*Extended Abstract*

# Towards an efficient smart camera trap for wildlife monitoring

*Delia Velasco-Montero*[1,*] *Jorge Fernández-Berni*[1] *Ricardo Carmona-Galán*[1] *Ariadna Sanglas*[2] *Francisco Palomares*[2]

[1] *Instituto de Microelectrónica de Sevilla, CSIC - Universidad de Sevilla, Cl. Américo Vespucio 28, 41092, Seville, Spain*
[2] *Estación Biológica de Doñana, CSIC, Cl. Américo Vespucio 26, 41092, Seville, Spain*
** E-mail: delia@imse-cnm.csic.es*

**Abstract:** This paper presents an AI-based smart camera-trap hardware system designed for wildlife monitoring. Our camera incorporates classification convolutional neural networks optimized for running on embedded platforms at the edge. We primarily focus on the task of blank-image filtering to alleviate subsequent manual or automatic analysis. System specifications in the proposed design enable real-time image processing and autonomous operation in the wild. Field tests conducted in Sierra de Aracena Natural Park (Spain) revealed challenges arising from environmental scene variations. To overcome these challenges, we employed transfer learning using diverse datasets and location-specific data. All in all, this study demonstrates the feasibility of building smart camera traps and emphasizes the importance of dataset diversity and adaptation to specific locations.

## 1 Introduction

Manual image classification is a major burden for wildlife conservationists, and the easy and cheap access to camera devices is making the amount of visual information unmanageable. Fortunately, artificial intelligence (AI) and software automation have emerged as a solution to alleviate this task. Automatic vision procedures such as individual identification, species classification, and false-positive ('blank images') filtering on collected data are examples of the possibilities that are being explored and tested on the field.

Currently, most automatic wildlife image procedures are cloud-based, utilizing data collected from cameras and stored and processed on servers. As an alternative, we propose the design of **smart** cameras that leverage power-efficient AI implementations **on edge devices**. Our ultimate objective is real-time detection of relevant events and remote reporting. However, the deployment of these cameras at different target locations for remote animal monitoring is extremely challenging: camera-trap locations not seen during training leads to a non-negligible accuracy loss.

This paper is organized as follows. We first introduce (sect. 2) the proposed visual monitoring system, based on end-to-end integration of AI pipelines on embedded hardware. Our initial case study comprises a dataset of images collected at several locations of Sierra de Aracena Natural Park, Spain. Promising performance results were achieved. However, in situ testing of this first approach with our customized camera reveals a number of issues to be solved, such as robustness against illumination changes and limited model generalization to new scenarios (AI overfitted to the training dataset). To address these and other aspects, we introduced additional AI techniques in sect. 3 with a focus on improving model generalization to the specific location of interest for conservationists. In this second case study (in-situ images), we evaluated a new pipeline, obtaining better model accuracy while maintaining the same inference performance in terms of CNN processing time.

### *1.1 Related Work*

New public environmental datasets are constantly released, thereby expanding the available resources for research. For instance, LILA BC repository [17] contains a variety of datasets for biology conservation, including Snapshot Serengeti (SS) [19], Caltech Camera Traps [16], and North American Camera Trap Images [18]. Given the growing availability of large-size collection of camera trap images, Deep Learning (DL), and specifically its realization in the form of convolutional neural networks (CNNs), has emerged as the state-of-the-art method to assist wildlife conservationists. Previous works have demonstrated the effectiveness of CNNs in species identification/classification [9, 15, 21, 30], and individual detection tasks [5, 10, 12, 25, 28] – it is worth mentioning the large-sized, high-accuracy animal detection model MegaDetector [6, 14]. Furthermore, the integration of CNNs in comprehensive computer-vision pipelines for wildlife monitoring have also been investigated [6, 20, 22]. However, generalization to new locations still remains a remarkable challenge [5, 26]. In this regard, techniques such as data augmentation and transfer learning have proven to be critical to improve training performance. The present study is framed in the area of edge processing of wildlife data, which has been explored by a limited number of research works [7, 8, 13].

