Offline Biodiversity Recognition via Log-Linear Geo-Prior Fusion on Edge Devices

Wensheng WU
University of Canterbury
Christchurch, New Zealand

Course code: COSC681

January 20, 2026

Abstract

Existing biodiversity recognition tools often rely on cloud-based inference, which limits usability in connectivity-constrained environments and raises privacy concerns. This work presents an open-source and privacy-preserving biodiversity recognition system designed to operate fully offline on low-cost edge hardware.

The proposed approach introduces a log-linear geo-prior fusion strategy that integrates convolutional neural network (CNN) predictions with geographical context, improving fine-grained species classification under strict resource constraints. Accuracy analysis on the iNaturalist-2021 dataset shows that the proposed fusion strategy can significantly improve classification performance, achieving up to 89.24% Top-1 accuracy when combined with a high-capacity backbone such as ConvNeXt-Small.

For practical on-device deployment, the system is implemented using EfficientNet-B0 with log-linear geo-prior fusion on a Raspberry Pi Zero 2 W, achieving 83.26% Top-1 accuracy while satisfying latency, memory, and energy constraints. Compared to cloud-dependent solutions such as Google Lens and Seek, the system ensures full offline functionality and user privacy. In addition, the total hardware cost remains under NZD 160, representing a cost reduction of over 50$\times$ compared to specialized commercial devices such as AX Visio.

Overall, this work demonstrates the feasibility of combining context-aware learning with efficient system design to enable accessible and scalable biodiversity recognition for education and citizen-science applications in connectivity-limited environments.

Keywords: Edge AI, Offline Biodiversity Recognition, Privacy-Preserving AI, Geographical Prior, Log-Linear Fusion, Raspberry Pi.

Contents

1 Introduction

1.1 Motivation

Biodiversity is crucial for ecological balance and sustainable development. Accurate species identification is key for scientific research, conservation, and education. Recent advances in artificial intelligence (AI) have significantly improved biodiversity monitoring, enabling large-scale species classification through image-based deep learning, as seen in apps like Google Lens and Pl@ntNet.

However, most existing solutions rely on cloud-based inference, requiring continuous internet access. This introduces critical limitations: outdoor activities often take place in remote areas with unreliable connectivity, and cloud processing raises privacy concerns, especially for child-oriented applications. Additionally, the reliance on smartphones presents usability challenges for younger users.

These issues highlight the need for offline, privacy-preserving, and cost-effective biodiversity recognition systems that can function effectively on low-resource edge devices.

1.2 Problem Statement

Despite advances in AI-driven biodiversity recognition, existing tools face several limitations:

- **Network Dependency:** Cloud-based solutions are unusable in connectivity-constrained environments, common in outdoor education and fieldwork.
- **Privacy Risks:** Cloud processing involves uploading images and metadata, raising privacy concerns, especially in child-oriented applications.
- **High Hardware Cost:** Deployment often requires expensive smartphones or specialized devices, limiting accessibility in low-resource settings.

- **Limited Context Awareness:** Fine-grained species classification is difficult due to strong visual similarities, and few systems integrate geographical or temporal context offline.

These limitations highlight the need for a unified solution that operates offline, preserves privacy, leverages contextual information, and works on low-resource edge devices with strict cost and resource constraints (e.g., under NZD 160).

1.3 Research Objectives and Scope

This work aims to design and evaluate a resource-efficient biodiversity recognition framework for educational and citizen science applications. Specific objectives include:

- Design lightweight CNN architectures for large-scale species classification on edge devices.
- Explore an offline mechanism to incorporate geographical context, improving accuracy for visually similar species.
- Evaluate post-training quantization (PTQ) to reduce model size and cost while maintaining accuracy on devices like Raspberry Pi Zero 2 W.
- Develop an open-source, reproducible pipeline for transparency and community-driven development.

The scope is limited to fully offline inference, privacy-preserving operation, and cost-effective deployment on low-resource hardware, excluding cloud-assisted paradigms.

1.4 Contributions

The main contributions are:

- Design and implementation of an open-source, offline biodiversity recognition system on low-cost edge hardware, addressing privacy and accessibility in field environments.

- Adaptation and evaluation of a log-linear geo-prior fusion strategy for offline inference, balancing visual predictions and geographical context.
- Empirical analysis of post-training quantization, showing a fourfold reduction in model size with minimal accuracy loss on ARM-based devices.
- Integration of geographical priors into CNN models, improving fine-grained classification accuracy to 83.26% Top-1 accuracy, outperforming image-only baselines.
- System-level comparison with cloud-based solutions, demonstrating competitive performance at lower cost.

2 Background

2.1 Biodiversity Recognition Technologies

Biodiversity monitoring has shifted from manual taxonomic identification to AI-powered automated recognition. Traditional methods rely on human expertise, which is error-prone, especially for fine-grained classification of visually similar species[28]. Recent advances in deep learning and computer vision have enabled real-time species identification through large-scale convolutional neural networks (CNNs), supporting ecological research, citizen science, and education.

Mainstream solutions like Google Lens and Pl@ntNet rely on cloud-based models trained on extensive labeled datasets, offering high accuracy across diverse taxa. However, these solutions have key limitations: they require continuous internet connectivity, which is often unavailable in remote areas, and raise privacy concerns due to the transmission of images and metadata to external servers, especially in child-focused applications[29]. Additionally, smartphone dependency limits usability for younger users and restricts deployment in low-resource settings.

Hybrid approaches like Seek by iNaturalist improve accessibility by caching data for offline use, but still require network access for advanced features. High-end devices like AX Visio offer full offline functionality but are prohibitively expensive (around NZD 8,500), limiting scalability.

Edge AI developments, such as mobile-friendly architectures like MobileNet and EfficientNet, have enabled offline inference on resource-constrained devices, offering a potential solution. However, challenges remain in balancing accuracy, latency, and energy efficiency.

2.2 Limitations of Edge AI

Despite significant progress in model compression and hardware optimization, deploying large-scale biodiversity classifiers on low-power devices remains challenging. Three key limitations persist:

Memory Constraints: Devices such as Raspberry Pi Zero 2 W provide only 512 MB of RAM[6], making it infeasible to deploy standard CNNs (e.g., ResNet or Inception) without aggressive optimization strategies such as quantization or pruning[8].

Compute and Energy Limitations: Edge devices typically lack GPUs or NPUs, relying on low-frequency ARM CPUs for inference[14]. This results in high latency and elevated energy consumption, reducing the practicality of real-time species recognition in field conditions.

Accuracy Trade-Off: Fine-grained classification across tens of thousands of species requires capturing subtle visual differences[3]. When models are aggressively compressed to meet edge constraints, significant accuracy degradation often occurs[4]. Additionally, reliance on visual cues alone overlooks valuable contextual signals such as geography and seasonality, which can help disambiguate morphologically similar species.

These challenges underscore the need for an integrated approach that combines efficient visual recognition with contextual modeling while remaining fully compatible with resource-

constrained edge hardware. This motivates the development of our proposed framework, which introduces a novel log-linear geo-prior fusion strategy, leverages lightweight CNN architectures, and applies post-training quantization to enable scalable, privacy-preserving biodiversity recognition.

3 Related Work

In recent years, biodiversity monitoring has witnessed significant progress through the integration of advanced species classification models and context-aware AI systems. Existing approaches span a spectrum of techniques, including lightweight convolutional neural networks (CNNs) tailored for edge deployment, probabilistic modeling to incorporate contextual priors, and multimodal fusion strategies that combine visual inputs with auxiliary data sources. While these methods have achieved notable improvements in accuracy and robustness, they still exhibit critical limitations—such as insufficient optimization for ultra-low-power devices and inflexible mechanisms for integrating heterogeneous contextual cues. These gaps motivate the design objectives of this work, which aims to deliver a unified, resource-efficient solution for offline, context-aware species recognition on constrained edge platforms.

3.1 Model Efficiency for Edge AI

3.1.1 Lightweight Architectures

Deploying AI models for biodiversity monitoring on edge devices requires efficient architectures that meet strict memory, computation, and power constraints. Unlike cloud-based systems with powerful GPUs, edge devices typically have limited memory and low-frequency CPUs, making real-time, offline predictions with minimal energy consumption challenging, especially in remote environments.

Traditional deep CNNs like ResNet and Inception offer high accuracy but are too computationally expensive for edge deployment. This has led to the development of lightweight architectures that reduce complexity through innovations in convolution design, parameter compression, and network scaling. The goal is to balance accuracy with low latency and energy consumption, critical for biodiversity monitoring in resource-limited settings.

In recent years, a new generation of lightweight neural networks has emerged, pushing the boundaries of efficiency without sacrificing accuracy. For example, **EfficientFormer** [15] combines mobile-friendly convolutional layers with transformer blocks, achieving transformer-level accuracy while maintaining MobileNet-like speed. This hybrid design offers better feature representation for fine-grained classification, which is crucial for distinguishing visually similar species in the wild. Similarly, **MobileOne** [27] leverages structural re-parameterization, collapsing multiple training-time branches into a single inference-time graph. This approach yields extremely fast models that are well-suited for real-time inference on CPUs and embedded GPUs, making it an attractive candidate for ecological applications.

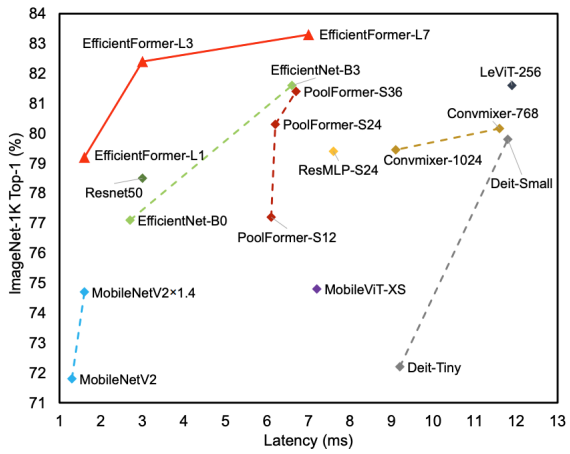


Figure 1: Inference Speed vs. Accuracy. All models are trained on ImageNet-1K and measured by iPhone 12. [15]

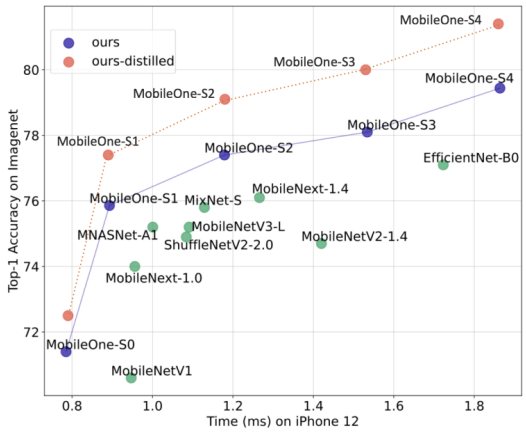
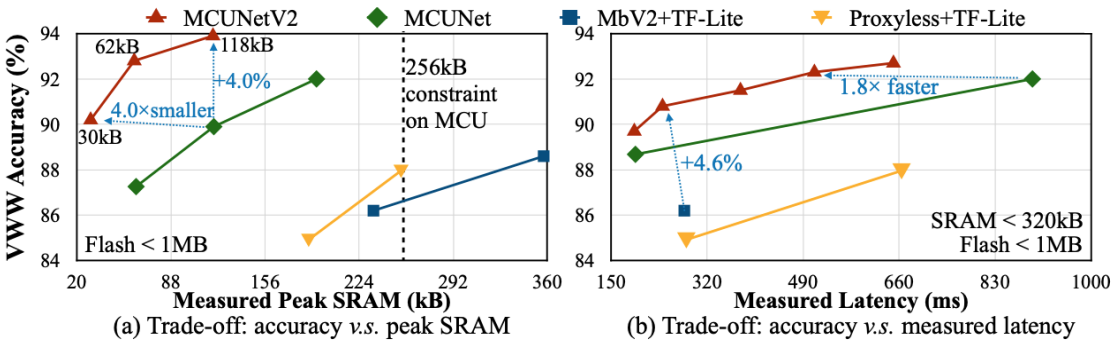


Figure 2: Top 1 accuracy vs Latency on iPhone 12. [27]

Meanwhile, transformer-based designs optimized for edge devices, such as **TinyViT** [30], distill knowledge from large-scale Vision Transformers into compact hybrids that integrate CNN-

like efficiency. These architectures exhibit robust generalization even under domain shifts, a common challenge in biodiversity datasets where species distribution and background environments vary widely. For ultra-constrained scenarios such as long-term wildlife monitoring on microcontrollers, **MCUNetV2** [17] pushes the efficiency frontier by coupling neural architecture search (NAS) with compiler-level optimization, enabling sub-megabyte models to run in real time with as little as 256KB SRAM—opening possibilities for deploying AI at an unprecedented scale in remote sensor networks.



(a) Trade-off: accuracy *v.s.* peak SRAM (b) Trade-off: accuracy *v.s.* measured latency

Figure 3: Left: MCUNetV2 has better visual wake word (VWW) accuracy vs. peak SRAM trade-off.Right: patch-based method expands the search space that can fit the MCU, allowing better accuracy vs. latency trade-off. [17]

As shown in Table 1, recent lightweight architectures such as EfficientFormer, TinyViT, MobileOne, and MCUNetV2 demonstrate strong performance by leveraging hybrid CNN–Transformer designs, large-scale distillation, or extreme memory optimizations. These models achieve higher accuracy or faster inference latency in specific conditions, making them promising candidates for fine-grained classification or deployment on ultra-constrained hardware like microcontrollers. However, their deployment often requires custom runtimes, immature quantization pipelines, or non-trivial engineering effort to adapt them for TensorFlow Lite and Raspberry Pi–class devices, which limits reproducibility in real-world edge scenarios.

In this work, we restrict our primary evaluation to three representative and practically deployable lightweight CNN models—MobileNetV2, MobileNetV3-Large, and EfficientNet-B0.

These architectures offer complementary trade-offs between parameter size, computational complexity, and accuracy, while enjoying mature support in the TensorFlow Lite ecosystem, stable post-training quantization workflows, and proven compatibility with ARM-based devices such as the Raspberry Pi Zero 2 W.

