International Journal of Intelligent Systems

WILEY

RESEARCH ARTICLE OPEN ACCESS

Resource-Constrained Keyword Spotting With Quantized CNNs: Implementation Strategies and Performance Trade-Offs

Patrik Medur[1] | Mart Lubbers[2] | Goran Mauša[1]

[1]University of Rijeka, Faculty of Engineering, Rijeka, Croatia | [2]Institute for Computing and Information Sciences, Radboud University, Nijmegen, the Netherlands

Correspondence: Goran Mauša (goran.mausa@uniri.hr)

Received: 30 January 2026 | **Revised:** 11 July 2026 | **Accepted:** 20 July 2026

Keywords: ESP32-S3-EYE | keyword spotting | LiteRT | quantization | tiny machine learning

ABSTRACT

Resource-constrained embedded devices require specialized optimization strategies to achieve reliable real-time keyword spotting performance. This work presents a comprehensive implementation framework for deploying quantized convolutional neural networks on the ESP32-S3-EYE development board, achieving practical capabilities within severe memory and computational limitations. The implementation utilizes Google's Speech Commands Dataset V0.02 with MFCC preprocessing, following the standard 12-class evaluation convention, 10 target keywords together with 2 auxiliary classes to ensure comparability with established KWS benchmarks. The deployed model achieves 89.58% accuracy in laboratory conditions with a quantized TensorFlow Lite (LiteRT) memory of 0.68 MB, enabling deployment on edge devices with limited memory resources. A dual-core system architecture implements FreeRTOS-based task distribution, dedicating one core to continuous audio capture, while the second handles feature extraction and inference operations. Voice activity detection with adaptive noise floor monitoring triggers processing when speech exceeds background noise levels. Comprehensive validation reveals substantial performance differences between controlled and real-world conditions. Testing with 13 non-native English speakers demonstrates 81.82% recognition accuracy. Synthetic speech evaluation using 10 AI voices achieves 90.06% accuracy, confirming that acoustic consistency significantly impacts recognition performance. The tiny machine learning framework enables practical keyword spotting on resource-constrained embedded systems for IoT applications.

1 | Introduction

Recent advances in microcontroller technology have enabled sophisticated machine learning applications on devices with ultra-low resources [1, 2], opening new possibilities for intelligent embedded systems. This shift toward edge-based artificial intelligence allows devices to process data locally, reducing communication overhead and improving response times while addressing privacy concerns associated with cloud-based processing. Voice-controlled systems have found applications in various domains, from home automation for controlling electrical appliances [3] to sophisticated wake word detection systems. However, implementing complex neural networks on edge devices requires careful optimization to accommodate strict

Abbreviations: CNN, convolutional neural networks; FFT, fast Fourier transform; FreeRTOS, free real-time operating system; IoT, internet of things; KWS, keyword spotting; MFCC, Mel-frequency cepstral coefficients; VAD, voice activity detection.

resource constraints, including limited memory, processing power, and energy consumption [4]. The challenge lies in maintaining acceptable performance while adapting models originally designed for resource-rich environments to operate within the severe limitations of embedded platforms.

Voice recognition systems usually operate in a two-phase architecture, balancing performance and resource efficiency. In the first phase, the edge device continuously monitors audio input using lightweight keyword spotting (KWS) models, consuming minimal power while waiting for specific wake words [5]. Upon detection, the system based on tiny machine learning activates a second phase employing a more advanced and computationally demanding speech recognition system, typically hosted on cloud servers or a more powerful local device. This architectural division addresses the fundamental challenge of embedded systems, always maintaining functionality within severe constraints of memory, computational capacity, and power consumption. By delegating complex natural language processing to resource-rich environments while keeping wake word detection local, the system achieves both responsiveness and energy efficiency essential for practical deployment.