### *1.2 Proposed Methodology*

Our approach includes these steps for integrating AI on edge devices:

1. **Data collection**: typically carried out by conservationists, who place commercial camera traps in the wild and then retrieve large volumes of data. The images must be manually labeled into categories extract useful information (and/or train AI models).
2. **Network training**: we can either design and/or train a CNN from scratch or use pre-trained weights on pre-defined architectures to speed up this step.
3. **System integration**: a suitable integration of the AI model on the targeted embedded platform is critical. Considering the number of software libraries, hardware processors, and optimization techniques available, selecting the optimal combination is not trivial.
4. **System testing**: finally, the experimental validation of the system requires conducting a series of on-site operational tests in diverse real-world scenarios. These scenarios always pose new challenges, demanding methodology refinement to finally meet the prescribed requirements.

Steps (1)–(3) are covered in sect. 2, while sect. 3 also applies step (4) for a specific wildlife scenario.

## 2 Hardware and AI for Animal Monitoring. Case Study I

This case study is the result of a collaboration with the mammal research group at Doñana Biological Station (CSIC) [1]. Their

**Fig. 1**: **Case study I.** Images captured by commercial camera traps at multiple locations of Sierra de Aracena Natural Park. The categories associated to these examples are *feral cat* (left), *other species* (middle), and *blank* (right)

research focuses on studying and analyzing population trends of various species in their natural habitats. One particular area of interest is the population of feral cats in Sierra de Aracena Natural Park. Over the years, biologists have collected an extensive dataset of images and video sequences using commercial camera traps (Bushnell Trophy Cam HD Brown model 119874) placed at different locations throughout the Park. 'Blank' captures – containing no animal – constitute a high proportion of the collected data. Therefore, two obvious categories to filter out irrelevant information are *animal* and *blank*. Note that commercial camera traps also register other species such as dogs, foxes, martens, and wild pigs. This admits a further categorization into three classes: *feral cat*, *other species*, and *blank*. Figure 1 illustrates examples of these categories.

### 2.1 Artificial Intelligence: Classification Networks

Over 8000 images of the three aforementioned categories were selected to compose a balanced dataset. The AI core of our smart camera is based on classification networks. We define three classification tasks. **(1) Blank filtering**, to distinguish between blank images and those containing relevant information. **(2) Species identification**, to distinguish between feral cats and other species. This would be a subsequent processing step after blank filtering. **(3) Combined species identification and blank filtering**, which implies direct classification of the images into the three categories considered: feral cats, other species, and blank.

We selected the well-known SqueezeNet [11] as the classification model. The reason is that this CNN is an energy-efficient and fast-response model suitable for deployment on embedded devices [29]. To adapt the original architecture to our particular requirements, we replaced the last 1000-category layer with a classifier featuring 2 outputs (for Tasks 1-2), or 3 outputs (for Task 3). We trained the model (whose weights had been obtained from ImageNet-based training) using TensorFlow and Keras [4]. For each task, the available data were split into a training dataset (80%) and a validation dataset (20%). The images were augmented – applying zooming, rotation, width/height shifting, and horizontal flipping – during model training. The training results for the three tasks are presented in Table 1, which includes precision, recall, and accuracy rates calculated on the validation data. For calculating these rates, we considered *Animal* as the 'positive' class and *Blank* as the 'negative' class*. Keep in mind that a high precision rate indicates a low number of false positives (FP), while a high recall indicates few false negatives (FN).

According to biologists' requirements, we prioritized the implementation of Task 1 (**blank filtering**) in our system. This task is relevant because it greatly reduces human labour in any wildlife scenario and is applicable to all species. Likewise, removing spurious images right after capturing extends the system lifetime in terms of SD-card memory saturation and alleviates the computational load of automatic off-site analysis. Although other approaches for blank filtering could be experimented and tested (i.e., image difference), our aim is developing a global solution based on CNNs, with the future intention of incorporating additional network outputs for extended classification tasks.

**Table 1** Classification performance rates of Tasks 1-3 on Case Study I

| Task | Precision | Recall | Accuracy |
|---|---|---|---|
| Task 1) Blank filtering | 0.96 | 0.75 | 0.84 |
| Task 2) Species identification | 0.99 | 0.69 | 0.77 |
| Task 3) Combined id. & blank filtering | 0.89 | 0.96 | 0.69 |

### 2.2 Embedded Hardware

We implemented the early prototype on an embedded CPU-based processing platform (Raspberry Pi 4B [24]), equipped with an off-the-shelf camera module [23] *. To simulate the operation of commercial camera traps, we incorporated a passive infrared (PIR) sensor to detect moving objects around the system before capturing and analyzing images. Finally, to enable field deployment, the hardware set-up was completed by an external battery providing 5 V. A picture of the system including labels of the hardware components is shown in Fig. 2).

