



TinyML-based Image Recognition for Real-time Capybara Detection on Resource-constrained Embedded Hardware in Precision Livestock Farming

Tung Chiun Wen*, Fabiano Gregolin, Késia Oliveira da Silva Miranda, Luana Maria Benicio, Miguel Ângelo Cyrillo Narbot

GBAZP - Grupo de Pesquisa em Bem-Estar, Ambiente e Zootecnia de Precisão, ESALQ/USP, Piracicaba, São Paulo 13418-900, Brazil.

How to cite this paper: Tung Chiun Wen, Fabiano Gregolin, Késia Oliveira da Silva Miranda, Luana Maria Benicio, Miguel Ângelo Cyrillo Narbot. (2026). TinyML-based Image Recognition for Real-time Capybara Detection on Resource-constrained Embedded Hardware in Precision Livestock Farming. *International Journal of Food Science and Agriculture*, 10(1), 40-51.
DOI: 10.26855/ijfsa.2026.03.006

Received: January 25, 2026
Accepted: February 22, 2026
Published: March 21, 2026

Corresponding author: Tung Chiun Wen, GBAZP - Grupo de Pesquisa em Bem-Estar, Ambiente e Zootecnia de Precisão, ESALQ/USP, Piracicaba, São Paulo 13418-900, Brazil.

© 2026 by the author(s).
This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives (CC BY-NC-ND) license, which permits non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited and is not modified or adapted.
<https://creativecommons.org/licenses/by-nc-nd/4.0/>

Abstract

The modernization of animal production increasingly relies on precision technologies to improve productivity, reduce operational costs, and maintain animal welfare and environmental sustainability. Among these technologies, artificial intelligence and computer vision have enabled automated monitoring of animals throughout the production chain. However, manual data collection remains a significant limitation for achieving efficient and scalable monitoring systems. This study aimed to develop and evaluate an image recognition system based on Deep Learning implemented through Tiny Machine Learning (TinyML) for the detection of capybaras in invasive animal containment systems. The methodology comprised four stages: image collection, dataset preparation and analysis, model training, and deployment on embedded hardware. A convolutional neural network based on the MobileNet architecture was trained and optimized using the Edge Impulse platform for deployment on resource-constrained edge devices. The results demonstrated that the MobileNet-based model achieved 97% accuracy during validation and 94.83% during testing on a holdout set of 116 images, with a capybara detection precision of 96.22% and sensitivity of 92.72%. The classifier was successfully deployed on an ESP32 microcontroller, enabling real-time inference on embedded hardware. These findings demonstrate the feasibility of integrating TinyML and computer vision for low-cost, intelligent monitoring systems. The proposed approach provides a practical solution for automated invasive animal detection and contributes to the advancement of embedded artificial intelligence in precision livestock farming.

Keywords

TinyML; convolutional neural network; MobileNet; edge inference; invasive species detection; precision livestock farming; Brazilian spotted fever; ESP32

1. Introduction

In recent decades, animal production systems have undergone a significant technological transformation driven by the emergence of Precision Livestock Farming (PLF), which integrates sensing technologies, automation, and artificial intelligence to improve productivity, reduce operational costs, and ensure environmental sustainability and animal welfare. To achieve this, investments in precision technologies are necessary at every stage, from rural

properties to slaughterhouses, enabling individual animal analysis and monitoring to generate relevant data. The application of devices such as IoT, robots, microchips, as well as techniques like precision animal production, cloud computing, and artificial intelligence, has been studied along with their challenges and prospects [1].

Precision livestock technologies offer benefits to animal protein producers by providing real-time monitoring of animal health and performance, enhanced management through data analysis, early disease detection, resource use efficiency, and improved product quality. These tools empower producers to make more informed decisions, promoting animal welfare, increasing productivity, and meeting the growing global demand for animal-based food products [2].

Improving operational efficiency has long been a central objective in livestock production systems. However, this objective has faced challenges in achieving the desired balance between quality and costs. One of the main limitations is the continued reliance on non-automated data collection, which increases operational costs and limits the scalability of monitoring systems. Technologies based on the IoT provide remote and precise monitoring, making the process not only smarter but also more economical [3].

Real-time monitoring alone is insufficient to enable intelligent livestock management. It is necessary to follow a cycle of observation, diagnosis, decision, and action, using data collected to optimize the process. During this cycle, the data are transmitted to an IoT cloud platform, where decision models determine the operational or health status of the animals. Machine Learning (ML) techniques evaluate whether intervention is required, which is then evaluated and applied by the user, restarting the cycle [4].

In this context, Deep Learning plays a fundamental role. It is a machine learning approach based on the principles of artificial neural networks. Its distinction from traditional neural networks lies in the depth of its layers, which allows it to uncover hidden structures in unlabeled and unstructured data. Deep learning networks that perform automatic feature extraction without human intervention have demonstrated improved performance compared with earlier algorithms in several applications [5, 6].

