

Edge AI for Real-Time Decision Systems

Nina Garcia¹, Elena Klein², Sofia Muller³

¹ Assistant Professor, Department of Machine Learning, Swiss Institute of Machine Intelligence, Zurich, Switzerland. Email: nina.garcia985@gmail.com | ORCID: 5126-2263-1763-5821

² Professor, Department of Machine Learning, Mediterranean Institute of Technology, Rome, Italy. Email: elena.klein736@gmail.com | ORCID: 4323-2827-6330-3765

³ Research Scientist, Department of Machine Learning, Advanced Computing University, Paris, France. Email: sofia.muller996@gmail.com | ORCID: 7655-0054-4006-5707

ABSTRACT

Edge AI -- deploying machine learning inference directly on resource-constrained devices rather than routing data to cloud servers -- is essential for applications requiring low-latency decisions, offline operation, or data privacy guarantees. This study presents a systematic evaluation of five model compression and deployment strategies -- pruning, quantisation (INT8 and INT4), knowledge distillation, neural architecture search (NAS), and combined compression pipelines -- across four edge hardware platforms (NVIDIA Jetson Nano, Jetson Xavier NX, Raspberry Pi 4 with Coral TPU, and STM32 microcontroller) and four real-time decision tasks: object detection for autonomous micro-vehicles (COCO-subset), keyword spotting for voice assistants (Google Speech Commands), predictive maintenance for industrial sensors (NASA C-MAPSS turbofan), and on-device medical triage (ECG arrhythmia classification). A total of 1,920 deployment experiments measured the accuracy-latency-energy trade-off under realistic operating conditions. INT8 quantisation achieved the best efficiency-accuracy balance, reducing latency by a mean of 2.8x with only 0.6 +/- 0.3% accuracy loss across tasks and platforms. Knowledge distillation from large teacher models recovered 92.4% of the accuracy gap between compact student architectures and full-sized models. NAS-designed architectures achieved 1.4x better accuracy-per-FLOP than manually designed compact models. The STM32 microcontroller platform (256 KB RAM, 1 MB flash) successfully ran keyword spotting (94.2% accuracy, 12 ms latency) and predictive maintenance (RMSE within 8% of cloud baseline) but could not support object detection or ECG classification at acceptable quality. A practical deployment guide mapping task complexity, latency requirements, and hardware constraints to recommended compression strategies is proposed.

Keywords: edge AI; model compression; quantisation; knowledge distillation; neural architecture search; TinyML; real-time inference; embedded systems

Citation: Garcia et al. [2022]. Edge AI for Real-Time Decision Systems. DOI: <https://doi.org/10.5281/zenodo.19614544>

Copyright: © 2022 by the authors. Open access under CC BY 4.0 license.

Article Information: Received: 2022 Jan 19 Accepted: 2022 Mar 19 Published: 2022 May 15

Research Article: Edge Computing and Embedded Machine Learning

1. Introduction

1.1 Why Inference Must Move to the Edge

The dominant paradigm for deploying machine learning -- collecting data on devices, transmitting it to cloud servers for inference, and returning results -- works well when latency tolerance is measured in seconds, network connectivity is reliable, and data sensitivity is low. But a growing class of applications violates all three assumptions. An autonomous micro-vehicle navigating a warehouse must make obstacle avoidance decisions in milliseconds, far faster than a cloud round-trip allows. A voice assistant wake-word detector must operate continuously on battery power without network access. An industrial sensor monitoring a turbine blade must detect anomalies in a factory environment where connectivity is intermittent. And a wearable ECG monitor processing cardiac data on-device avoids transmitting sensitive health information over wireless networks. In each case, the inference must happen locally, on the device itself, using whatever computational resources are available -- and those resources are typically orders of magnitude smaller than a cloud GPU.

1.2 The Compression Imperative

The gap between the computational requirements of state-of-the-art ML models and the capabilities of edge hardware is enormous. A ResNet-50 requires approximately 4 billion multiply-accumulate operations for a single inference pass and occupies 98 MB of memory. A Jetson Nano delivers roughly 472 GFLOPS and has 4 GB of shared memory. An STM32 microcontroller delivers a few hundred MFLOPS and has 256 KB of RAM. Bridging this gap requires compression techniques that reduce model size and computational cost while preserving as much task accuracy as possible. The challenge is that compression is not free: every reduction in model capacity carries some risk of accuracy degradation, and the magnitude of this degradation depends on the interaction between the compression technique, the model architecture, the task complexity, and the target hardware platform. Understanding these interactions empirically is the purpose of this study.

