Lightweight CNNs for Edge Vision: A Cross-Benchmark Study of Robustness and Efficiency

Saulo Roberto dos Santos[1], Emanuel Adler Medeiros Pereira[2], Erick de Andrade Barboza[1], Itallo Patrick Castro Alves da Silva[1], Márcio de Medeiros Ribeiro[1], Baldoino Fonseca dos Santos Neto[1]

[1]Instituto de Computação – Universidade Federal de Alagoas (UFAL)
57072-970 – Maceió – AL – Brazil

{srs,erick,ipcas,marcio,baldoino}@ic.ufal.br

[2]Centro de Tecnologia – Universidade Federal do Rio Grande do Norte (UFRN)
59078-900 – Natal – RN – Brazil

emanuel.pereira.111@ufrn.edu.br

Abstract. *Selecting lightweight CNNs for edge vision from clean accuracy alone can lead to suboptimal deployments, since practical performance also depends on corruption robustness, latency, throughput, and hardware constraints. We compare MobileNetV3-Small, EfficientNet-B0, and MCUNet on CIFAR-10, CIFAR-100, and Wake Vision under the CIFAR-C corruption-error protocol and float32 TensorFlow Lite CPU-only inference on two Android phones and a Raspberry Pi Zero 2 W. EfficientNet-B0 attains the highest clean accuracy and lowest absolute mCE (42.82–66.33); MobileNetV3-Small is the most efficient (P90 from 1.83 ms, 158.63 images/s, 3.58–3.78 MB). Friedman and Nemenyi tests confirm this ranking.*

1. Introduction

Edge-vision model selection must balance clean accuracy with corruption robustness, latency, throughput, and footprint on the target device [Hendrycks and Dietterich 2019, Korkmaz 2025]. We compare MobileNetV3-Small, EfficientNet-B0, and MCUNet on CIFAR-10, CIFAR-100, and Wake Vision under one float32 TensorFlow Lite CPU-only protocol, using common-corruption tests and measurements on two Android phones and a Raspberry Pi Zero 2 W. The main contributions of this paper are as follows:

1. A controlled comparison of three distinct design points — mobile-first MobileNetV3-Small, compound-scaled EfficientNet-B0, and MCU-first MCUNet — across CIFAR-10, CIFAR-100, and Wake Vision.
2. Joint reporting of clean accuracy, corruption robustness, footprint, and on-device behavior under a fixed float32 TensorFlow Lite CPU protocol on two Android smartphones and a Raspberry Pi Zero 2 W.
3. A reproducible selection protocol, a deterministic 400-image device benchmark and a Wake Vision-C construction, with rank consistency tested across corruption and device-task blocks.

2. Related Work

This study sits at the intersection of three lines of work: the design of lightweight backbones, their empirical benchmarking on edge hardware, and the evaluation of robustness under common corruptions. We review each and then state the gap they leave open.

2.1. Lightweight Architectures for Edge Vision

MobileNetV3 combines depthwise separable convolutions, squeeze-and-excitation, and architecture search to reduce mobile inference cost [Howard et al. 2019], whereas EfficientNet scales width, depth, and resolution jointly [Tan and Le 2019]. MCUNet targets a stricter regime by co-designing network and runtime for microcontroller SRAM budgets [Lin et al. 2020], and MCUNetV2 lowers peak memory through patch-based inference [Lin et al. 2021]. Such co-design implies that FLOPs and parameter counts do not determine device behavior [Korkmaz 2025], motivating direct measurement.

2.2. Comparative and On-device Benchmarking Studies

Because this paper contributes a comparison rather than an architecture, its closest relatives place several backbones under one protocol. A recent preprint [Shahriar 2025] compares seven lightweight CNNs — including MobileNetV3-Small and EfficientNet-B0 — across CIFAR-10, CIFAR-100, and Tiny ImageNet under a shared recipe, but reports only proxy costs (parameters, MACs, storage) without edge measurement. Measuring convolution variants of one backbone on a Raspberry Pi 5, a Coral board, and a Jetson Nano instead shows the depthwise-separable variant to be slowest despite having the fewest FLOPs [Korkmaz 2025]. The closest design shape to ours — three model families, three datasets, three boards, with Pareto fronts over accuracy, latency, and memory — evaluates only clean data and uses regression rather than significance testing [Rojas-Carrasco and Guinaldo 2026]. Device-centric benchmarks characterize hardware and runtime rather than architecture: MLPerf Tiny standardizes latency, energy, and accuracy for MCU-class tasks around fixed reference models [Banbury et al. 2021], and DeepEdgeBench compares MobileNet variants over five boards and runtimes [Baller et al. 2021]. This literature also warns against proxy costs: FLOPs-based latency prediction reaches about 22% accuracy against roughly 99% for kernel-level prediction, and the same network varies fourfold across backends [Zhang et al. 2021]. Runtime thus matters as much as the graph [TensorFlow Authors 2024, Google 2025], which is why we fix one CPU path. The same measure-on-the-device principle guides comparisons of frameworks on embedded CPUs and GPUs by energy, throughput, and accuracy [Alf et al. 2023], of MobileNet against DenseNet201 on a Raspberry Pi [Soares et al. 2024], and of embedded models through a dedicated Android harness [Miranda Filho et al. 2025]. None of these evaluates corrupted inputs.