Several studies have explored crucial optimization techniques and deployment strategies specifically for edge-based KWS systems. Zhang et al. [6] demonstrated that neural network architectures can be optimized to fit within the memory and compute constraints of microcontrollers without sacrificing accuracy, achieving 95.4% accuracy with a depthwise separable convolutional neural network (CNN) on resource-constrained devices. Sørensen et al. [7] implemented a 10-word KWS system based on a depthwise separable CNN classifier on a low-power ARM Cortex M4 microprocessor, targeting real-time applications with quantization-aware training and 8-bit deployment. Amoh and Odame [8] introduced the embedded gated recurrent unit specifically optimized for ARM Cortex M0+ microcontrollers, achieving 60 × faster execution and 10 × smaller size compared to a standard gated recurrent unit while maintaining 95.3% classification accuracy for KWS tasks. Bushur and Chen [9] demonstrated the implementation challenges of neural network architectures for KWS on resource-constrained edge devices, specifically highlighting that systems such as the ESP32-WROOM-32E with dual cores operating at 240 MHz and limited memory with 520 kB SRAM and 16 MB flash require careful optimization to ensure neural networks can fit within flash storage when inactive and execute efficiently in RAM during real-time operation. More recently, Kim et al. [10] proposed BC-ResNet, a broadcasted residual architecture that reaches up to 98.0% accuracy on Google Speech Commands with as few as 9.2 K parameters by operating on 40-dimensional log–Mel spectrogram inputs, representing the current best accuracy–oriented compact KWS. These studies provide valuable insights into edge-based specific optimization strategies.

Our previous research [11] established foundational optimization techniques for KWS models, focusing on neural network architecture design, quantization methods, and performance trade-offs. The present work extends this foundation by implementing and validating these optimization strategies on actual hardware. To align the evaluation with the conventions adopted across the KWS literature, the model architectures developed in that study were retrained for the standard 12-class Google Speech Commands configuration, 10 target keywords together with the "silence" and "unknown" classes rather than the broader 35 class used previously, ensuring that the deployed model is directly comparable with established benchmarks.

The primary objective of this paper is to deploy optimized CNN models for audio keyword detection on the ESP32-S3-EYE development board, demonstrating practical implementation of tiny machine learning models on resource-constrained embedded systems. The technical contribution of this paper is twofold. First, through a controlled on-device comparison with BC-ResNet, we show that model compactness alone does not guarantee efficient deployment, as shown by inference latency measures on deployed 8-bit quantized models. Second, we validate the deployed system beyond laboratory conditions, using non-native speakers and synthetic voices, to compare the gap between benchmark accuracy and real-world recognition performance. This research provides comprehensive implementation steps that bridge the gap between theoretical model development and practical embedded deployment. We present detailed strategies for managing computational resources, optimizing memory utilization, and maintaining real-time performance through a multithreaded architecture that efficiently distributes processing across available cores.

2 | Methodology

Deploying optimized KWS models on embedded hardware demands careful consideration of real-time processing constraints and memory limitations. This work extends theoretical optimization foundations [11] to practical implementation, encompassing hardware platform selection, real-time pipeline adaptation, and deployment strategies tailored for microcontroller environments. This study presents a complete implementation pipeline, as summarized in Figure 1, illustrating the transition from offline model development in Python's TensorFlow (TF) framework [11] to real-time deployment on the ESP32-S3-EYE platform. The pipeline begins with raw audio data from Google's Speech Commands Dataset, which undergoes preprocessing to extract Mel-frequency cepstral coefficient (MFCC) features that serve as compact spectral representations [12]. The resulting 13 × 50 MFCC matrices are fed into the CNN model for training and validation within the Python/TF environment. Following model training and evaluation, the optimized TF models undergo conversion to TF Lite (LiteRT) format, a lightweight framework specifically designed for mobile and embedded devices. This conversion process optimizes the model graph by fusing operations, folding constants, and removing unused nodes, while also eliminating training-specific operations and preparing the model for resource-constrained deployment [13]. Posttraining quantization transforms the models from 32-bit floating-point to 8-bit integer precision, achieving optimal memory–performance trade-offs essential for embedded systems. The quantized models are then converted to a C array format and integrated into the ESP32-S3-EYE firmware using the LiteRT for Microcontrollers runtime (TFLM). TFLM provides a C++ inference library optimized for microcontrollers, supporting only inference operations without the full TF framework overhead. This specialized runtime enables efficient model execution on devices with limited memory and computational resources [14]. The deployment results in real-time KWS inference operating

