazar, A., Gomez, A., and Diaz-Pulido, A. (2017). Recognition of mammal genera on camera-trap images using multi-layer robust principal component analysis and mixture neural networks. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 53–60. IEEE.
- [50] Giraldo-Zuluaga, J.-H., Salazar, A., Gomez, A., and Diaz-Pulido, A. (2019). Camera-trap images segmentation using multi-layer robust principal component analysis. *The Visual Computer*, 35:335–347.
- [51] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.
- [52] Gomez, A., Diez, G., Salazar, A., and Diaz, A. (2016). Animal identification in low quality camera-trap images using very deep convolutional neural networks and confidence thresholds. In *International symposium on visual computing*, pages 747–756. Springer.
- [53] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.
- [54] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.
- [55] Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

- [56] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.
- [57] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [58] He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567.
- [59] He, W., Luo, Z., Tong, X., Hu, X., Chen, C., and Shu, Z. (2023). Long-tailed metrics and object detection in camera trap datasets. *Applied Sciences*, 13(10):6029.
- [60] He, Z., Kays, R., Zhang, Z., Ning, G., Huang, C., Han, T. X., Millspaugh, J., Forrester, T., and McShea, W. (2016b). Visual informatics tools for supporting large-scale collaborative wildlife monitoring with citizen scientists. *IEEE Circuits and Systems Magazine*, 16(1):73–86.
- [61] Hinton, G., Srivastava, N., and Swersky, K. (2012). Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e*.
- [62] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- [63] Hinton, G. E. (2022). The forward-forward algorithm: Some preliminary investigations. *CoRR*, abs/2212.13345.
- [64] Horn, G. V., Aodha, O. M., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. (2018). The inaturalist species classification and detection dataset. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8769–8778.
- [65] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324.

- [66] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- [67] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310--7311.
- [68] Huang, Y., Xiao, D., Liu, J., Tan, Z., Liu, K., and Chen, M. (2023). An improved pig counting algorithm based on yolov5 and deepsort model. *Sensors*, 23(14):6309.
- [69] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. *arXiv:1602.07360*.
- [70] Iashin, V., Lee, H., Schofield, D., and Zisserman, A. (2025). Self-supervised learning on camera trap footage yields a strong universal face embedder. *arXiv preprint arXiv:2507.10552*.
- [71] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704--2713.
- [72] Janzen, M., Ritter, A., Walker, P. D., and Visscher, D. R. (2019). Eventfinder: a program for screening remotely captured images. *Environmental monitoring and assessment*, 191:1--10.
- [73] Jia, L., Tian, Y., and Zhang, J. (2023). Neural architecture search based on packed samples for identifying animals in camera trap images. *Neural Computing and Applications*, 35(14):10511--10533.
- [74] Jocher, G., Chaurasia, A., Laughing-Q, Borovec, J., SkalskiP, , and other contributors (2020). Yolov5 by ultralytics. <https://github.com/ultralytics/yolov5>. Accessed: 2025-05-16.
- [75] Jocher, G., Chaurasia, A., Laughing-Q, Borovec, J., SkalskiP, and other contributors (2024). Yolov11: Ultralytics official implementation. <https://github.com/ultralytics/ultralytics>. Accessed: 2025-05-16.

- [76] Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., and Kalantidis, Y. (2020). Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*.
- [77] Kay, J., Kulits, P., Stathatos, S., Deng, S., Young, E., Beery, S., Van Horn, G., and Perona, P. (2022). The caltech fish counting dataset: a benchmark for multiple-object tracking and counting. In *European Conference on Computer Vision*, pages 290–311. Springer.
- [78] Kilima, F. G., Kaijage, S., Luhanga, E., and Torney, C. (2025). A yolov4 method for wild animal counting and behaviour detection using small-sized camera-trap image dataset. *ICTACT Journal on Soft Computing*, 15(4):3722–3728.
- [79] Kim, J., Suh, Y., Lee, J., Chae, H., Ahn, H., Chung, Y., and Park, D. (2022). Embeddedpigcount: Pig counting with video object detection and tracking on an embedded board. *Sensors*, 22(7):2689.
- [80] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [81] Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer.
- [82] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [83] Kutugata, M., Baumgardt, J., Goolsby, J. A., and Racelis, A. E. (2021). Automatic camera-trap classification using wildlife-specific deep learning in nilgai management. *Journal of Fish and Wildlife Management*, 12(2):412–421.
- [84] Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. (2014). Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*.
- [85] Leorna, S. and Brinkman, T. (2022). Human vs. machine: Detecting wildlife in camera trap images. *Ecological Informatics*, 72:101876.
- [86] Li, J., Li, D., Xiong, C., and Hoi, S. (2022). Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR.