2.3. Common-corruption Robustness of Efficient Models

CIFAR-10-C and CIFAR-100-C established the corruption-error protocol and showed that clean-accuracy progress did not buy proportional resilience [Hendrycks and Dietterich 2019]; AugMix reports standard-training CIFAR-10-C and CIFAR-100-C error baselines of 26.9–30.8% and 53.3–59.3% [Hendrycks et al. 2020]. Later work stresses that robustness is multi-faceted [Hendrycks et al. 2021] and sensitive to image statistics [Geirhos et al. 2019, Burgert et al. 2025]. Three architecture-level benchmarks are directly relevant. RobustART evaluates 49 hand-designed networks — including the MobileNetV3 and EfficientNet families — plus over 1,200 subnetworks on ImageNet-C, and finds that for some lightweight architectures neither larger models nor extra data improve robustness [Tang et al. 2021]. A survey applying the mCE and relative-CE formulas we adopt

Table 1. Positioning against prior comparative studies. CR = common-corruption robustness; Dev. = measured on-device cost; Stat. = rank-based tests across conditions.

Study	Models / benchmarks	CR	Dev.	Stat.
[Shahriar 2025]	7 lightweight CNNs; CIFAR-10/100, TinyIN	—	—	—
[Rojas-Carrasco and Guinaldo 2026]	3 families × 3 datasets × 3 boards	—	✓	—
[Baller et al. 2021]	MobileNet variants; 5 edge boards	—	✓	—
[Tang et al. 2021]	49 CNNs + subnets; ImageNet-C	✓	—	—
[Jung et al. 2023]	6,466 cells; CIFAR-10-C/100-C	✓	—	—
[Alves da Silva et al. 2025]	3 CNNs × compression; CIFAR-*-C	✓	—	—
This study	MNV3-Small, EffB0, MCUNet; CIFAR-10/100-C, Wake Vision-C; 2 phones + Pi Zero 2 W	✓	✓	✓

shows robustness does not scale simply with capacity: EfficientNet-B2 (9.1M parameters) attains a lower ImageNet-C mCE than ResNet-50 (25.6M) [Wang et al. 2024]. And across 6,466 architectures on CIFAR-10-C and CIFAR-100-C, the rankings induced by different corruptions are mostly uncorrelated [Jung et al. 2023] — which motivates aggregating ranks over corruption blocks and testing them, a step not taken there. Efficiency transformations also interact with robustness: quantization, pruning, and clustering leave most compressed CIFAR-10-C variants at or below the uncompressed mCE, int8 quantization-aware training being the clear exception [Alves da Silva et al. 2025]. Porting the protocol to new datasets is established practice, from detection [Michaelis et al. 2019] to MangoLeafDB-C, which re-baselines on the worst clean-accuracy model for lack of an AlexNet reference [Andrade et al. 2025] — the decision we repeat for Wake Vision-C. In TinyML, a systematic review finds accuracy dominates the metrics in use and identifies the absence of standardized robustness benchmarking as an open gap [Pereira et al. 2026]: Wake Vision supplies fine-grained and natural out-of-distribution splits but no synthetic corruption suite [Banbury et al. 2026], and neither MCUNet nor MLPerf Tiny reports corruption results [Lin et al. 2020, Banbury et al. 2021].

2.4. Research Gap

Table 1 positions these studies along the three axes this paper combines. Robustness benchmarks cover architecture space but stop at proxy costs; edge benchmarks measure devices but only on clean data; and neither group applies rank-based testing across conditions. We found no published mCE for MobileNetV3-Small, EfficientNet-B0, or MCUNet on CIFAR-10-C and CIFAR-100-C, nor any corruption suite for Wake Vision.