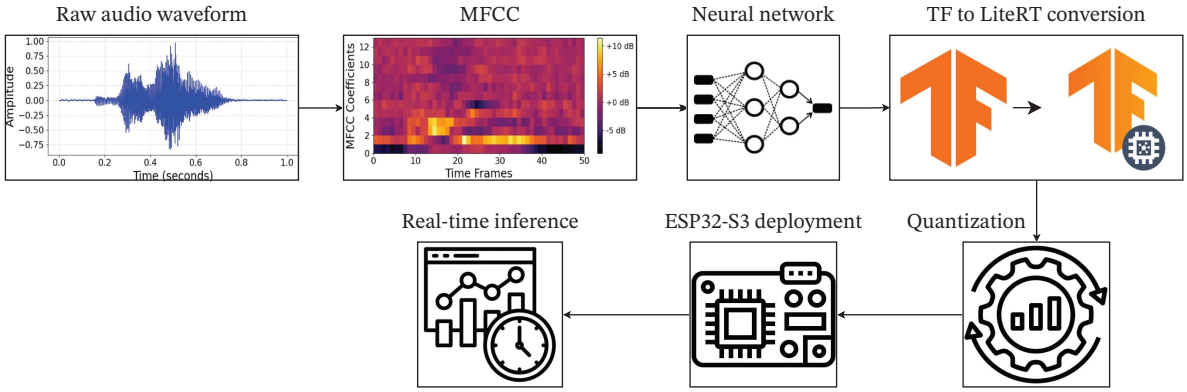

FIGURE 1 | Complete KWS pipeline: the workflow demonstrates the transition from offline model training with MFCC-processed audio data in TensorFlow to real-time deployment on ESP32-S3 hardware through LiteRT conversion, quantization, and TFLM integration for embedded inference.

within the microcontroller's computational and memory constraints, demonstrating a successful transition from offline model development to practical embedded implementation.

2.1 | Implementation Platform

The ESP32-S3-EYE development board serves as the deployment platform for this study, featuring a dual-core Xtensa LX7 processor running at 240 MHz, 512 kB SRAM, 8 MB PSRAM, and 8 MB Quad SPI flash. The board's I^2S microphone and sufficient memory resources make it well suited for prototyping real-time audio processing and neural network inference. This platform represents typical constraints found in modern IoT devices, providing a realistic environment for evaluating KWS implementation strategies.

The dual-core architecture enables efficient task distribution, with one core dedicated to continuous audio capture while the second handles feature extraction and neural network inference. The ESP32-S3's support for TFLM, combined with hardware-accelerated integer operations, makes it particularly suitable for deploying quantized neural networks. In addition, the platform's widespread adoption in commercial IoT applications ensures that optimization strategies developed on this hardware translate directly to real-world deployment scenarios, making performance results practically relevant for embedded KWS systems.

2.2 | Dataset and Preprocessing

The proposed framework utilizes the Google Speech Commands Dataset V0.02 [15], with an MFCC preprocessing pipeline established in previous work [11], demonstrating optimal performance for resource-constrained KWS applications. The dataset contains 105,829 audio recordings, each 1 s long, across 35 command words; 2618 speakers, with each contributor typically recording individual words multiple times, providing robust diversity for embedded deployment validation.

