TinyML Challenge 2023: Scalable and High-Performance TinyML Solutions for Wildlife Monitoring

Final Report from Team AI4D-Lab Anglophone Africa

Fatma Issa[1], Jabhera Matogoro[2], Zephania Reuben[3], Ramadhani Massawe[4], Rogers Kalunde[5], Paul Mkai[6], Madaraka Marco Masasi[7], Ipyana Issah Mwaisekwa[8]

AI4D-Lab Anglophone Africa, University of Dodoma, P.O.Box 41218, Iyumbu, Dodoma-Tanzania
{Fatma.issa, japhera.matogoro, zephania.reuben, ramadhani.massawe, rogers.kalunde, paul.mkai, madaraka.marco, ipyana.mwaisekwa}@ai4dlab.ac.tz

Abstract:

The integration of artificial intelligence and machine learning in resource-constrained devices, known as TinyML, holds the potential to revolutionize wildlife monitoring and conservation. In this journal article, we explore the application of TinyML in the context of sustainable wildlife management. We investigate how TinyML technologies, optimized for low-cost and low-power operation, are transforming the way we monitor and protect our planet's invaluable biodiversity. Using open-sourced images, both edge impulse platform and Python programs have been employed to develop TinyML models for the detection and classification of wild animals.

The model's performance was evaluated before and after deployment to a microcontroller. In the pre-deployment phase, accuracy, speed, and overall metrics were assessed in a simulated environment. Microcontrollers presented challenges, such as limited resources and memory constraints, which required fine-tuning following deployment in order to achieve optimal performance.

The contribution of this work is a successful tool that can be used to detect and classify wild animals at different environment and habitats in resources limited areas. The tool will assist in conservation of wild life and it can be helpful in evaluating the impact of environmental changes towards wildlife.

1. Introduction

The rapid advancement of Artificial Intelligence (AI) and Machine Learning (ML) in recent years has opened doors to innovative applications across a myriad of fields. One domain that is witnessing a remarkable

transformation is the integration of these technologies with resource-constrained devices, a concept known as Tiny Machine Learning (TinyML) [1][2]. TinyML empowers these devices ranging from microcontrollers and edge computing devices to Internet of Things (IoT) sensors with the capacity to execute machine learning algorithms [3]–[5]. It is within this fascinating intersection of technology and conservation that TinyML finds its profound application, particularly in the context of wildlife monitoring [6], [7]. Tiny Machine Learning, is an innovation that allows machine learning models to operate on devices with constrained computational resources. These resource-constrained devices include microcontrollers, edge devices, and IoT sensors[8]–[10]. One of the key promises of TinyML is to enable these devices to make intelligent decisions and process data without the need for cloud-based or data center processing. In the context of wildlife monitoring, this technology promises to provide real-time insights and data-driven conservation decisions in remote and resource-limited environments.

Challenges in Wildlife Monitoring

Wildlife conservation has always presented a myriad of challenges, and the accurate and efficient monitoring of biodiversity remains a critical concern. Traditional methods, often marked by high costs and energy consumption, are gradually being augmented and, in some cases, supplanted by the capabilities of TinyML[11], [12]. This article explores the immense potential of TinyML in revolutionizing the way we monitor and manage Earth's invaluable wildlife resources.

Wildlife monitoring has always been a pivotal element of conservation efforts. Understanding animal behavior, species distribution, and population dynamics is essential for informed decision-making [12]. However, conventional monitoring methods have had their limitations, ranging from prohibitive costs to environmental impacts and the practical challenges of maintaining continuous human presence in remote, often harsh, ecosystems.

TinyML Solutions for Wildlife Conservation

One avenue of innovation in wildlife monitoring involves the use of camera trap datasets containing images captured in natural habitats. These images showcase diverse wildlife species and their activities. The intersection of TinyML and wildlife monitoring offers concrete solutions to longstanding conservation challenges [13]. By leveraging TinyML models tailored to the specific needs of wildlife monitoring, ensuring

that these models are lightweight, efficient, and capable of running on devices with minimal power and processing capacity [5], [12], [14]. This enables real-time monitoring in remote areas while minimizing financial and environmental costs.

Generally, this paper explores the use of TinyML, with a particular focus on wildlife monitoring. TinyML delves to revolutionize conservation practices, offering low-cost, low-power models capable of monitoring and safeguarding our planet's precious biodiversity. The subsequent sections will provide a detailed methodology, results, discussion, and conclusion, shedding further light on the significance of TinyML in sustainable wildlife management.

