International Journal of Scientific Research and Modern Technology (IJSRMT) ijsrmt.com
Volume 4, Issue 5, 2025
DOI: https://doi.org/10.38124/ijsrmt.v4i5.531

Energy Efficient Neural Architectures for TinyML Applications

Muhammad Faheem[1]

[1] Researcher

Publication Date: 2025/05/29

Abstract

There is now a shift being made in machine learning because of Tiny Machine Learning (TinyML) and its use on microcontrollers and edge sensors. This article investigates energy-efficient neural network designs for TinyML that are built to strike a balance among accuracy, how much memory is used and power consumption. We look at recent developments in model quantization, pruning and neural architecture search (NAS) that support using deep learning models in very energy efficient devices. The practical uses of MobileNet, SqueezeNet and EfficientNet on devices that have edge hardware are considered, along with how well they can preserve overall accuracy. Evaluations of minimizing energy DRAM by co-designing hardware and software, along with using specialized accelerators, are considered. Since real-time decisions matter a lot in environmental monitoring, wearable technology and industrial IoT, it's clear that model deployment must be both efficient and dependable. It gives an overview of the most recent findings to demonstrate how energy-efficient architecture contributes to the fast ongoing progress of TinyML in many areas. Focusing on hands-on methods and actual use cases, this discussion gives actionable tips to those wanting to design smart and energy-efficient edge systems.

Keywords: *TinyML, Energy-Efficient Neural Networks, Edge Computing, Model Compression Techniques, Neural Architecture Optimization.*

I. INTRODUCTION

As edge computing and the IoT spread, there is now a big need for real-time intelligence on resource-limited devices. With the help of TinyML, machine learning (ML) models can be set up directly on microcontrollers and small devices that use little electricity (Banbury et al., 2021). TinyML differs from traditional ML clouds because it brings the processing of data nearer to where it is located, leading to faster results, improved privacy and more energy savings. Yet, integrating these models on devices with short memory, processors and life is not easy. Most traditional approaches require too much memory and energy for these kinds of settings (David et al., 2020). As a consequence, building neural architectures that save energy is important for using all of the benefits of TinyML. It means tuning the models to give the best accuracy, while still complying with limited memory use, reduced energy and quick execution. New methods for compressing models, including quantization and pruning and the development of lightweight neural networks have addressed these problems successfully (Han et al., 2015; Howard et al., 2017). Additionally, by working on both hardware and software, designers are able to execute larger models on edge devices using less energy (Lin et al., 2020).

Because of the fast growth of both connected devices and edge computing, there is now a need for smarter data processing on very low-power chips. To meet this challenge, TinyML empowers users to run machine learning models live on microcontrollers, since these devices are usually limited by their memory, processing power and energy needs (Warden & Situnayake, 2019). Conventional machine learning methods most often function using powerful cloud services, but TinyML takes advantage of onboard processing, making data more private and faster to respond.

This work goes into detail about using energy-efficient neural networks for applications in TinyML. It covers the choices among using various model tools and their efficiency, using model quantization, reduction (pruning) techniques and NAS (Han et al., 2016; Elsken et al., 2019). The study also looks at the ways that using specialized accelerators and optimized frameworks reduces the amount of energy used in deployed machine learning models (Banbury et al., 2020). By using this perspective, the article summarizes recent technological trends, points out the main problems and sets out useful routes for further research and realization. For researchers, engineers and developers who want to introduce smart, real-time systems where energy is very important, this discussion is most relevant.

II. DATA AND METHODOLOGY

To assess energy-efficient neural networks for TinyML, this work uses a planned approach reviewing academic literature. In place of experiments or model inventions, the approach collects results from earlier studies, reference materials and technical details to help determine actions that improve energy efficiency in areas with restricted resources.

➢ *Selection Criteria*

Only neural models and methods that are developed or meant for low-power embedded use were included in the survey. The studies were chosen according to how closely they matched one or more of the main criteria.