This study adopts the standard 12-class evaluation convention introduced by Warden [15]. Target keywords are as follows: "yes," "no," "up," "down," "left," "right," "on," "off," "stop," and "go," together with 2 auxiliary classes, "silence" and "unknown." This is the configuration most widely reported in the embedded KWS literature and replaces the broader 35-word formulation used in our preliminary study [11], directly enabling comparison with reference architectures. The auxiliary classes are constructed as follows. The unknown class is populated by randomly sampling utterances from the remaining 25 nontarget words. The silence class is generated by segmenting the background noise recordings provided with the dataset into one-second clips. Class proportions follow the Warden [15] convention of approximately balanced representation, with the auxiliary classes each contributing roughly 10% of the data and the ten target words making up the remainder.

The preprocessing steps follow the established framework from our previous research [11], transforming raw audio into 13×50 MFCC feature matrices suitable for CNN input. Key preprocessing parameters include a 25 ms window size, a 20 ms step size, and 13 cepstral coefficients, operating on audio downsampled to 8 kHz. These parameters were selected as an efficiency trade-off for always-on microcontroller deployment rather than as a maximal accuracy configuration. Whereas for accuracy-oriented architectures such as BC-ResNet, which use a higher-dimensional log Mel spectrogram, our 13-coefficient MFCC representation at 8 kHz yields a substantially smaller input tensor, which in turn reduces the size of the first convolutional layer, the peak activation memory held in the tensor arena, and the number of multiply-accumulate operations per inference. This configuration is therefore optimized specifically for resource-constrained deployment.

The dataset undergoes 10-fold cross-validation with 80/10/10 splits for training, validation, and testing. This three-way partitioning enables model optimization on training data, hyperparameter selection using validation data, and unbiased performance assessment on held-out test data. Speaker independence is enforced across partitions, with all utterances from a given speaker confined to a single split so that no speaker appears in both the training and test data.

Since native English speakers are rare in the target deployment region, the deployed system was evaluated on locally collected recordings from predominantly non-native speakers. These data are used only for on-device testing, never for training, and reflect the realistic user population; the native-trained, non-native tested setting is treated as a deliberate robustness assessment.

While the training pipeline employs standard floating-point processing with full-featured libraries, embedded deployment necessitates real-time preprocessing adaptations to accommodate ESP32-S3 computational constraints. These implementation-specific modifications, including simplified filtering operations and integer-based arithmetic, are detailed in

the next section to ensure compatibility between trained models and deployed systems while meeting real-time performance requirements.

2.3 | KWS System Architecture

The selection of CNN architecture is based on a comprehensive analysis of different designs for KWS models [11], in which we tested 10 architectures, ranging from simple single-layer setups to complex multilayer networks with up to three convolutional blocks and 1024 dense layer neurons, providing performance baselines for embedded deployment.

The deployed model is a compact 2D CNN that takes the 13 $\times$ 50 single-channel MFCC matrix as input. It comprises 3 convolutional blocks; each block applies a 2D convolution with a 3 $\times$ 3 kernel size and 32/64/128 filters, followed by ReLU activation and 2D max pooling for spatial dimensionality reduction. The resulting feature maps are flattened and passed to one dense layer of 128 units with ReLU activation, with dropout 0.5 applied for regularization, and a final softmax layer over the 12 output classes. The network contains 684,172 trainable parameters.

A critical aspect of embedded deployment involves model optimization through posttraining quantization [16]. Converting the trained TF model to quantized LiteRT format transforms 32-bit floating-point weights and activations to 8-bit integers, enabling efficient deployment on embedded systems with limited memory and computational resources.

