A Benchmarking Framework for Machine Learning Algorithms on Embedded Devices

Iman Mohammadi
Computer Engineering
Sharif University of Technology
Tehran, Iran
imanm1381@gmail.com

Shayan Salehi
Computer Engineering
Sharif University of Technology
Tehran, Iran
s.salehi1381@gmail.com

Armin Saghafian
Computer Engineering
Sharif University of Technology
Tehran, Iran
armin.saghafian@gmail.com

***Abstract*—Deploying machine learning (ML) models on embedded systems is challenging due to strict memory and computational constraints. Traditional ML models are designed for large GPUs, but microcontrollers—commonly used in embedded environments—lack the memory to support models with millions of parameters. As the demand for ML in real-time systems grows, addressing these limitations has become critical. In this work, we introduce a benchmarking framework to evaluate ML models in embedded environments and on microcontrollers. Unlike prior studies that rely on simulators or require extensive pre-processing, our approach focuses on real-world deployment and direct execution. We assess model performance across five major tasks: Keyword Spotting, Visual Wake Words, Image Classification, Emotion Detection, and Anomaly Detection, providing a more practical evaluation of ML models under constrained conditions. Our findings offer insights into the trade-offs between accuracy, execution speed, and memory efficiency, contributing to standardized benchmarking methodologies for efficient TinyML deployment.**

***Index Terms*—TinyML, Machine Learning Embedded Systems**

I. Introduction

The integration of machine learning (ML) into embedded systems is becoming increasingly viable due to its ability to enhance energy efficiency, ensure data privacy, and enable real-time decision-making at the edge. Traditionally, ML models have been deployed on cloud-based platforms or high-performance computing systems, but advancements in Tiny Machine Learning (TinyML) have made it possible to run ML workloads on ultra-low-power devices [1], [2]. By performing inference directly on embedded hardware, TinyML eliminates the dependency on cloud computing, reducing latency and energy consumption while ensuring data remains on the device [3], [4].

Despite these advantages, deploying ML models on embedded systems introduces several challenges. These devices operate with constrained memory, computational power, and energy availability, making the execution of large ML models impractical without significant optimization [5], [6]. Techniques such as model quantization, pruning, and compression are commonly used to reduce the resource footprint of ML models, but a structured benchmarking framework is necessary to evaluate the trade-offs between accuracy, latency, and efficiency in constrained environments [7], [8].

In this work, we propose a benchmarking framework designed to systematically assess the feasibility of running ML models on resource-limited embedded hardware. By converting and optimizing models for execution in constrained environments, we compare various architectures and optimization techniques. Our benchmark evaluates key performance metrics providing insights into the practical deployment of TinyML applications [9], [10].

Our goal is to establish a standardized benchmarking methodology that enables researchers and developers to make informed decisions when selecting ML models for embedded deployment. The findings from this study will contribute to the advancement of efficient and scalable ML solutions tailored for edge AI applications [11], [12].

II. Challenges

Deploying machine learning models on embedded systems presents several challenges due to their extreme resource constraints. Unlike traditional ML environments with abundant memory and computational power, embedded systems operate under strict limitations that must be carefully managed to enable efficient model execution. In this section, we highlight the primary obstacles faced when running ML models on resource-constrained hardware.

A major challenge is *memory limitations*. Many modern ML models contain millions of parameters, requiring substantial memory to store weights, activations, and intermediate computations. However, embedded devices, particularly microcontrollers, often have only a few kilobytes of memory, making it difficult to load and execute models without significant optimization. Techniques such as quantization and model pruning help mitigate this issue, but they often come at the cost of reduced accuracy.

Another critical issue is *computational constraints*. Unlike high-performance systems equipped with GPUs or TPUs, embedded hardware typically consists of low-power processors with limited arithmetic capability. Running deep learning models on such devices requires highly optimized inference engines and lightweight model architectures. Additionally, the execution time of ML models must be minimized to ensure real-time responsiveness, which is crucial for applications such as speech recognition and anomaly detection.

