Optimization of Deep Learning Models for Bird Identification in the Pantanal Using Embedded Devices

Isis Milena Daron[1], **Julia Milioranza Gomes**[1], **Thiago Meirelles Ventura**[1,2]

[1]Instituto de Computação - Universidade Federal de Mato Grosso (UFMT)

[2]Instituto Nacional de Pesquisa do Pantanal, Cuiabá, MT, Brasil

{isis.daron, julia.gomes1}@sou.ufmt.br, thiago@ic.ufmt.br

Abstract. *Autonomous biodiversity monitoring in remote ecosystems such as the Pantanal requires on-device inference under strict resource constraints. To this end, deep learning models must demand reduced memory footprint and computational cost compared to standard architectures. This work evaluates post-training quantization, knowledge distillation, and pruning individually and in hybrid combinations applied to BirdNET v2.4 adapted for Pantanal bird species. The best-performing configuration achieved 96.72% accuracy at 3.63× compression, while hybrid pipelines reached 11.55× at 87.99%, with KD-first sequences outperforming pruning-first by up to 7.5 percentage points. The identified profiles support on-device bioacoustic inference without cloud dependency.*

1. Introduction

Covering approximately 160,000 km² across Brazil, Bolivia, and Paraguay, the Pantanal is the world's largest wetland and harbors exceptional biodiversity and requires high-priority environmental protection [Junk et al. 2006, Alho and Sabino 2011]. Passive acoustic monitoring combined with deep learning classifiers has emerged as a scalable approach to automated avian biodiversity assessment [Stowell 2021, Kahl et al. 2021]. Yet state-of-the-art recognition models are computationally demanding, representing a fundamental barrier to field deployment [Zhao et al. 2022]. Most AI-based wildlife monitoring systems depend on cloud infrastructure or high-performance computing [Mohaimenuzzaman et al. 2023], making continuous data transmission infeasible in remote areas with limited connectivity. Autonomous, on-device inference is therefore a prerequisite for practical field deployment.

Model compression techniques, including quantization, knowledge distillation, and pruning, reduce model complexity without retraining from scratch [Gholami et al. 2021]. Although their individual merits are documented in computer vision and speech domains, no prior work has evaluated these techniques, alone or in combination, for bioacoustic models tailored to this specific regional fauna. This work addresses that gap by applying individual and hybrid optimization configurations to BirdNET v2.4 [Kahl et al. 2025], a bird species classification model adapted for Pantanal avifauna, with deployment on constrained embedded platforms as the guiding objective.

The main contributions are: (i) a comparative evaluation of quantization, knowledge distillation, and pruning in individual and hybrid configurations, with an empirical characterization of how technique application order affects the accuracy-compression trade-off; and (ii) two deployment-oriented profiles for remote ecological monitoring under contrasting resource budgets.

2. Background and Related Work

Developed by the Cornell Lab of Ornithology, BirdNET [Kahl et al. 2021] is a deep convolutional neural network for automated bird species identification from audio recordings. Capable of classifying over 6,000 species worldwide in its v2.4 release [Wood and Kahl 2024], its strong recognition performance has established it as one of the leading tools for acoustic biodiversity monitoring in both research and field deployment contexts. Deploying such models on resource-constrained devices, however, requires reducing their computational footprint, an objective that can be approached through compression strategies, of which this work evaluates three widely adopted ones.

Post-training quantization applies compression directly to a trained model by mapping floating-point weights and activations to lower-precision numerical formats, reducing memory footprint and computational cost without additional training iterations [Krishnamoorthi 2018, Gholami et al. 2021], with common schemes including INT8 integer arithmetic and Float16 reduced-precision formats [Jacob et al. 2018, Krishnamoorthi 2018].

In turn, knowledge distillation [Hinton et al. 2015] trains a compact student network to reproduce the behavior of a larger teacher by minimizing the divergence between their output distributions. Temperature scaling applied to the softmax generates soft probability distributions that reveal inter-class similarities beyond what hard labels convey, providing richer supervisory signal [Gou et al. 2021].