To contextualize these lightweight baselines within a broader design space, we additionally include ResNet-50, ResNet-101, ConvNeXt-Tiny, and ConvNeXt-Small in our analysis. Although these models fall outside the strict micro-model category, they provide useful reference points for understanding how modern high-capacity backbones scale in terms of accuracy and computational cost. Their inclusion allows us to contrast classical lightweight CNNs with deeper residual networks and contemporary transformer-inspired architectures.

By focusing primarily on deployable lightweight models, while benchmarking against larger backbones for completeness, we ensure that the evaluation remains both reproducible and directly transferable to real-world biodiversity monitoring applications on low-power edge devices.

Table 1: Comparison of lightweight and high-capacity models based on ImageNet Top-1 accuracy

Model	Params (M)	GFLOPs	ImageNet Top-1 (%)
MCUNetV2	<1	<0.1	71.8
MobileNetV3-Large	5.4	0.22	75.2
MobileNetV2	3.5	0.30	72.2
EfficientNet-B0	5.3	0.39	77.1
EfficientFormer-L1	12.3	1.3	79.2
MobileOne-S4	14.8	2.98	79.4
TinyViT-11M	11	2.3	80.7
ResNet-50	25.6	4.1	76.2
ResNet-101	44.5	7.8	77.4
ConvNeXt-Tiny	28.6	4.5	82.1
ConvNeXt-Small	50.2	8.7	83.1

3.1.2 Quantization and Compression

Model compression is a key enabler for efficient on-device inference under the strict memory, computational, and energy constraints of edge platforms such as the Raspberry Pi Zero 2 W. Full-precision CNN models (FP32) typically exceed the practical limits of such devices, motivating the use of quantization and related compression techniques to reduce model size, latency, and energy consumption.

Among existing approaches, post-training quantization (PTQ) and quantization-aware training (QAT) are the two dominant paradigms. PTQ is particularly attractive for edge deployment due to its simplicity and hardware friendliness, as it does not require retraining. Recent advances in the transformer literature demonstrate that carefully designed PTQ methods can achieve near-lossless accuracy on large-scale vision models [34, 31, 16, 33]. While these techniques are primarily developed for transformer architectures, their underlying principles extend naturally to convolutional neural networks.

Lightweight CNNs such as MobileNetV2/V3 and EfficientNet-B0 are particularly amenable to 8-bit quantization due to their reliance on depthwise separable convolutions and channel-wise nonlinearities[23]. In practice, per-channel INT8 quantization often preserves accuracy with minimal degradation while achieving approximately a fourfold reduction in model size and improved CPU-side efficiency through ARM NEON integer instructions[12]. In contrast, architectures with heavy normalization and activation components (e.g., ConvNeXt variants with LayerNorm and GELU) tend to be more sensitive to quantization, highlighting the importance of backbone selection for edge deployment.

Within the TensorFlow Lite ecosystem, several quantization strategies are available, including full-integer INT8, mixed-precision, FP16, and dynamic range quantization (DRQ)[26]. For Raspberry Pi Zero 2 W, FP16 acceleration is not supported at the hardware level, limiting

its practical benefit. Dynamic range quantization, adopted in this work, offers a pragmatic compromise by quantizing weights to INT8 while retaining floating-point activations, resulting in substantial size reduction and improved loading efficiency without requiring calibration data. In summary, the combination of lightweight CNN architectures and dynamic range INT8 quantization provides an effective balance between model compactness, inference speed, and classification accuracy. This design choice enables fully offline biodiversity recognition on low-cost edge hardware while satisfying strict memory and energy constraints.

3.2 Context-Aware Strategies Beyond Vision

3.2.1 Metadata-Enhanced Learning

Beyond visual features, incorporating contextual cues such as geographic location and time of observation has become a promising strategy for improving fine-grained species classification. These metadata elements provide strong priors that can help distinguish visually similar species and enhance robustness under real-world conditions (see Figure 4).

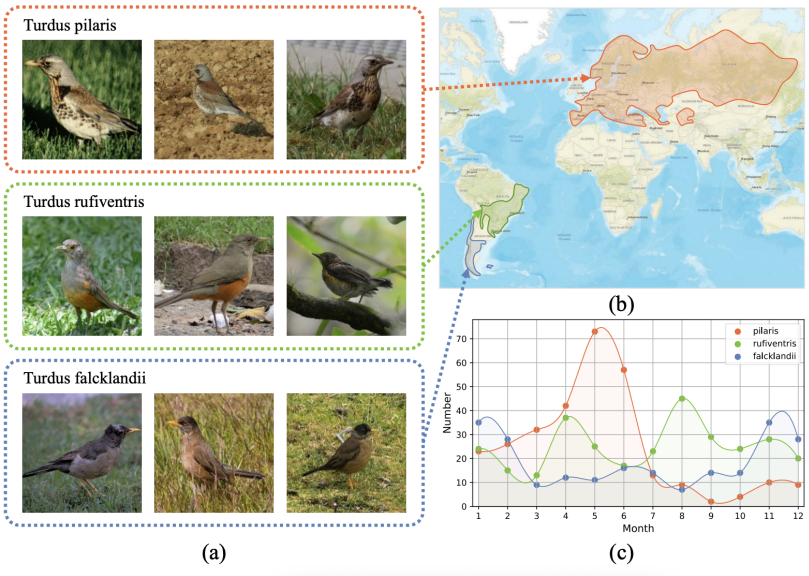


Figure 4: The pilaris, rufiventris, and falcklandii are species belonging to the genus Turdus. (a): They are visually similar and hard to recognize based on their appearance. (b): Their habitats vary widely. (c): Their activity frequency varies throughout the year, so the data amount at different times is different. Adapted from [32]

A foundational contribution in this area is the work by Mac Aodha et al. on **Presence-Only Geographical Priors** [1]. This approach introduces a spatio-temporal prior estimated from presence-only data—easier to obtain than presence-absence datasets—which predicts the likelihood of a species occurring at a given location and time. The prior is modeled using a neural embedding framework that jointly represents location, time, and photographer identity along with species labels. At inference, the prior is combined with image classifier predictions under a Bayesian Fusion:

$$P(y|I, x) \propto P(y|I) \cdot P(y|x),$$

where $P(y|I)$ denotes image-based predictions and $P(y|x)$ the spatio-temporal prior. This method significantly improves top-1 accuracy across datasets such as iNat2018 and NABirds without requiring retraining of the image classifier. Unlike grid-based or nearest-neighbor approaches, this embedding strategy (Figure 5) scales efficiently to millions of observations while capturing fine ecological patterns such as seasonal migrations.

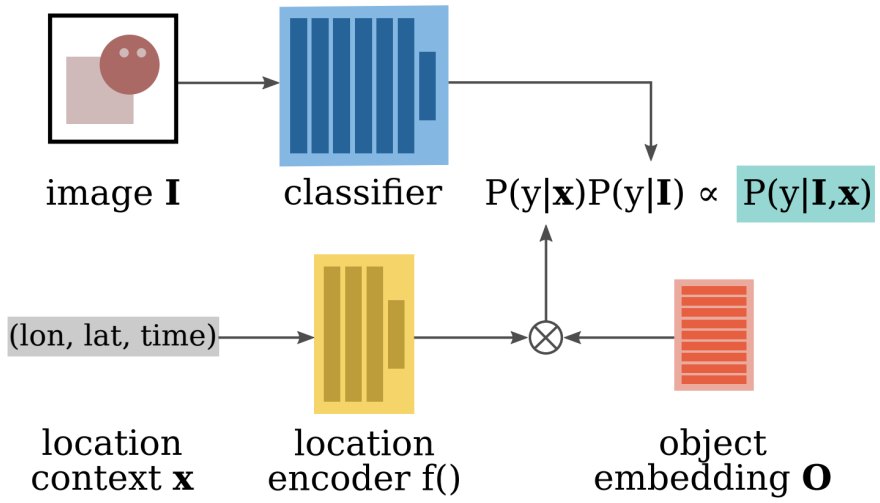


Figure 5: Illustration of the log-linear fusion framework combining image-based predictions $P(y|I)$ and spatio-temporal priors $P(y|\mathbf{x})$, where $\mathbf{x}$ includes metadata such as longitude, latitude, and time. Adapted from [1]

Building upon this direction, Yang et al. propose a **Dynamic MLP framework** [32] to fully

exploit multimodal information, addressing limitations of earlier fusion methods that operate in a single dimension (e.g., concatenation or additive fusion). Their method employs a dynamic multi-layer perceptron whose weights are conditioned on metadata (latitude, longitude, date), enabling instance-wise, high-dimensional interactions between visual and contextual features. This dynamic projection adaptively refines image embeddings, improving decision boundaries for visually similar species. Experiments on multiple fine-grained datasets, including iNaturalist 2017–2021, demonstrate consistent state-of-the-art performance, with up to +5.8% top-1 accuracy gains over previous multimodal baselines. Moreover, t-SNE visualizations show that dynamic MLP produces more discriminative clusters than concatenation-based models, confirming its effectiveness in leveraging metadata. The architecture is shown in Figure 6.

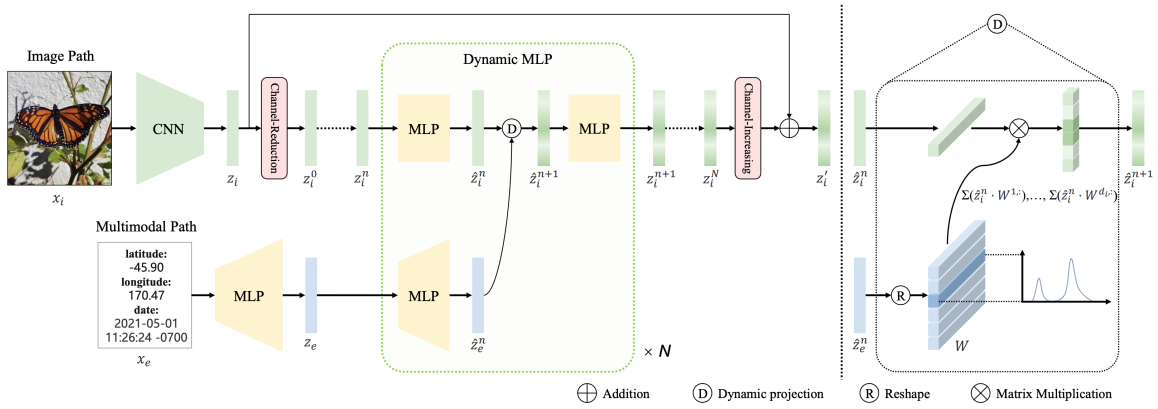


Figure 6: An overview of Dynamic MLP framework. [32]

These advances underscore the growing importance of metadata-enhanced learning for biodiversity monitoring. However, deploying such approaches on edge devices introduces new challenges, including storage of prior models, real-time inference under strict compute budgets, and robustness to missing or noisy metadata—factors addressed by the hybrid framework proposed in this work.

3.2.2 Hybrid Deep-Structured Models

Hybrid deep-structured models, broadly defined as approaches that integrate deep neural networks with additional sources of structured or contextual information, have emerged as a promising paradigm for enhancing robustness, interpretability, and adaptability. These models extend the representational power of deep learning by embedding domain-specific hierarchies, symbolic constraints, or multimodal knowledge into the learning pipeline, thereby improving generalization and decision-making under uncertainty.

One representative example is the Hierarchy-Guided Neural Network (HGNN) introduced by Elhamod et al., which embeds taxonomic hierarchy into the prediction pipeline by constraining classifier outputs according to phylogenetic relationship (Figure 7). This integration of hierarchical structure not only improves generalization across species classes but also provides biologically interpretable decision-making, a key advantage for ecological monitoring tasks [7].

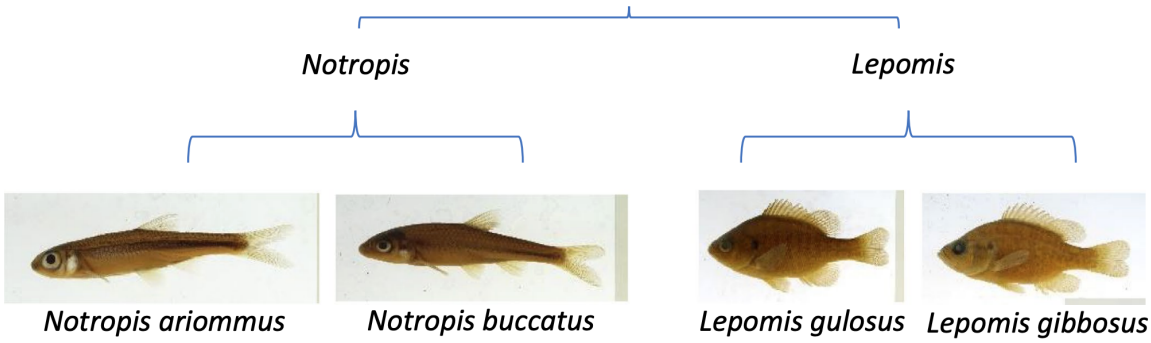


Figure 7: Species that belong to the same genus exhibit features that are similar because of common ancestry [7]

More recently, research has explored multimodal fusion paradigms, particularly integrating Large Language Models (LLMs) with visual backbones for plant disease detection. For instance, Roumeliotis et al. developed a hybrid GPT-4o and ResNet-50 pipeline, enabling both zero-shot and few-shot learning scenarios on the PlantVillage dataset. This multimodal framework achieved an impressive 98.12% accuracy on apple leaf classification (Figure 8), signifi-

cantly outperforming conventional CNN-based baselines while enhancing adaptability to novel classes [21].