3. Methodology

The protocol below is deliberately uniform: datasets, training recipe, export precision, and measurement procedure are fixed across the three architectures, so that any observed difference reflects the models rather than the pipeline.

3.1. Experimental Pipeline

The workflow comprised four stages: (i) supervised training and checkpoint selection, (ii) corruption-based robustness evaluation, (iii) TensorFlow Lite export, and (iv) on-device benchmarking, implemented respectively in TensorFlow/Keras, a Kotlin/Jetpack Compose Android application, and a Python inference script for the Raspberry Pi. The

Table 2. Wake Vision-C construction.

Item	Specification
Source examples	Official test split, excluding `person == -1`; the original RGB `uint8` image is corrupted before resizing.
Corruption operator	`imagecorruptions==1.1.2` with `scikit-image==0.22.0`.
Corruption set	The 15 common CIFAR-C corruptions (noise, blur, weather, contrast, elastic transform, pixelate, JPEG).
Severity and preprocessing	$s \in \{1, \ldots, 5\}$; resized bilinearly to 224×224, cast to float32, then architecture-specific preprocessing.

complete reproducibility package, including code, analysis scripts, and benchmarking artifacts, is publicly available[1].

3.2. Datasets and Corruption Protocol

We use three datasets representing complementary edge-vision scenarios. CIFAR-10 and CIFAR-100 [Krizhevsky 2009] provide 10 and 100 natural-image classes; inputs are resized to 160×160 and the last 5,000 training images serve as validation. Wake Vision [Banbury et al. 2026] is a binary person-detection benchmark; we use the official `train_quality`, `validation`, and `test` splits from `tensorflow_datasets`, resize to 224×224, and exclude ambiguous `person == -1` samples from the test split. Common corruptions are not used for training and do not alter task labels; they provide a controlled stress test for non-adversarial distribution shift, covering sensor noise, blur, weather-like overlays, contrast changes, pixelation, and JPEG artifacts [Hendrycks and Dietterich 2019]. All reported robustness metrics and statistical tests use the 15 common CIFAR-C corruptions at five severity levels. For CIFAR-10-C and CIFAR-100-C, we used the pre-generated benchmark corruptions. For Wake Vision-C, the same 15 corruptions were applied on-the-fly to each clean test image with `imagecorruptions` [Michaelis et al. 2019]. Precomputing all 75 corrupted 224×224 RGB conditions for approximately 55k test examples would require about 0.6 TB uncompressed, so each corrupted image is generated, evaluated, and discarded; Table 2 specifies the full construction.

3.3. Architectures and Training Strategy

We compare three design points: **MobileNetV3-Small** for mobile efficiency, **EfficientNet-B0** as a compound-scaled baseline, and **MCUNet** for MCU-class deployment. This compact panel does not rank every modern edge architecture; it contrasts mobile-first, compound-scaled, and MCU-first design goals. All models start from ImageNet-pretrained weights with a task-specific head. MCUNet was converted from PyTorch to Keras before fine-tuning, which is required for the common comparison path but does not reproduce its native deployment pipeline. So that observed differences reflect architectural properties rather than pipeline variation, all models were trained under one shared protocol implemented in a single function (`train_head_and_finetune`) and summarized in Table 3. Training proceeded in two stages: head-only optimization with the backbone frozen, then partial fine-tuning as specified there. All models were trained with a single fixed seed (`seed = 42`), so results reflect one outcome per architecture–dataset pair and do not estimate seed-level variance: the tests reported below assess ranking consistency across conditions, not the significance of seed-level training differences.

[1] `https://github.com/saulolv/Lightweight-Analyses`

Table 3. Shared training protocol and per-architecture preprocessing.