2. Architecture and Design

In this section we describe all the steps of our proposed solution including prototype designing, data preprocessing, model development, training and deployment. Also, analysis of pretrained models will be explained based on the performances during the model development phase.

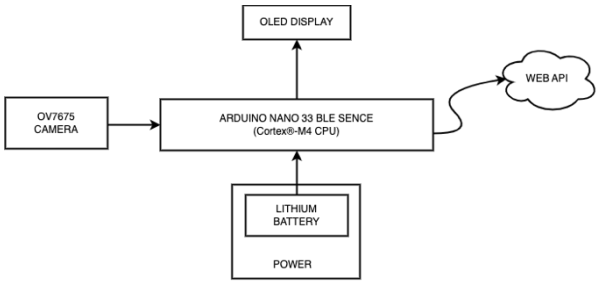

Figure 1: Architecture Design - Block diagram

A. Prototype

Our prototype consists of a TinyML Microcontroller Arduino Nano 33 BLE Sense Lite, OV7675 camera, Grove OLED display and Arduino TinyML shield all powered by a Lithium battery. The deployed model in a microcontroller was classifying images from the camera in real time and disply the inference results on the OLED and Web Platform.

i. Arduino Nano BLE Sense Lite

The Arduino Nano 33 BLE Sense is built upon the nRF52840 microcontroller and runs on Arm® Mbed™ OS. The Arduino Nano 33 BLE Sense combines a tiny form factor, different environment sensors and the possibility to run AI using TinyML and TensorFlow™ Lite [15][4].

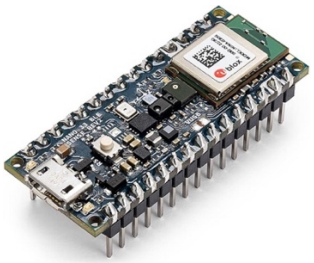

Figure 2: Arduino Nano 33 BLE Sense

ii. OV7675 Camera

The Arducam OV7675 camera provides support for resolutions of up to 640x480 pixels, ensuring the capture of clear and sharp images. Its versatile range of features enhances the functionality of TinyML projects. Its user-friendly interface, superior image quality, it is an ideal choice for various applications where visual data is crucial. It uses power 1.5V ~ 3.0V, and resolution and frame of 640x480/320x240/160x120@15fps [4].

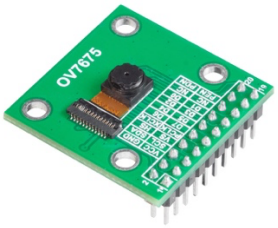

Figure 3: OV7675 Camera

iii. Organic Light Emission Diode (OLED) Display

Monochrome OLED display with 1.12" diagonal and 128 x 128 px resolution. It has deprived of 16 gray scale adjustment to increase the maximum speed of the I2C bus from 100 KHzto200 KHz. The screen based on the SH1107G controller works with voltages of 3.3 V and 5 V, communicates via the I2C bus.

Figure 4: Grove OLED Display

iv. Arduino TinyML Shield

It is a custom Arduino shield to make it easy to attach your components together and enhance TinyML projects [15].

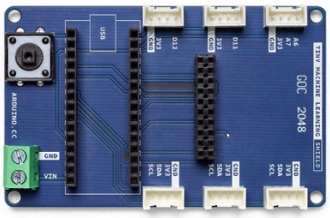

Figure 5: Arduino TinyML shield

v. Battery

The Arduino Nano 33 BLE typically utilizes a Lithium Polymer (LiPo) battery for power. LiPo batteries are rechargeable, lightweight, and have a high energy density. When selecting a battery, it's important to consider

compatibility with the Arduino Nano 33 BLE's operating voltage of 3.3V. The battery's capacity determines the duration of operation on a single charge. This enhances the tinyML models that consume minimal power while maintaining high performance.