Finally, pruning reduces model complexity by eliminating redundant parameters, building on early work by [LeCun et al. 1989] that demonstrated removing connections decreases storage requirements and mitigates overfitting. While structured pruning removes entire filters or layers to produce dense networks, unstructured pruning operates at the individual weight level, creating sparse representations while preserving the original layer structure, with magnitude-based criteria eliminating weights with the smallest absolute values [Han et al. 2015].

These strategies have been explored in the context of bird recognition. Höchst et al. [Höchst et al. 2022] deployed an EfficientNet-based CNN on a NVIDIA Jetson Nano using TensorRT optimization with FP16 quantization, reporting up to 95.2% mean average precision on European forest soundscapes and demonstrating practical feasibility of on-device species identification without cloud connectivity. Huang et al. [Huang et al. 2024] introduced TinyChirp, deploying an INT8-quantized bird detector on a microcontroller with only kilobytes of available memory, extending field data collection intervals from two to eighteen weeks. Silva et al. [Silva et al. 2026] implemented a CNN-RNN system on a SoC-FPGA with mixed-precision quantization-aware training, reaching 87.4% accuracy at only 1.4 W. Wang et al. [Wang et al. 2023] applied decoupled knowledge distillation with attention-guided data augmentation for fine-grained bird image classification, achieving 87.6% accuracy with 67% fewer parameters than the teacher.

Collectively, these works confirm that individual optimization techniques can meet embedded deployment constraints in specific scenarios. However, none addresses bioacoustic models adapted to Pantanal bird species, nor evaluates how combining these techniques sequentially, and in varying orders, affects the accuracy-compression trade-off in this domain. The present work targets precisely that gap.

3. Methodology

The experimental design follows a progressive structure: baseline establishment through fine-tuning, individual technique assessment, hybrid combination evaluation, and consolidated analysis of deployment-suitable configurations.

3.1. Dataset

The dataset comprises 6,415 audio segments representing 10 bird species (Table 1) native to the Pantanal wetland ecosystem, obtained through the Computational Bioacoustics Research Unit (CO.BRA) at the Federal University of Mato Grosso from recordings sourced at Xeno-Canto[1]. The CO.BRA research group segmented original recordings into standardized 3-second clips at 48,000 Hz. All segments were subsequently loaded as mono-channel audio and amplitude-normalized via peak normalization using the librosa library [McFee et al. 2015]. No additional data augmentation was applied.

Table 1. Target bird species from the Pantanal wetland ecosystem.

Scientific Name	Common Name
Anodorhynchus hyacinthinus	Hyacinth Macaw
Campylorhynchus turdinus	Thrush-like Wren
Cantorchilus guarayanus	Fawn-breasted Wren
Cercomacra melanaria	Mato Grosso Antbird
Crypturellus undulatus	Undulated Tinamou
Furnarius rufus	Rufous Hornero
Ortalis canicollis	Chaco Chachalaca
Saltator coerulescens	Blue-grey Saltator
Synallaxis albilora	White-lored Spinetail
Theristicus caerulescens	Plumbeous Ibis

Stratified splitting was applied at the original audio file level rather than at the segment level. Since the CO.BRA preprocessing pipeline segments each continuous recording into consecutive 3-second clips, adjacent segments share background acoustic conditions and recording characteristics, introducing temporal autocorrelation. Splitting at the segment level would therefore allow correlated samples to appear in both training and test subsets, artificially inflating evaluation metrics. File-level stratification ensures complete independence between subsets, as all segments derived from a single recording are assigned exclusively to one partition. This procedure yielded 260 training files, 62 validation files, and 63 test files across 385 unique recordings, corresponding to a 70%/15%/15% split. The segment-level distribution comprised 3,905 training, 1,411 validation, and 1,099 test segments across the 6,415 total segments, with segment-level proportions differing slightly from file-level proportions due to variable recording durations.