Hyperparameter / Setting	Value
Shared across all architectures	
Input resolution	160×160 (CIFAR-10/100); 224×224 (Wake Vision)
Optimizer	AdamW ($\lambda = 10^{-4}$), Adam as fallback
Learning rate	10^{-3} (head), 10^{-5} (fine-tune)
Label smoothing	$\varepsilon = 0.1$
Callbacks	Early stopping (patience 8) and LR reduction on plateau (patience 3), both on val. accuracy
Data augmentation	None (`USE_AUGMENTATION = False`)
Fine-tune ratio	Bottom 20% of backbone frozen, top 80% unfrozen; BatchNorm in inference mode
Model selection	Best validation accuracy (`ModelCheckpoint`)
Export precision	float32, no quantization
Per-architecture preprocessing	
MobileNetV3-Small	Keras rescaling to $[-1, 1]$
EfficientNet-B0	ImageNet-compatible preprocessing (Keras built-in)
MCUNet	Explicit ImageNet normalization: $(x/255 - \mu)/\sigma$

3.4. Robustness Metrics

For an architecture $m \in \mathcal{M}$, reference b (MCUNet), corruption $c \in C$ from the 15-element common set, and severity $s \in S = \{1, \ldots, 5\}$, let $E_{m,c,s} = 1 - \mathrm{Acc}_{m,c,s}$ and let $E_{m,\mathrm{clean}}$ be its clean error. Following the corruption-error framework [Hendrycks and Dietterich 2019], we compute:

$$CE_{m,c}^{(b)} = \frac{\sum_{s=1}^{5} E_{m,c,s}}{\sum_{s=1}^{5} E_{b,c,s}}, \qquad mCE_{m}^{(b)} = \frac{1}{|C|} \sum_{c \in C} CE_{m,c}^{(b)}, \tag{1}$$

$$RCE_{m,c}^{(b)} = \frac{\sum_{s=1}^{5} E_{m,c,s} - E_{m,\mathrm{clean}}}{\sum_{s=1}^{5} E_{b,c,s} - E_{b,\mathrm{clean}}}, \qquad RmCE_{m}^{(b)} = \frac{1}{|C|} \sum_{c \in C} RCE_{m,c}^{(b)}. \tag{2}$$

Relative mCE thus discounts clean error. MCUNet is the internal baseline because it has the highest clean error on all datasets, and is fixed at 100 by definition, so values below 100 are better than it.

3.5. On-device Benchmarking

We benchmarked float32 TensorFlow Lite CPU inference on all devices, with no hardware delegate or post-training quantization, measuring latency, throughput, and memory; accuracy is reported separately on the full evaluation splits. The Android app (Kotlin/Jetpack Compose) loads each model, runs 10 warm-ups, and times `Interpreter.run()` per image with up to four CPU threads, reporting mean and P90 latency, standard deviation, throughput, and memory over the 400-image subset; we emphasize P90 because it captures tail behavior relevant to latency-sensitive applications. The Raspberry Pi Zero 2 W uses a Dockerized Python TFLite interpreter with `num_threads=1`, giving a reproducible conservative baseline that avoids thread-dependent LiteRT variability, with the same warm-up procedure.

Deployment scope and 400-image subset. This float32 CPU-only protocol is a control, not peak performance under INT8, GPU/NPU delegates, or native MCU runtimes; no accelerator delegate is loaded on the Pi. Each device/model pair is timed over 400 images drawn deterministically, without shuffling, from the same export, so every model on a device processes the same workload; because these runs report only latency, throughput, and memory, the outcome does not depend on which images are selected. Clean accuracy and all corruption metrics are computed on the full evaluation splits.

3.6. Statistical Testing

Friedman and Nemenyi tests assess whether model rankings remain consistent over repeated blocks [Demšar 2006]. Robustness uses 45 dataset–corruption blocks (3 datasets $\times$ 15 corruptions), ranked by corruption error (lower is better). Efficiency uses 9 dataset–device blocks (3 datasets $\times$ 3 devices), ranked separately by P90 latency (lower) and throughput (higher). Because mean-rank post-hoc verdicts depend on the compared pool [Benavoli et al. 2016], the reported significances are conditional on this three-model panel.

4. Results

Results follow the three axes under comparison — footprint, on-device cost, and corruption robustness — and are interpreted jointly in Section 4.4.

4.1. Model Size

Model size was determined mainly by architecture and varied only slightly across datasets: 3.58–3.78 MB for MobileNetV3-Small in TensorFlow Lite format, 5.39–5.50 MB for MCUNet, and 15.33–15.79 MB for EfficientNet-B0 — about $4.2\times$ the storage of MobileNetV3-Small and $2.8\times$ that of MCUNet, a markedly higher cost on memory-constrained targets before runtime is considered.