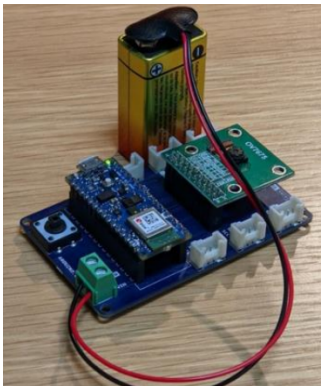

Figure 6: Lithium Battery

B. Data Preprocessing

Starting Data were provided by the ITU through the links provided at the Challenge website page beginning of the challenge. Before model development, the collected images (Datasets) underwent a rigorous preprocessing stage. Edge Impulse's tools, MakeSense-Ai [16] facilitated this crucial step. Feature extraction and engineering techniques were applied to identify relevant features, optimizing the model's ability to classify species. The following preprocessing tasks were performed

i. Image Resizing

Images were resized to a standardized resolution to ensure consistency across the dataset. This resizing not only reduced data size but also prepared images for efficient processing on the resource-constrained Arduino Nano 33 BLE Sense.

ii. Data Augmentation

To enhance model robustness, data augmentation techniques were applied. These included random rotations, flips, and brightness adjustments. Augmented data expanded the dataset and improved the model's ability to generalize to different environmental conditions and species variations.

iii. Labeling

Images were labeled according to the 10 wildlife species they contained on the model trained using edge impulse. MakeSense-Ai [16] was used to label images and Edge Impulse's labeling tools facilitated this process, ensuring that each image was associated with the correct species category because the accurate labeling is critical for training a supervised machine learning model.

C. Model Development

This phase is a multifaceted process, marked by a deliberate and meticulous approach to building models that are both accurate and resource-efficient. Feature extraction from

preprocessed wildlife images is the cornerstone of model development, as it involves capturing relevant information such as color histograms, texture descriptors, and object recognition features, enabling the model to differentiate between wildlife species. During model training different algorithms and pretrained models were used and optimized for resource efficiency through meticulous hyperparameter tuning, ensuring that the models are well-suited for deployment on devices like the Arduino Nano 33 BLE Sense.

D. Model Training

The Artificial Intelligence (AI) models for this study were trained using Google Colab and Edge Impulse. Tensor Flow Lite library was used to train mode in Google Colab environment. Convolutional neural network (CNN) architecture was chosen to meet the resource constraints of edge devices. To achieve high performance on image classification and recognition with a relatively modest computation cost inception v3 was used [17], [18]. Hyperparameters were fine-tuned, and transfer learning techniques were employed to ensure efficient model training.

Edge Impulse is a platform that simplifies the process of developing machine learning models for small devices, known as TinyML [2], [19]. It provides a GUI-based pipeline for running data through different stages of processing: feature engineering, model building, testing and deployment. The platform also has libraries of pre-trained models and, which can be used to speed up and simplify the development process. Edge Impulse provides tools for data preprocessing, such as noise reduction and feature extraction, making it easier to work with raw sensor data. You can then choose from a variety of pre-built neural network architectures or create custom ones using the platform's visual interface [19]. Edge Impulse supports automatic hyperparameter tuning and data augmentation, streamlining the training process. Once your model is trained, it can be optimized to run efficiently on microcontrollers or edge devices, taking into account the limited computational resources. The platform also offers libraries and deployment options for various hardware platforms, enabling you to easily integrate your TinyML model into your target application.

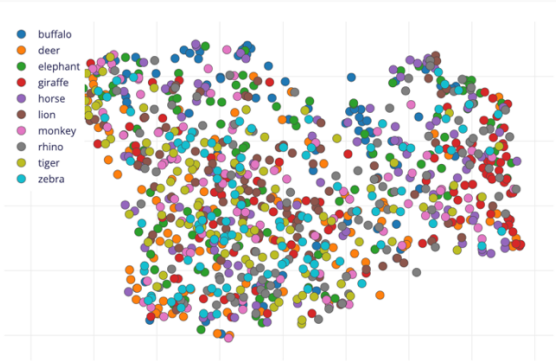

Figure 7: Feature generation

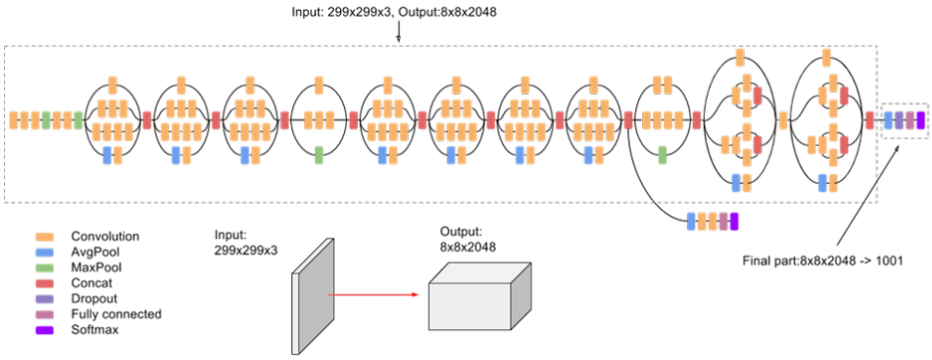

Figure 8: The architecture of CNN (Inception3).