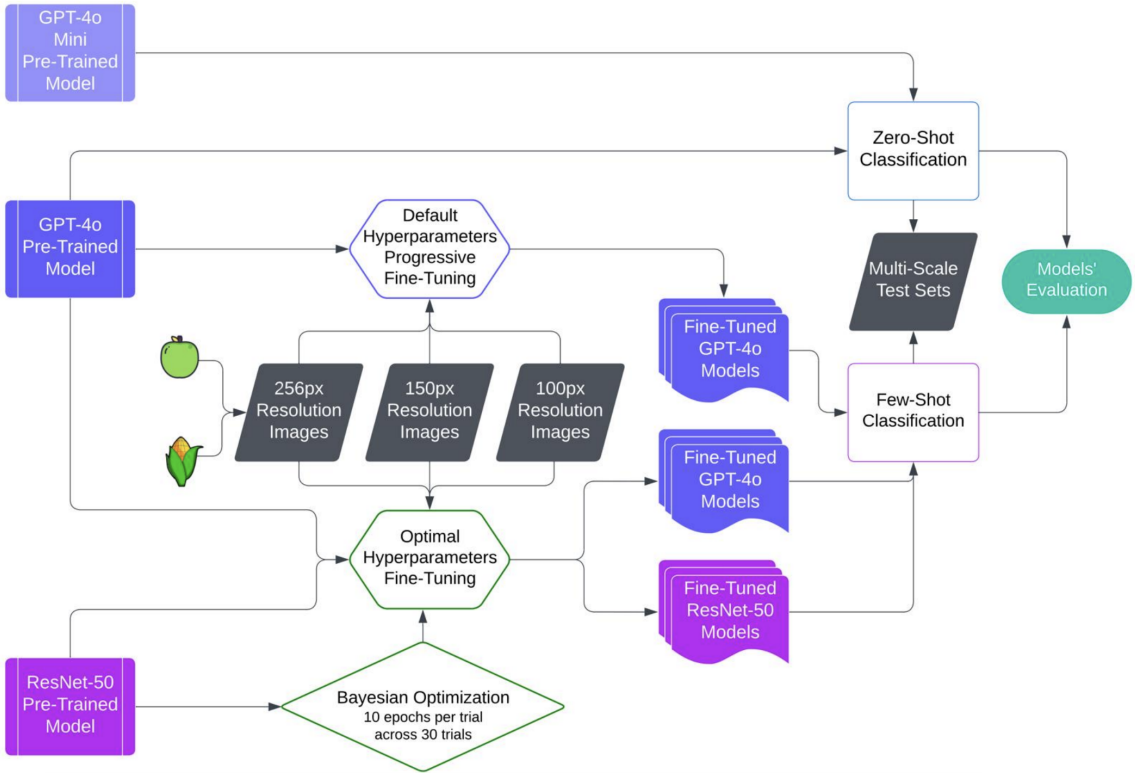


Figure 8: Fine-Tuning and Evaluation Framework for Multimodal Plant Leaf Disease Classification Using GPT-4o and ResNet-50. [21]

Overall, these hybrid strategies—ranging from graph-based reasoning and taxonomic guidance to multimodal LLM integration—highlight an emerging trend: combining structural knowledge and cross-modal learning with deep feature extraction can substantially enhance accuracy, interpretability, and efficiency in biodiversity monitoring and related ecological applications.

3.3 Summary and Research Gap

Recent advances in biodiversity monitoring have been driven by lightweight neural architectures for edge inference, model compression techniques such as post-training quantization, and metadata-enhanced learning that incorporates geographic and temporal context. Together, these developments have improved both recognition accuracy and computational efficiency.

A wide range of visual backbones has been explored for fine-grained species classification. Lightweight models, including MobileNet variants and EfficientNet-B0, provide favorable accuracy–efficiency trade-offs for mobile and embedded deployment. In contrast, higher-capacity architectures such as ResNet and ConvNeXt achieve stronger representational performance but incur prohibitive computational costs, making them unsuitable for real-time inference on ultra-low-power edge devices. Among deployable candidates, EfficientNet-B0 offers a particularly balanced combination of accuracy, model size, and quantization stability for edge platforms such as the Raspberry Pi Zero 2 W.

Beyond purely visual approaches, context-aware classification has demonstrated clear benefits by integrating geographic and temporal priors to disambiguate visually similar species. However, most existing methods rely on fixed-weight Bayesian fusion schemes, limiting adaptability when contextual reliability varies across environments.

More complex hybrid frameworks that incorporate structured ecological knowledge or large language models further improve generalization and interpretability, but their computational and memory demands exceed the constraints of fully offline edge deployment.

Research Gap. Despite these advances, existing approaches lack a unified framework that simultaneously (1) operates fully offline on low-cost edge hardware, (2) flexibly balances visual and contextual information, and (3) remains compatible with reproducible, resource-efficient deployment pipelines. This gap motivates the development of an edge-oriented biodiversity recognition system that combines lightweight CNN backbones, adaptive fusion mechanisms, and practical quantization strategies.

4 Research Question and Critical Functionalities

4.1 Research Question

The overarching research question guiding this work is:

> *How can a biodiversity recognition system be designed to operate fully offline on low-cost edge devices while preserving user privacy and maintaining competitive classification accuracy under strict resource constraints?*

This question emerges from the limitations identified in existing approaches: reliance on cloud-based inference, lack of privacy protection, high hardware cost, and inadequate use of contextual information in edge deployments. Addressing this question requires integrating efficient visual recognition, context-aware modeling, and model optimization techniques into a unified framework.

4.2 Critical Functionalities

To answer the research question, the proposed system must deliver the following critical functionalities:

1. **Fully Offline Operation:** Ensure all inference processes run locally without any dependence on cloud connectivity, enabling usability in remote environments and safeguarding user privacy.

2. **Lightweight CNN Backbone:** Employ computationally efficient neural network architectures (e.g., MobileNet, EfficientNet) optimized for resource-constrained hardware such as Raspberry Pi Zero 2 W.

3. **Context-Aware Prediction:** Incorporate geographical priors alongside visual recognition using an adaptive fusion strategy to improve fine-grained species classification accuracy.

4. **Model Optimization:** Apply post-training quantization and other compression techniques to reduce model size and inference latency without significant accuracy degradation.

5. **Open-Source and Extensible Framework:** Provide publicly available code and reproducible pipelines to ensure transparency, foster educational use, and support future enhancements.

6. **Cost-Effectiveness:** Achieve low deployment cost (under NZD 160) while delivering real-time inference performance on edge devices with limited memory (512 MB RAM).

5 Methodology

5.1 Dataset

We use the iNaturalist 2021 dataset, a large-scale benchmark for fine-grained species classification, widely used in biodiversity and ecological AI research. This dataset contains 2.68 million training images across 10,000 species categories, organized into 13 high-level taxonomic super-categories (e.g., plants, insects, birds). Each image is accompanied by metadata such as GPS coordinates and timestamp, which provides essential context for modeling geographical priors.[20]

5.1.1 Data Split

Unlike the official competition setting, which keeps the test set private, our project adopts a custom split. Specifically, the original validation set (100,000 images) is divided into:

- Validation set: 50,000 images
- Test set: 50,000 images

This ensures that evaluation is performed on unseen samples while preserving consistency with the iNaturalist distribution. The resulting data partitions are:

- Training set: 2,686,843 images
- Validation set: 50,000 images
- Test set: 50,000 images

5.1.2 Category Distribution

The dataset exhibits a pronounced long-tailed distribution, with plants and insects accounting for the majority of samples, while other groups such as arachnids and mollusks remain under-represented[10]. This distribution significantly impacts model performance and motivates the use of geographical priors to mitigate class imbalance. Table 2 summarizes the distribution of species and image samples in the iNaturalist 2021 subset used in this work.

Super Category	Species Count	Train Images	Train Mini Images	Val Images	Test Images
Plants	4,271	1,148,702	213,550	42,710	x
Insects	2,526	663,682	126,300	25,260	x
Birds	1,486	414,847	74,300	14,860	x
Fungi	341	90,048	17,050	3,410	x
Reptiles	313	86,830	15,650	3,130	x
Mammals	246	68,917	12,300	2,460	x
Ray-finned Fishes	183	45,166	9,150	1,830	x
Amphibians	170	46,252	8,500	1,700	x
Mollusks	169	44,670	8,450	1,690	x
Arachnids	153	40,687	7,650	1,530	x
Animalia	142	37,042	7,100	1,420	x
Total	10,000	2,686,843	500,000	100,000	500,000

Table 2: Distribution of species and image samples in the original iNaturalist 2021 subset used in this project[20]. In our project, the validation set was further divided equally into a 50,000-image validation set and a 50,000-image test set.

The class imbalance and sample allocation are visualized in Figure 9, illustrating the distribution of image counts per species for the training, mini-training, and validation subsets. The curve clearly demonstrates the long-tail nature of the dataset.

Geographic diversity is another defining characteristic of the iNaturalist dataset. Figure 10 shows the global distribution of images for the training, mini-training, validation, and test subsets, highlighting coverage across multiple continents but with varying density.

5.1.3 Preprocessing

All images were resized to 224×224 pixels and normalized using ImageNet mean and standard deviation. Standard data augmentation techniques were applied during training, including ran-

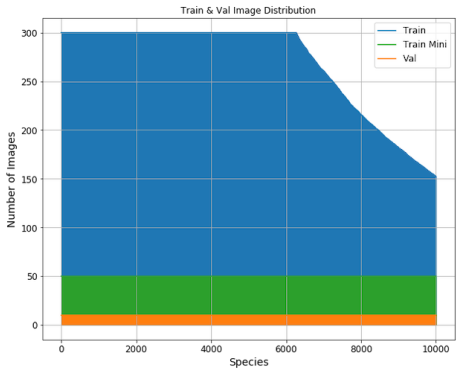


Figure 9: Train and validation image distribution across 10,000 species (long-tail pattern)[20].

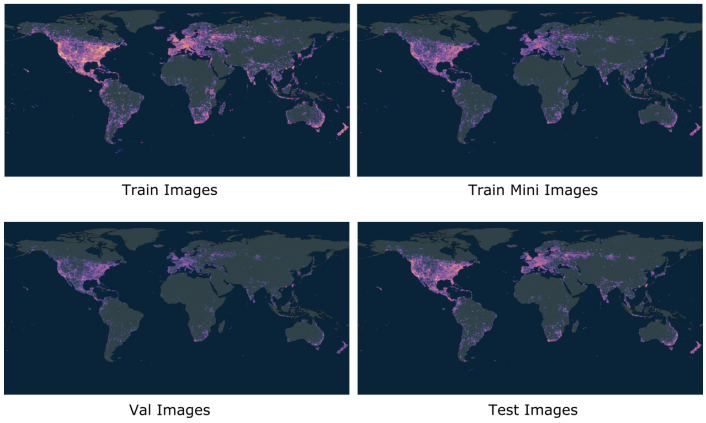


Figure 10: Geographic Distribution of Images[20]

dom cropping, horizontal flipping, and color jitter. For geographical prior integration, metadata such as latitude, longitude, and observation date were extracted and encoded. Images without valid location data were assigned a uniform prior.

5.2 Visual Backbone Models

To evaluate the effectiveness of the proposed fusion strategy across a diverse range of visual representations, we adopt multiple convolutional neural network (CNN) backbones that span different architectural families and accuracy–efficiency trade-offs. Specifically, we consider EfficientNet-B0, MobileNetV3-Large, MobileNetV2, ConvNeXt-Tiny, ConvNeXt-Small, as well as ResNet-50 and ResNet-101 [25, 11, 23, 18]. Together, these models cover lightweight mobile-oriented architectures and more expressive high-capacity networks, enabling a comprehensive comparison under resource-constrained deployment settings.

From a functional perspective, all selected backbones act as hierarchical feature extractors that transform an input image $\mathbf{I} \in R^{H\times W\times 3}$ into a compact semantic representation:

$$\mathbf{f} = \Phi_\theta(\mathbf{I}), \tag{1}$$

where Φ_θ denotes the backbone network parameterized by θ, and $\mathbf{f} \in R^d$ represents the resulting feature embedding passed to the downstream classification or fusion modules.

The selected backbones differ in depth, convolutional design, and computational complexity, allowing us to analyze how architectural choices and quantization interact with on-device inference efficiency, model size, and predictive performance.

5.2.1 EfficientNet-B0

EfficientNet-B0 introduces a principled compound scaling strategy that uniformly scales network depth, width, and input resolution using a set of learnable coefficients. Instead of scaling architectural dimensions independently, the model jointly optimizes them under a fixed computational budget. Formally, the scaled network parameters are defined as

$$d = \alpha^\phi, \qquad w = \beta^\phi, \qquad r = \gamma^\phi, \tag{2}$$

where d, w, and r denote depth, width, and resolution multipliers, respectively, and ϕ is the global scaling factor. The coefficients (α, β, γ) are determined via grid search under the constraint

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2, \tag{3}$$

ensuring that each increment of ϕ approximately doubles the computational cost[25].

EfficientNet-B0 serves as the baseline model ($\phi = 0$), consisting of 5.3M parameters and requiring 390M FLOPs at 224×224 resolution. Despite its compact size, it achieves a strong 77.1% Top-1 accuracy on ImageNet. This efficiency arises from its use of inverted bottleneck MBConv layers, squeeze-and-excitation (SE) attention, and depthwise separable convolutions, combined with the optimized scaling methodology.

Table 3 provides a stage-by-stage decomposition of EfficientNet-B0, illustrating how different MBConv configurations contribute to the overall capacity and representational hierarchy of the network.

Table 3: EfficientNet-B0 baseline network architecture. Each stage i applies operator $\hat{\mathcal{F}}_i$ to inputs of resolution $\hat{H}_i \times \hat{W}_i$, producing $\hat{C}_i$ output channels over $\hat{L}_i$ repeated layers. Adapted from [25].