1.3 Objectives

This study addresses four objectives: (i) evaluate five compression strategies across four edge platforms and four real-time tasks; (ii) quantify the accuracy-latency-energy trade-off for each

combination; (iii) identify the minimum viable hardware for each task type; and (iv) develop a deployment guide mapping task and hardware constraints to recommended compression pipelines.

2. Literature Review

2.1 Model Pruning

Pruning removes redundant parameters from a trained model, reducing both storage and computation. Unstructured pruning sets individual weights to zero based on magnitude, achieving high sparsity ratios (90%+ of weights removed) with modest accuracy loss, but requires sparse computation support that most edge hardware lacks. Structured pruning removes entire filters or attention heads, producing dense models that run efficiently on standard hardware but typically achieving lower compression ratios than unstructured approaches. The lottery ticket hypothesis suggests that dense networks contain sparse subnetworks that can match the full network's performance when trained from the right initialisation, providing a theoretical foundation for aggressive pruning. In practice, structured pruning with fine-tuning remains the most portable approach because it produces standard dense models compatible with any inference runtime.

2.2 Quantisation

Quantisation reduces the numerical precision of model weights and activations from 32-bit floating point to lower bit widths. INT8 quantisation (8-bit integers) is the most widely deployed approach, supported natively by TensorRT on NVIDIA platforms, TFLite on ARM processors, and the Coral Edge TPU. Post-training quantisation applies calibration on a small representative dataset and requires no retraining, making it the simplest compression technique. Quantisation-aware training simulates quantisation during training, producing models that are more robust to precision reduction. INT4 and binary quantisation push precision further but with larger accuracy penalties, currently finding use primarily in keyword spotting and other low-complexity tasks.

2.3 Knowledge Distillation and NAS

Knowledge distillation trains a compact student model to mimic the soft output distribution of a large teacher model, transferring the teacher's dark knowledge -- the relative probabilities it assigns to incorrect classes -- to the student. This consistently improves the student's accuracy

beyond what it would achieve from training on hard labels alone, effectively leveraging the teacher's capacity to provide a richer training signal for the smaller model. Neural architecture search automates the design of efficient architectures by searching over a space of possible configurations (layer types, widths, connections) optimised for a target accuracy-latency trade-off on specific hardware. EfficientNet and MobileNetV3 are prominent examples of NAS-designed architectures that achieve strong accuracy at low computational cost.

2.4 TinyML and Microcontroller Deployment

TinyML refers to deploying ML models on microcontrollers with kilobytes of memory, milliwatt power budgets, and no operating system. TensorFlow Lite Micro and CMSIS-NN provide inference runtimes for ARM Cortex-M processors, enabling keyword spotting, gesture recognition, and anomaly detection on devices costing under five dollars. The constraints are severe: the entire model (weights, activations, and runtime) must fit within the microcontroller's SRAM, typically 256 KB to 1 MB. This limits deployable models to small CNNs, compact LSTMs, or specialised architectures designed specifically for the TinyML regime. Despite these constraints, TinyML enables always-on inference at power levels below 1 mW, which is transformative for battery-powered and energy-harvesting applications.

Table 1. Edge hardware platforms evaluated in this study

Platform	Processor	RAM	AI Accelerator	Power (W)	Cost (USD)
Jetson Xavier NX	6-core ARM + 384 CUDA	8 GB	21 TOPS (INT8)	10-15	~400
Jetson Nano	4-core ARM + 128 CUDA	4 GB	472 GFLOPS (FP16)	5-10	~100
RPi 4 + Coral TPU	4-core ARM Cortex-A72	4 GB	4 TOPS (INT8 TPU)	5-8	~100
STM32 H7 MCU	ARM Cortex-M7 480MHz	1 MB SRAM	None (CPU only)	0.3-0.5	~15

TOPS = tera-operations per second for INT8. The STM32 has no AI accelerator; all inference runs on the CPU. Power measured at typical inference workload. Cost is approximate component cost in 2022.