A. Data Validation and Performance Assessment

On this phase, data validation and performance assessment hold a pivotal role. To ensure that the models are not only accurate but also reliable, a separate set of data known as validation data was used. This data, distinct from the training dataset, allows the model to be rigorously tested on previously unseen scenarios, mimicking real-world conditions. Several performance metrics, including accuracy, precision, recall, and the F1 score, were employed to quantitatively evaluate the model's capabilities. The Accuracy measures overall correctness, precision assesses accurate positive identifications, recall evaluates the model's ability to identify all instances of a particular species, and the F1 score balances precision and recall. These metrics and the model's consistently high performance across these metrics offer a comprehensive view of the model's performance and its suitability for informed choices and decision-making.

B. Model Deployment

The models were converted and optimized using TensorFlow Lite Micro, enabling seamless deployment on microcontrollers that supports TinyML. The models were exported in form of Tensor Flow (.tf) file from Google Colab, as Arduino Library, and Firmware from Edge Impulse. The models were deployed in Arduino Nano BLE 33 Sense using Arduino IDE Software. The Microcontroller was connected to computer using Universal Serial Bus (USB). The device operated continuously, capturing and analyzing images. When an animal was detected and identified, the inferences were shown on the Arduino IDE Serial Monitor and Terminal or Command Prompt (CMD) in real time as shown in the figure 9.

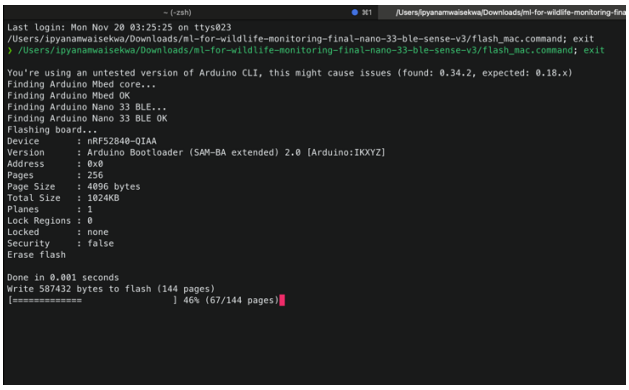

Figure 9: Model deployment using terminal by firmware flash file.

C. Real-Time Monitoring

The Arduino Nano 33 BLE Sense, now equipped with the trained TinyML model, was used in detection of animal's species for real-time monitoring. The device operated continuously, capturing and classifying images as in figure 10. When an animal was detected and identified, the device transmitted this information wirelessly to Edge Impulse web API using the terminal and display inferences on the prototype's OLED display.

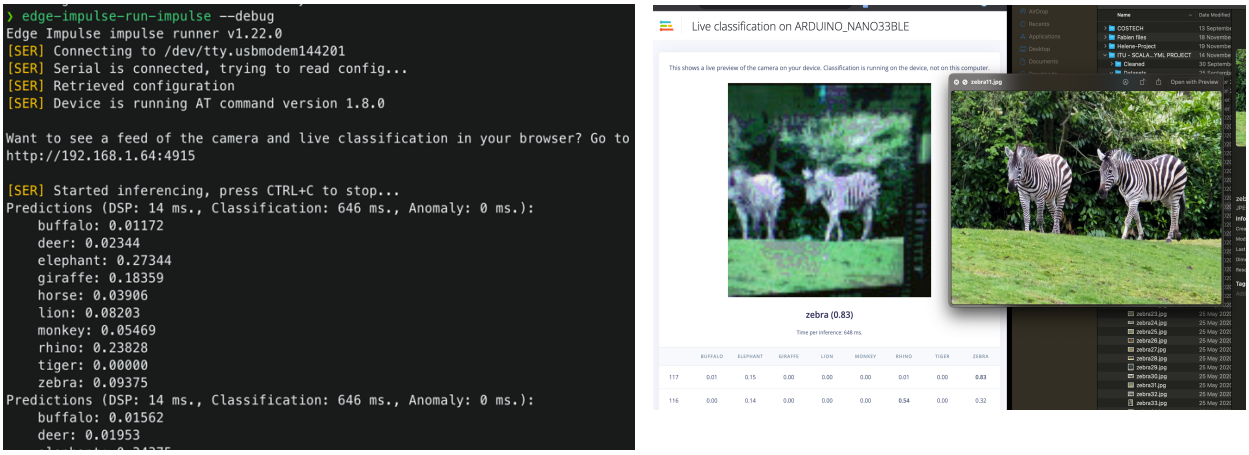

Figure 10: Model running using the terminal and web API.