This technology can be implemented in intelligent embedded devices using Deep Learning models via TinyML. TinyML refers to the deployment of machine learning models on highly resource-constrained devices, such as microcontrollers and IoT devices, focusing on lightweight techniques that enable localized, real-time intelligence across a wide range of applications [7]. This approach enables IoT devices to perform local inference without relying on external computational resources, reducing latency and energy consumption [8]. In the context of wildlife and livestock monitoring, recent studies have demonstrated the viability of deploying convolutional neural networks on microcontrollers for animal identification tasks, including the use of MobileNet and SqueezeNet architectures optimized for edge inference [9, 10]. Despite these advances, the application of TinyML for automated detection of invasive wildlife species in containment systems remains underexplored, representing a significant research gap addressed by the present study.

In several regions of South America, the increasing interaction between wildlife and human environments has raised serious public health concerns, particularly involving *Hydrochoerus hydrochaeris* (capybaras). These mammals serve as amplifying hosts for *Rickettsia rickettsii*, transmitted by *Amblyomma sculptum*, the primary tick vector of this pathogen in Brazil [11]. *Rickettsia rickettsii* is the causative agent of Brazilian Spotted Fever (BSF), the most lethal tick-borne disease in the Western Hemisphere, with a case fatality rate of up to 40% in untreated patients [12, 13]. Between 2010 and 2022, Brazil registered over 2,500 confirmed BSF cases, with São Paulo state accounting for the majority of fatalities [14]. The proximity of capybara populations to human and livestock environments has been identified as a key epidemiological risk factor, reinforcing the need for automated, scalable monitoring technologies capable of detecting the presence of these animals in managed areas.

The objective of this study was to develop and experimentally evaluate an embedded image recognition system based on Tiny Machine Learning for capybara identification in systems designed for invasive animal containment.

2. Methods

The research was conducted in four stages: (1) image collection; (2) data analysis and preparation; (3) training and modeling; and (4) implementation on embedded devices. Figure 1 illustrates the research flow.

The images were collected from the Shutterstock® image bank, a repository containing over 300 million licensed photographs. Although this source provides controlled licensing and consistent image quality, it introduces a potential domain shift between training data and real-world deployment conditions, given that field images captured by a 2 MP embedded camera differ substantially from high-resolution stock photographs [15]. To partially mitigate this limitation, strict variability criteria were applied in image selection to maximize diversity in animal posture,

background composition, and lighting conditions. Future studies should incorporate images captured directly by embedded sensors under field conditions to reduce this distributional discrepancy and improve model robustness. Images were manually selected and initially categorized into three classes: (i) capybara (*Hydrochoerus hydrochaeris*), (ii) deer (*Subulo gouazoubira*), and (iii) puma (*Puma concolor*), which were later reorganized into two classes for the final classification model. Each image was individually inspected to verify whether the animal's position and visibility were suitable for classification. Images meeting these criteria were subsequently downloaded. The selection criteria were medium-sized images (from 200x200 pixels to 800x800 pixels) and large-sized images (800x800 pixels and above); exclusion of images with icons, paintings, montages, people, or brands; and prioritization of images with animals in their natural habitat. Figure 2 illustrates some of the selected images.

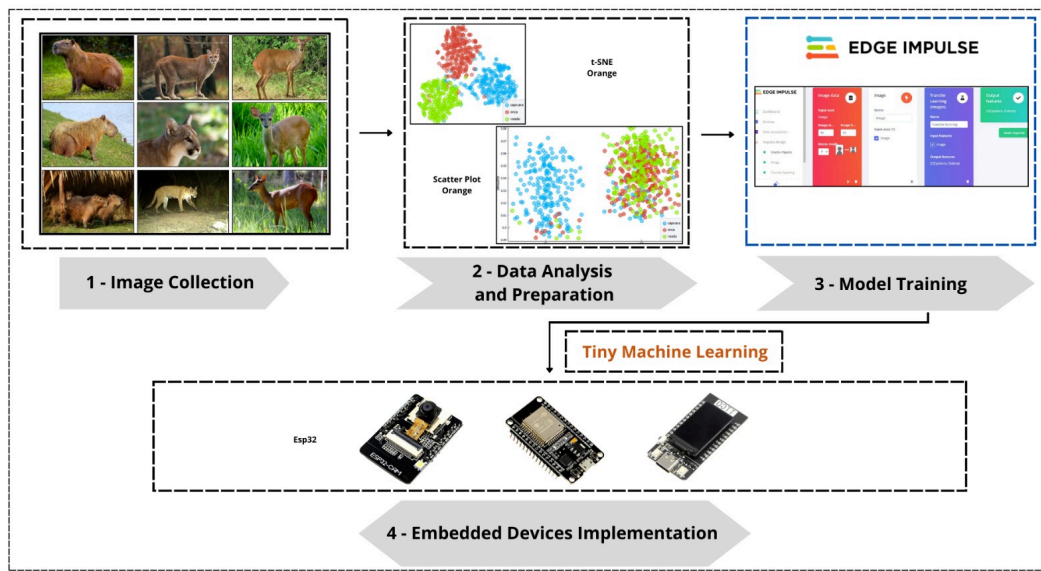


Figure 1. Research steps.



Figure 2. Example of images that comprise the library.

For the dataset construction, the Makesense® platform was employed to annotate the images for label generation, to assign class labels to each image for model training – Figure 3 – an essential step for training machine learning algorithms, especially for tasks related to computer vision, such as object classification and detection.