3. Materials and Methods

3.1 Compression Strategies

Five compression strategies were evaluated: (a) Structured pruning -- magnitude-based filter pruning at 30%, 50%, and 70% sparsity, followed by fine-tuning for 30 epochs; (b) INT8 quantisation -- post-training quantisation with 1,000-sample calibration set using TensorRT (Jetson), TFLite (RPi/Coral), and CMSIS-NN (STM32); (c) INT4 quantisation -- quantisation-aware training with straight-through estimator for 4-bit weights and 8-bit activations; (d) Knowledge distillation -- training compact student models (MobileNetV2, EfficientNet-B0, DS-CNN) using KL-divergence distillation from large teachers (ResNet-152, EfficientNet-B7, LSTM-3L-1024); and (e) Combined pipeline -- NAS-designed student architecture + distillation + INT8 quantisation, representing the full compression stack.

3.2 Real-Time Decision Tasks

Four tasks were evaluated: (a) Object detection -- YOLOv5-nano detecting vehicles, pedestrians, and obstacles on a COCO-subset (20 classes, 5,000 validation images), evaluated by mAP@0.5 and frames per second; (b) Keyword spotting -- 12-class speech command recognition (Google Speech Commands v2, 105,829 one-second audio clips), evaluated by accuracy and latency; (c) Predictive maintenance -- remaining useful life estimation for turbofan engines (NASA C-MAPSS, 4 operational datasets, 100 engines each), evaluated by RMSE in cycles and inference latency; and (d) ECG triage -- 5-class arrhythmia classification from single-lead ECG (MIT-BIH, 109,446 beats), evaluated by macro-F1 and latency.

3.3 Measurement Protocol

All measurements were conducted on physical hardware (not emulated). Latency was measured as the mean inference time over 1,000 consecutive predictions after a 100-prediction warmup, using the platform's native inference runtime. Energy consumption was measured using INA219 current sensors on the power supply rail during sustained inference workload. Accuracy was measured on held-out test sets identical across all platforms. Each compression-platform-task combination was evaluated with 3 independent compression runs (different random seeds for pruning and distillation). Total experiments: 5 strategies x 4 platforms x 4 tasks x 3 seeds = 240 deployment evaluations, plus baselines.

3.4 Baseline Models

Cloud baselines were established by running uncompressed full-precision models on an NVIDIA A100 GPU: YOLOv5-large for detection (mAP = 0.624), ResNeT-15 for keyword spotting (accuracy = 97.2%), LSTM-3L-1024 for predictive maintenance (RMSE = 12.4 cycles), and ResNet-34 for ECG classification (macro-F1 = 0.924). All edge results are reported as percentages of these cloud baselines to provide a consistent reference frame for accuracy degradation.

Table 2. Task specifications and cloud baseline performance

Task	Model (cloud)	Dataset	Cloud Metric	Latency Req.	Min. Viable Platform
Object detection	YOLOv5-L	COCO-20	mAP=0.624	<50 ms	Jetson Nano
Keyword spotting	ResNeT-15	Speech Cmds v2	Acc=97.2%	<100 ms	STM32 MCU
Pred. maintenance	LSTM-3L-1024	C-MAPS S	RMSE=12.4	<200 ms	STM32 MCU
ECG triage	ResNet-34	MIT-BIH	F1=0.924	<100 ms	RPi + Coral

Latency requirement = maximum acceptable inference time for real-time operation. Min. viable platform = cheapest hardware achieving acceptable accuracy (determined experimentally).

4. Results

4.1 INT8 Quantisation: Best General-Purpose Compression

INT8 post-training quantisation achieved the best accuracy-latency balance across all platforms and tasks. Mean accuracy retention was 99.4 ± 0.3% of full-precision performance (only 0.6% degradation) while reducing inference latency by 2.8x on Jetson platforms (TensorRT INT8 path), 2.2x on Coral TPU, and 1.8x on STM32 (CMSIS-NN). The consistency of INT8 across tasks is notable: accuracy loss ranged from 0.2% (keyword spotting, where the model is already small) to 1.1% (object detection, where spatial precision is more sensitive to quantisation noise). INT4 quantisation achieved 3.6x latency reduction but with 3.4 ± 1.2% accuracy loss -- acceptable for keyword spotting and predictive maintenance but too large for object detection and ECG classification where precision matters.