4.2. On-device Efficiency

Table 4 reports P90 latency and throughput on all devices. MobileNetV3-Small is fastest, MCUNet is second, and EfficientNet-B0 is slowest in every device–dataset block. MobileNetV3-Small stays below 52 ms P90 on every device and is the only model above 8 images/s on the Raspberry Pi, where EfficientNet-B0 falls to 1.39–2.83 images/s. Memory follows the same ordering: 19.89–61.68 MB for MobileNetV3-Small, 19.67–75.38 MB for MCUNet, and 43.79–78.19 MB for EfficientNet-B0. Across the nine device–dataset blocks, Friedman tests give $\chi^2_F = 18.00$ and $p = 1.23 \times 10^{-4}$ for both P90 latency and throughput; Nemenyi ranks MobileNetV3-Small first, MCUNet second, and EfficientNet-B0 third. The run-level analysis (300 blocks) gives the same ranking ($\chi^2_F = 453.84$, $p \approx 2.8 \times 10^{-99}$).

4.3. Robustness to Common Corruptions

Figure 1 and Table 5 show that EfficientNet-B0 has the highest clean accuracy and the lowest absolute mCE on every benchmark, followed by MobileNetV3-Small and MCUNet. Relative mCE, however, inverts this ordering on Wake Vision, an effect analyzed in Section 4.4. Over the 45 dataset–corruption blocks, Friedman yields $\chi^2_F = 104.46$ and $p = 2.08 \times 10^{-23}$; Nemenyi ranks EfficientNet-B0 first, MobileNetV3-Small second, and MCUNet third (Figure 2).

Table 4. On-device efficiency. P90 latency and throughput per device. MNV3-S = MobileNetV3-Small; EffB0 = EfficientNet-B0.

		P90 latency (ms)			Throughput (images/s)		
Dataset	Model	S24+	A14	Pi	S24+	A14	Pi
CIFAR-10	MNV3-S	1.83	4.53	25.63	158.63	69.95	34.06
CIFAR-10	MCUNet	5.98	16.87	182.70	102.37	38.54	5.39
CIFAR-10	EffB0	12.87	39.85	350.34	59.94	21.45	2.83
CIFAR-100	MNV3-S	2.05	4.58	25.75	152.26	71.20	33.90
CIFAR-100	MCUNet	7.93	16.91	182.96	85.44	38.69	5.38
CIFAR-100	EffB0	17.04	39.60	350.85	47.08	20.93	2.83
Wake Vision	MNV3-S	4.07	8.85	51.02	53.52	30.91	8.69
Wake Vision	MCUNet	17.18	33.40	462.20	32.81	18.26	1.91
Wake Vision	EffB0	36.11	77.15	663.06	20.80	10.73	1.39

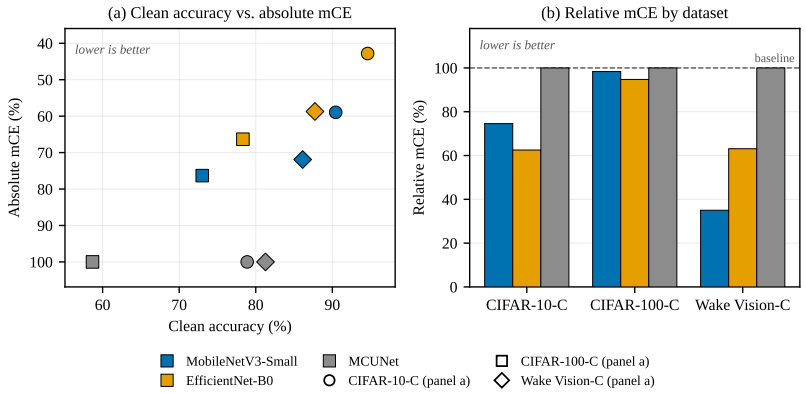


Figure 1. Robustness summary: (a) clean accuracy versus absolute mCE; (b) relative mCE by dataset. Color denotes architecture in both panels; marker shape denotes dataset in (a). Lower is better; exact values in Table 5.

Table 5. Clean error, mCE, and relative mCE (%) with MCUNet as the baseline. Lower is better for all metrics.