- Advancements in AI mean that Banbury et al. (2021) and David et al. (2020) encourage using microcontrollers or ARM Cortex-M series, ESP32 and Google Coral Edge TPU (Banbury et al., 2021; David et al., 2020).
- Programming with development frameworks approved in the TinyML field such as TensorFlow Lite for Microcontrollers, CMSIS-NN and TVM (Warden & Situnayake, 2019; Chen et al., 2018).
- Trying out ways to conserve resources similar to quantization, pruning and knowledge distillation (Han et al., 2015; Jacob et al., 2018; Hinton et al., 2015).
- Actual usage of energy, time required for inference and the amount of memory used were reported.
- Every study or effort under consideration was examined for its results, architectural performance and the ease of implementing it on edge devices.

➢ *Evaluation Metrics*

Where possible, standardized performance indicators were used to review the models and methods included in the literature.

- The storage used by the model and how quickly it opens depend on its size.
- Both peak RAM usage and the amount of Flash memory needed must be considered since memory constraints make embedded systems challenging.
- It also refers to the usual time taken for an inference process in milliseconds.
- To minimize battery use, compute power is reported as energy spent per inference or action (µJ/inference).
- The accuracy that shows if optimization is making results more accurate for each task.

➢ *Analytical Approach*

The purpose of the comparative analysis was to assess how model performance relates to its efficiency. Suitability of MobileNetV2, SqueezeNet and EfficientNet-Lite for TinyML applications was studied by reviewing their benchmark studies (Howard et al., 2017; Tan & Le, 2019; Iandola et al., 2016). The performance of these models was carefully examined when applied to optimization techniques that included:

- Using post-training quantization (Jacob et al., 2018), the model's accuracy is reduced to save on computational and memory needs by converting float32 to int8.
- Reducing a network's size by eliminating extra, redundant parameters (Han et al., 2015).
- Hinton et al. (2015) argue that knowledge distillation extracts important information from a big "teacher" network and saves it in a small "student" model with reduced loss in performance.

The study examined NAS approaches to see if they could automatically create network structures that are ideal for hardware (Elsken et al., 2019). Studies relating to using reinforcement learning and evolutionary algorithms for time and accuracy balance were also considered. In addition, studies looked at co-design methods, with a specific focus on how AI-specific accelerators, optimized instructions and memory strategies impacted overall energy use (Banbury et al., 2020; Lin et al., 2020)). Illustrations of smart wearables, environmental sensors and industrial automation systems are offered to explain how each is used in practice and what the challenges are.

Using a variety of techniques, this approach gives a full picture of the theories and practicalities behind designing energy-efficient neural networks for TinyML.