- [87] Li, Y., Wang, T., Kang, B., Tang, S., Wang, C., Li, J., and Feng, J. (2020). Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10991--11000.
- [88] Lila.science (2021a). Caltech camera traps. <http://lila.science/datasets/caltech-camera-traps>. Accessed: 2021-02-17.
- [89] Lila.science (2021b). Snapshot serengeti. <http://lila.science/datasets/snapshot-serengeti>. Accessed: 2021-02-17.
- [90] Lila.science (2022). Wcs camera traps. <https://lila.science/datasets/wcscameratraps>. Accessed: 2022-03-17.
- [91] Lin, K.-H., Khorrami, P., Wang, J., Hasegawa-Johnson, M., and Huang, T. S. (2014a). Foreground object detection in highly dynamic scenes using saliency. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1125--1129. IEEE.
- [92] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980--2988.
- [93] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014b). Microsoft coco: Common objects in context. In *Eur. conf. on computer vision*, pages 740--755. Springer.
- [94] Liu, C., Li, H., Wang, Z., and Xu, R. (2025). Is a pure transformer effective for separated and online multi-object tracking? *ACM Transactions on Multimedia Computing, Communications and Applications*, 21(12):1--21.
- [95] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21--37. Springer.
- [96] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012--10022.

- [97] Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., and Yu, S. X. (2019). Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2537--2546.
- [98] Lonsinger, R. C., Dart, M. M., Larsen, R. T., and Knight, R. N. (2024). Efficacy of machine learning image classification for automated occupancy-based monitoring. *Remote Sensing in Ecology and Conservation*, 10(1):56--71.
- [99] Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.
- [100] Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- [101] Maile, R. E., Duggan, M. T., and Mousseau, T. A. (2023). The successes and pitfalls: Deep-learning effectiveness in a chernobyl field camera trap application. *Ecology and Evolution*, 13(9):e10454.
- [102] Miao, Z., Liu, Z., Gaynor, K. M., Palmer, M. S., Yu, S. X., and Getz, W. M. (2021). Iterative human and automated identification of wildlife images. *Nature Machine Intelligence*, 3(10):885--895.
- [103] Miguel, A., Beery, S., Flores, E., Klemesrud, L., and Bayraksicsmith, R. (2016). Finding areas of motion in camera trap images. In *2016 IEEE international conference on image processing (ICIP)*, pages 1334--1338. IEEE.
- [104] Nazir, S. and Kaleem, M. (2024). Object classification and visualization with edge artificial intelligence for a customized camera trap platform. *Ecological Informatics*, 79:102453.
- [105] Norman, D. L., Bischoff, P. H., Wearn, O. R., Ewers, R. M., Rowcliffe, J. M., Evans, B., Sethi, S., Chapman, P. M., and Freeman, R. (2023). Can cnn-based species classification generalise across variation in habitat within a camera trap survey? *Methods in Ecology and Evolution*, 14(1):242--251.
- [106] Norouzzadeh, M. S., Morris, D., Beery, S., Joshi, N., Jojic, N., and Clune, J. (2021). A deep active learning system for species identification and counting in camera trap images. *Methods in Ecology and Evolution*, 12(1):150--161.
- [107] Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C., and Clune, J. (2018). Automatically identifying, counting, and describing

- wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, 115(25):E5716--E5725.
- [108] Örn, F. (2021). *Computer Vision for Camera Trap Footage: Comparing Classification with Object Detection*. Master's thesis, Uppsala University.
- [109] Pantazis, O., Bevan, P., Pringle, H., Ferreira, G. B., Ingram, D. J., Madsen, E., Thomas, L., Thanet, D. R., Silwal, T., Rayamajhi, S., et al. (2024). Deep learning-based ecological analysis of camera trap images is impacted by training data quality and size. *arXiv preprint arXiv:2408.14348*.
- [110] Pantazis, O., Brostow, G. J., Jones, K. E., and Mac Aodha, O. (2021). Focus on the positives: Self-supervised learning for biodiversity monitoring. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 10583--10592.
- [111] Park, S., Cho, M., Kim, S., Choi, J., Song, W., Kim, W., Song, Y., Park, H., Yoo, J., Seo, S. B., et al. (2024). Exploring the potential application of a custom deep learning model for camera trap analysis of local urban species. *Landscape and Ecological Engineering*, 20(4):595--604.
- [112] Pícek, L., Šulc, M., Matas, J., Jeppesen, T. S., Heilmann-Clausen, J., Læssøe, T., and Frøslev, T. (2022). Danish fungi 2020-not just another image recognition dataset. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1525--1535.
- [113] Prudic, K. L., McFarland, K. P., Oliver, J. C., Hutchinson, R. A., Long, E. C., Kerr, J. T., and Larrivée, M. (2017). ebutterfly: leveraging massive online citizen science for butterfly conservation. *Insects*, 8(2):53.
- [114] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779--788.
- [115] Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6517--6525. IEEE.
- [116] Redmon, J. and Farhadi, A. (2018). Yolo3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

- [117] Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137--1149.
- [118] Ren, X., Han, T. X., and He, Z. (2013). Ensemble video object cut in highly dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1947--1954.
- [119] Reynolds, S. A., Beery, S., Burgess, N., Burgman, M., Butchart, S. H., Cooke, S. J., Coomes, D., Danielsen, F., Di Minin, E., Durán, A. P., et al. (2024). The potential for ai to revolutionize conservation: a horizon scan. *Trends in Ecology & Evolution*.
- [120] Rolnick, D., Aspuru-Guzik, A., Beery, S., Dilkina, B., Donti, P. L., Ghassemi, M., Kerner, H., Monteleoni, C., Rolf, E., Tambe, M., et al. (2024). Position: Application-driven innovation in machine learning. In *Forty-first International Conference on Machine Learning*.
- [121] Rowcliffe, J. M., Jansen, P. A., Kays, R., Kranstauber, B., and Carbone, C. (2016). Wildlife speed cameras: measuring animal travel speed and day range using camera traps. *Remote Sensing in Ecology and Conservation*, 2(2):84--94.
- [122] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211--252.
- [123] Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432.
- [124] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510--4520.
- [125] Santamaria, J. D., Isaza, C., and Giraldo, J. H. (2025). Catalog: A camera trap language-guided contrastive learning model. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1197--1206. IEEE.
- [126] Santos, E. M. d., Cunha, F., Colonna, J. G., and Carvalho, J. R. H. (2023). Monitoramento ambiental não invasivo utilizando dados de sensores e técnicas de aprendizagem de máquina. *Computação Brasil*, 1(50):24--28.