3. Results and Discussion

The section provides an overview of the outcomes obtained from each Machine Learning technique employed in this study. The process of model development involved two distinct approaches: the first approach involved the building of a Python-based model on Google Colab, while the second approach involved utilizing Edge Impulse. Two distinct datasets were employed in these methodologies, yielding disparate outcomes. The average performance outcomes of the Colab Notebook are provided and may be accessed at [20]. Similarly, the average performance results obtained by Edge Impulse can be found at [21].

A. Transfer Learning on Notebook

The model underwent training using a dataset consisting of 10 classes that are prevalent across the African continent. The list comprises several species of African wildlife, including the buffalo, eland, elephant, giraffe, hyaena brown, impala, leopard, lion, rhino, and zebra shown in figure 11. The training dataset comprises 56,446 photos, whereas the testing dataset contains 14,106 images. The model constructed inside this framework was configured with various parameters and underwent multiple training iterations, ultimately achieving an F1 score of 79% accuracy. The conversion and profiling of the model from an unquantized (Float 32) representation to an Optimized (int8) representation were performed using Tensorflow Lite. This was done with the purpose of accommodating the model within a micro-controller, as depicted in figure 12.

```
[ ]
    # Define the class names
    class_names = ['buffalo','eland', 'elephant','giraffe','hyaenabrown','impala','leopard','lion','rhino', 'zebra']
```

Figure 11: Animal species classes

```
[ ]  # Evaluate the model
     valid_data.reset()
     y_pred = model.predict(valid_data, steps=np.ceil(valid_data.samples / 64))
     y_true = valid_data.classes

     f1 = f1_score(y_true, np.argmax(y_pred, axis=1), average='weighted')
     print(f"F1 Score : {f1}")

     221/221 [==============================] - 704s 3s/step
     F1 Score : 0.7996056498384594

     print("Classification Report:")
```

Figure 12: Model training results

Furthermore, despite the favourable outcomes of the training, the dimensions of the model exceeded the RAM capacity that a microcontroller could accommodate. The Arduino Nano 33 BLE Sense is equipped with a maximum random-access memory (RAM) capacity of 256 kilobytes (Kb), which represents its upper limit for storing and accessing data during operation. The deployed Model exhibited success; nevertheless, it failed to generate inferences due to insufficient

scratch space for the Neural Network's operations.

Classes	*precision*	*recall*	*f1-score*	*support*
0	0.84	0.23	0.36	647
1	0.97	0.36	0.52	718
2	0.87	0.53	0.66	1485
3	0.97	0.79	0.87	1124
4	0.86	0.56	0.68	599
5	0.93	0.91	0.92	5903
6	0.13	0.97	0.24	230
7	0.60	0.52	0.55	196
8	0.43	0.82	0.56	814
9	0.95	0.92	0.94	2390
Accuracy			0.78	14106
Macro avg	0.76	0.66	0.63	14106
Weighted avg	0.88	0.78	0.80	14106

Table 1: Performance of the model based on Accuracy, Precision, Recall, and F1 Score

B. Transfer learning on Edge Impulse

This approach involved the utilization of the Edge Impulse framework. The outcomes derived from these methodologies exhibit variations in terms of precision and performance measures, owing to the disparities in the datasets employed and the neural networks utilized for training the models. The model used in Edge Impulse consists of ten (10) distinct species, containing a total of 2500 samples.

Different Neural Network settings such as Training Cycle, Learning Rate, Batch size, Argumentation with various pre-built models, namely FOMO (Faster Objects, More Objects) MobileNetV2 0.35, FOMO MobileNetV2 0.1,

MobileNetV2 SSD FPN-Lite, and YOLOv5. This study demonstrates disparities in accuracy between MobileNetV1 models and MobileNetV2 models, specifically in relation to Training Accuracy when comparing optimized and unquantized models. Consequently, these discrepancies also extend to the accuracies seen during deployment. The MobileNetV2 model demonstrated superior performance in both training and testing stages. However, a notable issue arose about the latency and RAM requirements for on-device performance after deploying the model. These requirements exceeded the capacity of the microcontroller, as depicted in Figure 14 and 15.