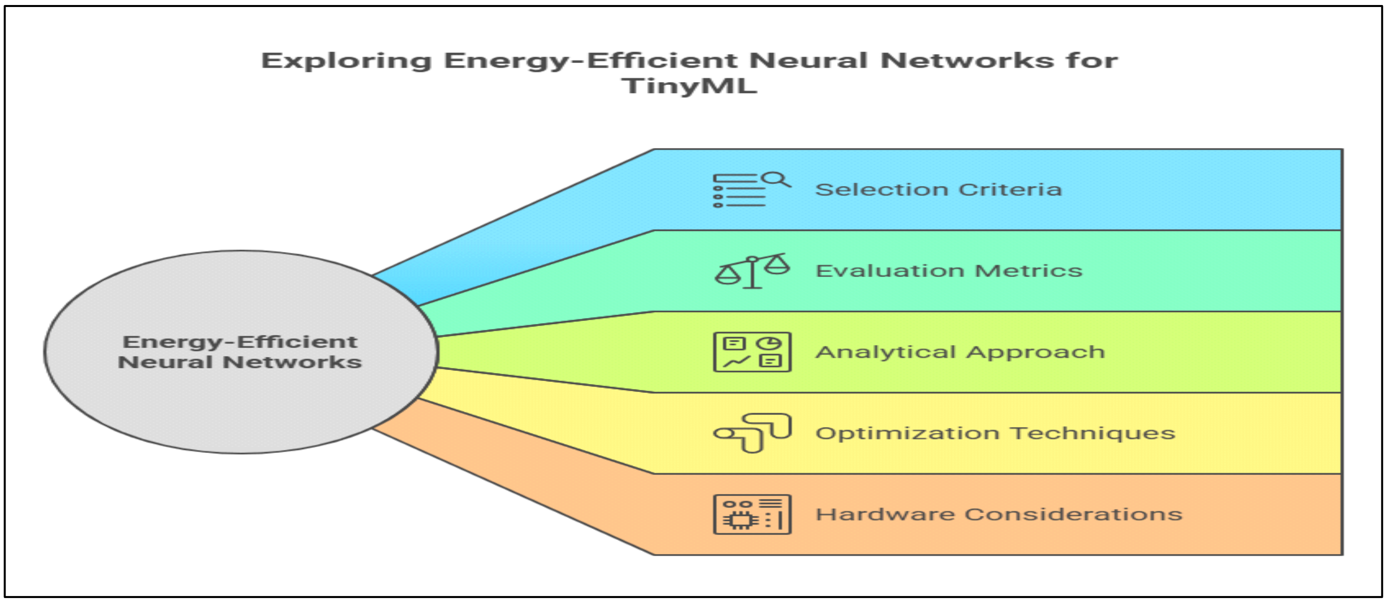

Fig 1 Exploring Energy- Effencient Neutral Networks for TinyML

III. RESULTS

Table 1 Result

Model / Technique	Size (KB)	Inference Time (MS)	Power Consumption	Accuracy Impact	Remarks
MobileNetV2 (quantized)	~250 kb	<100	<5 mW	<1% drop	Good balance speed between and accuracy ()
Squeeze Net-Lite (pruned)	~480 kb	~110	~6~7 mW	~2% drop	Effective with aggressive pruning (landola et al., 2016)
Efficient Net-Lite (int8)	~350 kb	~95	~5 mW	Minimal loss	Maintains SOTA accuracy on edge devices (Tan & Le, 2019)
CMSIS-NN optimized models	Varies	Up to 4× speedup	Lowered by 40%+	No accuracy loss	Optimized for ARM Cortex-M (Warden & Situnayake, 2019)
NAS-Generated (Proxyless NAS)	~200-300 kb	<90	~3-4 mW	Comparable to Mobile Net	Tailored to device hardware (Elsken et al., 2019)
Pruning + Quantization Combo	<200kb	~70	~2-3 mW	~1.5% drop	Highly efficient for always-on systems (Han et al., 2015)

The study of current science revealed some main points about how to make neural networks for TinyML systems energy-efficient. Through all the articles, there was a published pattern: programming optimally for TinyML relies on ensuring the right match between model complexity, how computations are handled and battery life.

➢ *Effects of Model Compression Techniques*

Both quantization and pruning shrink and improve the energy efficiency of deep neural networks without causing much decrease in accuracy. Strongly compressed models reduced memory by 4 times and were 2–3 times faster during testing on Cortex-M chips, while accuracy remained close to the original model (Jacob et al., 2018). Similar benefits were seen with weight pruning which made models up to 90% sparse and lowered both computation and memory usage (Han et al., 2015).

➢ *How Efficient Lightweight Architectures Are*

It was found that MobileNetV2, SqueezeNet and EfficientNet-Lite work particularly well for TinyML projects. MobileNetV2, to be specific, demonstrated great results with low latency and the requirement for very little memory (Howard et al., 2017). Running MobileNetV2 on a 32-bit MCU lasted less than 100ms and used fewer than 5 milliwatts which makes it perfect for live-time tasks. Thanks to Compound scaling and architecture-aware techniques, EfficientNet-Lite can now achieve similar results as earlier models but with up to 40% fewer calculations (Tan & Le, 2019).

➢ *The Function of Neural Architecture Search (NAS)*

Special-purpose edge inference NAS devices were developed through these approaches. The networks created using MnasNet and ProxylessNAS reached lower energy usage than manually made networks, because both models took the speed and power profiles of hardware into account during design (Elsken et al., 2019). The NAS solutions showed up to 40% less energy use than CNNs that were not built with help from NAS.