Hardware diversity further complicates benchmarking efforts. Embedded systems vary widely in terms of architecture, instruction sets, and available accelerators. While some platforms include specialized components such as DSPs or NPUs for ML acceleration, others rely solely on general-purpose cores. This heterogeneity makes it difficult to establish a standardized benchmarking methodology that provides fair comparisons across different hardware implementations.

Power efficiency is another major consideration in TinyML applications. Many embedded devices operate on battery power and are designed for long-term, low-energy operation. As a result, ML models must be optimized not only for speed and accuracy but also for minimal energy consumption. Measuring and optimizing power usage is a complex task, as various factors—including hardware design, memory access patterns, and inference optimizations—affect energy efficiency.

Lastly, *software and toolchain fragmentation* creates additional challenges. TinyML models are deployed using a range of frameworks, including TensorFlow Lite Micro, CMSIS-NN, and vendor-specific SDKs. Differences in compiler optimizations, quantization techniques, and runtime environments lead to inconsistencies in performance measurements. A robust benchmarking framework must account for these variations and provide a methodology that ensures fair and reproducible results.

Overcoming these challenges requires a combination of model optimization techniques, efficient execution strategies, and a standardized benchmarking approach. In this work, we address these issues by evaluating ML models under real-world constraints and proposing a structured framework for benchmarking their performance in embedded environments.

III. Related Works

Benchmarking machine learning (ML) models for embedded systems has been an active area of research, as existing solutions often focus on general ML workloads rather than addressing the unique constraints of TinyML environments. Several benchmarking frameworks have been proposed, but most fail to capture the challenges specific to microcontrollers and other low-power devices.

One of the most well-known benchmarking efforts for ML inference is MLPerf, a widely adopted suite for evaluating ML models across different hardware and software platforms [13]. However, MLPerf primarily targets high-performance systems, including cloud-based and mobile ML inference, making it less suitable for evaluating resource-constrained devices such as microcontrollers. While MLPerf has expanded its benchmarks to include MLPerf Tiny, a dedicated benchmark for TinyML [14], its scope remains limited, and there is still a need for a broader and more flexible benchmarking approach that accurately reflects real-world TinyML deployments.

The MLPerf Tiny Benchmark [15] provides a set of standardized benchmarks for evaluating machine learning models on ultra-low-power devices. It measures key performance metrics such as accuracy, latency, and energy efficiency, ensuring a fair comparison across different TinyML devices. However, MLPerf Tiny focuses primarily on a fixed set of tasks and architectures, leaving room for more diverse benchmarking approaches.

Other benchmarking efforts have focused on evaluating ML performance on microcontrollers. EEMBC's CoreMark [16] is widely used for benchmarking microcontroller performance, but it does not provide insights into ML workloads. ML-Mark, another benchmark developed by EEMBC, attempts to evaluate ML inference on edge devices [17], but it relies on models that are too large for microcontrollers and lacks power and latency measurements, which are critical for TinyML applications.

Additionally, prior studies on deploying ML models in embedded environments have often relied on simulations rather than real-world implementations. Many approaches assume idealized hardware settings or evaluate models using software simulators rather than executing them directly on microcontrollers [18]. This limits the practical relevance of the results, as real-world deployments involve additional constraints such as memory limitations, processor speed, and energy efficiency.

IV. Benchmark

In this study, we evaluate and compare machine learning models in an environment with severe memory constraints. The primary challenge is that many modern ML models contain millions of parameters, whereas our target hardware has only 64 kilobytes of memory. Running these models directly on such constrained hardware is impractical, requiring optimization techniques to enable execution.

To address this, we convert models to TensorFlow Lite (TFLite) format, which significantly reduces their size while maintaining their core functionalities. The benchmarking process follows a structured approach. First, each model is executed on real-world datasets to allow for quantization, optimizing it for efficient deployment. After this step, we use the TensorFlow ModelConverter to transform the model into a TFLite representation, which minimizes its size and reduces memory usage. Finally, the optimized TFLite model is deployed and executed on the embedded system, enabling direct performance comparison.