- [127] Schall, M., Knöfel, F. L., König, N. E., Kubeler, J. J., von Klinski, M., Linne-
mann, J. W., Liu, X., Schlegelmilch, I. J., Woyciniuk, O., Schild, A., et al. (2026).
Gorillawatch: An automated system for in-the-wild gorilla re-identification and pop-
ulation monitoring. In *2026 IEEE/CVF Winter Conference on Applications of Com-
puter Vision (WACV)*, pages 8364--8375. IEEE.
- [128] Schindler, F. and Steinhage, V. (2021). Identification of animals and recogni-
tion of their actions in wildlife videos using deep learning techniques. *Ecological
Informatics*, 61:101215.
- [129] Schneider, D., Lindner, K., Vogelbacher, M., Bellaafkir, H., Farwig, N., and
Freisleben, B. (2024). Recognition of european mammals and birds in camera trap
images using deep neural networks. *IET Computer Vision*, 18(8):1162--1192.
- [130] Schneider, S., Greenberg, S., Taylor, G. W., and Kremer, S. C. (2020). Three
critical factors affecting automated image species recognition performance for camera
traps. *Ecology and Evolution*, 10(7):3503--3517.
- [131] Schneider, S., Taylor, G. W., and Kremer, S. (2018). Deep learning object detec-
tion methods for ecological camera trap data. In *2018 15th Conference on computer
and robot vision (CRV)*, pages 321--328. IEEE.
- [132] Schneider, S., Taylor, G. W., Linquist, S., and Kremer, S. C. (2019). Past, present
and future approaches using computer vision for animal re-identification from camera
trap data. *Methods in Ecology and Evolution*, 10(4):461--470.
- [133] Seabold, S., Perktold, J., et al. (2010). Statsmodels: econometric and statistical
modeling with python. *scipy*, 7(1):92--96.
- [134] Shahinfar, S., Meek, P., and Falzon, G. (2020). “how many images do i need?”
understanding how sample size per class affects deep learning model performance
metrics for balanced designs in autonomous wildlife monitoring. *Ecological Infor-
matics*, 57:101085.
- [135] Shashidhara, B. M., Mehta, D., Kale, Y., Morris, D., and Hazen, M. (2020).
Sequence information channel concatenation for improving camera trap image burst
classification. *arXiv preprint arXiv:2005.00116*.
- [136] Simões, F., Bouveyron, C., and Precioso, F. (2023). Deepwild: Wildlife iden-
tification, localisation and estimation on camera trap videos using deep learning.
Ecological Informatics, 75:102095.

- [137] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [138] Smith, J., Wycherley, A., Mulvaney, J., Lennane, N., Reynolds, E., Monks, C.-A., Evans, T., Mooney, T., and Fancourt, B. (2024). Man versus machine: cost and carbon emission savings of 4g-connected artificial intelligence technology for classifying species in camera trap images. *Scientific Reports*, 14(1):14530.
- [139] Sokolova, M., Japkowicz, N., and Szpakowicz, S. (2006). Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015--1021. Springer.
- [140] Stevens, S., Wu, J., Thompson, M. J., Campolongo, E. G., Song, C. H., Carlyn, D. E., Dong, L., Dahdul, W. M., Stewart, C., Berger-Wolf, T., et al. (2024). Bioclip: A vision foundation model for the tree of life. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19412--19424. IEEE.
- [141] Straka, J., Hruz, M., and Picek, L. (2024). The hitchhiker's guide to endangered species pose estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 50--59.
- [142] Swanson, A., Kosmala, M., Lintott, C., Simpson, R., Smith, A., and Packer, C. (2015a). Data from: Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna.
- [143] Swanson, A., Kosmala, M., Lintott, C., Simpson, R., Smith, A., and Packer, C. (2015b). Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. *Scientific data*, 2.
- [144] Swinnen, K. R., Reijniers, J., Breno, M., and Leirs, H. (2014). A novel method to reduce time investment when processing videos from camera trap studies. *PloS one*, 9(6):e98881.
- [145] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1--9.
- [146] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818--2826.