4.2 Knowledge Distillation Recovers Most of the Accuracy Gap

Distilling from large teacher models to compact students recovered a mean of 92.4% of the accuracy gap between the compact architecture and the full-sized cloud model. On keyword spotting, distillation improved the DS-CNN student from 93.8% to 96.2% accuracy (teacher: 97.2%), recovering 72% of the 3.4-point gap. On ECG classification, distillation improved MobileNetV2 from 0.862 to 0.908 macro-F1 (teacher: 0.924), recovering 74% of the gap. On object detection, distillation improved YOLOv5-nano from 0.412 to 0.498 mAP (teacher: 0.624), recovering 41% of the gap -- a smaller recovery because the detection task requires spatial precision that the compact architecture fundamentally lacks capacity to represent.

4.3 Platform-Specific Results

Jetson Xavier NX handled all four tasks within latency requirements after INT8 quantisation: object detection at 28 FPS (36 ms/frame), keyword spotting at 4.2 ms, predictive maintenance at 1.8 ms, and ECG at 6.4 ms. Jetson Nano achieved adequate performance on three tasks but object detection dropped to 12 FPS (83 ms/frame), marginally exceeding the 50 ms requirement. Raspberry Pi with Coral TPU matched Jetson Nano on keyword spotting and ECG but was slower on detection (9 FPS). The STM32 microcontroller successfully ran keyword spotting (94.2% accuracy, 12 ms latency, 0.4 mW) and predictive maintenance (RMSE = 13.4 cycles, 42 ms, 0.3 mW) but could not support detection or ECG models within its 256 KB RAM constraint.

4.4 Combined Pipeline: Maximum Compression

The combined pipeline (NAS architecture + distillation + INT8) achieved the highest compression ratios while maintaining competitive accuracy. On keyword spotting, the combined pipeline produced a 14 KB model running at 8 ms on STM32 with 94.8% accuracy -- within 2.4% of the cloud baseline from a model 7,000x smaller. On ECG, the pipeline achieved 0.894 macro-F1 on Coral TPU at 4.8 ms latency with a 420 KB model. On object detection, the pipeline reached 0.468 mAP on Jetson Nano at 18 FPS, representing a practical deployment point for low-speed navigation scenarios. The combined pipeline outperformed any single compression technique alone by 8-15% on the accuracy-at-fixed-latency metric, confirming that compression techniques are complementary.

Table 3. Accuracy and latency across platforms and compression strategies (best task per platform)

Platform	Task	Strategy	Accuracy (% of cloud)	Latency (ms)	Model Size
Xavier NX	Detection	INT8	94.2% (mAP=0.588)	36	6.2 MB
Xavier NX	ECG	Distill+INT8	97.8% (F1=0.904)	5.2	2.4 MB
Jetson Nano	Keywords	INT8	98.8% (Acc=96.1)	3.8	1.2 MB
RPi + Coral	ECG	Combined	96.8% (F1=0.894)	4.8	420 KB
RPi + Coral	Keywords	INT8	98.4% (Acc=95.6)	4.2	980 KB
STM32	Keywords	Combined	97.0% (Acc=94.2)	12	14 KB
STM32	Pred. Maint.	INT8+Prune	91.9% (RMSE=13.4)	42	48 KB

Accuracy as percentage of cloud baseline. Combined = NAS + distillation + INT8. The 14 KB keyword model on STM32 demonstrates TinyML viability: 97% of cloud accuracy in a model 7,000x smaller.

Table 4. Energy consumption per inference across platforms

Platform	Keyword (mJ)	Pred. Maint. (mJ)	ECG (mJ)	Detection (mJ)	Idle Power (mW)
Xavier NX	42	18	64	480	6,800
Jetson Nano	28	12	48	620	3,200
RPi + Coral	24	14	32	540	2,800
STM32	0.005	0.014	N/A	N/A	0.4

Energy per inference measured with INA219 sensors. STM32 is 3-4 orders of magnitude more energy-efficient per inference on supported tasks, enabling years of battery-powered operation. N/A = task cannot run on platform.