3.2. Base Model and Baseline Development

BirdNET v2.4 [Kahl et al. 2025], developed by the Cornell Lab of Ornithology and Chemnitz University of Technology, was selected as the base model. It comprises 227 layers, approximately 49 MB of storage, accepts 144,000-sample raw waveform inputs (3

[1] www.xeno-canto.org

seconds at 48 kHz) internally converted to mel-spectrograms, and classifies 6,522 species. The pre-trained model was obtained in both Keras and TFLite format from the official repository.

Three fine-tuning strategies were evaluated to adapt the model to the 10-class target task, all replacing the original output layer with a 10-class dense layer: partial fine-tuning (last 20 layers unfrozen), full fine-tuning (all layers except BatchNormalization), and feature extraction (backbone frozen, classification head appended atop the preserved original dense layer).

Experiments were conducted using TensorFlow 2.16.1 [Abadi et al. 2015], with TensorFlow Lite (TFLite) for model conversion and the TensorFlow Model Optimization Toolkit (tfmot) for pruning, running on Python 3.11.9 under Windows with a fixed random seed (42) and batch size 32 throughout.

The three strategies were trained with the Adam optimizer, categorical cross-entropy loss, batch size 32, and up to 50 epochs, with early stopping (patience: 15) and learning rate decay on plateau (patience: 7, factor: 0.5, minimum: 1×10^{-7}). The learning rate was set to 1×10^{-5} for both fine-tuning approaches and 1×10^{-3} for feature extraction. These training settings are referred to as the *standard training configuration* and reused in subsequent optimization stages unless stated otherwise.

3.3. Optimization Techniques

The selection of optimization techniques evaluated in this work was informed by a systematic literature review of post-training optimization for embedded deep learning [Daron et al. 2026], which synthesized 32 studies meeting predefined quality criteria under a structured protocol. That review identified quantization as the most prevalent technique (62.5% of studies), followed by knowledge distillation (50.0%) and pruning (46.9%), with 43.8% of studies employing hybrid combinations. These findings motivated the three techniques investigated here and the hybrid strategies explored in Section 3.4.

3.3.1. Quantization

Three post-training quantization methods were implemented via the TFLite converter interface, covering the main precision reduction schemes available for deployment: INT8 full quantization, INT8 weight-only quantization, and Float16 quantization. Table 2 summarizes the numerical format applied to weights and activations, the calibration requirement, and the theoretical compression factor for each method. For INT8 full quantization, a representative calibration dataset of 1,000 audio segments was constructed through stratified sampling (100 samples per species) from the training set.

Table 2. Quantization methods and configurations.

Method	Weights	Activations	Calibration	Compression
INT8 Full	INT8	INT8 (static)	Required	4×
INT8 Weight-Only	INT8	Dynamic[a]	Not required	4×
Float16	Float16	Float32 (CPU)	Not required	2×

[a] Activations are handled dynamically at inference time and remain in floating point.

3.3.2. Knowledge Distillation

A compact student network was trained to replicate the soft probability distributions produced by the fine-tuned baseline (teacher). Following VGGNet principles [Simonyan and Zisserman 2015], the architecture comprises three convolutional blocks with 64, 128, and 256 channels, each containing three consecutive 3×3 convolution layers followed by batch normalization and ReLU activation, with dropout rates of 0.25, 0.3, and 0.4 applied after each pooling stage. A 128-unit fully connected layer precedes the 10-class output, yielding approximately 2.91 million parameters (38.4% of the teacher's count). The teacher's mel-spectrogram preprocessing layers were reconstructed in the student to ensure identical time-frequency representations.

Training employed a combined loss of categorical cross-entropy and Kullback-Leibler divergence, evaluating three temperature-weighting configurations: T=2.0, α=0.5; T=4.0, α=0.7; and T=6.0, α=0.5. Models were trained for up to 100 epochs following the standard training configuration.

3.3.3. Pruning

Magnitude-based unstructured pruning was applied via the TensorFlow Model Optimization Toolkit (tfmot) through a polynomial decay schedule incrementally increasing sparsity from 0% to target levels of 40%, 50%, or 60% over 15 epochs, excluding `MelSpecLayerSimple`, `DepthwiseConv2D`, and `InputLayer` instances. Training followed the standard configuration defined in Section 3.2.