- [147] Tabak, M. A., Falbel, D., Hamzeh, T., Brook, R. K., Goolsby, J. A., Zoromski, L. D., Boughton, R. K., Snow, N. P., VerCauteren, K. C., and Miller, R. S. (2022). Cameratrappdetector: Automatically detect, classify, and count animals in camera trap images using artificial intelligence. *BioRxiv*, pages 2022--02.
- [148] Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Newton, E. J., Boughton, R. K., Ivan, J. S., Odell, E. A., Newkirk, E. S., Conrey, R. Y., Stenglein, J., et al. (2020). Improving the accessibility and transferability of machine learning algorithms for identification of animals in camera trap images: Mlwic2. *Ecology and evolution*, 10(19):10374--10383.
- [149] Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Sweeney, S. J., VerCauteren, K. C., Snow, N. P., Halseth, J. M., Di Salvo, P. A., Lewis, J. S., White, M. D., et al. (2019). Machine learning to classify animal species in camera trap images: Applications in ecology. *Meth. in Ecology and Evolution*, 10(4):585--590.
- [150] Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *Int. Conference on Machine Learning*, pages 6105--6114.
- [151] Tan, M. and Le, Q. (2021). Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, pages 10096--10106. PMLR.
- [152] Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781--10790.
- [153] TEAM Network (2011). Terrestrial vertebrate protocol implementation manual, v. 3.1. Tropical Ecology, Assessment and Monitoring Network, Center for Applied Biodiversity Science, Conservation International, Arlington, VA, USA.
- [154] Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. (2019). Fixing the train-test resolution discrepancy. *Advances in neural information processing systems*, 32.
- [155] Tuia, D., Kellenberger, B., Beery, S., Costelloe, B. R., Zuffi, S., Risse, B., Mathis, A., Mathis, M. W., van Langevelde, F., Burghardt, T., et al. (2022). Perspectives in machine learning for wildlife conservation. *Nature communications*, 13(1):1--15.
- [156] Tydén, A. and Olsson, S. (2020). Edge machine learning for animal detection, classification, and tracking. Master's thesis, Linköping University.
- [157] Van Horn, G. and Perona, P. (2017). The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*.

- [158] Vargas-Felipe, M., Pellegrin, L., Guevara-Carrizales, A. A., López-Monroy, A. P., Escalante, H. J., and Gonzalez-Fraga, J. A. (2021). Desert bighorn sheep (*ovis canadensis*) recognition from camera traps based on learned features. *Ecological Informatics*, 64:101328.
- [159] Varghese, R. and Sambath, M. (2024). YoloV8: A novel object detection algorithm with enhanced performance and robustness. In *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pages 1--6. IEEE.
- [160] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [161] Velasco-Montero, D., Fernández-Berni, J., Carmona-Galán, R., Sanglas, A., and Palomares, F. (2024). Reliable and efficient integration of ai into camera traps for smart wildlife monitoring based on continual learning. *Ecological Informatics*, 83:102815.
- [162] Vélez, J., McShea, W., Shamon, H., Castiblanco-Camacho, P. J., Tabak, M. A., Chalmers, C., Fergus, P., and Fieberg, J. (2023). An evaluation of platforms for processing camera-trap data using artificial intelligence. *Methods in Ecology and Evolution*, 14(2):459--477.
- [163] Villa, A. G., Salazar, A., and Vargas, F. (2017). Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *Ecological informatics*, 41:24--32.
- [164] Viswanathan, A. S., Bock, A., Bent, Z., Peyton, M. A., Tartakovsky, D. M., and Santos, J. E. (2026). Two-stage wildlife event classification for edge deployment. *Sensors*, 26(4):1366.
- [165] Wang, X., Lian, L., Miao, Z., Liu, Z., and Yu, S. (2021). Long-tailed recognition by routing diverse distribution-aware experts. In *International Conference on Learning Representations*.
- [166] Wang, Y., Zhang, Y., Feng, Y., and Shang, Y. (2022). Deep learning methods for animal counting in camera trap images. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 939--943. IEEE.