Stage i	**Operator** $\hat{\mathcal{F}}_i$	**Resolution** $\hat{H}_i \times \hat{W}_i$	**#Channels** $\hat{C}_i$	**#Layers** $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 + Pooling + FC	7×7	1280	1

5.2.2 MobileNetV3-Large

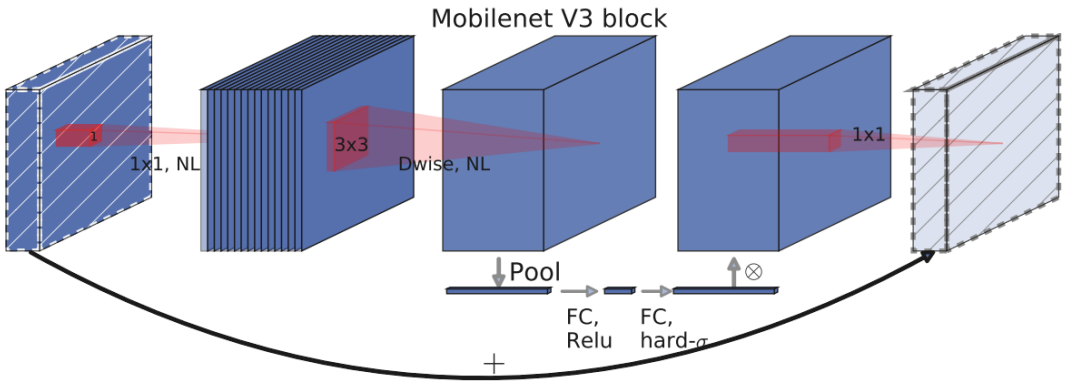


Figure 11: MobileNetV3 block structure. Adapted from [11].

MobileNetV3-Large combines advances from Neural Architecture Search (NAS), attention mechanisms, and efficient activation functions to achieve strong performance under strict computational budgets[11]. Designed explicitly for mobile inference, it integrates three key components: (1) resource-aware NAS that explores layer configurations under latency constraints, (2) Squeeze-and-Excitation (SE) blocks that provide channel-wise attention, and (3) the lightweight

h-swish activation function that improves accuracy while maintaining computational efficiency. With 5.4M parameters and only 219M FLOPs, MobileNetV3-Large attains a 75.2% Top-1 accuracy on ImageNet. Compared to EfficientNet-B0, it sacrifices a small amount of accuracy in exchange for approximately 44% lower FLOPs, making it well suited for real-time edge applications where inference speed is a critical requirement.

The SE attention mechanism rescales each channel using

$$\mathbf{z} = \sigma(W_2\, \delta(W_1 \cdot \mathrm{GAP}(\mathbf{x}))), \tag{4}$$

where GAP denotes global average pooling, δ is the ReLU activation, and σ is the sigmoid function. The input feature $\mathbf{x}$ is modulated as $\tilde{\mathbf{x}} = \mathbf{z} \odot \mathbf{x}$, allowing the network to emphasize informative channels with minimal overhead.

To further reduce computational cost, MobileNetV3 replaces swish with the piecewise approximation h-swish:

$$\text{h-swish}(x) = x \cdot \frac{\mathrm{ReLU6}(x+3)}{6}, \tag{5}$$

which is both hardware-friendly and empirically effective.

Figure 11 illustrates the MobileNetV3 block design, which combines depthwise separable convolutions, SE attention, and the NAS-discovered expansion ratios and kernel sizes.

5.2.3 MobileNetV2

MobileNetV2 represents an earlier yet influential generation of efficient CNNs, introducing two key innovations that significantly reduce computational cost: inverted residual blocks and linear bottlenecks[23]. Unlike traditional residual architectures that expand channel dimensions before projection, MobileNetV2 applies a depthwise–separable convolution within a narrow in-

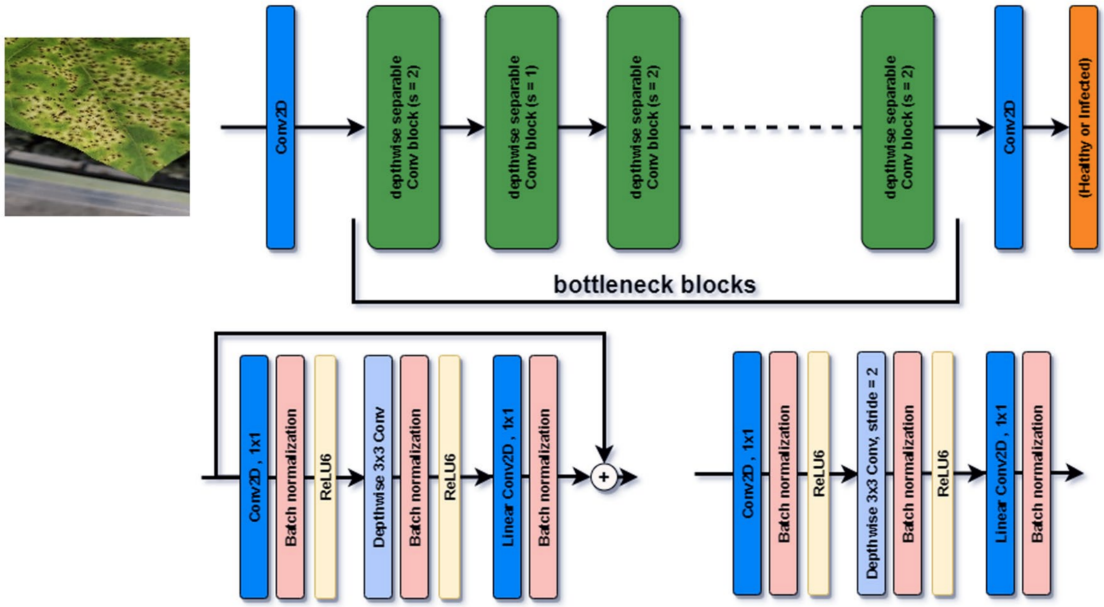


Figure 12: The architecture of the MobileNetV2 with a sample input image and 19 residual bottleneck layers. Adapted from [24].

termediate representation, improving efficiency while maintaining feature expressiveness. Formally, each block performs a nonlinear transformation only in the high-dimensional space, while the projection to a lower-dimensional bottleneck remains strictly linear, ensuring that information is not prematurely compressed.

With 3.4M parameters and approximately 300M FLOPs, MobileNetV2 achieves around 71.8% Top-1 accuracy on ImageNet. Although it is less accurate than newer architectures such as EfficientNet-B0 and MobileNetV3-Large, its conceptual simplicity, strong efficiency, and historical impact make it a valuable baseline for lightweight model comparison.

5.2.4 ConvNeXt-Tiny and ConvNeXt-Small

ConvNeXt represents a modernized convolutional architecture that revisits the design principles of classical ResNets through the lens of Vision Transformer (ViT) developments[18]. The model incorporates several architectural refinements—such as a patchify convolutional stem, large-kernel depthwise convolutions, GELU activation, pre-activation LayerNorm, and simplified bottleneck structures—which collectively enhance optimization stability and representa-

tion capacity.

Formally, a ConvNeXt block applies the sequence

$$\mathbf{y} = \mathbf{x} + \mathrm{Conv}_{1\times1}\Big(\mathrm{GELU}\big(\mathrm{DWConv}_{7\times7}(\mathbf{x})\big)\Big),$$

which parallels the residual formulation of ResNet while integrating modern design choices such as depthwise separability and non-linear expansion.

In this work, we evaluate two ConvNeXt variants[18]:

- **ConvNeXt-Tiny** — 28M parameters, 4.5 GFLOPs, 82.1% ImageNet Top-1 accuracy.
- **ConvNeXt-Small** — 50M parameters, 8.7 GFLOPs, 83.1% ImageNet Top-1 accuracy.

ConvNeXt-Small provides approximately a 1% accuracy gain over the Tiny variant at the cost of substantially increased computation, making it a stronger but less efficient backbone. Although both models exceed the real-time compute budget of devices such as the Raspberry Pi Zero 2 W, they remain compatible with TensorFlow Lite and support INT8 quantization, allowing them to serve as high-capacity reference baselines for fine-grained biodiversity recognition tasks. Their inclusion enables analysis of performance upper bounds under our fusion framework.

5.2.5 ResNet-50 and ResNet-101

The ResNet architecture introduced the concept of *residual learning*, in which skip connections enable identity mappings across layers to alleviate the vanishing-gradient problem that historically limited the depth of convolutional networks. Formally, instead of learning a direct mapping $\mathcal{H}(\mathbf{x})$, each block learns a residual function $\mathcal{F}(\mathbf{x})$ such that[9]:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}) + \mathbf{x}, \tag{6}$$

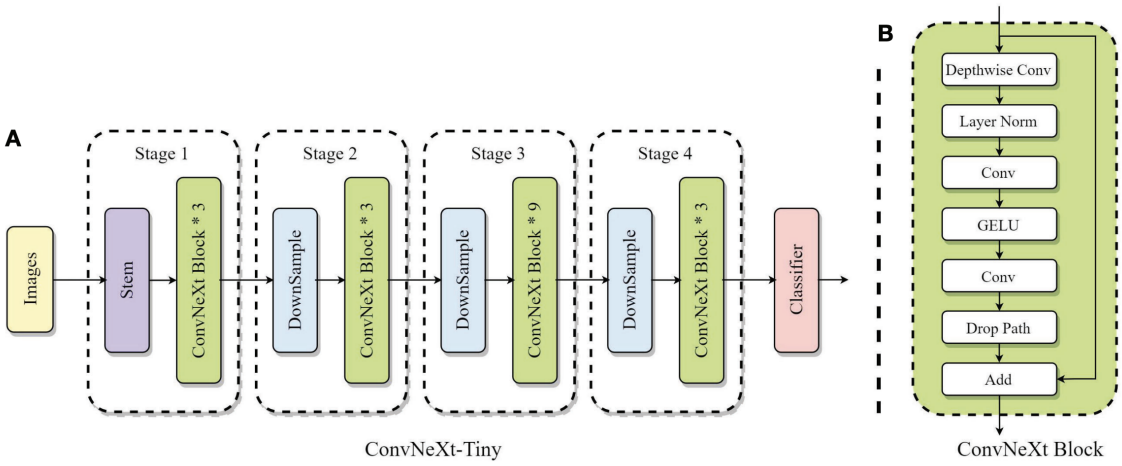


Figure 13: (A) ConvNeXt-Tiny overall network structure; (B) ConvNeXt Block structure. Adapted from [13].

which stabilizes gradient flow, facilitates optimization of very deep models, and encourages feature reuse across layers.

ResNet-50 consists of 50 layers composed of bottleneck blocks using a $1 \times 1 \rightarrow 3 \times 3 \rightarrow 1 \times 1$ convolutional structure, providing a balanced trade-off between accuracy and computational cost for large-scale visual recognition. ResNet-101 extends this design to 101 layers by stacking additional bottleneck blocks, substantially increasing representational capacity and achieving higher ImageNet Top-1 accuracy. These deeper residual networks remain strong baselines for fine-grained recognition tasks, including biodiversity datasets such as iNaturalist.

In this project, ResNet-50 and ResNet-101 serve as mid- and high-capacity reference architectures to contextualize the performance of lightweight models under compression and quantization. Whereas ResNet-50 represents a practical depth–complexity compromise, ResNet-101 provides a deeper performance ceiling that illustrates the limits of depth scaling under edge computing constraints. Their TensorFlow Lite conversions allow direct comparison against efficient architectures such as MobileNet and EfficientNet on the Raspberry Pi Zero 2 W, highlighting the trade-offs between conventional deep CNNs and modern lightweight designs.

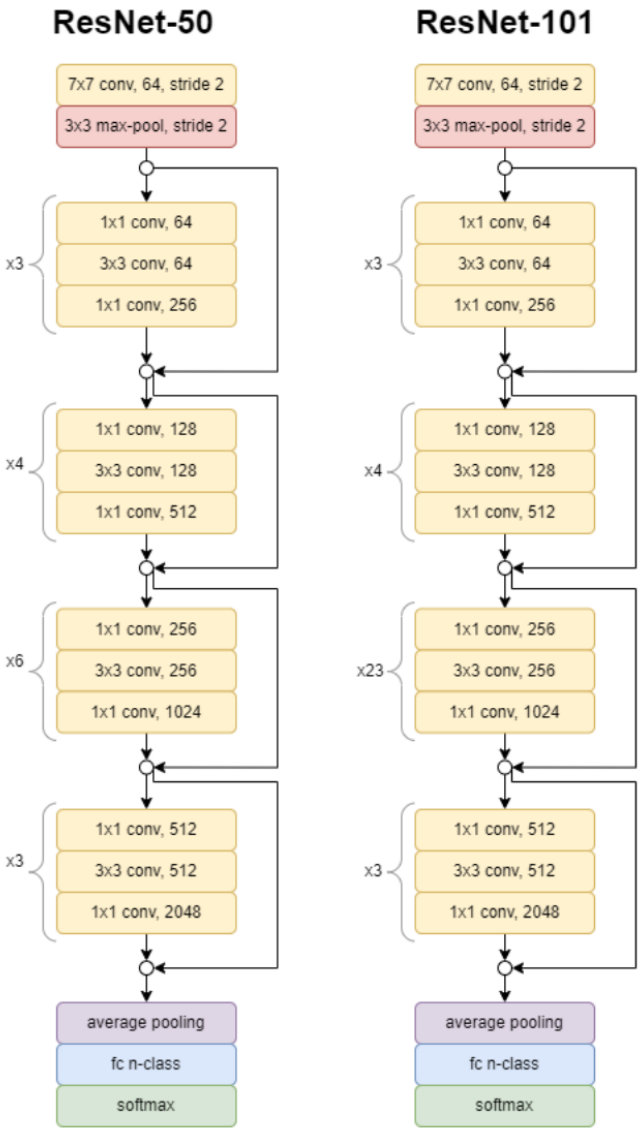


Figure 14: Visualization of the ResNet-50 and ResNet-101 architectures. Adapted from [19].

5.2.6 Comparison and Rationale

Table 4 summarizes the core quantitative characteristics of all evaluated CNN architectures, including ImageNet Top-1 accuracy, parameter count, computational cost (FLOPs), and INT8 quantization support. The selected models span a wide range of architectural families and capacity levels, enabling a systematic examination of how model scale and design principles affect performance under edge deployment constraints.

MobileNetV2, MobileNetV3-Large, and EfficientNet-B0 represent lightweight, depthwise-separable convolutional architectures designed for mobile and resource-limited environments. EfficientNet-B0 achieves the highest accuracy among these models due to its compound scaling strategy, whereas MobileNetV3-Large offers a strong accuracy–efficiency trade-off through the use of SE modules and NAS-guided design.