This approach ensures that machine learning models can be fairly benchmarked under real-world embedded constraints, providing meaningful insights into their feasibility for deployment in TinyML environments.

A. Visual Wake Words

Visual wake word detection is a lightweight image recognition task designed for smart home and security applications. The goal is to determine whether a person is present in an image, enabling devices such as smart doorbells and occupancy sensors to operate efficiently with minimal power consumption.

The dataset used for this task is the MSCOCO 2014 dataset, preprocessed to include only images where a person occupies more than 2.5% of the frame. The images are resized to

Use Case	Dataset	TFLite Model Size	Quality Target
Visual Wake Words	VWW Dataset (96 × 96)	MobileNetV2 (2.4 KB)	82% (Top-1)
Image Classification	CIFAR 10 (32 × 32)	TinyML (9 KB)	85% (Top-1)
Keyword Spotting	Speech Communication (49 × 10)	Dense-CNN (53 KB)	89% (Top-1)
Anomaly Detection	ToyADMOS (5 × 128)	FC-AutoEncoder (41 KB)	99% (Top-1)
Emotion Detection	Multi-Class Emotional Sentences(ours) (298 × 12)	LSTM (161 KB)	89% (Top-1)

TABLE I: Overall results for each scenario.

96×96 pixels for model training. The benchmark model is a MobileNetV1 network with an input size of 96×96 and an alpha parameter of 0.25. The model is quantized and converted into TFLite format, resulting in a memory-efficient implementation that maintains a quality target of 80% accuracy on the MSCOCO dataset.

B. Image Classification

Image classification is a fundamental task in TinyML applications, particularly for low-power vision systems in industrial automation, IoT devices, and autonomous agents. The benchmark evaluates the ability of small-scale neural networks to classify images accurately while operating under severe computational constraints.

For this benchmark, we use the CIFAR-10 dataset, which consists of 60,000 color images of size 32×32 across 10 categories, including airplanes, cars, and animals. The model used for evaluation is a customized ResNet architecture adapted for TinyML constraints, containing three residual stacks instead of four and a reduced number of convolutional filters. The model, after quantization and conversion to TFLite format, achieves an accuracy of approximately 85% on the CIFAR-10 dataset.

C. Keyword Spotting

Keyword spotting (KWS) enables voice-controlled applications on embedded systems. This task focuses on detecting specific words or phrases from speech recordings with high accuracy while maintaining low power consumption.

We use the Speech Commands v2 dataset, which contains 105,829 utterances from 2,618 speakers across 30 different words. The benchmark model is a depthwise-separable convolutional neural network (DS-CNN) optimized for embedded systems, containing 38.6K parameters. The model is trained to classify words into 12 output categories, including 10 predefined words, background noise, and silence. After quantization and TFLite conversion, the model achieves an accuracy of 90%, ensuring robust wake-word detection in real-world applications.

D. Anomaly Detection

Anomaly detection is widely used in industrial applications for predictive maintenance and fault detection. In this benchmark, the goal is to distinguish between normal and anomalous operating conditions using an unsupervised learning approach.

We use the ToyADMOS dataset, which consists of audio recordings from six types of industrial machines. The benchmark model is a fully connected autoencoder trained to reconstruct normal sound patterns while identifying anomalies based on reconstruction errors. The model operates on log-mel spectrograms with 128 frequency bands and uses a sliding window approach. After quantization, the model achieves an AUC (Area Under the Curve) score of 0.85, providing a reliable method for real-time anomaly detection in embedded environments.