### 2.3 Smart Camera-Trap Application

Our system aims to provide the basic functionality of a commercial camera trap plus on-device image processing to make decisions. The workflow of the application running on the device is as follows:

| **Trigger** → | **Capture frames** → | **Analysis** → | **Real-time actions** |
|---|---|---|---|
| (PIR sensor) | (camera) | (CNN) | (alarm) |

When the PIR sensor detects activity, the camera captures a burst of images, which constitute the input for the trained CNN. If animal presence is identified, an alarm is sent to park staff, along with a

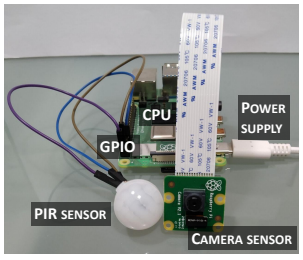


**Fig. 2**: Main hardware components for the developed camera-trap vision system

*For Task 2, 'positive' class represents 'Cat' while 'negative' represents 'Other Species'.

*We plan to use a more energy-efficient embedded device in the future: [3].

thumbnail image and the corresponding timestamp. Additional real-time actions, for example using actuators, could also be incorporated in the future thanks to the versatility of Raspberry Pi 4B to integrate peripheral devices. Furthermore, the system is able to detect nighttime periods by analyzing the captured images according to thresholds associated to the median and standard deviation of pixels.

The application offers different configurable parameters for user customization, such as the number of images to take, the CNN weights, or the PIR sensitivity. Concerning the software employed, we combined multiple libraries for image processing [2], camera access, GPIO reading, and network inference. We optimized the CNN execution to be run on our hardware platform [29].

### *2.4 System Performance*

We conducted laboratory measurements to determine the system performance. The CNN **processing time** (including image loading, pre-processing, and inference) is approximately 90 ms, thereby enabling provision of real-time alarms. Concerning **power consumption**, for a 5-V power supply, the system demands around 0.43 mA when idle and 0.86 mA during image processing. Solar power can be easily integrated in the future to further extend the system operation time.

## 3 Field Tests and System Evolution. Case Study II

After developing a first version of the smart camera trap and conducting laboratory tests, we carried out field tests at **a specific location of the Sierra de Aracena Natural Park**. This location was chosen because of the high probability of feral cat passing. Interestingly, this location had not been registered by any of the commercial camera traps that provided the aforementioned images for the training dataset. Figure 3 shows an example of image at this location – completely new from the perspective of the CNN –, which we refer to as 'Case study II'. This allowed us to evaluate the system adaptability to untrained scenarios, providing valuable insights for further refinement and optimization. Note that exclusively feral cats were observed at this specific location, thus making the task of blank filtering the only feasible option.

**Fig. 3**: **Case study II.** Image captured by our smart camera at a specific location – completely new for the trained CNN – in the park (dataset II). This image corresponds to a positive example (*feral cat*).

### *3.1 On-site Results, System Limitations*

In this field test, we configured the camera to apply the CNN trained on dataset I (as described in sect. 2) on a burst of 10 frames whenever motion was detected by the PIR sensor. Over a period of approximately 26 hours of continuous operation in the field, the camera captured more than 35000 images. Out of these, we manually classified more than 10000 images (dataset II). We ensured that this dataset covered a wide range of illumination scenarios and vegetation changes – more on this later on. Although, the labeled dataset was unbalanced (13% cats – 87% blanks), many distinct situations were observed – e.g., individuals located far away from the camera or camouflaged.

**Table 2** Classification performance rates of the CNN trained and tested on various datasets

| Training // Test datasets | Precision | Recall | Accuracy | F1-score |
|---|---|---|---|---|
| Dataset I // Dataset II | 0.19 | 0.81 | 0.52 | 0.31 |
| SS // Dataset I | 0.72 | 0.84 | 0.73 | 0.78 |
| SS // Dataset II | 0.13 | 0.92 | 0.18 | 0.23 |

For the aforementioned CNN trained on dataset I (sect. 2) operating on these manually labeled frames (dataset II) as testing data, we obtained the accuracy results reported in the first row of Table 2 – also including f1-score*. These results (with low precision due to many FP) highlight the performance loss resulting from forcing inference on an unseen location [5]. To delve deeper into the network operation and its performance, we leveraged **explainable DL** techniques; specifically, we applied the Grad-CAM algorithm [27]. This analysis led to the following observations:

- Most FP in this scenario were due to network misinterpretation of vegetation changes as cat individuals. See, for instance, a fallen leaf in the foreground – initially not there – in Fig. 4, which is incorrectly identified by the network as an animal. This leaf is present in a high number of negative frames.
- Regarding FN, we can stress the difficulty of the dataset: detecting distant or camouflaged individuals can be challenging even for the human eye.