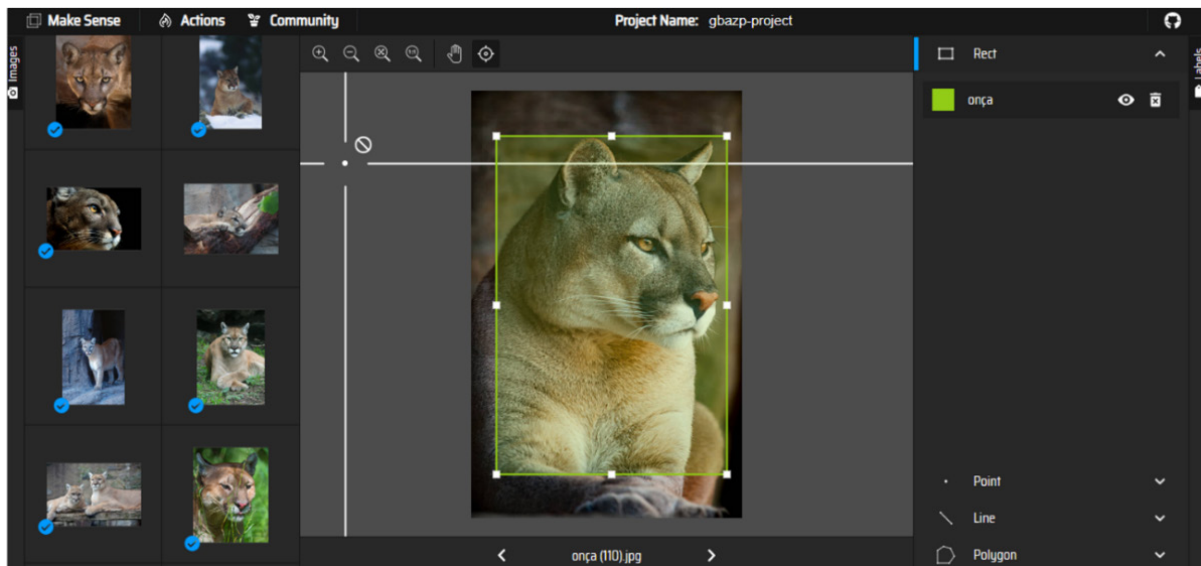


Figure 3. Annotation and labeling process on the Makesense® platform.

This stage involved the analysis and classification of images into distinct categories to form a consistent and reliable dataset, allowing the machine learning algorithm to perform more precise training. For this purpose, the Orange Data Mining software was used – Figure 4 – a free No-Code tool based on Python, which utilizes objects (widgets) to handle and analyze data, apply machine learning algorithms, and employ statistical analysis tools.



Figure 4. Screen of information in Orange Data Mining.

For a preliminary analysis of the dataset’s quality, the dataset was loaded through the “Import Images” widget in Orange Data Mining, along with the respective classifications, for cluster analysis. This aims to assess the quality of the groupings using unsupervised learning techniques.

Upon completion of the data loading, the deep learning architecture SqueezeNet was selected. The “Image Embeddings” widget was used to apply this architecture and generate a feature map for each image class.

After obtaining the feature map, the t-SNE was applied, a stochastic dimensionality reduction technique often used alongside PCA preprocessing (Principal Components Analysis), which allowed for the visualization of cluster concentration across the three initially labeled classes: capivara, puma, and deer, as illustrated in Figure 5.

Subsequently, the k-means algorithm was used to evaluate the possibility of increasing or reducing the number of clusters.