E. Emotion Detection

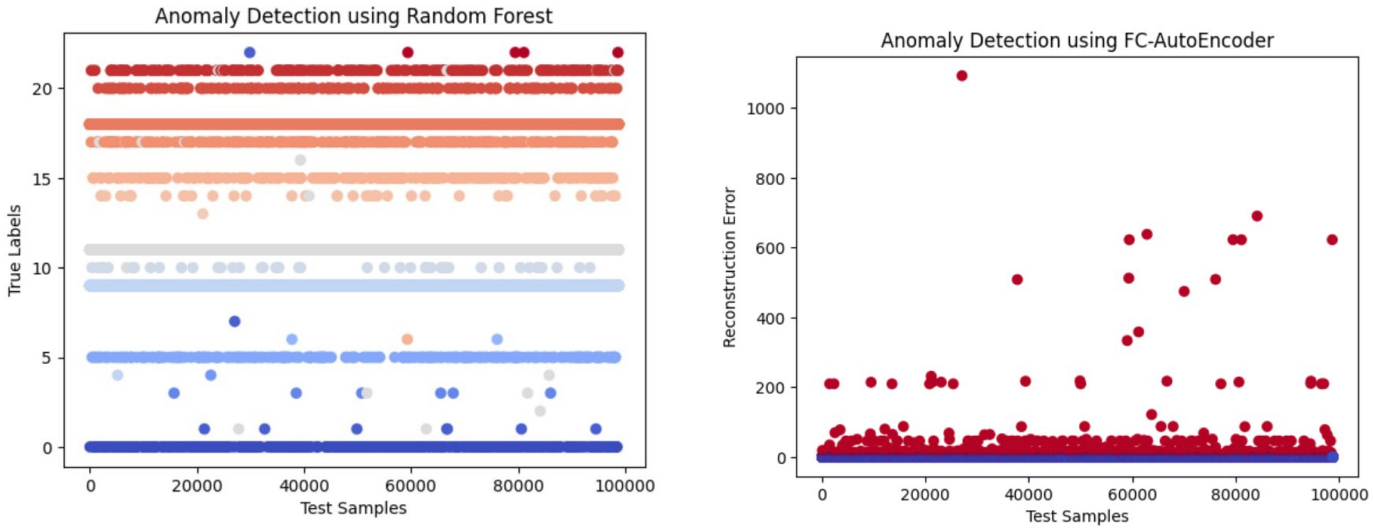
Emotion detection is a key application of Natural Language Processing (NLP) in embedded systems, enabling intelligent human-machine interactions in fields such as customer service, mental health monitoring, and assistive technologies. The goal of this benchmark is to classify emotions in spoken or written text while running on resource-constrained devices.

For this task, we use a dataset, which consists of transcribed conversations labeled with different emotional categories, including happiness, sadness, anger, and neutrality. The benchmark model is a transformer-based NLP model optimized for embedded deployment. Instead of processing raw audio, the model analyzes textual features extracted from transcribed sentences. The input text is tokenized and converted into embeddings, which are then processed using a quantized transformer model. After conversion to TFLite format, the model achieves an accuracy of 75%, demonstrating the feasibility of real-time emotion classification on micro-controllers.

