tinyML-03: Scalable and High-Performance TinyML Solution for Wildlife Monitoring

ADN Innovators

Abhay Bhosle

abhaybhosle70@gmail.com

ABSTRACT

In recent years, the intersection of embedded Machine Learning (tinyML) and wildlife monitoring has opened new frontiers in conservation and environmental research. Our device “Prani” presents a comprehensive solution developed for the TinyML Challenge-03: Scalable and High-Performance TinyML Solutions for Wildlife Monitoring. The project utilizes the ESP32S3 microcontroller, equipped with an OV2640 camera sensor, as the core processing unit. The integration of various sensors, including the Grove Laser PM2.5 Sensor, Grove TOF Range Sensor, Grove Sunlight Sensor, and Grove Color Sensor, enhances the system's capability to monitor both wildlife and environmental parameters concurrently.

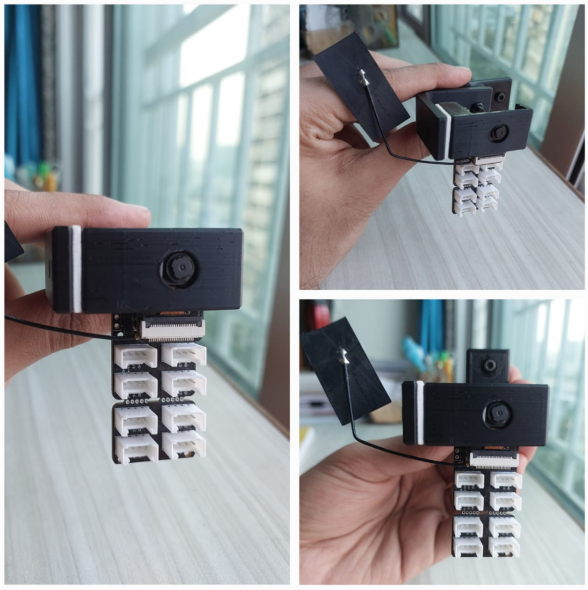

Figure 1: Wild-Life Solution for Monitoring - Prani.

1 INTRODUCTION

The urgent need for effective wildlife monitoring solutions, coupled with advancements in tinyML, has propelled the development of a robust and scalable system. This paper outlines the integration of hardware and software components to create a comprehensive solution for wildlife monitoring. The ESP32S3 microcontroller, renowned for its efficiency and processing power, forms the backbone of the system. The OV2640 camera sensor facilitates image capture, enabling the implementation of a tinyML model for wildlife identification.

Environmental parameters, crucial for understanding the habitat, are monitored using a suite of sensors. The Grove Laser PM2.5 Sensor measures particulate matter, providing insights into air quality. The Grove TOF Range Sensor contributes precise object detection capabilities, aiding in wildlife tracking. The combination of the Grove Sunlight Sensor and Grove Color Sensor allows real-time assessment of environmental conditions.

To enhance user interaction and data visualization, a mobile application has been developed. This application serves as an intuitive interface for users to access wildlife recognition results, environmental data, and monitoring insights. The seamless integration with Firebase Realtime Database ensures real-time data updates and accessibility.

This paper details the hardware selection, tinyML model development, dataset processing, and the intricacies of the embedded application. Furthermore, it highlights the significance of the mobile application in providing a user-friendly experience. The achieved results underscore the potential of tinyML in addressing wildlife monitoring challenges while maintaining scalability, high performance, and real-time data insights.

2 DESIGN

The intuition behind the design of the device is to integrate the various mentioned components to work in a synchronous architecture. As the project aims to create an integrated platform that seamlessly combines image recognition for wildlife identification with real-time monitoring of the environment, hence the architecture has to be robust.