### *3.2 Large datasets. Transfer Learning*

In order to improve the model performance, we decided to enhance the training dataset and adapt it to the application scenario. The larger and more diverse the dataset, the better the model generalization. Therefore, we train the CNN using the well-known S. Serengeti (SS) dataset – the largest camera trap dataset with 1.2 million images [19]. By specifically employing animal/blank images from the first two seasons of SS to train the CNN, the model achieves a validation accuracy of 0.80 in blank filtering.

One straightforward approach consists on directly using this trained CNN on dataset II images. However, as mentioned earlier, the very fact that the CNN must deal with an unseen location, together with the changes in illumination, vegetation, etc. at this location, which lie out of the scope of those reflected in the SS dataset, lead to many FP and subsequent loss in accuracy and precision rates. For instance, if we apply our SS-trained model to datasets I-II, we obtain the results in the second and third rows of Table 2. While the

*Note that dataset II is unbalanced, which makes the accuracy rate less informative compared to precision and recall (and consequently f1-score).*

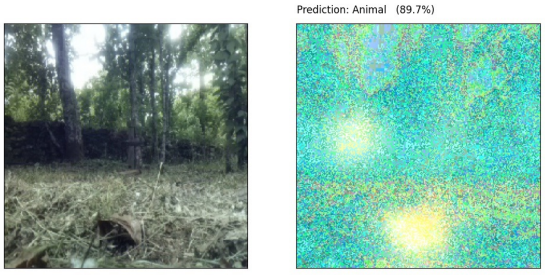


**Fig. 4**: Grad-CAM [27] explaination map (*right*) for a FP example (*left*). Yellowish pixels correspond to the areas that better *explain* the CNN output (shown over the image).

results on dataset I may be acceptable, when testing on dataset II, we still observed a significant number of FP primarily attributed to vegetation changes similar to shown in Fig. 4. This clearly calls for additional steps to address the challenges posed by environmental variations. In this direction, a more effective approach is to exploit Transfer Learning (TL) from this model trained on SS. TL is specially beneficial when the knowledge from a previous training must be conveyed to a new application scenario using a small dataset, particularly when this dataset shares similarities with the original one.

### 3.3 On-device Continuous Learning

The proposed solution we describe next involves (a) leveraging large-size datasets through TL, and (b) employing a dataset tailored to a **specific camera location**.

Given that there is no prior large-size dataset available for a specific location, we produce a synthetic dataset on-the-fly. These synthetic images are generated to simulate the background captured by the camera at that location, along with synthetically placed individuals (of the desired species). Data generation includes variations in position, size, and other parameters related to data augmentation. See examples of synthetic training images in Fig. 5. This approach aims to incorporate knowledge about the particular deployment location into the CNN model. The updated workflow for the application is as follows:

**Capture blanks → Train CNN**
↓
**Trigger → Capture frames → Analysis → Real-time actions**

It includes a first calibration period in which (i) the camera takes background images, (ii) generates synthetic data containing individuals (cats), and (iii) re-trains the network with these data (TL from SS). The main challenge was the integration of the training process into the embedded hardware.

The results for our testing dataset are reported in Table 3. The effectiveness of the proposed solution is evident when comparing Tables 3and 2.

### 3.4 Discussion

We found that the majority of errors occur when it comes to identifying distant or camouflaged individuals (FN), or when there are significant changes in illumination or vegetation (FP), suggesting the need for endowing the model with further information about the surveyed scene. For instance, we can periodically re-train the CNN with batches of synthetic data generated from temporally-distributed captured backgrounds (Continual Learning approach). It is also worth noting that processing field wildlife data poses challenges and difficulties. As an example, we conducted tests using the well-known

**Fig. 5**: Synthetic training data generated on-site by combining the background of the location with artificially placed individuals.