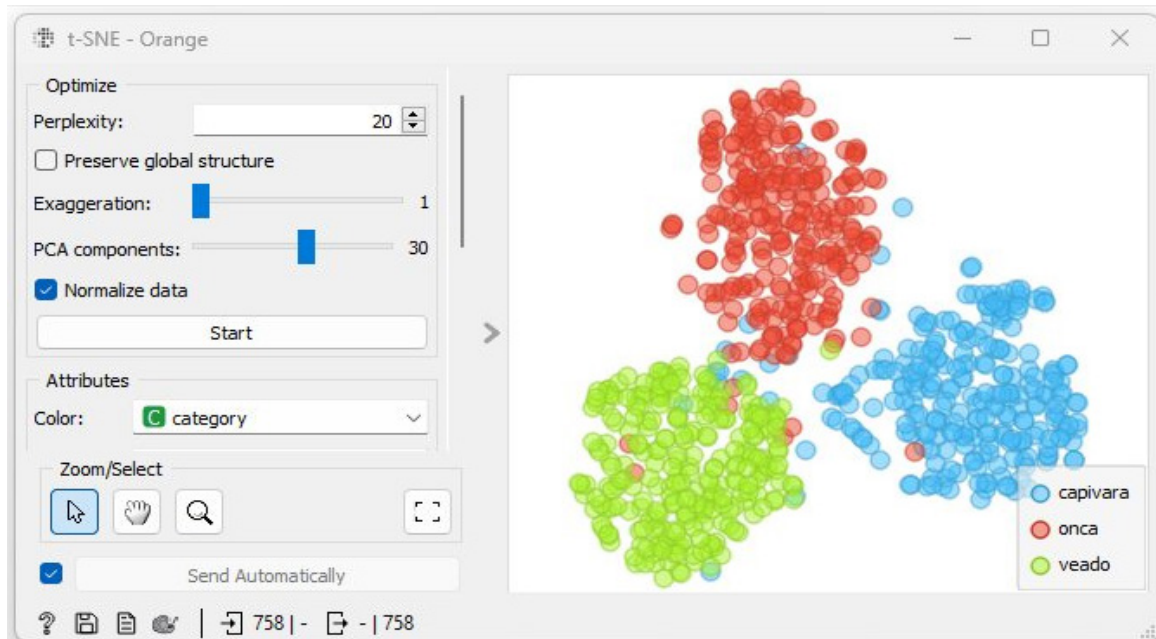


Figure 5. Clustering with t-SNE.

For neural network training, the Edge Impulse platform was employed. This platform is a free online graphical interface-based tool built on Python based on Python that facilitates dataset processing and model development in Deep Learning architectures to obtain the classification model and implement Tiny Machine Learning. This is achieved through optimizers and the creation of libraries for embedded systems. During the training and validation of the project, the MobileNet deep learning architecture was utilized.

Edge Impulse treats each project as an “impulse”. It is from this creation that the project is initiated, with the ultimate goal of developing an optimized and compact model that can be implemented in embedded systems. Edge Impulse incorporates Tiny Machine Learning modeling from the outset, enabling the creation of real-world datasets, training and evaluating machine learning models, and deploying them on any edge device [16].

Both in the training and testing of the classification model, metrics were extracted from the confusion matrix – Table 1 – which assesses the accuracy of a model obtained through supervised learning algorithms. In the training phase, the confusion matrix displays the model’s accuracy with the training data. In the testing phase, it shows the accuracy with the reserved data that was not used in training (HoldOUT), allowing for the evaluation of the model’s generalization capability [17, 18]. Where: TP: True Positive; TN: True Negative; FP: False Positive; FN: False Negative; N: number of samples tested.

Table 1. Confusion matrix

	Predictive class	
Real Class	TP	FN
	FP	TN

The metrics used for evaluation were accuracy, precision, and sensitivity (recall). Where:
 Accuracy: $((TP + TN) / N)$, Precision: $(TP / (TP + FP))$, Sensitivity: $(TP / (TP + FN))$

A system for the recognition and containment of capybaras was developed and implemented. This system is designed to automatically trigger the activation of a containment gate, with the goal of controlling the movement of these animals in specific areas – Figure 6.

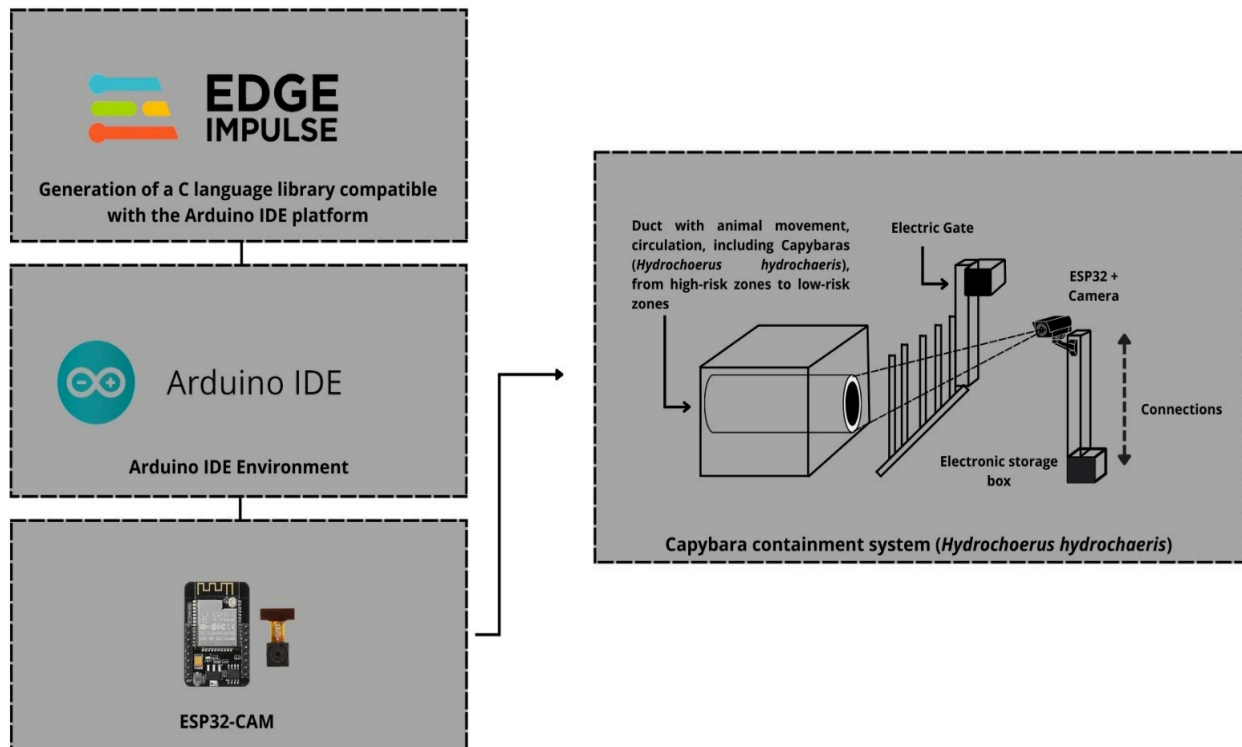


Figure 6. Deployment Process.