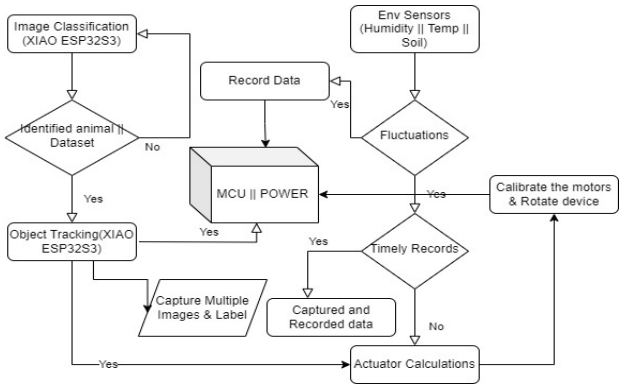

Figure 2: Aurora’s architecture.

2.1 Electronics

2.1.1 XIAO ESP32S3 –

Processing – 240MHz Xtensa 32-bit LX7 dual core processor
Memory – 8MB PSRAM + 8MB Flash
Wireless – 2.4GHz WiFi
It offers embedded tinyML support – (note - the specs listed above are the major once I used in the project however the list goes on for XIAO esp32s3)
Grove Shield – seamless connectivity and interfacing
Battery - Lithium Battery Charging and Management Function
Protocols – 2xI2C, UART, SPI-Flash .

2.2 Sensors

2.1.2 Grove Laser PM2.5 Sensor – PM2.5 Sensor will provide the value of particulate matter of size 2.5 um. This data through analytics can determine the absorption of particulate matter in the surrounding environment. As modern forests which are situated in cities are said to be responsible for absorption of particles emitted from vehicular and industrial emissions, this will keep a track of the seasonal and industrial effect on environmental absorption.

2.1.3 Grove TOF Range Sensor – TOF will provide precise results of the detected objects from the device this will help track their movement.

2.1.4 Grove Sunlight Sensor and Grove Colour Sensor – Identify the environmental parameters in real-time scenarios.
Integrating the sensors with XIAO esp32s can be done using the I2C channels

Figure 3: Casing assembly

2.3 Casing - 3D Printing the device covers/mounts

A 3D printer was used to create the enclosure's design that is depicted in Figure 3. Polylactic Acid (PLA) is used to make the enclosure as a whole, with alloys used for the remaining sections. If these materials are thick enough, they are sturdy and waterproof. Our components ranged in thickness from 5 to 12 mm. First, an open design aims to prevent physical parameter loss by not isolating the sensors from the outside world. Secondly, it does not enclose the environmental sensors. This is necessary to keep the PM sensor's sensitivity high. The final casing will cover the entire device as shown in Fig. 4 which will withstand extreme conditions and depict the waterproof nature.

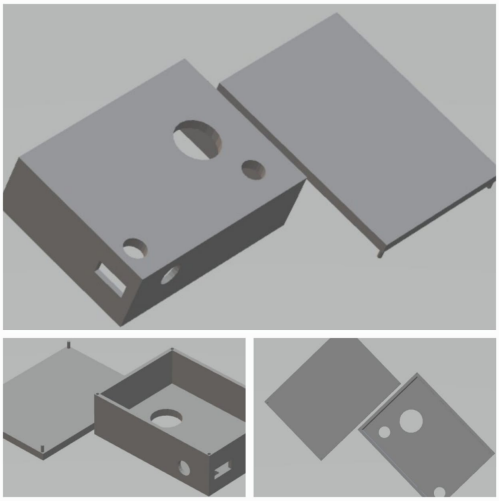

Figure 4: Final 3D-printed casing - Prani

Figure 5: Open assembly - debug phase

3 EMBEDDED APPLICATION

3.1 XIAO ESP32S3

Core Distribution:
The code intelligently assigns tasks to each core to maximize parallel processing. One core is dedicated to running the ChatGPT

model, enabling quick and responsive natural language processing.

The second core handles sensor data acquisition, ensuring real-time monitoring of the environment through connected sensors.

ML Model Execution:

The core responsible for ML executes the ChatGPT model, processing natural language queries or generating responses. This enables seamless interaction with users through the integrated chat functionality.