The pruning procedure started from the fine-tuned baseline, applied magnitude-based pruning with polynomial decay for 15 epochs, stripped pruning wrappers via `strip_pruning`, and converted the resulting model to TFLite. Since unstructured pruning retains the dense tensor layout, TFLite serialization without additional flags does not reduce file size. Each pruned model was therefore converted both with and without the `tf.lite.Optimize.EXPERIMENTAL_SPARSITY` flag[2], which enables sparse tensor encoding [Li and Ablavatski 2021]. Both versions were evaluated to determine the most suitable strategy for subsequent hybrid experiments.

3.4. Hybrid Approaches

Hybrid optimization strategies combined multiple techniques sequentially to investigate whether their combination could achieve superior compression-accuracy trade-offs compared to individual methods applied in isolation, while also assessing whether technique sequencing influences final model performance.

Four dual-technique combinations were evaluated: KD followed by INT8 weight-only quantization; KD followed by pruning; pruning followed by KD; and pruning followed by INT8 weight-only quantization. Two triple-technique combinations applied all three methods in two orderings: KD, pruning, then INT8 weight-only quantization; and pruning, KD, then INT8 weight-only quantization. Quantization was consistently applied last, as TFLite quantized conversion is irreversible and prevents further training.

[2] `https://www.tensorflow.org/lite/api_docs/python/tf/lite/Optimize`

3.5. Model Evaluation

Final models were evaluated in TFLite format on the held-out evaluation subset. Classification performance was assessed through accuracy, and macro-averaged F1-score. Compression is reported as model size in megabytes and compression ratio (fine-tuned baseline size / optimized size). Inference latency corresponds to the mean execution time over 100 timed runs following 10 warmup inferences, and memory footprint is quantified as the Resident Set Size (RSS) delta measured via `psutil` during interpreter instantiation and tensor allocation. All latency and memory measurements were obtained on a desktop CPU and should be interpreted as relative proxies across configurations rather than absolute deployment benchmarks for embedded hardware.

4. Results and Discussion

Results are organized progressively from baseline characterization through individual and hybrid compression strategies to the consolidated analysis of accuracy–compression trade-offs.

4.1. Baseline Model Development

Among the three fine-tuning strategies evaluated, Full and Partial Fine-Tuning achieved comparable accuracy (97.09% and 97.63%) at identical model sizes of 23.87 MB (2.07× compression). Feature Extraction achieved the highest accuracy (98.36%) but yielded a 49.56 MB model with negligible compression (0.99×), incompatible with deployment objectives. Partial Fine-Tuning provided the best performance–size balance and was adopted as the experimental baseline (Figure 1).