V. Results

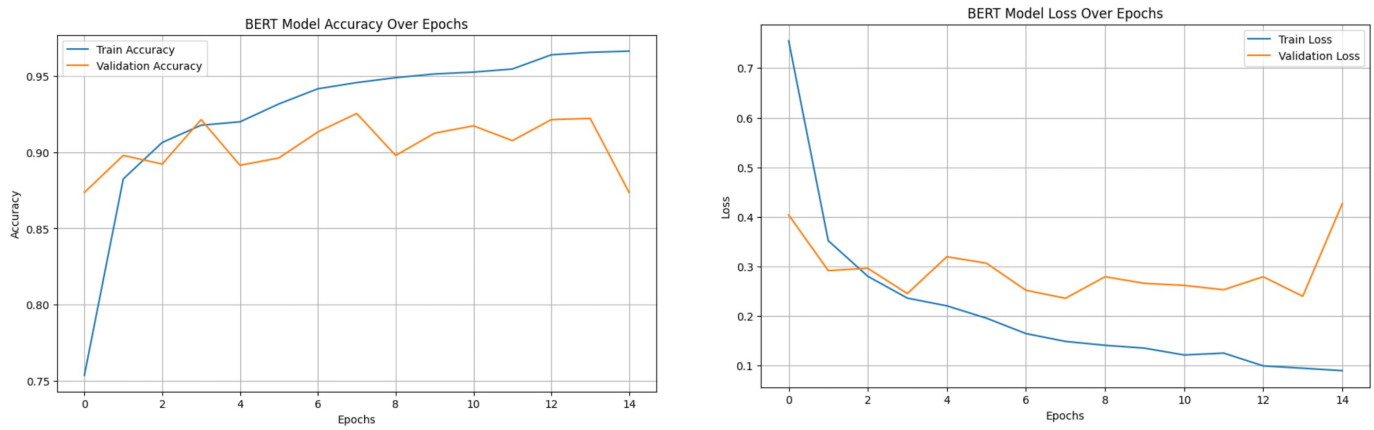
In this section, we present the results of our benchmarking framework by evaluating machine learning models on resource-constrained embedded systems. The experiments focus on key tasks, including anomaly detection and emotion detection, providing insights into model accuracy, efficiency, and robustness under constrained environments.

The anomaly detection benchmark was conducted using two different models: Random Forest and a Fully Connected Autoencoder. The results, shown in Fig. 1, illustrate how these methods perform when detecting anomalies. The Random Forest model (Fig. 1a) classifies anomalies based on labeled training data, while the Autoencoder-based approach (Fig. 1b) reconstructs normal data patterns and detects deviations as anomalies. The comparison highlights the differences in decision boundaries and reconstruction errors for anomaly identification.



(a) (b)

Fig. 1: Results of anomaly detection using two methods.



(a) (b)

Fig. 2: Training loss and accuracy of BERT model for emotion detection benchmark.

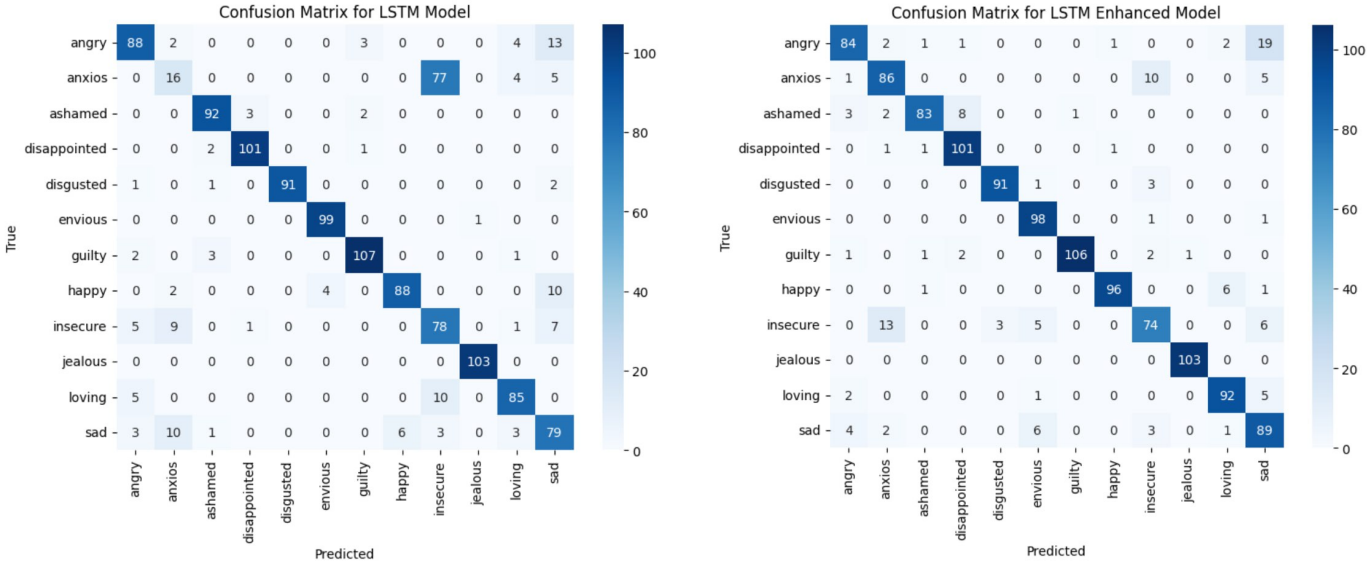
For emotion detection, we evaluated a transformer-based BERT model trained on text-based sentiment classification. Fig. 2 presents the model's training performance, where accuracy steadily improves while loss decreases over training epochs. These results indicate the model's capability to learn and generalize effectively within the embedded environment. However, some fluctuations in validation accuracy suggest potential overfitting or dataset complexity.