ResNet-50 and ResNet-101 serve as classical residual CNN baselines, providing mid- to large-capacity reference points. ConvNeXt-Tiny and ConvNeXt-Small extend this paradigm by revisiting and modernizing ResNet-style architectures with design elements inspired by Vision Transformers, enabling them to achieve higher accuracy at similar computational scales. By evaluating this diverse set of architectures under FP32 and INT8 quantization, we aim to understand how differences in representational capacity, convolutional design, and model scaling strategies influence the robustness and efficiency of our edge-focused fusion pipeline.

Table 4: Summary of evaluated CNN models, sorted by Top-1 Acc.

Model	Top-1 Acc. (%)	Parameters	FLOPs	INT8 Support
MobileNetV2 [23]	≈72.0	3.5M	300M	YES
EfficientNet-B0 [25]	77.1	5.3M	390M	YES
MobileNetV3-Large [11]	75.2	5.4M	219M	YES
ResNet-50 [9]	76.0	25.6M	4.1G	YES
ResNet-101 [9]	77.4	44.5M	7.8G	YES
ConvNeXt-Tiny [18]	82.1	28M	4.5G	YES
ConvNeXt-Small [18]	83.1	50M	8.7G	YES

5.3 Quantization Techniques

To deploy large-scale biodiversity classification models on memory-constrained edge devices such as the Raspberry Pi Zero 2 W, we adopted post-training quantization (PTQ) strategies supported by TensorFlow Lite. PTQ eliminates the need for retraining and significantly reduces storage footprint and computational cost, making it highly suitable for resource-limited environments[12].

5.3.1 Adopted Quantization Methods

FP32 (Baseline): The full-precision format uses 32-bit floating-point representation for both weights and activations. While this configuration provides the highest accuracy, it results in large model sizes (e.g., EfficientNet-B0 $\approx$ 64 MB) and high memory and latency costs, which limit its usability on low-power devices[2].

INT8 Dynamic Range Quantization (DRQ): We applied dynamic range quantization, which converts all model weights from FP32 to INT8 while keeping activations at floating-point. This

method does not require a representative dataset, making it fast and convenient for deployment. DRQ typically achieves a substantial reduction in model size compared to FP32 and improves loading and inference efficiency on CPU-bound systems.

5.3.2 Practical Considerations

Model Size vs. Accuracy Trade-off: Quantization provides substantial model size reduction (approximately 4× compared to FP32), lowering storage requirements and improving loading time[5]. Although a slight loss in accuracy is expected, this trade-off is acceptable for edge deployment.

Deployment Complexity: Dynamic range quantization does not require a representative calibration dataset, simplifying the optimization pipeline and accelerating deployment.

Hardware Compatibility: FP32 ensures maximum compatibility but incurs the highest memory and latency overhead. INT8 quantization is particularly effective on ARM-based processors leveraging NEON SIMD instructions for integer arithmetic.

FP16 Was Not Applied: Although TensorFlow Lite supports FP16 quantization, we did not adopt it because the Raspberry Pi Zero 2 W (ARM Cortex-A53 CPU) lacks dedicated FP16 hardware acceleration. Consequently, FP16 would not deliver meaningful runtime improvements over FP32, while introducing additional conversion steps.

5.4 Geo Prior Integration

5.4.1 Motivation

Fine-grained species classification on large-scale datasets such as iNaturalist 2021 remains challenging due to high inter-class similarity and limited discriminative features in visual con-

tent alone. Prior work has shown that incorporating contextual information, such as geographic location and time of capture, significantly improves classification accuracy. Following this idea, we integrate a geographical prior (Geo Prior) model into our baseline classifier predictions to exploit species distribution patterns.

As illustrated in Figure 4, species from the same genus (e.g., Turdus) exhibit strong visual similarity, while their geographical ranges and seasonal patterns differ significantly. This highlights the potential of leveraging spatio-temporal priors to disambiguate visually similar classes.

5.4.2 Geo Prior Model

We implement a fully connected network (FCNet) to model species distribution as a function of geographical and temporal metadata, following the presence-only prior approach of Mac Aodha *et al.*[1]. The model takes as input

$$\mathbf{x} = \big\{ \sin(\pi \cdot \text{lon}),\ \cos(\pi \cdot \text{lon}),\ \sin(\pi \cdot \text{lat}),\ \cos(\pi \cdot \text{lat}),\ \sin(\pi \cdot \text{date}),\ \cos(\pi \cdot \text{date}) \big\}, \tag{7}$$

where longitude and latitude are normalized to $[-1, 1]$, and the date is encoded cyclically to capture seasonal patterns.

Network Architecture The architecture consists of:

- An initial dense layer followed by ReLU activation.
- Four residual blocks, each with two dense layers and skip connections, producing an embedding of size 256.
- An output layer predicting class probabilities for all species using a sigmoid activation.

Loss Function The model is trained using **presence-only weighted binary cross-entropy** for location-based labels. To improve generalization, random geographic and temporal points are sampled, and the network is penalized for predicting positive labels on these synthetic negatives, as in the original work:

$$\mathcal{L} = \mathcal{L}_{\text{obs}}(y, \hat{y}) + \mathcal{L}_{\text{rand}}(0, \hat{y}), \tag{8}$$

where $\mathcal{L}_{\text{obs}}$ and $\mathcal{L}_{\text{rand}}$ correspond to observed and random samples, respectively.

5.4.3 Fusion Strategies

Bayesian Fusion (baseline): Following Mac Aodha *et al.*[1], we first consider the Bayesian product rule:

$$P(y \mid I, x) \propto P(y \mid I) \cdot P(y \mid x). \tag{9}$$

This formulation assumes equal confidence in both sources of information, simply multiplying the image-based prediction and the Geo Prior probability.

Log-Linear Fusion (proposed): While Bayesian Fusion offers a solid approach, it enforces a fixed balance between image and prior models, which may not be ideal in dynamic real-world scenarios. To introduce more flexibility, we adapt a Log-Linear Fusion strategy, where the log-probability space is interpolated:

$$\log P(y \mid I, x) \propto \alpha \log P(y \mid I) + (1 - \alpha) \log P(y \mid x), \tag{10}$$

with $\alpha \in (0, 1)$ acting as a tunable hyperparameter. In our experiments, α was adjusted based on model performance, with values closer to 1 prioritizing visual features and values closer to 0

emphasizing geographical context. This dynamic fusion mechanism allows the system to adapt to varying levels of context reliability and the visual clarity of species. By experimenting with different α values, we find that α can be optimized to balance accuracy and robustness to noisy metadata, ensuring better performance across various species and environmental conditions.

5.5 Evaluation Metrics

To comprehensively evaluate model performance under edge deployment constraints, we employ a set of metrics grouped into three categories: Accuracy Metrics, Efficiency Metrics, and Runtime Metrics. These metrics capture trade-offs between accuracy, resource utilization, and inference speed, which are critical for offline biodiversity classification on low-power devices[2].

5.5.1 Accuracy Metrics

- **Top-1 Accuracy**

  Measures the proportion of test samples for which the model's highest-confidence prediction matches the ground truth:

$$\text{Top-1 Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1\big(\arg\max_{k} \hat{y}_i^{(k)} = y_i\big)$$

- **Top-5 Accuracy**

  Calculates the percentage of samples where the correct label appears among the top five predictions[22]:

$$\text{Top-5 Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1\big(y_i \in \text{Top-5}(\hat{y}_i)\big)$$

- **Fusion-Aware Accuracy**

When applying Geo Prior integration, image-based predictions $\hat{y}_i^{image}$ are combined with geographical priors $\hat{y}_i^{geo}$ using the Log-Linear Fusion strategy:

$$\hat{y}_i^{fusion}(k) = \alpha \cdot \log\left(\hat{y}_i^{image}(k)\right) + (1-\alpha) \cdot \log\left(\hat{y}_i^{geo}(k)\right)$$

The final accuracy is then computed using the fused predictions:

$$\text{Fusion-Aware Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1\Big(\arg\max_k \hat{y}_i^{fusion}(k) = y_i\Big)$$

5.5.2 Efficiency Metrics

These metrics quantify the resource requirements of each model variant, which is crucial for determining feasibility on edge devices.

- **Model Size**

  Indicates the storage footprint of the model (in MB). Reducing model size is critical for devices with limited storage and helps shorten load times[12]. FP32 models serve as baselines, while INT8 quantization typically achieves 3×–4× compression.

  Model size estimation in megabytes:

$$\text{Model Size (MB)} = \frac{n_{\text{params}} \times b}{8 \times 10^6} \tag{11}$$

  where n_{params} is the total number of parameters, and b is the number of bits per parameter (e.g., $b = 8$ for INT8 quantization).

- **Computational Complexity**

Reported as the total number of multiply–accumulate operations (MACs) per inference:

$$\text{MACs}_{conv} = H_{out} \times W_{out} \times C_{in} \times K_h \times K_w \times C_{out}$$

where H_{out} and W_{out} are the spatial dimensions of the output feature map, C_{in} and C_{out} are the input and output channels, and K_h and K_w are the kernel dimensions. The total MACs are summed over all convolutional and fully connected layers, and FLOPs are computed as $FLOPs = 2 \times MACs$.

- **RAM Usage**

  Peak memory consumption during inference includes model weights and intermediate activations:

$$\text{RAM}_{peak} = \text{Model Size} + \text{Intermediate Activations} + \text{Runtime Overhead}$$

  Model size is determined by parameter count and precision (4 bytes for FP32, 1 byte for INT8). Intermediate activations are approximated by the maximum memory required to store the largest feature map during inference.

5.5.3 Runtime Metrics

These metrics characterize the real-time performance and energy behaviour of the deployed models on the Raspberry Pi Zero 2 W. All measurements follow the methodology described in Section 7.4.3, which distinguishes between system idle power and active inference power.

- **Latency**

  Latency is measured as the wall-clock time required to perform a single inference. For each model, we execute N repeated inferences and record the per-inference runtimes t_i. The

average latency is computed as:

$$\text{Latency}_{avg} = \frac{1}{N}\sum_{i=1}^{N} t_i,$$

and the p50 and p90 values are obtained from the ordered set $\{t_i\}$ to capture median and tail-latency behaviour. Throughput is reported as

$$FPS = \frac{1}{\text{Latency}_{avg}}.$$

- **Energy Consumption**

Energy and power consumption are measured using externally recorded voltage and current values, enabling system-level characterization of inference-related energy usage. For each experimental configuration, two operating states are considered:

– *Idle power* $P_{\text{idle}} = V_{\text{idle}} I_{\text{idle}}$, measured with the system fully initialized but not performing inference;

– *Active power* $P_{\text{active}} = V_{\text{active}} I_{\text{active}}$, recorded during continuous, steady-state inference.

The net power attributable to neural network inference is defined as:

$$P_{\text{net}} = P_{\text{active}} - P_{\text{idle}}.$$

Given the measured inference throughput (FPS), the energy cost per inference (in millijoules) is computed as:

$$E_{\text{inf}} = 1000 \times \frac{P_{\text{net}}}{FPS}.$$

To quantify energy efficiency independently of absolute power draw, we further report throughput-per-watt:

$$FPS/W = \frac{FPS}{P_{\text{net}}}.$$

Battery-Life Assumptions. Battery-life estimates reported in Section 7.4.3 adopt P_{active} as a worst-case assumption corresponding to continuous inference. This conservative setting represents a stress-test scenario and provides a lower bound on achievable runtime. By isolating inference-related power consumption, these metrics enable fair comparison across models while serving as a basis for subsequent analysis under more realistic, intermittent usage patterns.

6 Implementation

6.1 Visual Backbone Model

We adopted a three-stage fine-tuning strategy to optimize lightweight CNN backbones for large-scale biodiversity classification on the iNaturalist 2021 dataset (10,000 species). This approach ensures progressive adaptation while balancing accuracy and computational cost.

6.1.1 Training Strategy

The training pipeline consisted of three stages:

- **Stage 1 – Classifier Warm-Up**

  The backbone network remains frozen while only the classification head is trained using low-resolution inputs (224×224). This quickly aligns the model with the new label space and prevents catastrophic weight updates.

- **Stage 2 – Full Network Fine-Tuning**

All layers are unfrozen and trained at the same input resolution, allowing comprehensive feature refinement. This stage captures domain-specific representations critical for fine-grained species recognition.

- **Stage 3 – High-Resolution Refinement**

  Input resolution is increased (e.g., 300×300), and only a subset of higher layers is fine-tuned. This enhances detail sensitivity for visually similar species while mitigating overfitting and excessive computation.

6.1.2 Optimization and Regularization

Models were trained using stochastic gradient descent with momentum and a cosine learning rate schedule with warm-up. Label smoothing was applied to reduce overconfidence, and RandAugment served as the primary data augmentation strategy. Progressive input resizing combined with staged unfreezing improved convergence stability and final accuracy on the long-tailed distribution of iNaturalist classes.

6.1.3 Backbone Configurations

We evaluated a diverse set of CNN backbones spanning lightweight, mid-capacity, and high-capacity architectures, including MobileNetV2, MobileNetV3-Large, EfficientNet-B0, ResNet-50, ResNet-101, ConvNeXt-Tiny, and ConvNeXt-Small. This selection enables a systematic analysis of how model capacity and architectural design interact with geographical prior integration and quantization under edge deployment constraints.

All models followed a unified training pipeline with a staged fine-tuning schedule, differing only in the number of unfrozen layers and the depth of data augmentation to accommodate architectural characteristics. After convergence, each FP32 model was validated on a held-out test set and subsequently converted to dynamically quantized variants for deployment and

evaluation on resource-constrained edge hardware.

6.1.4 Quantized Models

To examine the accuracy–efficiency trade-off on resource-constrained edge devices, we derive two deployable variants from each trained backbone, covering lightweight, mid-capacity, and high-capacity CNN architectures.

- Baseline (FP32).

  Each trained model is exported as a TensorFlow SavedModel and converted into a TFLite FP32 artifact without additional optimization. These full-precision models serve as accuracy upper bounds but incur the largest memory footprint and inference latency.

- Quantized (INT8, Dynamic Range Quantization).