- [167] Wei, W., Luo, G., Ran, J., and Li, J. (2020). Zilong: A tool to identify empty images in camera-trap data. *Ecological Informatics*, 55:101021.
- [168] Whytock, R. C., Suijten, T., van Deursen, T., Świeżewski, J., Mermiaghe, H., Madamba, N., Mouckoumou, N., Zwerts, J. A., Pambo, A. F. K., Bahaa-el din, L., et al. (2023). Real-time alerts from ai-enabled camera traps using the iridium satellite network: A case-study in gabon, central africa. *Methods in Ecology and Evolution*, 14(3):867--874.
- [169] Whytock, R. C., Świeżewski, J., Zwerts, J. A., Bara-Słupski, T., Koumba Pambo, A. F., Rogala, M., Bahaa-el din, L., Boekee, K., Brittain, S., Cardoso, A. W., et al. (2021). Robust ecological analysis of camera trap data labelled by a machine learning model. *Methods in Ecology and Evolution*, 12(6):1080--1092.
- [170] Wightman, R. (2019). Pytorch image models. <https://github.com/rwightman/pytorch-image-models>.
- [171] Wightman, R., Touvron, H., and Jégou, H. (2021). Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*.
- [172] Wilber, M. J., Scheirer, W. J., Leitner, P., Heflin, B., Zott, J., Reinke, D., Delaney, D. K., and Boulton, T. E. (2013). Animal recognition in the mojave desert: Vision tools for field biologists. In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 206--213. IEEE.
- [173] Willi, M., Pitman, R. T., Cardoso, A. W., Locke, C., Swanson, A., Boyer, A., Veldthuis, M., and Fortson, L. (2019). Identifying animal species in camera trap images using deep learning and citizen science. *Methods in Ecology and Evolution*, 10(1):80--91.
- [174] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645--3649. IEEE.
- [175] Wu, B., Liu, C., Jiang, F., Li, J., and Yang, Z. (2023). Dynamic identification and automatic counting of the number of passing fish species based on the improved deepsort algorithm. *Frontiers in Environmental Science*, 11:1059217.
- [176] Xi, T., Wang, J., Qiao, H., Lin, C., and Ji, L. (2021). Image filtering and labelling assistant (ifla): Expediting the analysis of data obtained from camera traps. *Ecological Informatics*, 64:101355.

- [177] Xiong, Y., Liu, H., Gupta, S., Akin, B., Bender, G., Kindermans, P.-J., Tan, M., Singh, V., and Chen, B. (2020). Mobilenets: Searching for object detection architectures for mobile accelerators. *arXiv preprint arXiv:2004.14525*.
- [178] Yang, D.-Q., Li, T., Liu, M.-T., Li, X.-W., and Chen, B.-H. (2021a). A systematic study of the class imbalance problem: Automatically identifying empty camera trap images using convolutional neural networks. *Ecological Informatics*, 64:101350.
- [179] Yang, D.-Q., Ren, G.-P., Tan, K., Huang, Z.-P., Li, D.-P., Li, X.-W., Wang, J.-M., Chen, B.-H., and Xiao, W. (2021b). An adaptive automatic approach to filtering empty images from camera traps using a deep learning model. *Wildlife Society Bulletin*, 45(2):230–236.
- [180] Yang, D.-Q., Tan, K., Huang, Z.-P., Li, X.-W., Chen, B.-H., Ren, G.-P., and Xiao, W. (2021c). An automatic method for removing empty camera trap images using ensemble learning. *Ecology and Evolution*, 11(12):7591–7601.
- [181] Yang, Z., Tian, Y., Wang, L., and Zhang, J. (2025). Enhancing generalization in camera trap image recognition: Fine-tuning visual language models. *Neurocomputing*, 634:129826.
- [182] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.
- [183] Yousif, H., Yuan, J., Kays, R., and He, Z. (2017). Fast human-animal detection from highly cluttered camera-trap images using joint background modeling and deep learning classification. In *2017 IEEE international symposium on circuits and systems (ISCAS)*, pages 1–4. IEEE.
- [184] Yousif, H., Yuan, J., Kays, R., and He, Z. (2019). Animal scanner: Software for classifying humans, animals, and empty frames in camera trap images. *Ecology and evolution*, 9(4):1578–1589.
- [185] Yu, X., Wang, J., Kays, R., Jansen, P. A., Wang, T., and Huang, T. (2013). Automated identification of animal species in camera trap images. *EURASIP Journal on Image and Video Processing*, 2013:1–10.
- [186] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022). Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21. Springer.