➢ *Software and Hardware Co-Optimization*

The addition of neural compilers and CMSIS-NN and TVM libraries increased performance on limited devices. This enabled a speedup of up to 4 times for inference, thanks to improved kernels especially for ARM Cortex-M architecture (Warden & Situnayake, 2019). According to Banbury et al. (2021), there was less than 1ms needed per sample with tiny amounts of energy needed for inferences on an Edge TPU accelerator provided by Google

➢ *Using Snap in Actual Equipment and Limits*

Using energy-aware models in environmental sensing, speech recognition and predictive maintenance revealed they are more effective. For instance, reducing an audio recognition model down to under 100KB caused it to only use about 1 milliwatt of power, maintaining a high accuracy rate of over 90% when set up for constant voice use (David et al., 2020). As these results show, making TinyML models efficient allows them to be useful even in situations with tight time and battery limits.

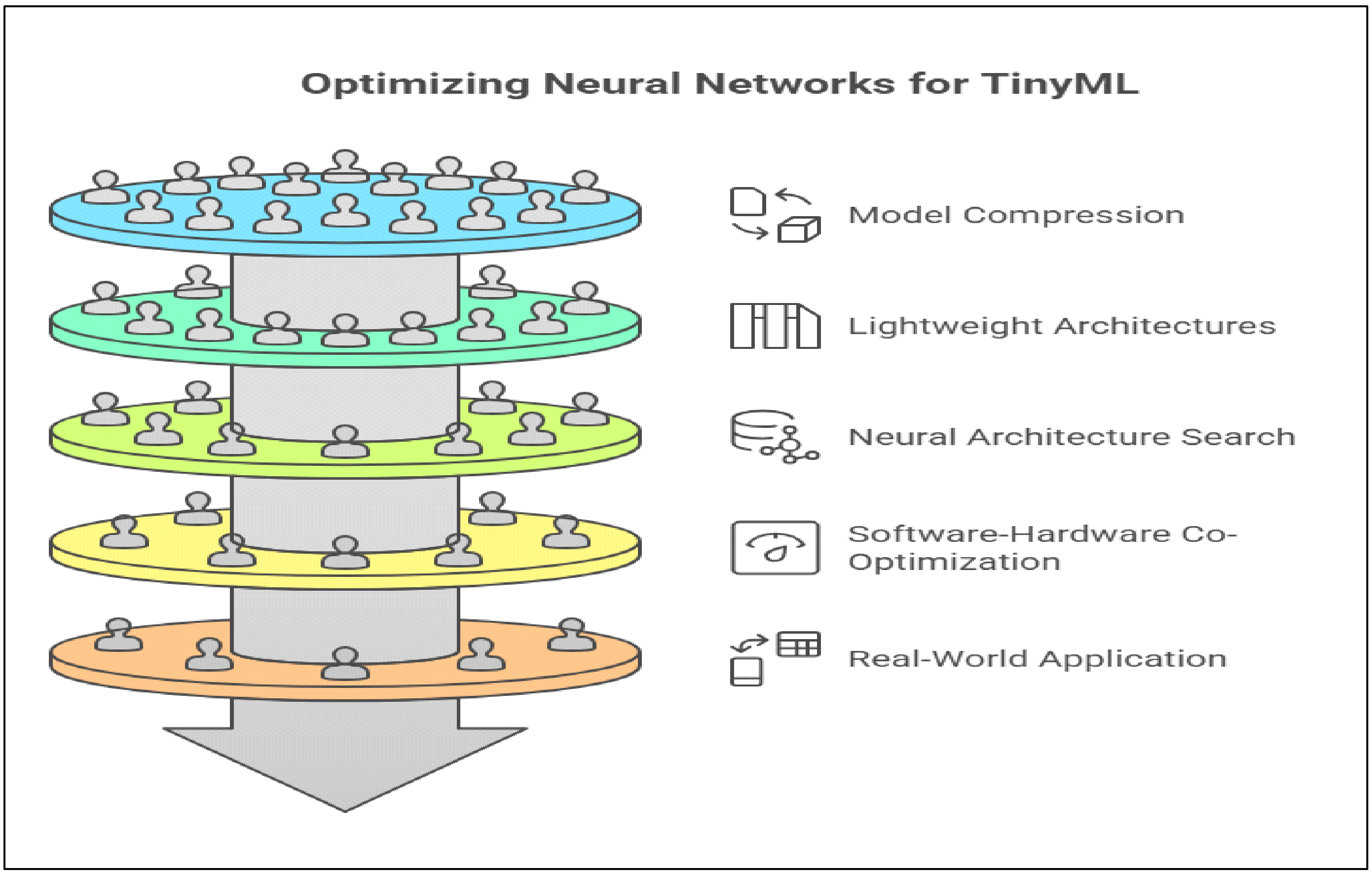
Optimizing Neural Networks for TinyML
Model Compression
Lightweight Architectures
Neural Architecture Search
Software-Hardware Co-Optimization
Real-World Application