  From the same SavedModel, an INT8 TFLite variant is produced using dynamic range quantization (DRQ), where model weights are quantized to INT8 while activations remain in floating-point. This approach requires no calibration dataset, substantially reduces model size, and typically improves loading time and CPU-side efficiency, making it well suited for deployment on devices such as the Raspberry Pi Zero 2 W.

Export workflow. Trained Keras models are first exported to the TensorFlow SavedModel format and subsequently converted to TFLite with the desired precision configuration. Although the conversion pipeline supports FP16 weight quantization, FP16 is not adopted in this work due to the lack of hardware acceleration on the target platform. Accordingly, all deployment and evaluation results focus on FP32 and INT8 (DRQ) variants.

6.2 Geo Prior Model

The Geo Prior model leverages spatio-temporal context to improve species classification in fine-grained biodiversity tasks. Unlike image-based backbones, this model predicts class likelihoods from latitude, longitude, and date features, enabling a probabilistic prior independent of visual cues.

6.2.1 Model Architecture

We adopt a fully connected residual network (FCNet) designed to encode geolocation and temporal signals into a low-dimensional embedding before projecting to species probabilities.

- **Input Representation:** Location (`lat`, `lon`) and date are normalized and encoded using `cos/sin` transforms, preserving angular continuity. Optional augmentation adds small perturbations to coordinates and dates during training.
- **Network Design:** A stack of dense layers with ReLU activations forms the embedding layer. Residual connections stabilize optimization and enhance feature propagation. The final layer outputs sigmoid-activated class scores across all species. An optional auxiliary branch predicts photographer identity when available, serving as an additional regularizer.
- **Negative Sampling Strategy:** Each batch combines real observations with randomly generated spatio-temporal samples. The model is penalized for assigning high probabilities to species at implausible locations or times, enforcing stronger spatial priors.

6.2.2 Training Setup

The model is trained using iNaturalist metadata parsed from JSON files containing species IDs, coordinates, and timestamps:

- **Optimization:** Optimizer: Adam with initial learning rate 0.0005, decayed exponentially by

2% per epoch. Batch size: 1024, facilitating stable estimation of class-conditional distributions. Training is conducted for 30 epochs.

- **Loss Functions:** The model is trained with several complementary loss terms. **Location-to-Object Loss** applies weighted binary cross-entropy to class predictions for observed locations. **Random Location Loss** computes the same loss for negative samples, driving probabilities toward zero for invalid coordinates. Finally, **Optional Photographer Loss** encourages consistency between user ID and species occurrence when such information is available.
- **Regularization:** Dropout in residual layers mitigates overfitting. Input augmentation introduces noise to location and date features to improve generalization.

6.2.3 Output Representation

At inference time, the Geo Prior model produces a species probability vector given a `(lat, lon, date)` triplet. This output acts as a contextual prior in the subsequent fusion mechanism, complementing visual predictions from CNN backbones. Unlike image models, Geo Prior predictions are invariant to visual ambiguity and provide strong disambiguation for species with geographically restricted ranges.

6.3 Fusion Mechanism

To enhance classification reliability under geographically constrained conditions, a fusion mechanism was implemented to integrate predictions from the visual model with those from the Geo Prior model. This process occurs during the inference phase, ensuring that spatial context complements visual evidence without altering the main CNN architecture.

6.3.1 Integration Workflow

The fusion mechanism operates after both models produce their probability distributions:

1. The visual model outputs class probabilities based on image features.

2. The Geo Prior model generates a complementary probability vector derived from encoded latitude, longitude, and date features.

3. The two probability vectors are combined using one of the following strategies:

   - **Bayesian Multiplication:** The fused probability is computed as the element-wise product of visual and prior probabilities, reinforcing classes consistent with both visual and geographic evidence.

   - **Log-Linear Combination:** A weighted approach in log space allows adjustable importance between the two sources:

$$\log \hat{P}(y) = \alpha \log P(y \mid I) + (1 - \alpha) \log P(y \mid x), \tag{12}$$

   where α determines the contribution of image-based versus prior-based predictions.

6.3.2 Practical Considerations

- **Invalid Location Handling:** For samples without valid geographic metadata, the system defaults to visual predictions to maintain robustness.

- **Numerical Stability:** When applying logarithmic operations, small constants are introduced to avoid undefined values.

- **Adjustable Weighting:** The relative contribution of the two models can be tuned via the fusion weight α, enabling scenario-specific optimization.

- **Computational Overhead:** The fusion process is lightweight, as it involves vector-level operations on model outputs, adding negligible latency compared to inference time.

6.3.3 Deployment

The fusion mechanism is applied dynamically during inference and evaluation. This design allows easy experimentation with different fusion strategies and weights without retraining any model components, ensuring flexibility for various deployment environments.

6.4 Hardware Setup

6.4.1 Target Device and Hardware Configuration

All deployment experiments were conducted on a Raspberry Pi Zero 2 W to simulate real-world resource-constrained edge devices. This platform was chosen to reflect the computational and energy limitations commonly found in low-cost educational hardware.

The Raspberry Pi Zero 2 W features a quad-core ARM Cortex-A53 CPU running at 1.0 GHz and 512 MB of RAM. The system runs Raspberry Pi OS Lite (64-bit) and boots from a 32 GB microSD card. For mobile operation and energy profiling, the device is powered by a 1200 mAh single-cell Li-ion battery through an external battery management module.

To enable image capture, on-device inference, and result visualization in a standalone form factor, the target device integrates a CSI-based camera module, a compact IPS display, and a battery management module. All major hardware components are commercially available off-the-shelf modules, ensuring ease of replication and reproducibility. A custom enclosure and physical buttons were fabricated using university-provided 3D printing facilities to integrate the components into a handheld device. As these fabrication resources were provided by the laboratory, the enclosure cost was excluded from the total hardware cost.

Table 5 summarizes the complete hardware bill of materials for the prototype AI camera device.

Table 5: Hardware bill of materials for the AI camera prototype.

Component	Model / Description	Approx. Cost (NZD)
Compute board	Raspberry Pi Zero 2 W	$35
Display module	1.3-inch IPS LCD with ST7789 driver (Waveshare)	$17
Camera module	Raspberry Pi CSI camera module (Sony IMX219, 8 MP)	$34
GPS module	L76K GPS module	$17
Power module	PiSugar 3 battery management module	$46
Storage	Micro SD card (32 GB)	$9
Enclosure	Custom 3D-printed case and buttons	–
Total		**$158**

6.4.2 Host Environment

Model training, fine-tuning, and quantization were performed on a high-performance workstation with the following specifications:

- **GPU:** NVIDIA RTX 4090 (24 GB VRAM)
- **CPU:** Intel Core Ultra 9 Processor 285K
- **RAM:** 62 GB
- **Operating System:** Linux Mint 22

The powerful GPU ensured efficient training of large-scale classification models, while ample RAM supported data preprocessing and augmentation workflows.

6.5 UI Design for Child Interaction

The user interface was designed to provide an intuitive and engaging experience for children aged 3 years and above. The design emphasizes simplicity, visual clarity, and playful interac-

tion to support outdoor learning activities.

Design Principles: The interface minimizes textual content, using large buttons and vivid icons to facilitate easy navigation. Gamification elements, such as badges and sound effects, are included to maintain engagement.

Layout: The main screen provides a live camera preview with a single capture button. After image capture, the result screen displays the predicted species name with an illustrative image. Navigation is limited to forward/backward actions to avoid complexity.

Interaction Flow: Children capture an image, wait for offline inference, and view predictions instantly. Optional audio feedback announces the species name for non-readers.

Accessibility: Large fonts and high-contrast colors ensure visibility under sunlight. Voice assistance and multi-language options enhance inclusivity.

7 Results

7.1 Accuracy and Efficiency Analysis

To enable deployment on resource-constrained edge devices such as the Raspberry Pi Zero 2 W, post-training dynamic range quantization (DRQ) was applied to all evaluated backbone models. This section analyses the impact of quantization on classification accuracy and model footprint.

Accuracy Trade-off: As shown in Table 6, DRQ introduces a small but consistent accuracy degradation across all architectures. EfficientNet-B0 experiences the largest Top-1 drop, from 73.77% to 70.84% (-2.93%), while MobileNetV3-Large and MobileNetV2 show more modest reductions of 1.99% and 0.31%, respectively. Heavier models such as ResNet-50, ResNet-101, and ConvNeXt-Small are minimally affected by quantization, with Top-1 accuracy drops below 0.1%. Top-5 accuracy follows a similar trend, remaining within a narrow margin across all models. Despite this loss, quantized models retain sufficient recognition performance for prac-

tical biodiversity applications, particularly when combined with geo-prior fusion (Section 7.3).

Table 6: Classification Accuracy on iNat2021 Validation Set.

Model Name	**Accuracy (%)**			
	FP32		**DRQ**	
	Top1	**Top5**	**Top1**	**Top5**
EfficientNet-B0	73.77	89.39	70.84	87.60
MobileNetV3-Large	71.63	87.70	69.64	86.75
MobileNetV2	68.62	86.25	68.31	86.14
ResNet-50	75.61	90.63	75.50	90.54
ResNet-101	77.85	91.89	77.80	91.85
ConvNeXt-Tiny	81.89	94.13	81.74	94.06
ConvNeXt-Small	83.33	94.81	83.29	94.78

Model Size Reduction: DRQ achieves an approximately $4\times$ reduction in model size across all architectures, as shown in Table 7. For example, EfficientNet-B0 is reduced from 64.07 MB to 16.15 MB, while MobileNetV3-Large shrinks from 47.95 MB to 12.07 MB. Similar reductions are observed for larger backbones such as ResNet-101 (240.09 MB $\rightarrow$ 60.20 MB) and ConvNeXt-Small (217.94 MB $\rightarrow$ 54.84 MB). Importantly, quantization does not change parameter count or FLOPs, indicating that compression is achieved purely through reduced numerical precision. These reductions significantly ease storage and memory constraints on ARM-based edge platforms.

Table 7: Model Complexity and Size of Baseline and Quantized Variants.

Model Name	Format	Model Size (MB)	Params	FLOPs
EfficientNet-B0	FP32	64.07	16.80M	0.81G
	DRQ	16.15	16.80M	0.81G
MobileNetV3-Large	FP32	47.95	12.57M	0.61G
	DRQ	12.07	12.57M	0.61G
MobileNetV2	FP32	57.28	15.02M	0.63G
	DRQ	14.41	15.02M	0.63G
ResNet-50	FP32	167.74	43.97M	8.20G
	DRQ	42.04	43.97M	8.20G
ResNet-101	FP32	240.09	62.94M	15.60G
	DRQ	60.20	62.94M	15.60G
ConvNeXt-Tiny	FP32	135.44	35.50M	9.00G
	DRQ	34.05	35.50M	9.00G
ConvNeXt-Small	FP32	217.94	58.13M	17.40 G
	DRQ	54.84	58.13M	17.40 G

Summary of Observations:

- **EfficientNet-B0** achieves the highest Top-1 and Top-5 accuracy in both FP32 and INT8 formats among lightweight models, but incurs higher FLOPs (0.81 G) and a larger memory footprint, making it suitable for scenarios where accuracy is prioritised over efficiency.
- **MobileNetV3-Large** offers the best balance between accuracy, model size, and computational cost. With only a modest accuracy gap relative to EfficientNet-B0 and substantially

lower FLOPs (0.61 G), it represents a strong candidate for resource-constrained edge deployment.

- **MobileNetV2** is the most compact among the evaluated lightweight models and shows minimal accuracy degradation under quantization. However, its overall accuracy remains lower than MobileNetV3-Large, limiting its effectiveness for fine-grained recognition tasks.
- **ResNet-50 and ResNet-101** maintain high accuracy and exhibit strong robustness to quantization, but their large parameter counts and high FLOPs (8.2 G and 15.6 G) make them impractical for real-time inference on ultra-low-power edge devices.
- **ConvNeXt-Small** delivers the highest overall accuracy, with negligible quantization impact, but its extremely high computational and memory requirements render it unsuitable for deployment on constrained platforms such as the Raspberry Pi Zero 2 W.

Overall, these results demonstrate that DRQ provides an effective trade-off between accuracy and deployment feasibility. When combined with appropriate backbone selection, quantization enables practical on-device biodiversity recognition without excessive loss in classification performance.

7.2 Multi-Stage Training Accuracy and Cost Analysis

Table 8: Accuracy and Training Time Across Multi-Stage Training

Model Name	**Accuracy (%)**			**Training Time (hours)**			
	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2	Stage 3	Total
EfficientNet-B0	37.43	71.04	76.11	2.84	13.72	1.02	**17.58**
MobileNetV3-Large	39.02	66.45	71.51	3.38	15.57	2.68	**21.63**
MobileNetV2	1.22	63.14	68.59	2.82	12.54	1.40	**16.76**
ResNet-50	27.12	52.59	75.45	2.78	21.97	2.30	**27.05**
ResNet-101	29.35	71.57	77.73	2.80	31.34	3.29	**37.43**
ConvNeXt-Tiny	23.07	78.26	81.95	2.77	54.02	5.60	**62.39**
ConvNeXt-Small	25.25	79.88	83.32	3.61	91.42	7.64	**102.67**

To train CNN backbones effectively on the large-scale iNaturalist 2021 dataset, a three-stage progressive training strategy was adopted (Section 6.1.1). Table 8 reports the Top-1 accuracy achieved at each training stage together with the corresponding training time.

Accuracy Progression: All models exhibit substantial accuracy gains as training progresses from Stage 1 to Stage 3, highlighting the effectiveness of the multi-stage strategy. Lightweight models benefit most from full fine-tuning in Stage 2. For example, EfficientNet-B0 improves from 37.43% in Stage 1 to 71.04% in Stage 2, indicating strong transferability of pretrained features. MobileNetV2 starts from a very low initial accuracy (1.22%), but rapidly improves once all layers are unfrozen, reaching 68.59% by Stage 3. Heavier backbones, such as ResNet-101 and ConvNeXt-Small, achieve the highest final accuracy, with ConvNeXt-Small reaching 83.32%, but require substantially longer training time.