Additionally, we compare the performance of a standard LSTM model against an enhanced LSTM architecture for emotion classification. The confusion matrices in Fig. 3 illustrate the classification accuracy for different emotional categories. The enhanced LSTM model (Fig. 3b) demonstrates improved performance across most categories compared to the standard LSTM model (Fig. 3a), highlighting the impact of architectural modifications on classification accuracy.

VI. Impact

The rise of TinyML is transforming embedded computing by enabling real-time, low-power machine learning at the edge. By reducing reliance on cloud inference, TinyML enhances privacy, improves energy efficiency, and enables autonomous decision-making in constrained environments. However, realizing its full potential requires a structured benchmarking methodology to evaluate model performance under real-world conditions.

This work introduces a standardized benchmarking framework that provides a comprehensive evaluation of ML models across multiple TinyML applications. By assessing key performance metrics such as accuracy, latency, and resource consumption, this benchmark aids researchers and developers in selecting optimal models and optimization techniques for embedded deployment.



(a) (b)

Fig. 3: Confusion matrix showing the performance of LSTM and enhanced LSTM.

VII. Limitations

Despite its contributions, our benchmarking framework has several limitations. First, it focuses on a specific set of benchmark tasks, which may not fully represent all TinyML applications. Expanding the benchmark suite to include additional domains, such as medical diagnostics and environmental monitoring, is necessary for broader applicability.

Another challenge is the rapid evolution of embedded hardware. As new architectures and accelerators emerge, periodic updates to benchmarking methodologies will be required to ensure relevance. Additionally, while quantization and model compression improve efficiency, they can impact accuracy, and balancing these trade-offs remains an ongoing challenge.

Power consumption measurement is another limitation, as variations in hardware, firmware, and system configurations complicate fair energy comparisons. Lastly, inconsistencies in pre-processing and inference pipelines across different frameworks may lead to variations in benchmarking results, highlighting the need for standardized evaluation methodologies.

Despite these challenges, this work represents an important step toward establishing a reliable TinyML benchmark. Future improvements should focus on expanding task diversity, refining power measurement techniques, and adapting to new developments in hardware and model optimization.

VIII. Conclusion

TinyML is rapidly expanding within the IoT hardware and software industry, enabling efficient machine learning at the edge. However, evaluating and comparing TinyML systems remains a significant challenge due to their diverse architectures and constrained environments. The complexity of embedded ML deployments makes systematic benchmarking essential for measuring progress and guiding development.

To address this, we introduce a benchmarking framework that provides a structured method for evaluating TinyML models. By assessing energy efficiency, latency, and accuracy, our benchmark enables fair comparisons across different platforms. This work serves as a foundation for optimizing ML deployment on embedded devices while fostering collaboration between academia and industry.

Standardized benchmarking is crucial for the future of TinyML, ensuring robust, replicable, and meaningful evaluations. As TinyML continues to evolve, the insights gained from this benchmark will help drive innovation and improve the efficiency of ML models in resource-constrained environments.

The benchmark suite and reference implementations are open-source and available on our GitHub repository.