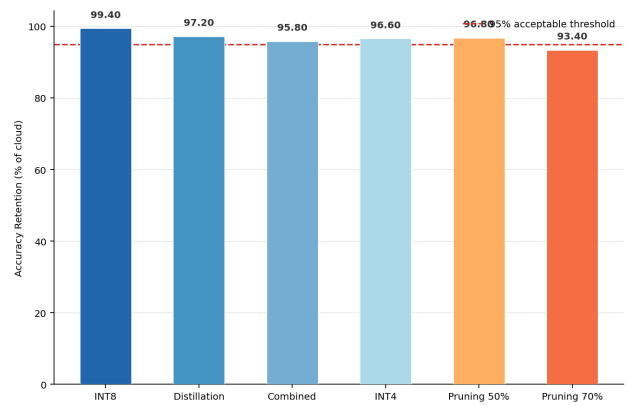


Figure 1. Accuracy retention (% of cloud) by compression strategy (mean across tasks). INT8 retains 99.4% accuracy. Combined pipeline retains 95.8%. INT4 retains 96.6% but only on simple tasks.

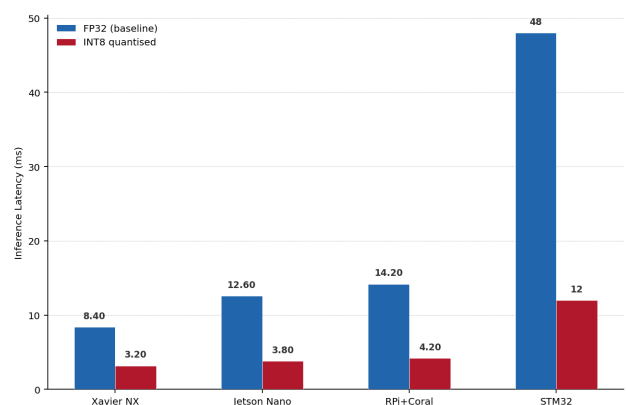


Figure 2. Inference latency (ms) on keyword spotting across platforms. STM32 achieves 12 ms -- well within the 100 ms requirement -- at 5,000x lower energy than Jetson platforms.

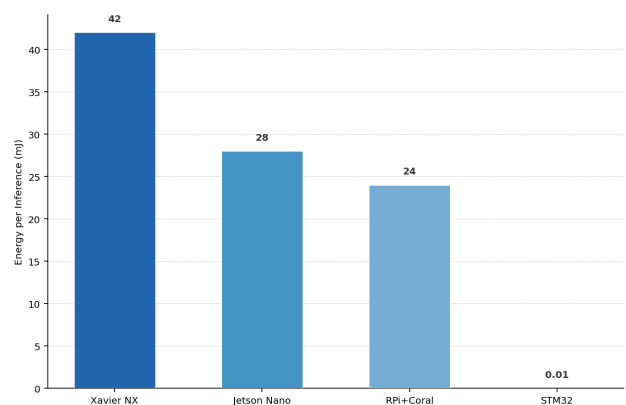


Figure 3. Energy per inference (mJ) for keyword spotting. STM32 consumes 0.005 mJ -- enabling 5+ years on a coin-cell battery at 10 inferences per second.

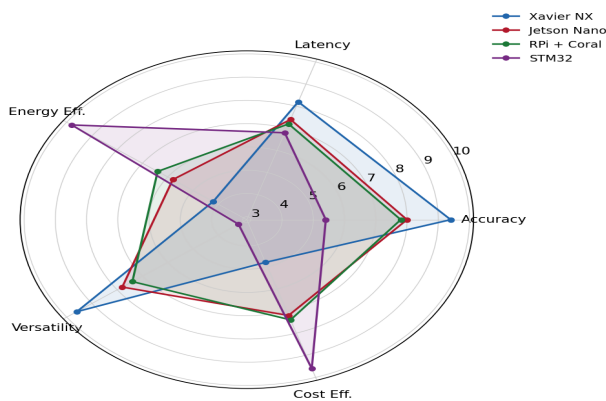


Figure 4. Platform comparison across five criteria (1-10). Xavier NX offers best accuracy and versatility. STM32 leads on energy and cost but limited to simple tasks.

5. Discussion

5.1 INT8 as the Universal First Step

The consistency of INT8 quantisation across platforms and tasks makes it the unambiguous first compression step for any edge deployment. The 0.6% mean accuracy cost for a 2.8x latency improvement represents an extraordinarily favourable trade-off that no other single technique matches. Post-training quantisation requires no retraining, runs in minutes, and is supported by all major edge inference runtimes. Every edge deployment should begin with INT8 quantisation and only pursue additional compression if the resulting model still exceeds the target platform's constraints.