Training Cost: Training cost varies significantly across architectures. Among lightweight models, MobileNetV2 requires the shortest total training time (16.76 hours), followed closely by EfficientNet-B0 (17.58 hours). MobileNetV3-Large requires a longer total time (21.63 hours), primarily due to extended convergence during Stage 2. Heavier models incur much higher costs: ResNet-101 requires 37.43 hours, while ConvNeXt-Small is particularly expensive, with over 100 hours of total training time dominated by Stage 2.

Accuracy–Cost Trade-off: Comparing final accuracy against total training time reveals clear trade-offs. EfficientNet-B0 achieves strong accuracy (76.11%) with relatively low training cost, representing a favourable balance between performance and efficiency. MobileNetV3-Large attains lower final accuracy while requiring longer training, suggesting that lower FLOPs do not necessarily translate to faster convergence. ConvNeXt-Small delivers the highest accuracy but at a prohibitive training cost, making it unsuitable for rapid iteration or resource-constrained

environments.

Summary: These results demonstrate that training cost is not solely determined by theoretical model complexity. Progressive multi-stage training substantially improves convergence for all architectures, but architectural design and optimization dynamics play a critical role in determining overall training efficiency. For edge-oriented applications, EfficientNet-B0 provides the most favourable accuracy–cost trade-off, while heavier models are better suited to scenarios where training resources are not a limiting factor.

7.3 Impact of Geo Prior Integration

To evaluate the effectiveness of incorporating contextual geographical information, we compared three configurations: baseline visual models without Geo Prior, Bayesian fusion of Geo Prior with image-based predictions, and our proposed Log-Linear fusion approach with adjustable weight parameter α. The results are summarized in Table 9 and Table 10.

Table 9: Accuracy improvement using Geo Prior.

Model Type	Format	Test Accuracy (%)					
		No Geo Prior	Geo Prior: Bayes	Geo Prior: Log-Linear (Ours)			
				0.2	0.3	0.4	0.5
EfficientNet-B0	FP32	73.77	82.92	83.01	**83.26**	83.20	82.95
	DRQ	70.84	80.82	81.21	**81.33**	81.22	80.84
MobileNetV3-Large	FP32	71.63	81.14	81.18	**81.53**	81.44	81.14
	DRQ	69.64	80.03	80.07	**80.45**	80.35	80.03
MobileNetV2	FP32	68.62	79.30	79.46	**79.70**	79.60	79.30
	DRQ	68.31	79.18	79.44	**79.60**	79.49	79.18
ResNet-50	FP32	75.61	84.20	84.15	**84.48**	**84.48**	84.20
	DRQ	75.50	84.01	84.02	**84.30**	84.27	84.01
ResNet-101	FP32	77.85	85.71	85.46	**85.85**	**85.85**	85.70
	DRQ	77.80	85.68	85.36	**85.80**	85.79	85.63
ConvNeXt-Tiny	FP32	81.89	88.24	87.88	88.31	**88.42**	88.24
	DRQ	81.74	88.11	87.76	88.19	**88.23**	88.11
ConvNeXt-Small	FP32	83.33	89.15	88.59	89.15	**89.24**	89.18
	DRQ	83.29	89.14	88.56	89.10	**89.20**	89.15

Table 10: Model Size Comparison of GeoPrior FCNet

Model Variant	Format	Model Size (MB)	Params	FLOPs
FCNet	FP32	11.79	3.08M	6.17M
	INT8	2.96	3.08M	6.17M

Overall Accuracy Improvement: Incorporating Geo Prior information consistently improves classification accuracy across all evaluated backbones and numerical formats. As shown in Table 9, Bayesian fusion yields substantial gains over purely visual models, with improvements of 8–10 percentage points in Top-1 accuracy. For example, EfficientNet-B0 (FP32) improves from 73.77% without Geo Prior to 82.92% with Bayesian fusion, demonstrating that geographical context provides strong complementary cues for fine-grained species recognition.

Log-Linear Fusion vs. Bayesian Fusion: The proposed Log-Linear fusion further improves upon Bayesian fusion by introducing a tunable weighting parameter α to balance visual and contextual predictions. Across most models, optimal performance is achieved with α in the range of 0.3–0.4. EfficientNet-B0 (FP32) reaches a peak accuracy of 83.26% at $\alpha = 0.3$, exceeding Bayesian fusion by 0.34 percentage points. Although the absolute gains are modest, the Log-Linear formulation provides greater robustness to noisy or incomplete metadata by preventing excessive reliance on geographic priors.

Effect on Quantized Models: Quantized (DRQ) models exhibit similar relative improvements from Geo Prior integration. For instance, EfficientNet-B0 (DRQ) improves from 70.84% to 80.82% with Bayesian fusion and further to 81.33% using Log-Linear fusion. Comparable trends are observed for MobileNetV2 and MobileNetV3-Large, indicating that the proposed fusion strategy remains effective under reduced numerical precision and is well suited for resource-constrained, offline deployment.

Efficiency of Geo Prior FCNet: The Geo Prior network is highly lightweight. As shown in Table 10, FCNet requires only 6.17M FLOPs per inference, which is negligible compared to the computational cost of CNN backbones such as EfficientNet-B0. INT8 quantization reduces the model size from 11.79 MB to 2.96 MB (**75%** reduction) without affecting accuracy, ensuring

minimal memory and latency overhead on edge devices.

Summary: Overall, these results demonstrate that Geo Prior integration provides a consistent and architecture-agnostic accuracy improvement. The proposed Log-Linear fusion offers flexible control over the contribution of contextual information while maintaining robustness under quantization. Combined with the extremely low computational cost of FCNet, this approach enables accurate, fully offline biodiversity recognition suitable for low-power educational and citizen-science applications.

7.4 Efficiency Analysis

7.4.1 Inference Latency and Throughput

Inference latency and throughput were measured on a Raspberry Pi Zero 2 W (Cortex-A53, 1.0 GHz, 4 cores) using TensorFlow Lite with batch size set to one. Both single-threaded and four-threaded executions were evaluated to assess the effect of CPU parallelism. Table 11 reports average latency, p90 latency, and throughput (FPS) for FP32 and dynamic range quantized (DRQ) models.

Multi-threading consistently improves throughput across all architectures, achieving speedups of approximately 2.5–3.0$\times$ compared with single-thread execution. Among lightweight backbones, MobileNetV3-Large and MobileNetV2 achieve the lowest latency under FP32 execution, reaching 116.3 ms (8.60 FPS) and 114.2 ms (8.76 FPS) respectively with four threads. EfficientNet-B0 incurs significantly higher latency (753.1 ms, 1.33 FPS), reflecting its higher computational complexity and making it more suitable for offline inference.

The impact of DRQ is model-dependent. For EfficientNet-B0, DRQ reduces latency and improves throughput, while for MobileNetV2 and MobileNetV3-Large it introduces additional overhead, resulting in slower inference than FP32. This suggests that extremely lightweight

models are dominated by kernel dispatch and quantization overhead, which can outweigh the computational savings of quantized weights.

Heavier architectures such as ResNet-50, ResNet-101, and ConvNeXt-Small remain impractical for real-time inference on the Raspberry Pi Zero 2 W, even under DRQ, due to their high latency. Overall, MobileNetV3-Large in FP32 provides the most practical balance between latency and throughput for real-time edge deployment on this platform.

Table 11: Inference latency, FPS, and load time on Raspberry Pi Zero 2 W.

Model Name	Format	Threads	Avg Latency (ms)	p90 (ms)	FPS
EfficientNet-B0	FP32	1	903.95	869.74	1.11
	FP32	4	753.08	700.12	1.33
	DRQ	1	626.26	627.06	1.60
	DRQ	4	461.50	499.07	2.17
MobileNetV3-Large	FP32	1	224.87	179.40	4.45
	FP32	4	116.28	91.29	8.60
	DRQ	1	239.81	237.09	4.17
	DRQ	4	205.18	244.22	4.87
MobileNetV2	FP32	1	221.25	212.60	4.52
	FP32	4	114.18	108.81	8.76
	DRQ	1	311.95	275.62	3.21
	DRQ	4	182.61	192.22	5.48
ResNet-50	DRQ	1	1731.15	1717.97	0.58
	DRQ	4	653.47	661.67	1.53
ResNet-101	DRQ	1	3213.36	3143.91	0.31
	DRQ	4	1192.17	1141.10	0.84
ConvNeXt-Tiny	DRQ	1	6484.90	6427.48	0.15
	DRQ	4	5243.91	5251.54	0.19
ConvNeXt-Small	DRQ	1	11302.33	13253.08	0.09
	DRQ	4	10884.12	12836.39	0.09

7.4.2 Energy Efficiency

Energy efficiency was evaluated using the methodology described in Section 5.5.3 by separating the system's idle power from the additional power consumed during inference. The Raspberry Pi Zero 2 W exhibited an approximately constant idle power of 1.0 W, while continuous inference consumed 2.5 W, resulting in a net inference power of 1.5 W. Based on the measured

throughput under the four-thread configuration, energy per inference and throughput-per-watt were derived accordingly.

As shown in Table 12, the impact of dynamic range quantization (DRQ) on energy efficiency is strongly model-dependent. For EfficientNet-B0, DRQ improves both energy-per-inference and FPS/W, reflecting reduced arithmetic cost in a moderately compute-bound model. In contrast, for lightweight architectures such as MobileNetV2 and MobileNetV3-Large, DRQ leads to higher energy consumption per inference and lower throughput-per-watt compared with FP32, indicating that quantization overhead dominates the overall execution cost.

Heavier backbones, including ResNet-50, ResNet-101, and ConvNeXt-Small, exhibit very low throughput-per-watt even under DRQ, confirming that their computational demands exceed the energy budget of ultra-low-power edge devices. Overall, these results demonstrate that energy efficiency on CPU-only edge platforms is primarily determined by the interaction between model architecture and quantization strategy, rather than by numerical precision alone.

Table 12: Energy efficiency metrics for FP32 and DRQ inference on the Raspberry Pi Zero 2 W (4-thread configuration, $P_{\text{net}} = 1.5\,\text{W}$).

Model	**Energy / inference (mJ)**		**FPS/W**	
	FP32	DRQ	FP32	DRQ
MobileNetV2	171.2	273.7	5.84	3.65
MobileNetV3-Large	174.4	308.0	5.73	3.25
EfficientNet-B0	1127.8	691.2	0.89	1.45
ResNet-50	n/a	980.4	n/a	1.02
ResNet-101	n/a	1785.7	n/a	0.56
ConvNeXt-Tiny	n/a	7894.7	n/a	0.13
ConvNeXt-Small	n/a	16666.7	n/a	0.06

7.4.3 Battery Life

Table 13: Battery life under a 30-second capture–infer–display cycle.

Model Name	Format	Battery Life (h)
EfficientNet-B0	FP32	4.68
	DRQ	4.55
MobileNetV3-Large	FP32	4.66
	DRQ	4.62
MobileNetV2	FP32	4.65
	DRQ	4.60
ResNet-50	DRQ	4.40
ResNet-101	DRQ	4.25
ConvNeXt-Tiny	DRQ	3.93
ConvNeXt-Small	DRQ	3.70

Table 13 shows that, under a 30-second capture–infer–display cycle, the measured battery lifetime spans a relatively narrow range from approximately 3.70,h to 4.68,h across all evaluated models. Lightweight CNN architectures such as EfficientNet-B0 and MobileNet variants achieve the longest battery lifetimes, consistently exceeding 4.6,h, whereas reminder heavier architectures, including ConvNeXt-Tiny and ConvNeXt-Small, exhibit shorter lifetimes below 4.0,h.

For models evaluated in both FP32 and DRQ formats, the impact of quantization on end-to-end battery life is limited. The maximum observed difference between FP32 and DRQ configurations is 0.13,h (EfficientNet-B0), corresponding to less than 3MobileNetV2 and MobileNetV3-Large. These small variations fall within the expected measurement variability for low-duty-

cycle operation and indicate that numerical precision has only a marginal effect on overall battery lifetime in this setting.

Among DRQ-only models, a gradual decrease in battery life is observed as model complexity increases. ResNet-50 (4.40,h) outperforms ResNet-101 (4.25,h), while ConvNeXt-based models incur a more pronounced reduction, with ConvNeXt-Tiny and ConvNeXt-Small reaching 3.93,h and 3.70,h, respectively. However, even in these cases, the absolute difference relative to the most energy-efficient models remains within approximately one hour.

Overall, these results confirm that under intermittent inference workloads, where inference events are sparse relative to idle periods, total battery lifetime is dominated by baseline system power, display usage, and power-conversion losses. Consequently, differences in model architecture and numerical precision manifest only as secondary effects at the system level. In such scenarios, the advantages of model compression and quantization are more effectively reflected in per-inference latency and energy measurements rather than in end-to-end battery lifetime alone.

7.5 Comparative Evaluation

Table 14 compares the proposed system with existing biodiversity recognition tools across accuracy, species coverage, cost, data storage strategy, and open-source availability.

Unlike cloud-dependent solutions such as Google Lens, Pl@ntNet, and Seek by iNaturalist, the proposed system offers fully offline operation with local data storage, ensuring privacy and reliability even in connectivity-constrained environments. While Seek by iNaturalist provides partial offline support, it still requires cloud synchronization for extended features, limiting usability in remote settings.

In terms of species coverage, Google Lens recognizes millions of species, whereas our system supports approximately 10,000 classes—sufficient for educational and citizen science applica-

tions—at a significantly lower cost (NZD 160) compared to proprietary hardware like AX Visio (NZD 8,500).

A further advantage is its open-source nature, which ensures transparency, reproducibility, and scalability, whereas other listed solutions remain closed. Additionally, the system features standalone, child-friendly hardware, eliminating smartphone dependency to provide a safe and distraction-free outdoor learning experience. Collectively, these characteristics position the proposed system as a cost-effective, privacy-preserving, and educationally tailored alternative to existing biodiversity recognition tools.

Table 14: Comparison of Biodiversity Recognition Solutions.