Dataset	Model	Clean Error	mCE	Rel. mCE
CIFAR-10	MobileNetV3-Small	9.56	58.95	74.58
CIFAR-10	EfficientNet-B0	5.39	42.82	62.52
CIFAR-10	MCUNet	21.12	100.00	100.00
CIFAR-100	MobileNetV3-Small	26.99	76.32	98.36
CIFAR-100	EfficientNet-B0	21.68	66.33	94.75
CIFAR-100	MCUNet	41.32	100.00	100.00
Wake Vision	MobileNetV3-Small	13.87	71.91	35.01
Wake Vision	EfficientNet-B0	12.27	58.72	63.11
Wake Vision	MCUNet	18.73	100.00	100.00

4.4. Discussion

Capacity helps only where spatial structure survives. EfficientNet-B0 has the lower CE in 42 of the 45 cells of Table 6, but the margin is strongly corruption dependent: averaged over datasets, its advantage is 22.2 CE points under noise and 16.2 under brightness and contrast, against 8.2 under the digital corruptions and 5.2 under

Table 6. Per-corruption CE (%). MCUNet is the baseline, fixed at 100. Lower is better. MNV3 = MobileNetV3-Small; EffB0 = EfficientNet-B0; MCU = MCUNet.

	CIFAR-10-C			CIFAR-100-C			Wake Vision-C		
Corruption	MNV3	EffB0	MCU	MNV3	EffB0	MCU	MNV3	EffB0	MCU
Gaussian Noise	72	53	100	87	76	100	68	48	100
Shot Noise	68	49	100	84	73	100	72	51	100
Impulse Noise	90	46	100	101	69	100	70	47	100
Defocus Blur	49	34	100	69	60	100	66	72	100
Glass Blur	74	82	100	89	99	100	62	51	100
Motion Blur	58	45	100	74	68	100	69	61	100
Zoom Blur	49	36	100	70	62	100	78	73	100
Snow	45	37	100	68	62	100	63	48	100
Frost	39	29	100	59	55	100	68	58	100
Fog	78	47	100	85	69	100	90	59	100
Brightness	33	19	100	55	44	100	69	55	100
Contrast	75	55	100	81	73	100	100	70	100
Elastic Transform	57	47	100	76	70	100	71	67	100
Pixelate	54	39	100	70	62	100	65	60	100
JPEG Compression	53	43	100	76	67	100	68	61	100

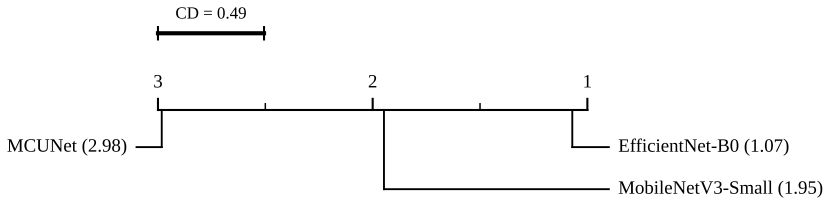


Figure 2. Nemenyi critical-difference diagram for corruption error.

blur — and all three reversals belong to the blur family (glass blur on both CIFAR sets, defocus blur on Wake Vision-C). Capacity therefore helps where a corruption perturbs or rescales intensity but leaves spatial structure intact, and helps little where local structure is destroyed. Two design differences fit this. Compound scaling spends about $7\times$ more multiply–accumulate operations than MobileNetV3-Small, over width as well as depth [Tan and Le 2019], providing redundant channels over which independent perturbations average out — consistent with the capacity–robustness relation reported for this family [Wang et al. 2024] and with lightweight architectures being where it weakens [Tang et al. 2021]. MobileNetV3-Small instead filters each channel independently in its depthwise separable blocks [Howard et al. 2019], so channel-local perturbations propagate rather than being attenuated. That blur is the one family able to invert the ordering agrees with texture-versus-shape reliance being architecture dependent [Geirhos et al. 2019, Burgert et al. 2025] and per-corruption rankings being largely uncorrelated [Jung et al. 2023].

The two robustness metrics can rank models in opposite orders. The capacity that lowers absolute mCE does not protect the relative metric. On Wake Vision, MobileNetV3-Small obtains the best relative mCE (35.01) while EfficientNet-B0 keeps the best absolute mCE (58.72). Relative mCE discounts each model's own clean error, so it measures the slope of degradation rather than its level. On a binary task decided by a coarse, large-area person region, MobileNetV3-Small already commits most of its errors on clean data (13.87% clean error) and corruption adds proportionally little, whereas EfficientNet-B0's lower clean error (12.27%) rests on finer evidence that corruption removes. This is the absolute-versus-relative distinction introduced with the corruption-error protocol [Hendrycks and Dietterich 2019], and the same inversion arises when the

protocol is ported to a new dataset [Andrade et al. 2025]. It also accounts for the worst cell in the study — MobileNetV3-Small ties the baseline under Wake Vision contrast ($\mathrm{CE} = 100$) — since reducing global contrast removes the coarse intensity evidence a low-capacity binary classifier relies on. Absolute mCE is thus the criterion when deployment must maximize accuracy under corruption, relative mCE when it must guarantee graceful degradation from a clean baseline.