3. Results and Discussion

Image acquisition was performed using the Shutterstock® image bank, which houses approximately 300 million images, from which a subset was retrieved for this study. Of these, around 11,000 are cataloged as photos of capybaras, 51,000 as pumas, and 590,000 as deer. Additionally, the IMAGEYE browser extension, version 1.6.1, was used for image capture.

The initial dataset consisted of 1,048 images, providing variability in animal posture, background composition, and lighting conditions for training a binary classification model. These images underwent a new manual analysis to verify compliance with the criteria. During this stage, only images that fully met the criteria were selected, resulting in the exclusion of 290 images and a final total of 758 images for the training set. The dataset comprised: 256 images of capybaras, 255 of pumas, and 247 of deer.

The collected images were labeled using the Makesense® platform to annotate the images and generate the labels, indicating the class to which each image belongs. Each image was individually uploaded and annotated to generate the labels. The dataset was then assigned classification labels for capybara, deer, and puma. The labeling process generated annotation files in .txt format, which were placed in a dedicated folder. Additionally, another folder was created containing only the collection of images in “.jpg” format.

In the analysis stage, Cluster analysis was performed using Orange Data Mining. SqueezeNet was used as the feature extractor. The t-SNE visualization revealed three partially overlapping clusters corresponding to capybara, puma, and deer classes (Figure 5).

K-means uses the “Silhouette Score” metric to evaluate the quality of a clustering. The results were as follows: for 4 classes, Silhouette Score = 0.103; for 3 classes, Silhouette Score = 0.124; and for 2 classes, Silhouette Score = 0.129. Based on these results, we opted for two classes (2 clusters), where the classes Puma (in red) and Deer (in green) from Figure 5 were grouped into a single set, aligning with the primary objective of identifying capybaras. Consequently, two classes were adopted: Capybara and Others (Deer + Puma), as illustrated by the Scatter Plot Widget – Figure 7.



Figure 7. Clustering into two classes in the scatter plot widget.

This stage ensured the preparation of a balanced dataset with two classes (Capybara vs. Others) for the neural network training stage used to obtain the classification model. To maintain the balance between the two classes [19], the ‘Others’ class was randomly constructed using 249 images (puma and deer).

The classification model used in this study was based on a convolutional neural network architecture derived from MobileNet, which is widely adopted for computer vision tasks on resource-constrained devices due to its computational efficiency.

During the training stage, the network parameters were optimized to minimize classification error using the training dataset.

In addition to the training dataset, a validation subset was used to monitor model performance and prevent overfitting during training. Periodically, the neural network was applied to the validation set, errors were measured, and weights were updated to improve the model’s generalization of the learning.

During the testing stage, the trained model was evaluated using a holdout dataset composed of images not used during the training phase [17]. In this stage, the trained convolutional neural network processes these new images using the parameters obtained during the training phase and evaluates the classification accuracy for each sample.

In Edge Impulse, a total of 505 images were loaded, divided into 2 classes: 256 images of Capybaras and 249 of Others. Using the holdout method, approximately 80% of images (389) were allocated for training and validation, while the remaining 116 images (approximately 20%) were reserved exclusively for the testing phase of the classification model.

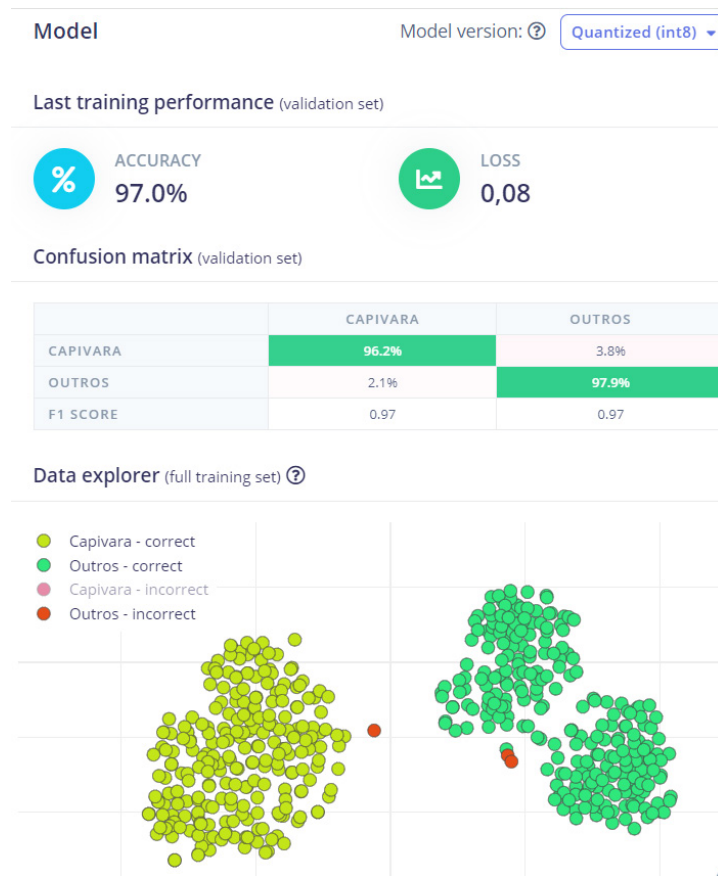
Each image was loaded into Edge Impulse at a resolution of 160×160 with the “Fit longest axis” option, utilizing the MobileNet deep learning architecture due to its suitability for embedded systems, given hardware limitations. The neural network was initially configured with the MobileNet architecture, a final dense layer of 16 neurons, dropout = 0.1, and the results were analyzed according to combinations of epochs (training cycles) and learning rate – Table 2.