Solution	**Acc. (%)**	**Species**	**Price (NZD)**	**Local Data Storage**	**Open Source**
Proposed System	81-89	10,000	160	✓	✓
AX Visio	80–90	9,000	8,500	✓	✗
Seek by iNaturalist	80–85	∼25,000	Free	✗	✗
Pl@ntNet	80–85	∼30,000	Free	✗	✗
Google Lens	85–90	Millions+	Free	✗	✗

8 Discussion

8.1 Key Findings

This study demonstrates that accurate biodiversity classification can be achieved on low-cost, resource-constrained edge devices without reliance on cloud-based computation. By combining lightweight visual backbones with geographical priors, the proposed system significantly improves robustness in fine-grained species recognition, particularly for visually similar taxa with distinct spatial distributions.

Across all evaluated lightweight models, the proposed log-linear geo-prior fusion consistently

outperforms conventional Bayesian fusion by enabling flexible weighting between visual and contextual information. This design improves stability under both FP32 and INT8 settings and reduces misclassification in long-tail categories. Using EfficientNet-B0 as the primary deployable backbone, the system achieves approximately 83% Top-1 accuracy on the iNaturalist 2021 dataset, corresponding to an improvement of over 9 percentage points compared to vision-only baselines.

Crucially, these accuracy gains are obtained while maintaining fully offline operation on a Raspberry Pi Zero 2 W. With a total hardware cost below NZD 160, the proposed system demonstrates the feasibility of an affordable, portable, and privacy-preserving AI solution for outdoor education and citizen-science applications in connectivity-limited environments.

8.2 Advantages of Our Method

The proposed system offers several advantages over existing cloud-dependent and mobile-only biodiversity recognition solutions.

- **Fully Offline Operation:** The system operates entirely offline, enabling reliable use in remote or outdoor environments where network connectivity is limited or unavailable. This contrasts with cloud-based services, which require continuous internet access and may fail in field-based educational settings.

- **Resource-Efficient Deployment:** By combining lightweight CNN backbones with post-training quantization, the proposed approach achieves competitive classification accuracy while operating within the memory and compute constraints of low-cost edge hardware. This allows deployment on devices with as little as 512 MB of RAM, without reliance on specialized accelerators.

- **Context-Aware Recognition:** The integration of geographical priors via log-linear fusion

enhances discrimination between visually similar species that exhibit distinct spatial or habitat-specific distributions. This context-aware design improves robustness in fine-grained recognition tasks common in biodiversity monitoring.

- **Education-Oriented System Design:** The standalone hardware configuration avoids dependence on smartphones, reducing distractions and privacy concerns while providing a simple and controlled interaction model. This design is particularly well suited for child-centered educational applications and guided outdoor learning activities.

8.3 Limitations of Our Method

Despite strong performance within the targeted edge deployment setting, several limitations of the proposed framework should be acknowledged.

- **Fixed global fusion weight.** The log-linear geo-prior fusion employs a globally fixed weighting parameter α to balance visual predictions and geographical priors. While this design simplifies deployment and reduces computational overhead on resource-constrained edge devices, it does not adapt to sample-level variability in visual quality or metadata reliability. As a result, the fusion strategy may be suboptimal for images with ambiguous visual cues or noisy contextual information.

- **Dependence on metadata availability and quality.** The effectiveness of geo-prior integration relies on the availability and accuracy of spatial and temporal metadata. When such information is missing, noisy, or intentionally disabled for privacy reasons, the system degrades to an image-only classifier. The current framework does not explicitly model uncertainty in metadata inputs.

- **Limited evaluation scope.** Experimental evaluation is conducted exclusively on the iNaturalist 2021 dataset. Although this dataset is large-scale and widely used, the absence of

cross-dataset or cross-region evaluation limits conclusions regarding the generalizability of the proposed approach to other ecological domains or geographic distributions.

- **Edge hardware constraints.** Deployment on ultra-low-power hardware such as the Raspberry Pi Zero 2 W imposes strict constraints on model capacity, latency, and memory usage. These constraints limit the feasibility of higher-capacity backbones, sample-adaptive fusion mechanisms, or more complex multimodal architectures.

- **Simplified system-level energy evaluation.** Due to time and experimental constraints, energy consumption and battery-life measurements were conducted without integrating a real-time GPS module. Instead, spatial and temporal inputs were obtained from annotated dataset metadata. Consequently, the reported energy results do not account for the additional power overhead of continuous sensor operation and may underestimate total system-level energy consumption in real-world deployments.

8.4 Future Research Directions

Building upon the current framework, several directions could further enhance system robustness, adaptability, and real-world applicability.

- **Sample-adaptive fusion strategies.** Future work could explore sample-adaptive or confidence-aware fusion mechanisms that dynamically adjust the weighting between visual predictions and contextual priors based on image quality or metadata reliability, while remaining compatible with edge resource constraints.

- **Metadata uncertainty modeling.** Incorporating uncertainty-aware modeling or reliability estimation for spatial and temporal metadata could improve robustness under missing or noisy contextual inputs and prevent over-reliance on unreliable priors.

- **Cross-dataset and cross-region evaluation.** Evaluating the proposed framework on additional biodiversity datasets or region-specific benchmarks would provide deeper insight into its generalization capability across varying species distributions and environmental conditions.
- **Hardware-aware optimization.** Exploring hardware-specific acceleration, neural architecture search (NAS), or lightweight pruning techniques could further reduce latency and energy consumption, enabling more adaptive models on next-generation edge platforms with integrated NPUs.
- **End-to-end system-level profiling.** Integrating real-time GPS and other sensors into the deployed system would enable comprehensive end-to-end energy and battery-life evaluation, providing more accurate characterization of real-world operational costs.
- **Extended interaction and educational features.** Future iterations could incorporate richer user interaction mechanisms, such as audio feedback, multilingual support, or adaptive educational content, to enhance usability in child-oriented and citizen-science applications.

9 Conclusion

This research successfully presents a fully offline biodiversity recognition system designed to operate on low-cost edge devices, while addressing crucial limitations such as privacy concerns and connectivity constraints. The proposed log-linear geo-prior fusion method improves classification performance, achieving up to 89.24% Top-1 accuracy with high-capacity models like ConvNeXt-Small, and 83.26

Through the integration of lightweight CNN architectures and dynamic range quantization (DRQ), we demonstrate a significant reduction in model size and energy consumption, making the system practical for deployment in resource-constrained environments. By combining

geographical context with CNN predictions, the system improves accuracy, particularly for visually similar species, and offers a scalable solution for biodiversity monitoring in remote or offline settings.

In comparison with existing cloud-based solutions such as Google Lens and Seek, our system provides full offline functionality, ensures user privacy, and significantly reduces hardware costs, achieving a 50× reduction compared to specialized commercial devices like AX Visio. This approach showcases the potential of edge AI in making biodiversity recognition accessible and cost-effective for educational and citizen science applications.

Future research should explore hybrid cloud-edge models to further enhance accuracy, address class imbalance with advanced techniques such as few-shot learning, and optimize edge device performance for larger datasets and more demanding environments. Additionally, expanding user interaction features such as multilingual support and gamified learning could improve the system's educational value and engagement.

10 Acknowledgments

Generative AI tools were used in a limited and supportive capacity during the preparation of this report, in accordance with course policy.

The AI system used was **ChatGPT (OpenAI)**.

AI assistance was applied to the following aspects:

- Language refinement and grammar correction across multiple sections of the report
- Improving clarity and readability of technical explanations
- Minor structural suggestions for organizing written sections
- Generating non-research decorative graphical elements used in the presentation slides

AI tools were **not used** for:

- Developing research ideas
- Designing model architectures or algorithms
- Making experimental decisions
- Generating code implementations
- Producing experimental results or analysis

All technical contributions, system design, model implementation, data processing procedures, and analytical conclusions were independently developed by the author.

All AI-assisted outputs were carefully reviewed, edited, and verified by the author, who takes full responsibility for the originality, accuracy, and academic integrity of this work.

References

[1] Oisin Mac Aodha, Elijah Cole, and Pietro Perona. *Presence-Only Geographical Priors for Fine-Grained Image Classification*. 2019. arXiv: 1906.05272 [cs.CV]. URL: https://arxiv.org/abs/1906.05272.

[2] Colby R. Banbury et al. *Benchmarking TinyML Systems: Challenges and Direction*. 2021. arXiv: 2003.04821 [cs.PF]. URL: https://arxiv.org/abs/2003.04821.

[3] Thomas Berg et al. "Birdsnap: Large-Scale Fine-Grained Visual Categorization of Birds". In: June 2014, pp. 2019–2026. DOI: 10.1109/CVPR.2014.259.

[4] Yu Cheng et al. "A Survey of Model Compression and Acceleration for Deep Neural Networks". In: (Oct. 2017). DOI: 10.48550/arXiv.1710.09282.

[5] Yu Cheng et al. *A Survey of Model Compression and Acceleration for Deep Neural Networks*. 2020. arXiv: 1710.09282 [cs.LG]. URL: https://arxiv.org/abs/1710.09282.

[6] Rocco De Marco et al. "Real-Time Dolphin Whistle Detection on Raspberry Pi Zero 2 W with a TFLite Convolutional Neural Network". In: *Robotics* 14 (May 2025), p. 67. DOI: 10.3390/robotics14050067.

[7] Mohannad Elhamod et al. "Hierarchy-guided Neural Networks for Species Classification". In: *Methods in Ecology and Evolution* 13 (Nov. 2021). DOI: 10.1111/2041-210X.13768.

[8] Song Han, Huizi Mao, and William J. Dally. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. 2016. arXiv: 1510.00149 [cs.CV]. URL: https://arxiv.org/abs/1510.00149.

[9] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[10] Grant Van Horn et al. *The iNaturalist Species Classification and Detection Dataset*. 2018. arXiv: 1707.06642 [cs.CV]. URL: https://arxiv.org/abs/1707.06642.

[11] Andrew Howard et al. *Searching for MobileNetV3*. 2019. arXiv: 1905.02244 [cs.CV]. URL: https://arxiv.org/abs/1905.02244.

[12] Benoit Jacob et al. "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference". In: (Dec. 2017). DOI: 10.48550/arXiv.1712.05877.

[13] Lingjie Jiang et al. "JujubeNet: A high-precision lightweight jujube surface defect classification network with an attention mechanism". In: *Frontiers in Plant Science* 13 (Jan. 2023). DOI: 10.3389/fpls.2022.1108437.

[14] ND Lane and P Georgiev. "Can deep learning revolutionize mobile sensing?" In: Feb. 2015.

[15] Yanyu Li et al. *EfficientFormer: Vision Transformers at MobileNet Speed*. 2022. arXiv: 2206.01191 [cs.CV]. URL: https://arxiv.org/abs/2206.01191.

[16] Ji Lin et al. *AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration*. 2024. arXiv: 2306.00978 [cs.CL]. URL: https://arxiv.org/abs/2306.00978.

[17] Ji Lin et al. *MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning*. 2024. arXiv: 2110.15352 [cs.CV]. URL: https://arxiv.org/abs/2110.15352.

[18] Zhuang Liu et al. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV]. URL: https://arxiv.org/abs/2201.03545.

[19] Paweł Nadachowski et al. "Classification of Glacial and Fluvioglacial Landforms by Convolutional Neural Networks Using a Digital Elevation Model". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* PP (Jan. 2024), pp. 1–17. DOI: 10.1109/JSTARS.2024.3470253.

[20] iNaturalist Open Source Software. *iNaturalist 2021 Competition*. https://github.com/visipedia/inat_comp/tree/master/2021. [Online; accessed 8-July-2025]. 2021.

[21] Konstantinos I. Roumeliotis et al. *Plant Disease Detection through Multimodal Large Language Models and Convolutional Neural Networks*. 2025. arXiv: 2504.20419 [cs.CV]. URL: https://arxiv.org/abs/2504.20419.

[22] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV]. URL: https://arxiv.org/abs/1409.0575.

[23] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV]. URL: https://arxiv.org/abs/1801.04381.

[24] Fereshteh Shahoveisi et al. "Application of image processing and transfer learning for the detection of rust disease". In: *Scientific Reports* 13 (Mar. 2023). DOI: 10.1038/s41598-023-31942-9.

[25] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG]. URL: https://arxiv.org/abs/1905.11946.

[26] TensorFlow. *TensorFlow Lite Model Optimization Toolkit*. https://www.tensorflow.org/model_optimization. [Online; accessed 8-July-2025]. 2024.

[27] Pavan Kumar Anasosalu Vasu et al. *MobileOne: An Improved One millisecond Mobile Backbone*. 2023. arXiv: 2206.04040 [cs.CV]. URL: https://arxiv.org/abs/2206.04040.

[28] Jana Wäldchen and Patrick Mäder. "Plant Species Identification Using Computer Vision Techniques: A Systematic Literature Review". In: *Archives of Computational Methods in Engineering* 25 (Apr. 2018), pp. 507–543. DOI: 10.1007/s11831-016-9206-z.

[29] Cheng Wang et al. "The Security and Privacy of Mobile Edge Computing: An Artificial Intelligence Perspective". In: *IEEE Internet of Things Journal* PP (Dec. 2023), pp. 1–1. DOI: 10.1109/JIOT.2023.3304318.

[30] Kan Wu et al. *TinyViT: Fast Pretraining Distillation for Small Vision Transformers*. 2022. arXiv: 2207.10666 [cs.CV]. URL: https://arxiv.org/abs/2207.10666.

[31] Guangxuan Xiao et al. *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models*. 2024. arXiv: 2211.10438 [cs.CL]. URL: https://arxiv.org/abs/2211.10438.

[32] Lingfeng Yang et al. *Dynamic MLP for Fine-Grained Image Classification by Leveraging Geographical and Temporal Information*. 2022. arXiv: 2203.03253 [cs.CV]. URL: https://arxiv.org/abs/2203.03253.

[33] Zhewei Yao et al. *ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers*. 2022. arXiv: 2206.01861 [cs.CL]. URL: https://arxiv.org/abs/2206.01861.

[34] Zhihang Yuan et al. *PTQ4ViT: Post-training quantization for vision transformers with twin uniform quantization*. 2024. arXiv: 2111.12293 [cs.CV]. URL: https://arxiv.org/abs/2111.12293.