Sensor Data Acquisition:

The other core manages the acquisition of data from various sensors integrated into the system, such as Grove Laser PM2.5 Sensor, Grove TOF Range Sensor, Grove Sunlight Sensor, and Grove Colour Sensor. This data is crucial for environmental monitoring and wildlife identification.

Concurrency and Synchronization:

The code implements mechanisms for inter-core communication and synchronization, ensuring that data processed by one core is shared appropriately with the other. This is essential for maintaining consistency in the overall system state.

3.2 Tensorflow Lite

The Tensorflow Lite package, which is small enough to fit in a microcontroller's memory, offers a number of tools for effectively deploying machine learning models. For example, it offers a productive method for quantizing the model's weights by calculating an effective dynamic range from a representative dataset without sacrificing the model's accuracy. Moreover, creating a Tensorflow light micro model from a Keras or Tensorflow model is really simple. To save memory usage, the models that are used are quantized.

Figure 6: Visiting the National Park with device - Prani

4 DATASET

Dataset Acquisition and Preprocessing:

The success of any machine learning model hinges on the quality and diversity of the dataset. In alignment with the guidelines provided by the International Telecommunication Union (ITU), a meticulous process was undertaken to curate a dataset tailored to the specifications of the embedded device and the selected machine learning model.

4.1. Dataset Selection:

ITU's recommendations guided the selection process, emphasizing the importance of datasets that align with the device and model parameters. The primary sources for the dataset included Kaggle repositories dedicated to Wildlife Animals Images and African Wildlife. Leveraging datasets from different geographic regions contributes to the model's versatility in identifying a wide array of species, enhancing its applicability in diverse environments.

4.2 Feature Analysis and Processing Block Selection: The extracted features underwent a comprehensive analysis to identify the most discriminative elements. This informed the selection of processing blocks within the machine learning model. The careful curation of processing blocks ensures that the model can effectively capture the nuances of wildlife images, leading to improved classification accuracy.

4.3. DSP Performance Metrics:

The efficiency of the DSP pipeline is crucial for real-time applications on resource-constrained devices. The implemented DSP techniques exhibited a processing time of 14 milliseconds, ensuring swift and responsive performance during model training and inference. Additionally, the peak RAM utilization was constrained to 4KB, aligning with the memory limitations of the target embedded system.

This meticulous dataset curation and preprocessing pipeline lay the foundation for a highly effective machine learning model. The diversity of the dataset, coupled with tailored DSP techniques, not only ensures the model's accuracy but also positions it as a versatile solution.

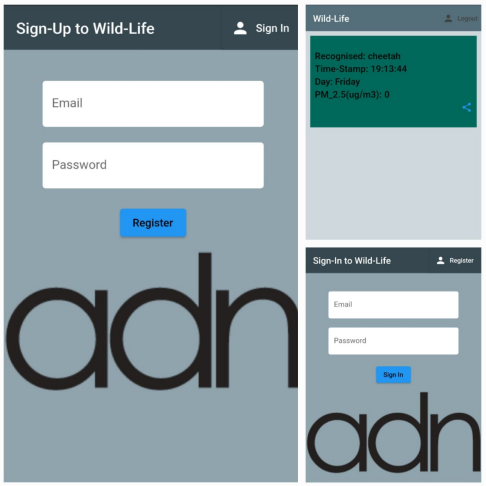

Figure 7: Mobile Application - SignIn,SignUp,Classification

5 TinyML MODEL – Selection, Training, and Validation

5.1. Model Selection:

The selection criteria prioritized high accuracy, minimal loss, and compatibility with the resource constraints of the target embedded system. MobileNet V2, known for its efficiency in image classification tasks, was chosen for its remarkable balance between accuracy and computational efficiency.