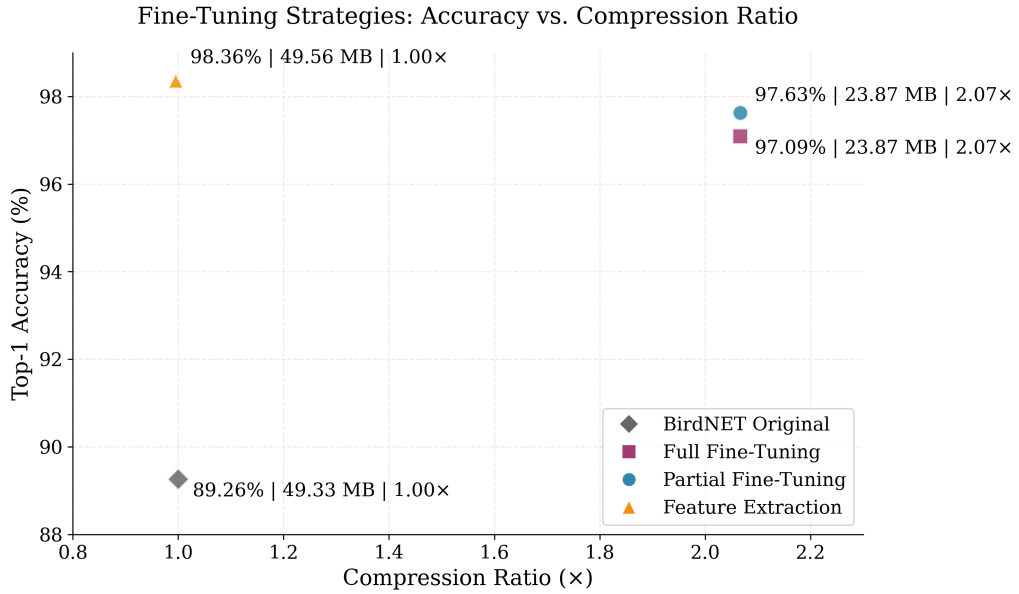


Figure 1. Fine-tuning strategies: accuracy vs. compression ratio.

4.2. Individual Optimization Techniques

4.2.1. Quantization

Table 3 reports the results for the three quantization configurations. Float16 preserved baseline accuracy (97.63%) with moderate compression (1.99×, 11.99 MB) and stable

latency (29.56 ms). INT8 weight-only quantization achieved the most favorable trade-off: 96.72% accuracy at 3.63× compression (6.57 MB), representing only 0.91 percentage points (pp) of degradation relative to the baseline, with latency increasing to 48.08 ms due to runtime dequantization operations. Full INT8 quantization caused catastrophic accuracy degradation to 29.03% despite comparable compression (3.40×), possibly due to insufficient calibration of activation distributions under static quantization.

Table 3. Quantization strategy performance.

Strategy	Accuracy (%)	F1 (%)	Size (MB)	RSS (MB)	Lat. (ms)	Δ Accuracy (pp)	Comp.
Baseline	97.63	96.95	23.87	23.88	29.67	—	1.00×
Float16	97.63	96.95	11.99	12.01	29.56	0.00	1.99×
INT8 Weight-Only	96.72	95.97	6.57	6.79	48.08	−0.91	3.63×
INT8 Full	29.03	25.79	7.01	7.05	68.69	−68.60	3.40×

4.2.2. Knowledge Distillation

As shown in Table 4, all three KD configurations share the same student architecture, yielding identical model size (8.06 MB), compression ratio (2.96×), and RSS (8.07 MB). Performance differences arose entirely from hyperparameter choices.

Table 4. Knowledge distillation configurations.

T	α	Accuracy (%)	F1 (%)	Size (MB)	RSS (MB)	Lat. (ms)	Δ Accuracy (pp)	Comp.
2.0	0.5	91.90	86.93	8.06	8.07	78.90	−5.73	2.96×
4.0	0.7	91.26	85.79	8.06	8.07	79.23	−6.37	2.96×
6.0	0.5	87.44	81.11	8.06	8.07	79.95	−10.19	2.96×

The T=2.0, α=0.5 configuration achieved the highest accuracy (91.90%), indicating that lower temperature scaling combined with balanced loss weighting better preserves the teacher's discriminative knowledge. The intermediate configuration (T=4.0, α=0.7) achieved comparable accuracy (91.26%), with a marginal 0.64 pp reduction. Increasing temperature to 6.0 degraded accuracy substantially to 87.44%, suggesting that stronger softening of the teacher's probability distributions was less effective under the evaluated configuration. Across all configurations, a consistent divergence between accuracy and F1-Score Macro indicates that knowledge distillation affects certain species more severely than others, a pattern absent in quantization-based approaches.

Among the evaluated KD configurations, T=2.0, α=0.5 achieved the best performance and was therefore retained as the representative KD configuration for the subsequent hybrid experiments.

4.2.3. Pruning

Magnitude-based unstructured pruning was assessed across three sparsity levels, with models serialized both with and without the `EXPERIMENTAL_SPARSITY` flag (Table 5).

Accuracy degradation remained acceptable up to 50% sparsity (95.00%, −2.63 pp) but declined substantially at 60% (87.44%, −10.19 pp), establishing a clear operational threshold. The 40% configuration achieved marginally higher accuracy (95.72%) but delivered less compression, offering no meaningful advantage over the 50% configuration.

Table 5. Pruning configurations under standard and sparse serialization.