- [187] Zhang, Y., Wei, X.-S., Zhou, B., and Wu, J. (2021). Bag of tricks for long-tailed visual recognition with deep convolutional neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(4):3447--3455.
- [188] Zhang, Z., He, Z., Cao, G., and Cao, W. (2016). Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification. *IEEE Transactions on Multimedia*, 18(10):2079--2092.
- [189] Zhou, B., Cui, Q., Wei, X.-S., and Chen, Z.-M. (2020). Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9719--9728.
- [190] Zhu, H., Tian, Y., and Zhang, J. (2022). Class incremental learning for wildlife biodiversity monitoring in camera trap images. *Ecological Informatics*, page 101760.
- [191] Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28--31. IEEE.
- [192] Zualkernan, I., Dhou, S., Judas, J., Sajun, A. R., Gomez, B. R., and Hussain, L. A. (2022). An iot system using deep learning to classify camera trap images on the edge. *Computers*, 11(1):13.
- [193] Zualkernan, I. A., Dhou, S., Judas, J., Sajun, A. R., Gomez, B. R., Hussain, L. A., and Sakhnini, D. (2020). Towards an iot-based deep learning architecture for camera trap image classification. In *2020 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*, pages 1--6. IEEE.

Heuristic Counting Solution for the iWildCam 2021 Challenge

In this Appendix, we provide the details of our final submission to the iWildCam 2021 challenge based on the MaxBoxCount heuristic.

As described in Section 6.3.1, the MaxBoxCount consists of two components: an image burst classifier based on an ensemble of EfficientNet-B2 [150] models, and a counting heuristic based on the bounding boxes generated by MegaDetectorV4 [13]. In Section A.1, we describe in detail how the classifiers were trained and how the ensemble aggregation was performed to produce sequence-level predictions. Finally, in Section A.2, we present the challenge results and include ablation studies to validate our design choices.

A.1 Species classifier

We trained two species classifiers based on the EfficientNet-B2 architecture: one trained using full images and another (bbox) trained with square crops containing animals, extracted from bounding boxes generated by MegaDetectorV4. We chose this approach following the iWildCam 2020 solutions, where the bbox model can provide better predictions due to the highlighted animal. On the other hand, the full image model has been shown to be more effective at classifying images with animal herds. We used only the provided camera-trap data to train the classifier models.

To train the bbox model, we treated all bounding boxes with a confidence score equal to or higher than 0.6¹ as independent training samples and applied square crops

¹The threshold of 0.6 for including a bounding box in the training set was selected based on the held-out validation set from the training data that was used to adjust the training hyperparameters.

around them, using the size of the largest bbox side. If no bounding box met this confidence threshold, we used the full image for that training instance. The image label was assigned as the bounding box label.

During inference, the species prediction assigned to each image is the weighted average of the predictions from the bounding box with the highest score, the full image, and their horizontally mirrored versions, following the approach used by the winning solution of iWildCam 2020: 0.15 (full image) + 0.15 (mirrored full image) + 0.35 (bbox) + 0.35 (mirrored bbox). To classify a sequence, we average the predictions of all nonempty images within the sequence.

Handling class imbalance. To address the class imbalance problem inherent in real-world applications, such as the one represented in iWildCam 2021 dataset, we applied the Balanced Group Softmax (BAGS) method. Following the original BAGS papers, we grouped classes into 4 softmax groups based to the number N of training instances: $N < 10$, $10 \leq N < 100$, $100 \leq N < 1000$, $N \geq 1000$. For each softmax group, we included the class “others” to represent instances from all classes not included in that group. During training, for each softmax, we undersampled the “others” category to ensure it contained at most eight times the number of instances belonging to other classes per batch, to prevent it from dominating the softmax. Additionally, we included a special softmax group for the foreground/background classification. For the final prediction, we remapped all predictions to the original softmax, ignoring the “others” class. As in the original paper, these predictions are not true probabilities since they do not sum up to one, but we consider the highest value as the BAGS prediction.

Image preprocessing. For image preprocessing, we applied a square crop centered on the bounding boxes or a random crop of aspect ratio sampled in $[3/4, 4/3]$ and area in $[65\%, 100\%]$ for full image, followed by a random horizontal flip and RandAugment ($N=6$, $M=2$). Then, the images were normalized and resized to match the input dimensions of the network. During the fix train/test resolution stage, we used inference preprocessing, which consists only of normalizing and resizing the image or square crop to the input dimensions of the network.