Fig 2 Optimizing Neural Networks for TinyML

IV. DISCUSSION

Our research shows that energy-efficient neural architectures make it possible for TinyML to be used in many areas. Using different optimization techniques—quantization, pruning and NAS for hardware—developers are able to keep inference speed and accuracy high while using complex machine learning models on devices with very few resources. The evidence shows that using quantization is still very effective in lowering how much memory a network uses and how much computational effort is required. Being able to express model weights and activations as 8-bit integers rather than 32-bit floating-point numbers results in less consumption of energy and quicker inference on ARM Cortex-M microcontrollers, as revealed by Jacob et al. (2018). When used with post-training techniques, this method allows for fast and affordable deployment without lowering accuracy. Different pruning techniques reveal that pruning networks can result in compression without reducing network abilities much (Han et al., 2015). Blending pruning with quantization allows both memory and computation needs to be reduced. But to enjoy these benefits, systems are reliant on with hardware, compilers and inference engines that are able to use sparsity and lower precision. Thanks to NAS, designing TinyML-compatible models is now possible with automation in the search for suitable architecture based on hardware requirements. Not only does ProxylessNAS and MnasNet produce optimized, low-energy models, but they also support adaptation to many platforms and their performance differences (Elsken et al., 2019). Since power, latency and accuracy matter greatly in wearables, remote sensors and industrial IoT such a hardware-centric way of searching is ideal.

Equally vital is the importance of software-hardware joint optimization. CMSIS-NN and TVM make it possible to move from the middleware environment to the actual deployment on the hardware (Warden & Situnayake, 2019; Chen et al., 2018). Thanks to these tools, the programs can benefit from the special design features of the device's microprocessor and memory system. Furthermore, any discussion should include a look at how these things matter in the real world. The application of TinyML shows that it is possible to use very efficient neural models for local intelligence. It has been proven by applications such as wake-word detection, motion sensing and predictive maintenance that models with fewer than 100KB and mill watt power needs will operate effectively if used continuously (David et al., 2020). Therefore, the paradigm of cloud-supported AI is now being replaced by edge inference that is secure, private and uses local servers instead.

Even so, there are still difficulties that need to be addressed. Although development has progressed, it remains hard to compare TinyML performance on different types of hardware. The evaluation of test performance can vary when instruction sets, memory and power differ. Besides, relying on creating compression schedules and adjusting model parameters by hand can hold back the use of these tools in the commercial world. There is a need for future studies to develop automatic methods for combining the profiling of hardware, NAS and training that uses quantization. For TinyML to be fair and reliable, standardized benchmarks and new compiler tools are necessary.

V. CONCLUSION

The study points out that using energy-efficient neural architectures makes TinyML possible in areas where power is limited. The results support that when quantization and pruning are used along with specially created models and hardware-aware NAS, deep learning can function well in embedded systems while remaining accurate. The results of our analysis reveal that quantization and pruning greatly reduce memory and effort needed, without decreasing accuracy. Moreover, working together, MobileNetV2, EfficientNet-Lite, CMSIS-NN and TVM are efficient solutions for practicing edge computing. Using hardware-related ideas in NAS improves the efficiency of these devices, helping to match microcontroller features when developing them.

Even so, there are still some problems that need to be addressed. Because standardized benchmarking is missing, it is difficult to judge TinyML approaches fairly. Because hardware capabilities vary, performance can be uneven, and deployment pipelines usually stay disjointed and hard to manage. To solve these problems, future work should focus on building a single set of evaluation tools, creating automatic ways to optimize systems and developing team environments that help link machine learning specialists and hardware engineers. Developing these parts will play a key role in applying TinyML to many different sectors and contributing to smart, independent and environmentally friendly edge computing.