			Standard			Sparse		
Target	**Accuracy(%)**	**F1 (%)**	**Size (MB)**	**RSS (MB)**	**Lat. (ms)**	**Size (MB)**	**RSS (MB)**	**Lat. (ms)**
40%	95.72	94.34	23.87	23.88	29.55	19.01	23.75	30.54
50%	95.00	93.32	23.87	23.88	29.44	15.96	19.39	30.38
60%	87.44	84.03	23.87	23.88	29.45	12.92	15.63	30.14

Standard: weights stored densely. *Sparse*: `EXPERIMENTAL_SPARSITY` flag enabled.

Without the sparse serialization flag, all pruned models retained the baseline size of 23.87 MB despite the induced sparsity, yielding no storage benefit. With the flag enabled, file size decreased proportionally to sparsity, reaching 15.96 MB at 50% and 12.92 MB at 60%, with a more pronounced gap between file size and RSS suggesting overhead from sparse tensor decompression at load time. Inference latency remained stable across serialization modes. Among the evaluated pruning configurations, the 50% sparsity configuration with `EXPERIMENTAL_SPARSITY` enabled was retained for the subsequent hybrid experiments.

4.3. Hybrid Optimization Approaches

Six hybrid configurations were evaluated, comprising four dual-technique and two triple-technique combinations (Table 6). Among dual combinations, KD + Pruning and Pruning + KD employed identical techniques but yielded markedly different results: applying KD first achieved 90.26% accuracy at 5.02× compression, while the reverse order degraded performance to 84.62% at only 2.96×, a difference of 5.64 pp. KD + INT8 achieved the most aggressive dual compression (11.55×, 2.07 MB) while maintaining 87.99% accuracy, approaching triple-technique ratios within a two-step pipeline.

Table 6. Hybrid optimization strategies: performance and compression.

Strategy	**Accuracy (%)**	**F1 (%)**	**Size (MB)**	**RSS (MB)**	**Lat. (ms)**	**Δ Accuracy (pp)**	**Comp.**
Baseline	97.63	96.95	23.87	23.88	29.67	—	1.00×
Dual Combinations							
Pruning + INT8	90.35	86.47	7.90	11.44	47.48	−7.28	3.02×
Pruning + KD	84.62	78.17	8.06	8.07	79.92	−13.01	2.96×
KD + Pruning	90.26	84.11	4.75	7.02	79.51	−7.37	5.02×
KD + INT8	87.99	82.17	2.07	2.09	618.08	−9.65	11.55×
Triple Combinations							
KD + Pruning + INT8	85.90	77.17	1.99	4.26	633.69	−11.74	11.99×
Pruning + KD + INT8	78.43	69.70	2.07	2.08	621.64	−19.20	11.56×

Combinations listed in application order.

Triple-technique combinations further reinforced the sequencing dependency. KD + Pruning + INT8 achieved 85.90% accuracy at 11.99× compression, while Pruning + KD + INT8 reached only 78.43% at a nearly identical ratio (11.56×), a difference of 7.47 pp from applying the same techniques in a different order. This pattern was consistent across all KD comparisons, suggesting that establishing a compact student prior to further optimization yields a more stable representational foundation than replicating an already-pruned teacher, with KD-first sequences outperforming pruning-first by 5 to 7.5 pp.

F1-Score Macro divergence increased progressively with combination depth, reaching 8.73 pp for KD + Pruning + INT8 compared to 0.75 pp for standalone INT8 weight-only quantization, indicating that hybrid pipelines introduce non-uniform performance degradation across species. Configurations involving KD combined with quantization exhibited a disproportionate latency increase to approximately 618–634 ms, compared to 79 ms for standalone KD and 48 ms for standalone INT8 weight-only. This behavior may reflect interactions between the student architecture and TFLite's INT8 execution pipeline, such as operator fallback or less optimized kernels; however, the exact cause was not isolated in the present experiments. Nonetheless, even the highest latency remains well within the 3-second audio segment duration, suggesting that the evaluated configurations may be viable for on-device inference.