The price of capacity grows as the device weakens. That capacity is also not paid at a fixed rate. The EfficientNet-B0 to MobileNetV3-Small P90 ratio is not device independent: it averages $8.1\times$ on the Galaxy S24+, $8.7\times$ on the Galaxy A14, and $13.4\times$ on the Raspberry Pi Zero 2 W. Two further measurements point at memory traffic rather than arithmetic. MCUNet is 3.3–$4.2\times$ slower than MobileNetV3-Small on the S24+ but 7.1–$9.1\times$ slower on the Pi; and moving from the 160×160 CIFAR configuration to the 224×224 Wake Vision configuration ($1.96\times$ the pixels) scales Pi latency by $1.98\times$ for MobileNetV3-Small and $1.89\times$ for EfficientNet-B0, but $2.53\times$ for MCUNet. On the weakest device latency thus tracks activation traffic and cache behavior rather than operation counts, as direct-measurement studies report for depthwise-heavy graphs [Korkmaz 2025] and cross-backend prediction [Zhang et al. 2021]. A FLOP-based selection would have mispredicted our ordering, and MCUNet — co-designed with an int8 microcontroller runtime — is penalized by this float32 path, not by its architecture [Lin et al. 2020, Lin et al. 2021].

Relation to published figures. Calibrating these observations against the literature requires care about protocol. Our clean accuracies sit a few points below published figures for the same architectures, as the shared protocol predicts (160×160 CIFAR inputs, bottom 20% of the backbone frozen, no augmentation, one seed): 94.61 and 78.32 on CIFAR-10 and CIFAR-100 for EfficientNet-B0 against 98.1 and 88.1 under full fine-tuning at native resolution [Tan and Le 2019], and 90.44 and 73.01 for MobileNetV3-Small against 96.12 and 82.43 at 224×224 [Shahriar 2025], with the ordering preserved. On Wake Vision our MobileNetV3-Small (86.13) exceeds the published MobileNetV2-0.25 baseline (84.89), while our MCUNet (81.27) stays 4.3 points below the MCUNet-320kb–1mb point (85.6), obtained with quantization-aware training in MCUNet's native pipeline rather than our float32 conversion path [Banbury et al. 2026]. Robustness figures demand more care: our mCE is normalized to MCUNet rather than AlexNet, so Table 5 is not comparable to published AlexNet-normalized ImageNet-C values — 63.5–69.8 for MobileNetV3-x1.4 [Tang et al. 2021], 66.83 and 58.87 for EfficientNet-B2 and B4 [Wang et al. 2024] — nor to the unnormalized CIFAR-C baselines above. The direction still agrees: the compound-scaled family is comparatively robust for its size.

The decision generalizes; the magnitudes do not. Taken together, the two rankings are stable and opposed across all 45 corruption and 9 device–task blocks, so what generalizes is the structure of the decision rather than a winner: capacity buys absolute robustness at a runtime cost that grows as the device weakens, the benefit concentrates in structure-preserving corruptions, and the two metrics can rank models in opposite orders. The magnitudes do not, being conditional on one seed, no augmentation, three datasets,

a float32 CPU path, and synthetic corruptions, whose relation to natural shift remains open [Hendrycks et al. 2021].

5. Conclusion and Future Work

Under the evaluated float32 TensorFlow Lite CPU protocol, EfficientNet-B0 is the predictive choice and MobileNetV3-Small the deployment-oriented one, consistently across corruption and device blocks. Future work should evaluate repeated seeds, INT8 exports, accelerator delegates, and energy behavior, testing the corruption- and device-dependent patterns reported here beyond a single recipe.

Acknowledgements

This study was financially supported by Alagoas State Research Support Foundation (FAPEAL) and National Council for Scientific and Technological Development (CNPq) project grant 404825/2023-0.

References

Alf, L. M., Hoffmann, R. B., Müller, C., and Griebler, D. (2023). Análise da execução de algoritmos de aprendizado de máquina em dispositivos embarcados. In *Anais do XXIV SSCAD*, pages 61–72.

Alves da Silva, I. P. C., Pereira, E. A. M., Barboza, E. d. A., dos Santos Neto, B. F., and Ribeiro, M. d. M. (2025). Evaluating the impact of compression techniques on the robustness of cnns under natural corruptions. In *IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*. IEEE.