Edge Impulse features the EON Tuner tool, which evaluates various configurations, adjustments, and architectures under different hardware constraints. We utilized the characteristics suited to the ESP32 hardware.

The EON Tuner ran for 17 minutes and 24 seconds and identified the optimal configuration: MobileNet architecture with 64 neurons in the final dense layer, learning rate of 0.0005, dropout rate of 0.5, 30 training epochs, and input image resolution of 160×160 pixels. The model was compiled using the Adam optimizer and categorical cross-entropy loss function, consistent with standard practices for classification tasks with neural networks [20]. This configuration achieved 97% accuracy on the validation set (Figure 8). The final model was exported as a compressed Arduino library (.zip), with a total size compatible with the ESP32 flash memory constraints (≤ 4 MB).

Table 2. Initial training results

Learning Rate	Epochs	Accuracy	Loss	Time (m:s)
0.0001	50	92%	0.23	4:25
0.0001	80	92%	0.23	6:17
0.0001	100	92%	0.23	8:05
0.0005	30	92%	0.22	4:02
0.0005	40	92%	0.22	4:51
0.0005	50	92%	0.22	5:15
0.00001	20	71%	0.58	3:21
0.00001	100	92%	0.27	7:51
0.00001	200	92%	0.22	11:03
0.00003	50	92%	0.25	5:41
0.00003	80	92%	0.24	7:41
0.00003	200	93%	0.23	11:40

**Figure 8. Model Validation.**

In the testing phase, the 116 images (55 Capybaras and 61 Others) were used to evaluate the classification model. The achieved accuracy was 94.83% – Figure 9.



Figure 9. Model Testing.

Applying the test results to the confusion matrix, a total of 116 images were tested, consisting of 55 Capybaras and 61 Others – Table 3. The per-class sensitivity was 92.72% for Capybaras (51 true positives out of 55) and 96.72% for Others (59 true negatives out of 61) – Table 4. The overall accuracy was calculated – Table 5.

Table 3. Confusion Matrix

		Predictive class	
		Capybara	Others
Real class	Capybara	51	4
	Others	2	59

Table 4. Accuracy with reserved data, precision, and sensitivity

Metrics	Calculation	Result	(%)
Accuracy	$(51+59) / 116$	0.9483	94.83
Capybara Precision	$51/(51+2)$	0.9622	96.22
Other Precision	$59/(59+4)$	0.9365	93.65
Capybara Sensitivity	$51/(51+4)$	0.9272	92.72
Other Sensitivity	$59/(59+2)$	0.9672	96.72

Table 5. Summary of classification metrics

Class	Precision	Sensitivity	F1 – Score	Accuracy
Capybara	96.22%	92.72%	94.44%	94.83%
Others	93.65%	96.72%	95.16%	94.83%

With the accuracy of the results obtained exceeding 90%, the next step was to implement the classifier model on the embedded ESP32 hardware. Before finalizing the hardware implementation, simulations of the model were conducted using the Edge Impulse online simulator. Three images of each animal species were selected for real-time testing – Figure 10 – displays some of the tested examples and their respective results.

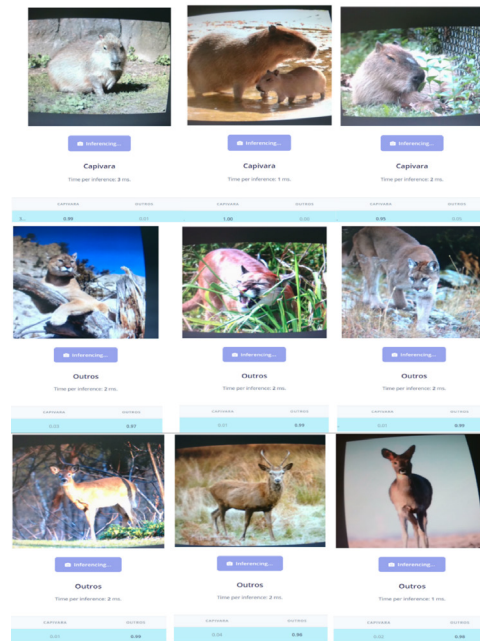


Figure 10. Online Simulation of the Model.