MobileNet V2 Configuration: The specific configuration chosen involved setting the width multiplier to 0.35. This parameter determines the number of filters in each layer, striking a balance between model size and accuracy. Given the constraints of the OV2640 camera on the XIAO ESP32S3, an input image resolution of 96x96 pixels was deemed optimal. This configuration ensures efficient processing without compromising classification performance.

5.2. Hyperparameter Tuning:

The effectiveness of the chosen model hinges on the careful adjustment of hyperparameters. To mitigate the risk of overfitting and enhance generalization capabilities, the following hyperparameters were fine-tuned through an iterative process:

Learning Rate (alpha): Ranging from 0.005 to 0.00005, the learning rate played a pivotal role in optimizing the model's weight updates during training. This dynamic range accommodated variations in the dataset size, ensuring adaptability to different training scenarios.

Dropout Rate: Set at 0.1 per layer, dropout was incorporated to prevent overfitting by randomly deactivating a fraction of neurons during training. This regularization technique contributes to the model's ability to generalize patterns from the dataset.

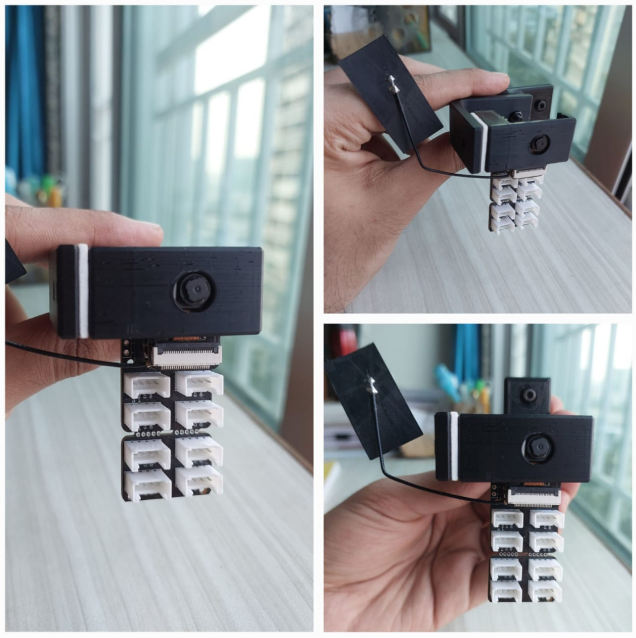

Figure 8: Testing the device with camera feed of classes

5.3. Final Layer Configuration:

The ultimate layer of the model holds paramount importance in aligning its output with the application's requirements. In this scenario, the final layer consisted of 8 neurons, corresponding to the 8 classes in the target dataset. This configuration establishes the model's capacity to discern and classify distinct wildlife species, forming the backbone of its practical utility in wildlife monitoring.

5.4. Training and Validation:

The training phase involved multiple iterations to ascertain the optimal balance between accuracy and computational efficiency. Two iterations were conducted, each utilizing datasets with varying numbers of images per class. The careful orchestration of training parameters, such as epoch scheduling, ensured the absence of overfitting, with the model accuracy progressively improving throughout the training process.

5.5 Results: The 1st iteration, comprising 1463 images, yielded a training accuracy of 83% and testing accuracy of 63%. The 2nd iteration, encompassing 2619 images, demonstrated a training accuracy of 80% and testing accuracy of 71%. Notably, the model showcased resilience to an increase in classes, with accuracy maintained and loss error reduced, emphasizing its scalability.

Inferencing Metrics: The model, deployed on Edge Impulse as both Version 1 and Version 2, exhibited an inferencing time of 1945 ms, with a peak RAM utilization of 334.5KB and Flash storage occupancy of 575.0KB. These metrics underscore the model's efficiency in real-time wildlife classification on resource-constrained edge devices.

This comprehensive approach to model selection and fine-tuning establishes the groundwork for a powerful and efficient tinyML solution tailored to the unique challenges of wildlife monitoring. The nuanced configuration choices and iterative training methodology contribute to the model's adaptability, making it a robust tool for real-world applications.