Andrade, G. V. d., Santos, S. R. d., Silva, I. P. C. A. d., Pereira, E. A. M., and Barboza, E. d. A. (2025). From code to field: Evaluating the robustness of convolutional neural networks for disease diagnosis in mango leaves. In *Intelligent Systems (BRACIS 2025), Part IV*, volume 16182 of *LNAI*, pages 291–305. Springer.

Baller, S. P., Jindal, A., Chadha, M., and Gerndt, M. (2021). DeepEdgeBench: Benchmarking deep neural networks on edge devices. In *IEEE Int. Conf. on Cloud Engineering (IC2E)*, pages 20–30.

Banbury, C., Njor, E., Garavagno, A. M., Mazumder, M., et al. (2026). Wake vision: A tailored dataset and benchmark suite for TinyML computer vision applications. *Journal of Data-centric Machine Learning Research*, 3.

Banbury, C., Reddi, V. J., Torelli, P., Holleman, J., et al. (2021). MLPerf tiny benchmark. In *NeurIPS Datasets and Benchmarks Track*, volume 1.

Benavoli, A., Corani, G., and Mangili, F. (2016). Should we really use post-hoc tests based on mean-ranks? *Journal of Machine Learning Research*, 17(5):1–10.

Burgert, T., Stoll, O., Rota, P., and Demir, B. (2025). ImageNet-trained cnns are not biased towards texture: Revisiting feature reliance through controlled suppression. In *Advances in Neural Information Processing Systems*, volume 38.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., et al. (2019). ImageNet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*.

Google (2025). XNNPACK: High-efficiency floating-point neural network inference operators for mobile, server, and web. `https://github.com/google/XNNPACK`. Software library, accessed 2026-08-18.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., et al. (2021). The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349.

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., et al. (2020). AugMix: A simple data processing method to improve robustness and uncertainty. In *International Conference on Learning Representations*.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324.

Jung, S., Lukasik, J., and Keuper, M. (2023). Neural architecture design and robustness: A dataset. In *International Conference on Learning Representations*.

Korkmaz, O. E. (2025). Revisiting convolutional design for efficient cnn architectures in edge-aware applications. *Scientific Reports*, 15:44351.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto.

Lin, J., Chen, W.-M., Cai, H., Gan, C., and Han, S. (2021). MCUNetV2: Memory-efficient patch-based inference for tiny deep learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 2346–2358.

Lin, J., Chen, W.-M., Lin, Y., Cohn, J., et al. (2020). Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 33:11711–11722.

Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., et al. (2019). Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*.

Miranda Filho, R., Matias, P., and de Freitas, R. (2025). HarpIA: a tool for comparative analysis of embedded AI models on android. In *Anais do XXII ENIAC*, pages 1938–1949.

Pereira, E., Barboza, E., Araújo, Í., dos Santos, S., et al. (2026). Robustness in TinyML: A systematic literature review. *ACM Transactions on Embedded Computing Systems*.

Rojas-Carrasco, P. and Guinaldo, M. (2026). Multi-objective evaluation of lightweight AI models on low-cost edge devices using pareto fronts. *Applied Sciences*, 16(8):3679.

Shahriar, T. (2025). Comparative analysis of lightweight CNNs for resource-constrained devices: Predictive performance, efficiency trade-offs, and initialization effects. arXiv preprint arXiv:2505.03303.

Soares, B., Silva, W., Ponciano, G., Stefanie, B., et al. (2024). Embedded neural networks for identifying *spodoptera frugiperda* in corn plantations. In *Anais do XXI ENIAC*, pages 376–387.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

Tang, S., Gong, R., Wang, Y., Liu, A., et al. (2021). RobustART: Benchmarking robustness on architecture design and training techniques. arXiv preprint arXiv:2109.05211.

TensorFlow Authors (2024). Tensorflow lite: Deploy machine learning models on mobile and embedded devices. `https://www.tensorflow.org/lite`. Accessed: 2026-04-18.

Wang, S., Veldhuis, R., Brune, C., and Strisciuglio, N. (2024). A survey on the robustness of computer vision models against common corruptions. arXiv preprint arXiv:2305.06024v4.

Zhang, L. L., Han, S., Wei, J., Zheng, N., et al. (2021). nn-Meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices. In *Proc. ACM MobiSys*, pages 81–93.