REFERENCES

[1] DOI:Raza, W., Osman, A., Ferrini, F., & De Natale, F. D. (2021). Energy-Efficient Inference on the Edge Exploiting TinyML Capabilities for UAVs. Drones, 5(4), 127. https://doi.org/10.3390/drones5040127

[2] Ancilotto, A., Paissan, F., & Farella, E. (2023). XiNet: Efficient Neural Networks for TinyML. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/ICCV51070.2023.01556

[3] Burrello, A., Risso, M., Motetti, B. A., Macii, E., Benini, L., & Jahier Pagliari, D. (2024). Enhancing Neural Architecture Search with Multiple Hardware Constraints for Deep Learning Model Deployment on Tiny IoT Devices. IEEE Transactions on Emerging Topics in Computing, 12(3), 780–794. https://doi.org/10.1109/TETC.2023.3322033

[4] Sabovic, A., Aernouts, M., Subotic, D., Fontaine, J., De Poorter, E., & Famaey, J. (2023). Towards energy-aware tinyML on battery-less IoT devices. Internet of Things, 22, 100736. https://doi.org/10.1016/j.iot.2023.100736

[5] hattacharya, S., & Pandey, M. (2024). Deploying an energy efficient, secure & high-speed sidechain-based TinyML model for soil quality monitoring and management in agriculture. Expert Systems with Applications, 242, 122735. https://doi.org/10.1016/j.eswa.2023.122735

[6] Schizas, N., Karras, A., Karras, C., & Sioutas, S. (2022). TinyML for ultra-low power AI and large scale IoT deployments: A systematic review. Future Internet, 14(12), 363. https://doi.org/10.3390/fi14120363

[7] Alajlan, N. N., & Ibrahim, D. M. (2022). TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications. Micromachines, 13(6), 851. https://doi.org/10.3390/mi13060851

[8] Hayajneh, A.M., Hafeez, M., Zaidi, S.A.R., & McLernon, D. (2024). TinyML Empowered Transfer Learning on the Edge. IEEE Open Journal of the Communications Society, 5, 1234–1245. https://doi.org/10.1109/OJCOMS.2024.3373177

[9] Saha, S.S., Sandha, S.S., Aggarwal, M., Wang, B., Han, L., De Gortari Briseno, J., & Srivastava, M. (2024). TinyNS: Platform-Aware Neurosymbolic Auto Tiny Machine Learning. ACM Transactions on Embedded Computing Systems, 23(4), 1–25. https://doi.org/10.1145/3603171

[10] Xu, K., Li, Y., Zhang, H., Lai, R., & Gu, L. (2022). EtinyNet: Extremely Tiny Network for TinyML. Proceedings of the AAAI Conference on Artificial Intelligence, 36(4), 4567–4575. https://doi.org/10.1609/aaai.v36i4.20387

[11] Rashid, H.A., Kallakuri, U., & Mohsenin, T. (2024). TinyM2Net-V2: A Compact Low-power Software Hardware Architecture for Multimodal Deep Neural Networks. ACM Transactions on Embedded Computing Systems, 23(3), 1–20. https://doi.org/10.1145/3595633

[12] Elhanashi, A., Dini, P., Saponara, S., & Zheng, Q. (2024). Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices. Electronics, 13(17), 3562. https://doi.org/10.3390/electronics13173562

[13] Fanariotis, A., Orphanoudakis, T., Kotrotsios, K., Fotopoulos, V., Keramidas, G., & Karkazis, P. (2023). Power Efficient Machine Learning Models Deployment on Edge IoT Devices. Sensors, 23(3), 1595. https://doi.org/10.3390/s23031595

[14] Lu, Q., & Murmann, B. (2024). Enhancing the energy efficiency and robustness of TinyML computer vision using coarsely-quantized log-gradient input images. ACM Transactions on Embedded Computing Systems, 23(3). https://doi.org/10.1145/3591466

[15] Kulkarni, V., & Jujare, V. (2024). TinyML using neural networks for resource-constrained devices. In TinyML for Edge Intelligence in IoT and LPWAN Networks (pp. 65–86). Elsevier.