For hardware implementation, the ESP32-CAM module was selected [22] due to its low cost (~US\$ 5-10), dual-core Xtensa LX6 processor (up to 240 MHz), 520 KB SRAM, 4 MB flash memory, integrated Wi-Fi/Bluetooth, and an OV2640 camera sensor (2 MP, maximum resolution 1600×1200 pixels). This module is natively supported by the Edge Impulse platform and programmable via Arduino IDE (version 2.x), making it suitable for TinyML inference at the edge [22]. The library generated by Edge Impulse integrates the trained classifier model and ESP32 hardware drivers into a compressed Arduino library. After deployment, the system performs real-time local image capture and inference, without requiring cloud connectivity. A retriggerable timer circuit was added to the system, activating the containment gate mechanism whenever the TinyML classifier detects the presence of a capybara, with a configurable trigger delay to prevent false actuation from transient detections.

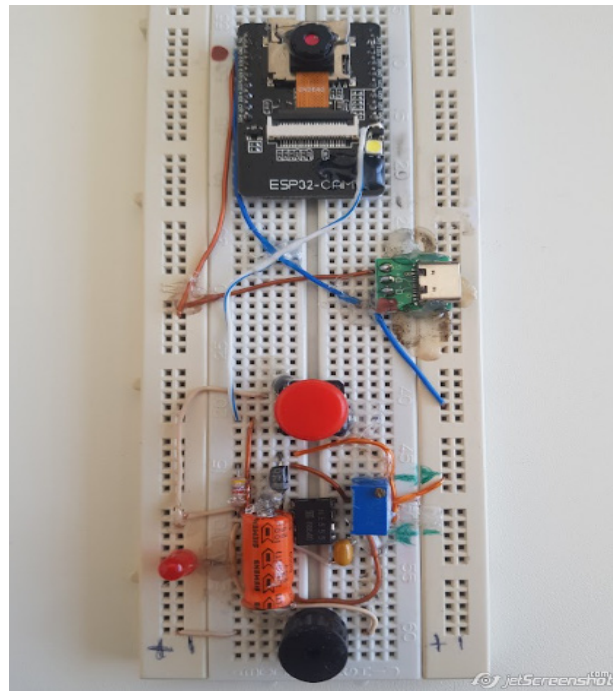


Figure 11. Embedded System with TinyML.

The results indicate that TinyML-based models can achieve reliable classification performance when deployed on low-power embedded hardware, consistent with previous studies. Chowdhery *et al.* [21] reported accuracy between 85% and 92% for MobileNet-based classifiers deployed on microcontrollers for wildlife species identification, while Zennaro *et al.* [9] demonstrated 89-93% accuracy in embedded animal monitoring systems using similar architectures. The 94.83% test accuracy achieved in the present study is therefore competitive within this reference range, particularly given the constraints of the ESP32-CAM hardware. In addition, the exploratory data analysis performed with Orange Data Mining supported the preparation of a structured two-class dataset, with both validation (97%) and testing (94.83%) results exceeding 90% accuracy. In real-world deployments of TinyML systems, performance variations may occur due to environmental conditions, hardware limitations, and variability in image acquisition. Performance variations in real-world deployments may be associated with hardware limitations of microcontrollers and the lower optical quality of embedded camera sensors compared with conventional imaging systems. Images from the dataset were used in both validation and testing phases. The Edge Impulse online simulation utilized a smartphone camera with a resolution above 5 MP, whereas the ESP32-CAM device has a 2 MP camera.

To characterize the computational footprint of the deployed model, inference latency, model size, and peak RAM usage were measured using the Edge Impulse profiling tool targeting the Espressif ESP-EYE (ESP32 240 MHz) hardware profile. The final quantized (int8) model occupied approximately 319.7 KB of flash memory, required 254.0 KB of peak RAM during inference, and achieved a total inference latency of approximately 5,217 ms per frame (15 ms for image processing and 5,202 ms for transfer learning inference) – all within the operational constraints of the ESP32-CAM (4 MB flash, 520 KB SRAM), consistent with microcontroller-class TinyML deployments reported by Banbury *et al.* [8] and David *et al.* [10]. Although an inference latency of 5,217 ms per frame (~0.19 fps) is substantially lower than the frame rates associated with continuous video processing, it is well-suited to the specific requirements of the proposed containment gate activation system. In this application, the system does not require high-frequency continuous detection; rather, it must reliably confirm the presence of a capybara within a decision window compatible with the mechanical response time of the containment gate. The observed latency is therefore operationally acceptable and consistent with event-triggered detection architectures reported in similar embedded wildlife monitoring systems [8, 10]. These metrics confirm that the proposed system is capable of autonomous edge inference without requiring cloud connectivity or external processing units, making it viable for deployment in remote livestock environments with limited infrastructure.

4. Conclusion

This study developed and validated a low-cost capybara recognition system based on Tiny Machine Learning and deep learning, deployed on embedded systems for invasive animal containment. The proposed approach demonstrated that image classification models can be successfully deployed on resource-constrained hardware, enabling real-time detection with accuracy exceeding 90%.

The results indicate that the integration of TinyML with computer vision techniques provides a solution for automated detection of capybaras in invasive animal containment systems. By enabling local inference on embedded devices, the system reduces dependence on external computational resources while maintaining reliable performance for practical applications in precision livestock environments.