**Table 3** Classification performance rates of the CNN trained on synthetic data generated in situ (transfer learning from SS) and tested on field data

| Training // Test datasets | Precision | Recall | Accuracy | F1-score |
|---|---|---|---|---|
| SS → on-site imgs // Dataset II | 0.53 | 0.46 | 0.88 | 0.49 |

**Table 4** Classification performance rates of MegaDetector tested on on-field real data

| Test dataset | Precision | Recall | Accuracy | F1-score |
|---|---|---|---|---|
| Dataset II | 0.28 | 0.88 | 0.70 | 0.43 |

MegaDetector model [6, 14] on the collected images, and this model also produces classification errors – the results are reported in Table 4 *. A detection example is shown in Fig. 6.


**Fig. 6**: Sample image analyzed with MegaDetector. This FP example clearly illustrates how challenging classification of data collected in the wild is.

## 4 Conclusions

The implementation of a smart camera-trap system using AI models on edge devices has the potential to significantly reduce human labor in wildlife monitoring scenarios and enables the provision of real-time alarms for quick actuation. We have developed a smart camera-trap based on a classification CNN tailored for execution on an embedded platform. The performance of the proposed system allows for real-time operation and unsupervised use. However, field tests revealed misclassifications caused by varying environmental conditions. This calls for robust and adaptable AI models to achieve reliable operation regardless of the deployment location. To enhance the model performance, we propose the application of TL from a larger and diverse dataset. In particular, we re-train the model with location-specific data. While inference is still far from being perfect, there is room for further refinement, for example confidence thresholding on CNN outputs, to enhance accuracy, precision, and recall rates.

Overall, this study demonstrates the potential of smart camera traps in wildlife monitoring, the importance of optimizing the system performance, the relevance of dataset diversity and adaptation to specificities of the deployment locations, and the ongoing need for refining techniques to address the challenges posed by real-world environmental conditions.

## 5 Acknowledgments

This work was supported by Project SEMIoTICS (PID2021-128009OB-C31), funded by MCIN/AEI /10.13039/501100011033 and EU "ERDF: A way of making Europe"; by Project ULTIMATE (TED2021-131835B-I00), funded by MCIN/AEI /10.13039/501100011033 and by the European Union NextGenerationEU/ PRTR; and by Project SUMHAL (LIFEWATCH-2019-09-CSIC-4, POPE 2014-2020), funded by Ministry of Science and Innovation of Spain through European Regional Development Fund.

*To calculate performance rates, we considered as 'positive' those predictions from MegaDetector including a detected object (regardless the number or position); otherwise, we assumed 'negative' predictions.

## 6 References


1 Doñana Biological Station. Conservation Biology and Global Change Research Group. http://www.ebd.csic.es/biologia-de-la-conservacion-y-cambio-global.

2 OpenCV. https://opencv.org/.

3 Raspberry Pi Zero W. https://www.raspberrypi.org/products/raspberry-pi-zero-w/.

4 Abadi, M. et al. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

5 Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

6 Sara Beery, Dan Morris, and Siyu Yang. Efficient pipeline for camera trap image review. *arXiv*, (1907.06772), 2019.

7 Sreedevi C.K and Sarita E. Automated Wildlife Monitoring Using Deep Learning. In *Proceedings of the International Conference on Systems, Energy & Environment (ICSEE) 2019*, 2019.

8 B. H. Curtin and S. J. Matthews. 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, 2019.

9 Alexander Gomez Villa, Augusto Salazar, and Francisco Vargas. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *Ecological Informatics*, 41:24 – 32, 2017. ISSN 1574-9541. doi: https://doi.org/10.1016/j.ecoinf.2017.07.004. URL http://www.sciencedirect.com/science/article/pii/S1574954116302047.

10 Yanhui Guo, Thomas Rothfus, Amira S. Ashour, Lei Si, du Chunlai, and Tih-Fen Ting. A varied channels region proposal and classification network for wildlife image classification under complex environment. *IET Image Processing*, 14, 11 2019. doi: 10.1049/iet-ipr.2019.1042.

11 Forrest Iandola, Song Han, Matthew Moskewicz, Khalid Ashraf, William Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1MB model size. *arXiv*, (1602.07360), 2016.

12 A. Loos, C. Weigel, and M. Koehler. Towards automatic detection of animals in camera-trap images. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1805–1809, 2018. doi: 10.23919/EUSIPCO.2018.8553439.