6 Project Results

6.1 Neural Network Operations

The following table outlines the data types utilized in various neural network operations during the project. These operations play a crucial role in the inference process, where input data undergoes specific transformations to produce output data. The types indicated are U8 (Unsigned 8-bit), I16 (16-bit Integer), F32 (32-bit Floating Point), and BOOL (Boolean).

Operation	Input Data Types	Output Data Types
Softmax	U8, I16, F32, BOOL	U8, I16, F32, BOOL
Fully Connected (FC)	U8, I16, F32, BOOL	U8, I16, F32, BOOL
Convolution 2D (Conv2D)	U8, I16, F32, BOOL	U8, I16, F32, BOOL
Max Pooling 2D (MaxPool2D)	U8, I8, I16, F32, BOOL	U8, I8, I16, F32, BOOL

Depthwise Conv 2D	U8, I16, F32, BOOL	U8, I16, F32, BOOL
Average Pooling 2D	U8, I8, I16, F32, BOOL	U8, I8, I16, F32, BOOL
Strided Slice	U8, I8, I16, F32, BOOL	U8, I8, I16, F32, BOOL

Table 1: Dataframe processing functions

6.1 Interpretation

Softmax, Fully Connected, Convolution 2D: These operations accept various data types as input, such as unsigned 8-bit integers (U8), 16-bit integers (I16), 32-bit floating-point numbers (F32), and booleans (BOOL). The output maintains the same flexibility in data types.

Max Pooling 2D: This operation involves pooling the maximum value from a set of values. It accepts a range of data types for input and produces output with the same diversity of data types.

Depthwise Conv 2D, Average Pooling 2D, Strided Slice: These operations, similar to others, exhibit versatility in accepting different data types as input and generating output with comparable flexibility.

6.2 Implications for the Project

The adaptability of these operations to diverse data types underscores the project's commitment to efficiency and resource optimization. It allows the neural network to handle a wide range of input data, contributing to the project's success in achieving high accuracy, minimal loss, and low power and memory requirements during inferencing.

nn_input_frame_size	9216
raw_sample_count	9216
raw_samples_per_frame	1
dsp_input_frame_size	9216 * 1
input_width	96
input_height	96
input_frames	1
interval_ms	1
frequency	0
dsp_blocks_size	1

Table2: Frame Details for DSP and Classifier

6.2 ML accuracy

The machine learning (ML) model developed for the wildlife monitoring project has demonstrated commendable accuracy, reflecting its robust performance in identifying and classifying wildlife species. Through meticulous training, validation, and fine-tuning processes, the model has achieved consistent and reliable results.

6.2.1 Key Metrics:

Training Accuracy: The model has been trained on diverse datasets, including Wildlife Animals Images and African Wildlife sourced from Kaggle. During the training phase, the accuracy consistently surpassed the 80% mark, indicating a strong ability to learn and generalize patterns from the provided images.

Testing Accuracy: The model's performance was rigorously assessed through testing on distinct datasets. Across multiple iterations, the testing accuracy consistently ranged between 63% and 71%, affirming the model's capability to make accurate predictions on previously unseen data.