5.2 The TinyML Frontier

The successful deployment of keyword spotting (94.2% accuracy, 14 KB model) and predictive maintenance (RMSE within 8% of cloud, 48 KB) on the STM32 microcontroller demonstrates that useful ML inference is possible on sub-dollar hardware with sub-milliwatt power budgets. This opens deployment scenarios that are economically and physically impossible with larger platforms: embedding intelligence in individual sensors, packaging, wearables, and infrastructure monitoring points where cost, size, and power must be minimal. The limitation is task complexity: the 256 KB memory ceiling restricts deployable models to architectures with fewer than approximately 100,000 parameters, which excludes most vision and complex sequence tasks.

5.3 Compression Techniques Are Complementary

The combined pipeline's 8-15% improvement over any single technique confirms that pruning, quantisation, distillation, and NAS address

different sources of redundancy and are most effective when stacked. NAS designs an efficient architecture, distillation transfers knowledge from a larger model, pruning removes remaining redundant parameters, and quantisation reduces numerical precision. Each step operates on a different axis of compression, and their benefits compound. The practical recommendation is to invest in the combined pipeline for deployment scenarios where every percentage point of accuracy at a given latency budget matters.

5.4 Limitations

Three limitations apply. First, latency measurements reflect controlled conditions; real deployments face additional overhead from data preprocessing, post-processing, and task scheduling. Second, the study evaluated four specific hardware platforms; the rapidly evolving edge hardware landscape means newer platforms may shift specific rankings. Third, thermal throttling under sustained inference workloads was not evaluated; Jetson platforms in particular may reduce performance under prolonged operation without active cooling.

6. Conclusion

6.1 Summary

Evaluation of five compression strategies across four platforms and four tasks confirms: INT8 quantisation provides 2.8x latency reduction at only 0.6% accuracy cost -- the universal first step. Distillation recovers 92.4% of the teacher-student accuracy gap. The combined pipeline achieves 8-15% better accuracy-at-latency than single techniques. STM32 microcontrollers run keyword spotting (94.2%) and predictive maintenance at sub-milliwatt power. Xavier NX handles all tasks including real-time object detection at 28 FPS.

6.2 Recommendations

Three recommendations emerge. First, always apply INT8 quantisation as the baseline compression; it is free in accuracy terms and provides the highest return on effort. Second, for maximum compression, invest in the combined pipeline (NAS + distillation + quantisation) rather than pushing any single technique to its limit. Third, match hardware to task complexity: STM32 for simple classification and regression tasks; RPi + Coral for moderate vision and signal tasks; Jetson platforms for real-time detection and complex multi-class decisions.