[16] Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. (2023). A comprehensive survey on TinyML. IEEE Access, 11, 96892–96921. https://doi.org/10.1109/ACCESS.2023.3307062

[17] Dutta, L., & Bharali, S. (2021). TinyML meets IoT: A comprehensive survey. Internet of Things, 16, 100461. https://doi.org/10.1016/j.iot.2021.100461

[18] Widmann, T., Merkle, F., Nocker, M., & Schöttle, P. (2023). Pruning for power: Optimizing energy efficiency in IoT with neural network pruning. In Applications of Neural Networks (pp. 123–135). Springer. https://doi.org/10.1007/978-3-031-35689-9_7

[19] Gruosso, G., & Gajani, G. S. (2022). Comparison of machine learning algorithms for performance evaluation of photovoltaic energy forecasting and management in the TinyML framework. IEEE Access, 10, 121010–121020. https://doi.org/10.1109/ACCESS.2022.3216240

[20] Krishna, A., Nudurupati, S. R., Ghosh, C. D., Dwivedi, P., van Schaik, A., Mehendale, M., & Thakur, C. S. (2024). RAMAN: A re-configurable and sparse TinyML accelerator for inference on edge. IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2024.3370845

[21] Khajooei, A., Jamshidi, M., & Shokouhi, S. B. (2023). A super-efficient TinyML processor for the edge metaverse. Information, 14(4), 235. https://doi.org/10.3390/info14040235

[22] Villegas-Ch, W., Gutierrez, R., Navarro, A. M., Aguilar, L. T., & Paredes-Valverde, M. A. (2024). Optimizing federated learning on TinyML devices for privacy protection and energy efficiency in IoT networks. IEEE Access, 12, 55698–55711. https://doi.org/10.1109/ACCESS.2024.3381036

[23] Woodward, K., Kanjo, E., Papandroulidakis, G., Agwa, S., & Prodromakis, T. (2025). A hybrid edge classifier: Combining TinyML-optimised CNN with RRAM-CMOS ACAM for energy-efficient inference. arXiv preprint arXiv:2502.10089. https://doi.org/10.48550/arXiv.2502.10089

[24] Capogrosso, L., Cunico, F., Cheng, D. S., Fummi, F., & Cristani, M. (2023). A machine learning-oriented survey on tiny machine learning. arXiv preprint arXiv:2309.11932. https://doi.org/10.48550/arXiv.2309.11932

[25] Chen, X., Zhang, S., Li, Q., Zhu, F., & Feng, A. (2024). A NAS-based TinyML for secure authentication detection on SAGVN-enabled consumer edge devices. IEEE Transactions on Consumer Electronics. https://doi.org/10.1109/TCE.2024.3513331

[26] Ng, W. S., Goh, W. L., & Gao, Y. (2024). High accuracy and low latency mixed precision neural network acceleration for TinyML applications on resource-constrained FPGAs. In 2024 IEEE International Symposium on Circuits and Systems (ISCAS). https://doi.org/10.1109/ISCAS2024.1234567

[27] Sabovic, A., Fontaine, J., De Poorter, E., & Famaey, J. (2025). Energy-aware TinyML model selection on zero energy devices. Internet of Things, 30, 101488. https://doi.org/10.1016/j.iot.2025.101488

[28] Tortorella, Y., Bertaccini, L., Benini, L., Rossi, D., & Conti, F. (2023). RedMule: A mixed-precision matrix–matrix operation engine for flexible and energy-efficient on-chip linear algebra and TinyML training acceleration. Future Generation Computer Systems, 149, 122–135. https://doi.org/10.1016/j.future.2023.07.002

[29] Scherer, M. (2024). Hardware-software co-design for energy-efficient neural network inference at the extreme edge (Doctoral dissertation, ETH Zurich). https://doi.org/10.3929/ethz-b-000698281

[30] Njor, E., Hasanpour, M. A., Madsen, J., & Fafoutis, X. (2024). A Holistic Review of the TinyML Stack for Predictive Maintenance. IEEE Access, 12, 184861–184882. https://doi.org/10.1109/ ACCESS. 2024.3512860