4.4. Synthesis and Deployment Considerations

Based on consolidated results measured on desktop CPU, two candidate profiles emerge for resource-constrained embedded hardware (Figure 2). INT8 weight-only quantization provides the most favorable profile when classification reliability is prioritized (96.72% accuracy, 3.63× compression, 6.79 MB RSS), while KD + INT8 represents a candidate for more memory-constrained scenarios, achieving 11.55× compression and 2.09 MB RSS at 87.99% accuracy. Pruning with sparse serialization was outperformed by quantization-based approaches on all deployment-relevant metrics, suggesting its greatest utility as a preprocessing step within hybrid pipelines rather than as a standalone strategy.

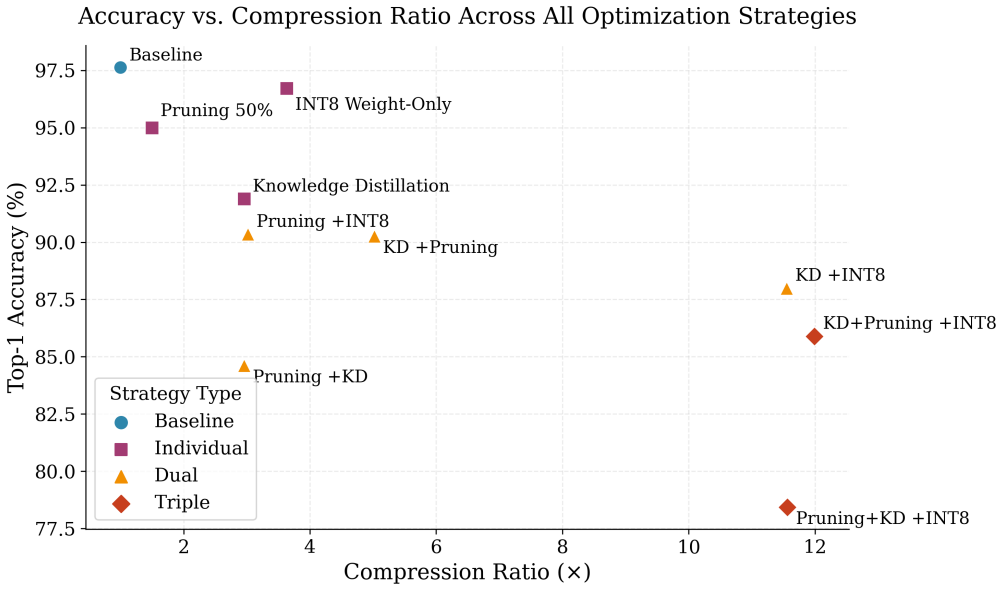


Figure 2. Accuracy vs. compression ratio across all evaluated strategies.

Together, the evaluated configurations establish a range of candidate optimization paths for bioacoustic models targeting remote ecological monitoring, from near-lossless compression to aggressive memory reduction, with technique sequencing emerging as a relevant design variable whose implications extend beyond this specific domain.

5. Conclusion

This work systematically evaluated model compression through post-training quantization, knowledge distillation, and pruning, applied individually and in hybrid combinations to BirdNET v2.4, adapted for 10 Pantanal bird species and targeting autonomous deployment on resource-constrained embedded hardware. Beyond identifying INT8 weight-only quantization as the most effective individual technique and KD + INT8 as the best trade-off among combinations, the experiments showed consistent performance differences across technique orderings, with KD-first pipelines outperforming pruning-first sequences by 5 to 7.5 percentage points at comparable compression ratios. The two deployment profiles identified, 6.79 MB RSS at 96.72% accuracy and 2.09 MB RSS at 87.99% accuracy, represent candidate configurations for passive acoustic monitoring devices in connectivity-limited environments such as the Pantanal, based on proxy measurements from desktop CPU execution. By reducing reliance on cloud infrastructure, these profiles represent a step toward scalable biodiversity monitoring in remote ecosystems such as the Pantanal.

These results should be interpreted in light of certain limitations. The evaluation was restricted to 10 bird species, and whether the observed accuracy-compression trade-offs generalize to broader taxonomic coverage remains an open question. Experiments also relied on a single random seed, so small differences between configurations may partly reflect training variability. Additionally, all latency and memory measurements were obtained on a desktop CPU rather than on target embedded hardware, so reported values serve as relative proxies rather than absolute deployment benchmarks.