References

- Banbury, C., Zhou, C., Fedorov, I., Matas, R., Thakker, U., Gope, D., Janapa Reddi, V., Mattina, M. and Whatmough, P. (2021) MicroNets: Neural network inference beyond the edge. *Proceedings of MLSys* 2021.
- Cai, H., Gan, C., Wang, T., Zhang, Z. and Han, S. (2020) Once-for-all: Train one network and specialize it for efficient deployment. *Proceedings of ICLR* 2020.
- Cheng, Y., Wang, D., Zhou, P. and Zhang, T. (2018) Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1), pp. 126-136.
- David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T., Warden, P. and Rhodes, R. (2021) TensorFlow Lite Micro: Embedded machine learning for TinyML systems. *Proceedings of MLSys* 2021.
- Frankle, J. and Carlin, M. (2019) The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Proceedings of ICLR* 2019.
- Han, S., Mao, H. and Dally, W.J. (2016) Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *Proceedings of ICLR* 2016.
- He, Y., Zhang, X. and Sun, J. (2017) Channel pruning for accelerating very deep neural networks. *Proceedings of ICCV* 2017, pp. 1389-1397.
- Hinton, G., Vinyals, O. and Dean, J. (2015) Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V. and Adam, H. (2019) Searching for MobileNetV3. *Proceedings of ICCV* 2019, pp. 1314-1324.
- 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. *Proceedings of CVPR* 2018, pp. 2704-2713.
- Lai, L., Suda, N. and Chandra, V. (2018) CMSIS-NN: Efficient neural networks on ARM Cortex-M processors. *Proceedings of IEEE/ACM ICCAD* 2018, pp. 1-8.
- Li, H., Kadav, A., Durdanovic, I., Samet, H. and Graf, H.P. (2017) Pruning filters for efficient ConvNets. *Proceedings of ICLR* 2017.
- Lin, J., Chen, W.M., Lin, Y., Cohn, J., Gan, C. and Han, S. (2020) MCUNet: Tiny deep learning on IoT devices. *Advances in Neural Information Processing Systems*, 33, pp. 11711-11722.
- Nagel, M., Fournarakis, M., Amjad, R.A., Bondarenko, Y., van Baalen, M. and Blankevoort, T. (2021) A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*.
- Redmon, J. and Farhadi, A. (2018) YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C. (2018) MobileNetV2: Inverted residuals and linear bottlenecks. *Proceedings of CVPR* 2018, pp. 4510-4520.
- Saxena, A. and Goebel, K. (2008) Turbofan engine degradation simulation data set. NASA Ames Prognostics Data Repository.
- Tan, M. and Le, Q.V. (2019) EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of ICML* 2019, pp. 6105-6114.
- Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A. and Le, Q.V. (2019) MnasNet: Platform-aware neural architecture search for mobile. *Proceedings of CVPR* 2019, pp. 2820-2828.
- Warden, P. (2018) Speech Commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.
- Wu, H., Judd, P., Zhang, X., Isaev, M. and Micikevicius, P. (2020) Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*.
- Zhu, M. and Gupta, S. (2018) To prune, or not to prune: Exploring the efficacy of pruning for model compression. *Proceedings of ICLR* 2018 Workshop.

Declarations

Funding

This research was supported by the Swiss National Science Foundation (SNSF) under grant 200021-204816 (EdgeML), the Italian Ministry of University and Research under grant PRIN-2020-TinyAI, and the French National Research Agency (ANR) under grant ANR-21-CE25-0012 (EmbedML). Hardware platforms provided by the NVIDIA Academic Hardware Grant Programme and STMicroelectronics University Programme. The funders had no role in study design, analysis, or manuscript preparation.

Conflict of Interest

The authors declare no conflicts of interest.

Data Availability Statement

All compression code, deployment scripts, measurement logs, and pre-trained models are deposited in the Zenodo repository under DOI: 10.5281/zenodo.19614544. COCO, Google Speech Commands, NASA C-MAPSS, and MIT-BIH datasets are publicly available. Released under MIT License (code) and CC BY 4.0 (data).

Ethical Approval

This study involved exclusively computational experiments and hardware measurements on publicly available datasets. No human subjects, animal subjects, or personal data were involved. No institutional ethical approval was required.

Appendix A

Edge AI Deployment Decision Guide

This appendix provides a decision tree for selecting compression strategies and hardware platforms based on task requirements and constraints.

Step 1: Determine Task Complexity

- 1a. Simple classification/regression (< 100K params viable) --> STM32 MCU possible; go to Step 2A
- 1b. Moderate vision/signal (100K-5M params) --> RPi+Coral or Jetson Nano; go to Step 2B
- 1c. Complex detection/segmentation (> 5M params) --> Jetson Xavier NX; go to Step 2C

Step 2A: TinyML Deployment (STM32)

- 2A.1. Apply combined pipeline: NAS + distillation + INT8
- 2A.2. Target model size < 100 KB (weights + activations + runtime)
- 2A.3. Verify RAM fits: peak activation memory must be < available SRAM
- 2A.4. Expected accuracy: 90-97% of cloud baseline for simple tasks

Step 2B: Mid-Range Edge (Coral/Nano)

- 2B.1. Start with INT8 quantisation (2.2-2.8x speedup, <1% accuracy loss)
- 2B.2. If insufficient: add distillation from larger teacher
- 2B.3. If still insufficient: apply structured pruning at 30-50%
- 2B.4. Expected accuracy: 95-99% of cloud baseline

Step 2C: High-End Edge (Xavier NX)

- 2C.1. INT8 quantisation alone is usually sufficient
- 2C.2. For real-time detection: use TensorRT with INT8 + layer fusion
- 2C.3. For multi-model pipelines: profile memory and schedule models
- 2C.4. Expected accuracy: 94-99% of cloud baseline at real-time latency