Figure 14: Training performance using MobileNetV2 model

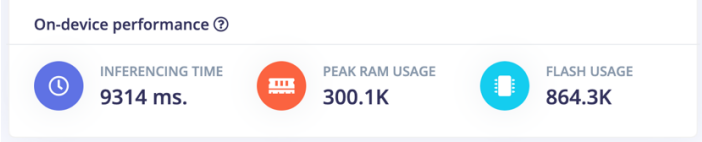

Figure 15: Expected on-device performance after Model deployment

Figure 16: Expected On-device performance using MobileNetV2 model

Thereafter deployment of MobileNetV2 models on the microcontrollers, the device failed to start inferences with request of working on the memory for a Neural Network to run smoothly as shown in figure 17.

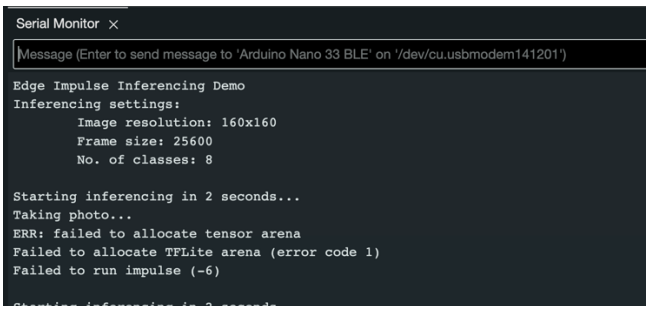

Figure 17: Failed to allocate TFLite arena

In addition to MobileNetV2, the MobileNetV1 model exhibited strong performance during both the training and testing phases. The deployment of the model after fine-tuning it using EON Tuner yielded good outcomes in terms of latency and RAM needs, as illustrated in Figures 18 and 19. The Model was effectively implemented on a microcontroller to perform real-time inferences, exhibiting high accuracy as demonstrated in Figures 20 and 21.

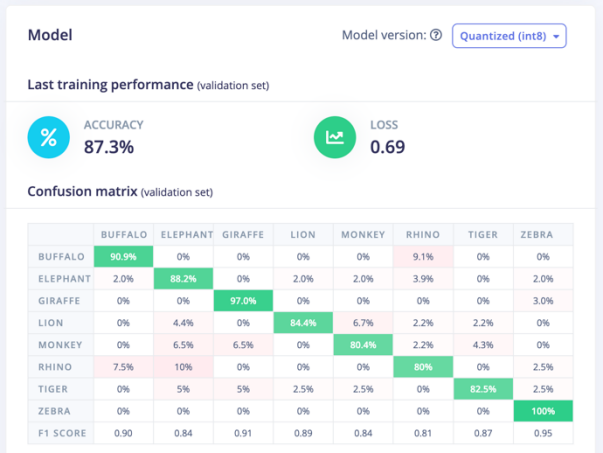

Figure 18: MobileNetV1 training performance results

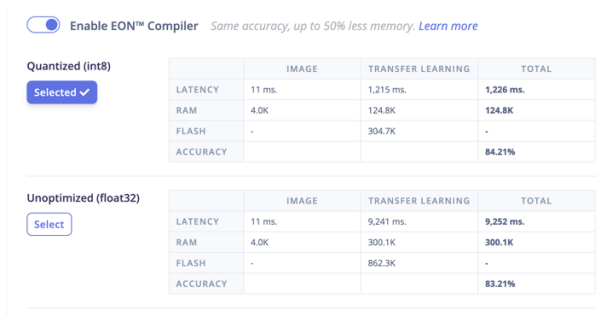

Figure 20: Model optimization for On-device performance.