Multi-stage training. We trained the models in multiple stages. First, each model was trained using the standard softmax with the default input resolution of EfficientNet-B2 (260 x 260). In the first stage, only the classifier layer was trained for 4 epochs, while the backbone – previously initialized with ImageNet weights – remained frozen. In the second stage, all layers were unfrozen, and all weights were fine-tuned for 20 epochs. In the third stage, to fix the train/test resolution discrepancy issue [154], we fine-tuned only the last inverted bottleneck block of the EfficientNet-B2 architecture and the classifier layer for 2 more epochs. This was done using a higher input resolution

(380 x 380) and applying the inference preprocessing steps to the images. Afterward, we removed the plain softmax and trained the BAGS header on top of the backbone, which was kept frozen. In the fourth stage, the BAGS header was trained for 12 epochs using training preprocessing with data augmentation at the inference resolution (380 x 380). Finally, in the fifth stage, the BAGS header was fine-tuned for 2 epochs using inference-time preprocessing. Table A.1 presents a summary of the parameters used.

Table A.1. Training hyperparameters for each stage of the classifiers training.

	Weights trained	Epochs	LR	Preprocessing	Resolution
Stage 1*	Classifier layer	4	0.01	Training	260 x 260
Stage 2	All	20	0.01	Training	260 x 260
Stage 3	Last block and classif.	2	0.001	Inference	380 x 380
Stage 4	BAGS header	12	0.01	Training	380 x 380
Stage 5	BAGS header	2	0.001	Inference	380 x 380

* For the bounding box model only.

Implementation details. To adjust the training hyperparameters, we held out a validation set from the training set based on the locations. However, to train the final models, we used all images available in the original training set. The models were initialized with ImageNet weights and trained with a batch size of 32 using SGD with an initial learning rate of 0.01 (0.001 during the fix train/test resolution stages) and momentum of 0.9. The learning rate was linearly warmed up from 0 to the initial value over one-third of the steps in the first epoch of each training stage, and then decayed to 0 following the cosine schedule. We also applied label smoothing with a value of 0.1.

A.2 Results

The iWildCam 2021 challenge used the mean columnwise root-mean-squared error (MCRMSE), described in subsection 2.8.3, to evaluate solutions for counting animals in camera traps. Table A.2 presents the final standings based on the private score, which was calculated using 50% of the test set. Our approach was ranked first in the challenge out of 42 teams that entered the competition.

As stated by the competition organizers [11], the MCRMSE metric tends to be a small number even when the error in counts is large. This occurs because camera traps typically have a small number of individuals and because the model is double penalized for both species and counting errors. The organizers provided some simple baselines, which we specify in Table A.2 for comparison purposes. One baseline uses

Table A.2. The iWildCam 2021 final results on the private test set.

Solution	Private score
1st place (ours)	0.0293278
2nd place	0.0293786
3rd place	0.0308570
4th place	0.0308756
5th place	0.0336513
6th place	0.0339028
Baseline - iWildCam 2020 Winner	0.0348171
7th place	0.0376347
8th place	0.0383956
Baseline - All zeros	0.0385147

the iWildCam 2020 winner’s species predictions combined with the maximum number of bounding boxes with confidence score higher than 0.8 across the sequence – in line with our solution. The all zeros baseline predicts zero for all instances, which performs surprisingly well. This highlights how challenging this counting problem is, given its inherent conditions.

We present an ablation study to demonstrate the effectiveness of certain design choices in our heuristic solution, as shown in Table A.3. This study considers only the species classifier at the image level used for the final submission. A well-known approach to improving classification predictions is using test-time augmentation; in our solution, we added a horizontal flip to both full image and bbox models. We also found that averaging the species predictions across images in a sequence provides better performance than the majority voting approach. Finally, we used a threshold of 0.8 for counting MegaDetectorV4 bounding boxes, as proposed by the organizers. We initially used 0.9 during the competition but found that it discarded too many valid bounding boxes.

Table A.3. An ablation study of our iWildCam2021 solution. The bold line represents our final solution.

Test-time aug.	Seq. aggreg.	MDv4 threshold	Private score
-	Voting	0.9	0.0314948
Horizontal flip	Voting	0.9	0.0305960
Horizontal flip	Average	0.9	0.0294853
Horizontal flip	Average	0.8	0.0293278