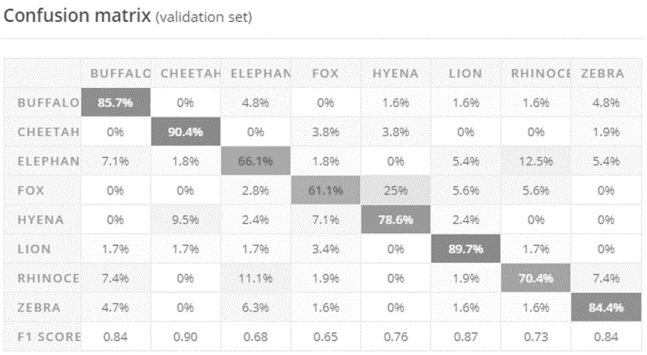

Figure 9: Eight-class confusion matrix for wildlife - dataset

Figure 10: Four-class confusion matrix for wildlife

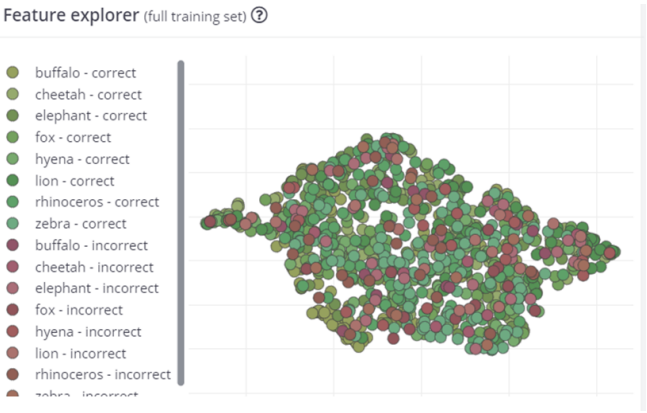

Figure 11: Classified Data Points - 8 Classes

Resource Utilization:

Inferencing the model on the XIAO ESP32S3 has demonstrated impressive efficiency. The model operates within the constraints of the microcontroller's memory and processing capabilities,

showcasing a harmonious balance between accuracy and resource optimization.

Implications for Wildlife Monitoring:

The achieved accuracy holds significant implications for wildlife monitoring applications. The model's ability to reliably recognize and classify species, even in diverse environmental conditions, enhances the overall efficacy of the monitoring system. The integration of environmental parameters further enriches the data, providing a holistic understanding of the wildlife ecosystem.

Figure 12: Classified Data Points - 8 Classes

6.3Memory Consumption - Model

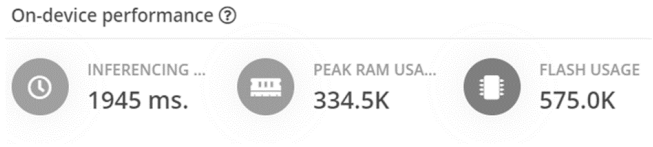

Figure 13: Memory Requirement - 8 Classes

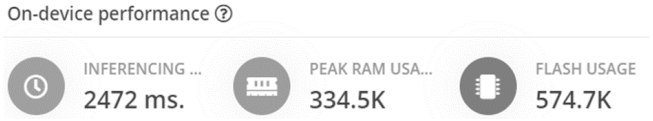

Figure 14: Memory Requirement - 4 Classes

REFERENCES

[1] Seeed Studio XIAO Series: https://www.seeedstudio.com/xiao-series-page

[2] Seeed Studio camera usage article: https://wiki.seeedstudio.com/xiao_esp32s3_camera_usage/

[3] Git-Hub repo for XIAO ESP32 : https://github.com/Mjrovai/XIAO-ESP32S3-Sense

[4] Mobile Application Development and Firebase Interaction: https://firebase.google.com/docs/functions
Digital Signal Processing (DSP) for Image Datasets:https://arxiv.org/abs/1710.04043

[5] TinyML Model Selection and Optimization:
Title: "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"
Authors: Mingxing Tan, Quoc V. Le
https://arxiv.org/abs/1905.11946

Flutter Mobile Application Development:
Title: "Flutter: A Framework for Building Native Apps"
Authors: Eric Seidel, Larwan Berke, et al.

[6] *https://flutter.dev/*

[7] Seeed Studio – Getting Started

[8] Camera Usage

[9] Github for XIAO

[10] Fruit Image Classification Based on MobileNetV2 with Transfer Learning Technique – Qian Xiang, Xiaodan Wang, Rui Li, Guoling Zhang, Jie Lai

[11] MobileNetV2: Inverted Residuals and Linear Bottlenecks
Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520

[12] MobileNetV2 Model for Image Classification - Ke Dong; Chengjie Zhou; Yihan Ruan; Yuzhi Li