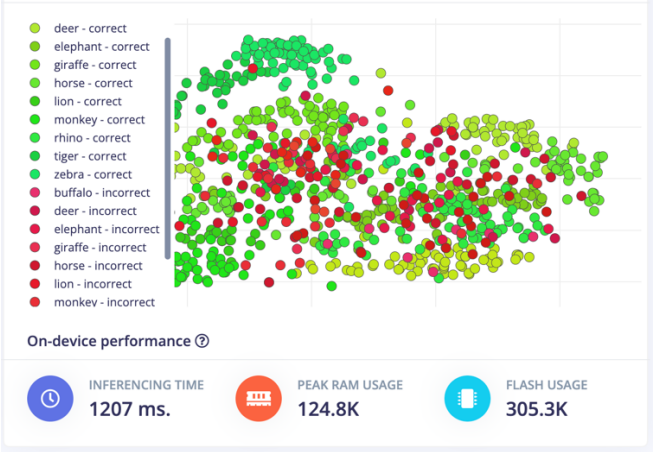

Figure 19: Features exploration and expected on-device performance results using MobileNetV1

The findings indicate that MobileNetV1 demonstrates favorable outcomes when implemented for doing inferences on the device. The model exhibits a high level of accuracy in accurately classifying animal species, as depicted in Figure #. Despite the increased accuracies, MobilNetV2 has faced issues in terms of adaptation. In contrast, MobileNetV1 has demonstrated good performance in both training and testing. Furthermore, even after the deployment of the model, the inferences remain accurate.

Model	Latency (ms)	RAM (Kb)	ROM (Kb)
MobileNetV1	1207	124.8	305.3
MobileNetV2	5615	721.6	660.6
MobileNetV2	9314	300.1	864.3

Table 2: Comparison of resource use

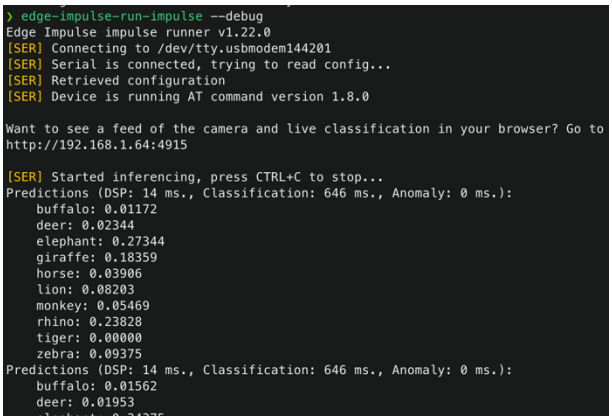

Figure 21: Inferences from a device displayed on the terminal

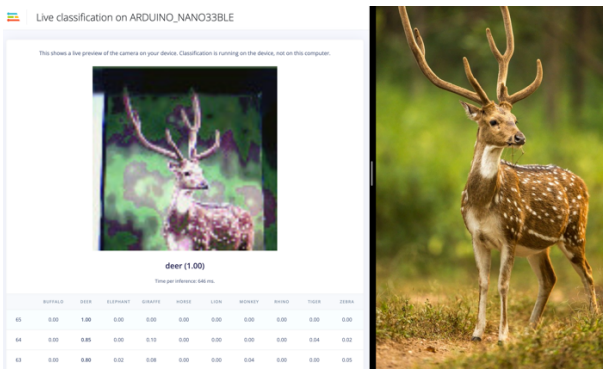

Figure 22: Live classification on the web API

4. Future works and Conclusion

The results obtained through the Edge Impulse training approach, particularly utilizing the MobileNetV1 model, showcase promising strides in the development of Tiny Machine Learning (TinyML) models for wildlife monitoring. Achieving an accuracy of 87.3% F1 Score underscores the effectiveness of this approach in accurately classifying animal species. Further optimization using the EON Tuner for the Arduino Nano 33 BLE Sense demonstrated remarkable efficiency, with an accuracy of 84.4%, peak RAM of 124 KB, ROM of 305 KB, and latency of 1226 ms. These metrics affirm the viability of deploying such models on resource-constrained embedded devices, showcasing the potential for low-cost, low-power, and reliable solutions in the field of wildlife conservation.

Looking ahead, our future endeavors aim to advance the capabilities of TinyML models by exploring Real-Time Operating Systems (RTOS) for resource-constrained embedded devices. The objective is to develop solutions (Embedded devices) that are not only low-cost and low-power but also scalable and adaptable, ensuring seamless integration into diverse ecosystems. The envisioned deployment of these solutions in the field holds the promise of revolutionizing wildlife monitoring and conservation efforts. By providing researchers and conservationists with effective tools for monitoring animal behavior and biodiversity, we aspire to contribute significantly to the preservation of our planet's precious resources.