Future work could extend the evaluation to a larger set of Pantanal species to assess whether the identified trade-offs generalize across broader taxonomic coverage, and validate the proposed profiles on physical embedded hardware for empirical latency and energy characterization. Quantization-aware training should also be explored to address the incompatibilities observed under full INT8 quantization, and investigating alternative student architectures may reduce the species-level performance gap introduced by knowledge distillation. These directions point toward scalable, connectivity-independent biodiversity monitoring solutions for remote ecosystems such as the Pantanal.

6. Acknowledgments

We thank the Computational Bioacoustics Research Unit (CO.BRA) at the Federal University of Mato Grosso for their support and guidance throughout this research.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B.,

Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Alho, C. J. R. and Sabino, J. (2011). A conservation agenda for the pantanal's biodiversity. *Brazilian Journal of Biology*, 71:327–335.

Daron, I. M., Gomes, J., Ventura, T. M., Teles, L. K. S., and Oliveira, A. G. d. (2026). A systematic review of post-training optimization for embedded deep learning to support biodiversity monitoring in the pantanal. In *Anais do Congresso de Inteligência Artificial de Mato Grosso*, Cuiabá, MT, Brasil.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. (2021). A survey of quantization methods for efficient neural network inference.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Han, S., Mao, H., and Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network.

Huang, Z., Tousnakhoff, A., Kozyr, P., Rehausen, R., Bießmann, F., Lachlan, R., Adjih, C., and Baccelli, E. (2024). Tinychirp: Bird song recognition using tinyml models on low-power wireless acoustic sensors.

Höchst, J., Bellafkir, H., Lampe, P., Vogelbacher, M., Mühling, M., Schneider, D., Lindner, K., Rösner, S., Schabo, D., Farwig, N., and Freisleben, B. (2022). *Bird@Edge: Bird Species Recognition at the Edge*, pages 69–86.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 2704–2713. IEEE.

Junk, W. J., da Cunha, C. N., Wantzen, K. M., Petermann, P., Strüssmann, C., Marques, M. I., and Adis, J. (2006). Biodiversity and its conservation in the pantanal of mato grosso, brazil. *Aquatic Sciences*, 68:278–309.

Kahl, S., Wood, C., and Klinck, H. (2025). Birdnet model v2.4.

Kahl, S., Wood, C. M., Eibl, M., and Klinck, H. (2021). Birdnet: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, 61:101236.

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342.

LeCun, Y., Denker, J., and Solla, S. (1989). Optimal brain damage. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Li, Y. and Ablavatski, A. (2021). Build fast, sparse on-device models with the new TF MOT pruning API. TensorFlow Blog. Accessed: 2025.

McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference*, SciPy, page 18–24. SciPy.

Mohaimenuzzaman, M., Bergmeir, C., West, I., and Meyer, B. (2023). Environmental sound classification on the edge: A pipeline for deep acoustic networks on extremely resource-constrained devices. *Pattern Recognition*, 133:109025.

Silva, R. L. d., Jacinto, G., Véstias, M., and Duarte, R. P. (2026). Real-time bird audio detection with a cnn-rnn model on a soc-fpga. *Electronics*, 15(2):354.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Stowell, D. (2021). Computational bioacoustics with deep learning: a review and roadmap. *CoRR*, abs/2112.06725.

Wang, K., Yang, F., Chen, Z., Chen, Y., and Zhang, Y. (2023). A fine-grained bird classification method based on attention and decoupled knowledge distillation. *Animals*, 13(2):264.

Wood, C. M. and Kahl, S. (2024). Guidelines for appropriate use of birdnet scores and other detector outputs. *Journal of Ornithology*, 165(3):777–782.

Zhao, T., Xie, Y., Wang, Y., Cheng, J., Guo, X., Hu, B., and Chen, Y. (2022). A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities. *Proceedings of the IEEE*, 110:334–354.