Despite the promising results, key limitations must be acknowledged. The use of stock photography as the sole training data source introduces a domain shift relative to images captured under field conditions by the embedded 2 MP sensor. Additionally, the relatively small dataset (758 images for training, 116 for testing) may limit model generalizability across diverse environmental scenarios such as nocturnal conditions, rain, or partial occlusion. The binary classification scheme (Capybara vs. Others) was effective for the intended application but may require expansion if the system is adapted to distinguish among multiple invasive species. Future studies should prioritize the acquisition of field-captured images using the target hardware, implementation of data augmentation strategies to simulate sensor-level noise, and evaluation of quantization-aware training to further reduce model size and inference latency. The integration of this system with IoT platforms for remote monitoring and alert generation represents a natural extension toward fully autonomous precision livestock management.

References

- [1] Sevegnani KB. Zootecnia de precisão: desafios para a produção animal. In: Tecnologia e inovação na agricultura: aplicação, produtividade, e sustentabilidade em pesquisa. Editora Científica; 2023. p. 258-71.

-
- [2] Berckmans D. Precision livestock farming technologies for welfare management in intensive livestock systems. *Rev Sci Tech*. 2014;33(1):189-207.
- [3] Goap A, Sharma D, Shukla AK, Krishna CR. An IoT based smart irrigation management system using Machine learning and open source technologies. *Comput Electron Agric*. 2018;155:41-9.
- [4] Krishnamoorthy R, Thiagarajan R, Padmapriya S, Mohan I, Arun S, Dineshkumar T. Applications of machine learning and deep learning in smart agriculture. In: *Machine learning algorithms for signal and image processing*. IEEE; 2022. p. 371-95.
- [5] LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436-44.
- [6] Ünal Z. Smart farming becomes even smarter with deep learning—A bibliographical analysis. *IEEE Access*. 2020;8:105587-609.
- [7] Lin J, Zhu L, Chen WM, Wang WC, Han S. Tiny machine learning: progress and futures. *IEEE Circuits Syst Mag*. 2023;23(3):8-34.
- [8] Banbury CR, Reddi VJ, Lam M, Fu W, Fazel A, Holleman J, et al. Benchmarking TinyML systems: challenges and direction. *arXiv [Preprint]*. 2021; arXiv:2003.04821. Available from: <https://arxiv.org/abs/2003.04821>
- [9] Zennaro M, Nduati R, Masera M, Nkenyereye L. Machine learning on low-power embedded devices for the African context: a review. *IEEE Access*. 2022;10:21450-62.
- [10] David R, Duke J, Jain A, Reddi VJ, Jeffries N, Li J, et al. TensorFlow Lite Micro: embedded machine learning for TinyML systems. *Proc Mach Learn Syst*. 2021;3:800-11.
- [11] Ramírez-Hernández A, Uchoa F, de Azevedo Serpa MC, Binder LC, Rodrigues AC, Szabó MPJ, et al. Clinical and serological evaluation of capybaras (*Hydrochoerus hydrochaeris*) successively exposed to an *Amblyomma sculptum*-derived strain of *Rickettsia rickettsii*. *Sci Rep*. 2020;10(1):924.
- [12] Lessa DAM, Moreira-Soto A, Ramos DGS, Krawczak FS, Pacheco TA, Martins MM, et al. Brazilian Spotted Fever: A re-emerging public health threat. *Front Public Health*. 2021;9:638000.
- [13] Labruna MB. Ecology of *Rickettsia* in South America. *Ann N Y Acad Sci*. 2009;1166:156-66.
- [14] SINAN/SVS/MS. Sistema de Informação de Agravos de Notificação: Febre Maculosa Brasileira. Ministério da Saúde do Brasil; 2023 [cited 2026 Apr 14]. Available from: <https://portalsinan.saude.gov.br>
- [15] Torralba A, Efros AA. Unbiased look at dataset bias. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit*. 2011:1521-8.
- [16] Hymel S, Banbury CR, Situnayake D, Elium A, Ward C, Kelcey M, et al. Edge Impulse: an MLOps platform for tiny machine learning. *arXiv [Preprint]*. 2022; arXiv:2212.03332. Available from: <https://arxiv.org/abs/2212.03332>
- [17] Raschka S. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv [Preprint]*. 2020; arXiv:1811.12808. Available from: <https://arxiv.org/abs/1811.12808>
- [18] Shahinfar S, Meek P, Falzon G. “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. *Ecol Inform*. 2020;57:101085.
- [19] Yadav S, Bhole GP. Handling imbalanced dataset classification in machine learning. 2020 IEEE Pune Sect Int Conf (PuneCon). 2020:1-6.
- [20] Kingma DP, Ba J. Adam: A method for stochastic optimization. *arXiv [Preprint]*. 2015; arXiv:1412.6980. Available from: <https://arxiv.org/abs/1412.6980>
- [21] Chowdhery A, Warden P, Shlens J, Howard A, Rhodes R. Visual Wake Words Dataset. *arXiv [Preprint]*. 2019; arXiv:1906.05721. Available from: <https://arxiv.org/abs/1906.05721>
- [22] Espressif Systems. ESP32-CAM: Technical Reference Manual. Espressif Systems Co., Ltd.; 2023.