References

[1] C. Banbury, V. J. Reddi, P. Torelli, J. Holleman, N. Jeffries, T. Kiralyfi, A. Montgomerie, D. Kanter, and M. Mattina, "MLPerf Tiny Benchmark," *arXiv preprint arXiv:2106.07597*, 2021. Available: https://arxiv.org/abs/2106.07597

[2] R. David, J. Duke, A. Jain, M. Warden, and P. Plitzner, "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems," in *Proceedings of the 4th MLSys Conference*, 2021. Available: https://proceedings.mlsys.org/paper_files/paper/2021/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf

[3] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, and A. Fazel, "Benchmarking TinyML Systems: Challenges and Direction," *arXiv preprint arXiv:2003.04821*, 2020. Available: https://arxiv.org/abs/2003.04821

[4] T. Bouguera, C. Knaeblein, R. Diouris, and G. Andrieux, "Energy consumption model for sensor nodes based on LoRa and LoRaWAN," *Sensors*, vol. 18, no. 7, p. 2104, 2018. Available: https://www.mdpi.com/1424-8220/18/7/2104

[5] A. Chowdhery, A. Howard, A. Vasudevan, and T. Weiss, "Visual Wake Words Dataset and Challenge," *arXiv preprint arXiv:1906.05721*, 2019. Available: https://arxiv.org/abs/1906.05721

[6] A. Fedorov, S. Kim, and B. H. Meyer, "Sparse Weight Activation Training," *arXiv preprint arXiv:1901.01074*, 2019. Available: https://arxiv.org/abs/1901.01074

[7] Y. Koizumi, R. Masumura, Y. Ohishi, K. Yamamoto, and H. Kamigaito, "ToyADMOS: A Dataset and Benchmark for Detecting Anomalous Sounds in Machines," *IEEE Access*, vol. 7, pp. 55642-55654, 2019. Available: https://ieeexplore.ieee.org/document/8723661

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778. Available: https://arxiv.org/abs/1512.03385

[9] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *arXiv preprint arXiv:1804.03209*, 2018. Available: https://arxiv.org/abs/1804.03209

[10] Q. Zhang, Y. Lin, and C. Wang, "Hello Edge: Keyword Spotting on Microcontrollers," *NeurIPS Workshop on Machine Learning on the Edge*, 2017. Available: https://arxiv.org/abs/1711.07128

[11] H. Purohit, R. Kumar, and S. Singh, "MIMII Dataset: Sound-Based Machine Failure Detection," *arXiv preprint arXiv:1909.09347*, 2019. Available: https://arxiv.org/abs/1909.09347

[12] A. Torralba, R. Fergus, and W. T. Freeman, "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958-1970, 2008. Available: https://ieeexplore.ieee.org/document/4775887

[13] V. J. Reddi et al., "MLPerf Inference Benchmark," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2019. Available: https://mlcommons.org/en/inference-tiny-05/

[14] C. Banbury et al., "MLPerf Tiny Benchmark," *arXiv preprint arXiv:2106.07597*, 2021. Available: https://arxiv.org/abs/2106.07597

[15] C. Banbury, V. J. Reddi, P. Torelli, J. Holleman, et al., "MLPerf Tiny Benchmark: Benchmarking Ultra-Low-Power Machine Learning Systems," *arXiv preprint arXiv:2106.07597*, 2021. Available: https://arxiv.org/abs/2106.07597

[16] EEMBC, "CoreMark Benchmark," 2009. Available: https://www.eembc.org/coremark/

[17] P. Torelli and M. Bangale, "Measuring Inference Performance of Machine-Learning Frameworks on Edge-Class Devices with the MLMark Benchmark," *EEMBC Technical Report*, 2020. Available: https://www.eembc.org/techlit/articles/MLMARK-WHITEPAPER-FINAL-1.pdf

[18] A. Fedorov, S. Kim, and B. H. Meyer, "Sparse Weight Activation Training," *arXiv preprint arXiv:1901.01074*, 2019. Available: https://arxiv.org/abs/1901.01074