13 Ankit Mathur and Saelig Khattar. Real-time wildlife detection on embedded systems. Technical report, Stanford University, June 2019. URL http://ilpubs.stanford.edu:8090/1165/.

14 MegaDetector. GitHub microsoft. CameraTraps repository. https://github.com/microsoft/CameraTraps/.

15 Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S. Palmer, Craig Packer, and Jeff Clune. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, 115(25):E5716–E5725, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1719367115. URL https://www.pnas.org/content/115/25/E5716.

16 Labeled Information Library of Alexandria: Biology and Conservation (LILA BC). Caltech Camera Traps. http://lila.science/datasets/caltech-camera-traps,.

17 Labeled Information Library of Alexandria: Biology and Conservation (LILA BC). Camera Traps Datasets . http://lila.science/category/camera-traps/,.

18 Labeled Information Library of Alexandria: Biology and Conservation (LILA BC). North American Camera Trap Images . http://lila.science/datasets/nacti,.

19 Labeled Information Library of Alexandria: Biology and Conservation (LILA BC). Snapshot Serengeti. http://lila.science/datasets/snapshot-serengeti,.

20 J. Parham, C. Stewart, J. Crall, D. Rubenstein, J. Holmberg, and T. Berger-Wolf. An animal detection pipeline for identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1075–1083, 2018. doi: 10.1109/WACV.2018.00123.

21 J. Parham, C. Stewart, J. Crall, D. Rubenstein, J. Holmberg, and T. Berger-Wolf. An animal detection pipeline for identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1075–1083, 2018. doi: 10.1109/WACV.2018.00123.

22 Noa Rigoudy, Abdelbaki Benyoub, Aurélien Besnard, Carole Birck, Yoann Bollet, Yoann Bunz, Nina De Backer, Gérard Caussimont, Anne Delestrade, Lucie Dispan, Jean-François Elder, Jean-Baptiste Fanjul, Jocelyn Fonderflick, Mathieu Garel, William Gaudry, Agathe Gérard, Olivier Gimenez, Arzhela Hemery, Audrey Hemon, Jean-Michel Jullien, Maden Le Barh, Isabelle Malafosse, Malory Randon, Romain Ribière, Sandrine Ruette, Guillaume Terpereau, Wilfried Thuiller, Valentin Vautrain, Bruno Spataro, Vincent Miele, and Simon Chamaillé-Jammes. The DeepFaune initiative: a collaborative effort towards the automatic identification of the french fauna in camera-trap images. *bioRxiv*, 2022. doi: 10.1101/2022.03.15.484324. URL https://www.biorxiv.org/content/early/2022/03/17/2022.03.15.484324.

23 RPi-camera. Raspberry Pi Documentation. Accessories. Camera. https://www.raspberrypi.org/documentation/accessories/camera.html.

24 RPi4. Raspberry Pi 4. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

25 S. Schneider, Graham W. Taylor, and S. C. Kremer. Deep Learning Object Detection Methods for Ecological Camera Trap Data. *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 321–328, 2018.

26 Stefan Schneider, Saul Greenberg, Graham W. Taylor, and Stefan C. Kremer. Three critical factors affecting automated image species recognition performance for camera traps. *Ecology and Evolution*, 10(7):3503–3517, 2020. doi: https://doi.org/10.1002/ece3.6147. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ece3.6147.

27 Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. doi: 10.1109/ICCV.2017.74.

28 A. Singh, M. Pietrasik, G. Natha, N. Ghouaiel, K. Brizel, and N. Ray. Animal detection in man-made environments. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1427–1438, 2020. doi: 10.1109/WACV45572.2020.9093504.

29 Delia Velasco-Montero, Jorge Fernández-Berni, Ricardo Carmona-Galán, and Ángel Rodríguez-Vázquez. Optimum selection of DNN model and framework for edge inference. *IEEE Access*, 6:51680–51692, 2018. doi: 10.1109/ACCESS.2018.2869929.

30 Marco Willi, Ross T. Pitman, Anabelle W. Cardoso, Christina Locke, Alexandra Swanson, Amy Boyer, Marten Veldthuis, and Lucy Fortson. Identifying animal species in camera trap images using deep learning and citizen science. *Methods in Ecology and Evolution*, 10(1):80–91, 2019. doi: 10.1111/2041-210X.13099. URL https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13099.