In conclusion, this work stands as a dedication to the collective efforts of researchers and conservationists who tirelessly work towards enhancing our understanding of wildlife and championing sustainable practices. The integration of TinyML is indeed at the forefront of a technological revolution in

conservation, playing a crucial role in fostering sustainable practices and strengthening global initiatives aimed at protecting and preserving our biodiversity. As we embark on the path toward deploying these solutions, we anticipate a future where technology becomes an invaluable ally in safeguarding the delicate balance of our natural world.

5. References

[1] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the Extreme Edge: A Survey on Reformable TinyML," *ACM Comput. Surv.*, vol. 55, no. 13s, Jul. 2023, doi: 10.1145/3583683.

[2] P. Warden and D. Situnayake, "TinyML," https://www.oreilly.com/library/view/tinyml/9781492052036/ch04.html.

[3] M. Rovai Professor and U. -Brazil Shawn Himel, "SciTinyML Scientific use of machine learning on low power devices Regional Workshop-Africa Introduction to Edge Impulse."

[4] "SciTinyML Scientific Use of Machine Learning on Low Power Devices TinyML Kit Overview-HW and SW Installation & Test," 2022.

[5] T. Delsart, "'An ultra-low-power embedded convolutional neural network for acoustic monitoring of forest ecosystems.'" [Online]. Available: http://hdl.handle.net/2078.1/thesis:30718

[6] S. O. Ooko, M. Muyonga Ogore, J. Nsenga, and M. Zennaro, "TinyML in Africa: Opportunities and Challenges," in *2021 IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6. doi: 10.1109/GCWkshps52748.2021.9682107.

[7] H. R. Sabbella, A. R. Nair, V. Gumme, S. S. Yadav, S. Chakrabartty, and C. S. Thakur, "An Always-On tinyML Acoustic Classifier for Ecological Applications," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 2393–2396. doi: 10.1109/ISCAS48785.2022.9937827.

[8] N. S. Yamanoor, S. Yamanoor, and S. Thyagara, "Low-Cost, Open, Citizen Science with IoT and TinyML," in *2023 Intermountain Engineering, Technology and Computing (IETC)*, 2023, pp. 300–304. doi: 10.1109/IETC57902.2023.10152224.

[9] T. Delsart, "'An ultra-low-power embedded convolutional neural network for acoustic monitoring of forest ecosystems.'" [Online]. Available: http://hdl.handle.net/2078.1/thesis:30718

[10] A. P. Van Der Burgt, M. S. Thesis, D. J. W. Kamminga, and E. Molenkamp, "Robust evaluation and optimized fine-tuning of machine learning algorithms deployed on the edge," 2023.

[11] N. Van Dijk, "Advanced wildlife camera trapping using embedded AI machine vision," 2023.

[12] R. Gotthard and M. Broström, "Edge Machine Learning for Wildlife

Conservation-A part of the Ngulia Project Maskininlärning i Noden för Bevarandet av Djurlivet på Savannen." [Online]. Available: www.liu.se

[13] N. Tekin, A. Aris, A. Acar, S. Uluagac, and V. C. Gungor, "A review of on-device machine learning for IoT: An energy perspective," *Ad Hoc Networks*, vol. 153, p. 103348, 2024, doi: https://doi.org/10.1016/j.adhoc.2023.103348.

[14] E. K. Ronoh, S. Mirau, and M. A. Dida, "Human-Wildlife Conflict Early Warning System Using the Internet of Things and Short Message Service," 2022. [Online]. Available: www.etasr.com

[15] Arduino, "Arduino TinyML Kit," https://store.arduino.cc/products/arduino-tiny-machine-learning-kit.

[16] P. Skalski, "Make Sense Ai," https://www.makesense.ai/.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, "Rethinking the Inception Architecture for Computer Vision."

[18] "1-s2.0-S1877050919318587-main".

[19] Edge Impulse, "Get started with Edge Impulse," https://docs.edgeimpulse.com/docs/.

[20] AI4D Lab, "Project's Model Github Repository," https://drive.google.com/file/d/1CHIWiGsqp1XFhYkLwwiwlNbHbu64yGtS/view?usp=share_link. Accessed: Oct. 20, 2023. [Online]. Available: https://drive.google.com/file/d/1CHIWiGsqp1XFhYkLwwiwlNbHbu64yGtS/view?usp=share_link

[21] AI4D-Lab Impulse Repository, "Edge Impulse Studio Project Model Repository," https://studio.edgeimpulse.com/public/311894/latest.