Prior investigation identified the smallest architecture surpassing 80.0% accuracy (Model 4 from [11]) as the initial candidate for the embedded implementation due to its compact 0.16 MB quantized size, suitable for severely memory-constrained environments. However, practical testing on the ESP32-S3-EYE revealed that the platform's 8 MB PSRAM and dual-core architecture can accommodate larger models while maintaining acceptable inference times for real-time operation. Consequently, we selected the architecture with the optimal accuracy–efficiency trade-off (Model 7 from [11]), retrained for the 12 class configuration. This model achieves 90.04% accuracy and a 0.9033 F1-score at a size of 7.9 MB. Quantized LiteRT reduces the model to 0.68 MB, a reduction of approximately 91% while maintaining 89.58% accuracy and a 0.9037 F1-score, remaining well within the platform's memory capacity.

Comparing our selected model against a recognized architecture, we retrained BC-ResNet [10] within the same cross-validation harness and 12-class configuration. To enable a controlled comparison under identical input conditions, BC-ResNet is adapted to operate on the same 13 $\times$ 50 MFCC dataset used throughout this work rather than its original 40 bin log Mel spectrogram at 16 kHz. In its full precision float 32 form, BC-ResNet performs comparably to the selected model, reaching 89.75% accuracy and a 0.8997 F1-score with 1.1 MB size. However, after quantized LiteRT conversion, the two architectures differ. BC-ResNet's accuracy falls to 85.35% with an F1-score of 0.8569, at a size of 142.9 kB, a degradation of approximately 4.4% points, whereas the selected model retains 89.58% accuracy (F1 0.9037) with a drop of less than half a percentage point, as seen in Table 1. Although BC-ResNet yields a smaller quantized footprint, the selected architecture was selected for deployment because of its markedly greater robustness to quantization. Its quantized size of 0.68 MB remains well within the platform's 8 MB PSRAM and imposes no practical memory constraint.

Latency measurements reveal a second divergence between the two architectures. Although BC-ResNet is by far the more compact model, it executes approximately 2.5 times slower on the ESP32-S3. This arises because the selected model consists exclusively of standard convolution, pooling, and dense operators backed by optimized TFLM kernels, whereas BC-ResNet's broadcasted residual design decomposes into nonstandard operators such as `SPACE_TO_BATCH_ND` and `BATCH_TO_SPACE_ND` that fall back to unoptimized reference kernels. Execution time was measured on device using the `esp_timer` high-resolution timer. The MFCC preprocessing, identical for both models, requires (100.79 $\pm$ 0.10) ms per 1 s audio window. Inference adds (145.19 $\pm$ 0.01) ms for the selected model and (365.22 $\pm$ 0.04) ms for BC-ResNet, yielding total processing times of approximately 246 and 466 ms per decision, respectively. Both remain within a 1 s decision window, but the selected model leaves substantially more headroom for audio capture and system tasks, reinforcing its selection for deployment.

While model optimization addresses memory and computational constraints, effective embedded deployment also requires careful system architecture design to coordinate audio capture, feature extraction, and inference in real time. The ESP32-S3-EYE implementation employs a multithreaded architecture using the free real-time operating system (FreeRTOS), an open-source real-time operating system that provides deterministic task scheduling and resource management. The dual-core processor enables efficient task distribution, with Core 0 dedicated to continuous audio capture through the I^2S interface using direct memory access buffering to minimize CPU overhead. Core 1 handles computationally intensive operations, including feature extraction and neural network inference.

Voice activity detection (VAD) monitors audio energy levels against an adaptive noise floor, triggering processing when speech exceeds twice the background noise level. The VAD

TABLE 1 | Selected vs. recognized architecture model comparison.

Model	Precision	Accuracy (%)	F1-score	Size	Latency (inference)[a]
Selected Model 7	FP32	90.04	0.9033	7.9 MB	—
Selected Model 7	INT8	89.58	0.9037	0.68 MB	(145.19 $\pm$ 0.01) ms
BC-ResNet	FP32	89.75	0.8997	1.1 MB	—
BC-ResNet	INT8	85.35	0.8569	142.9 kB	(365.22 $\pm$ 0.04) ms

[a]On-device inference latency of the deployed INT8 models on the ESP32-S3-EYE at 240 MHz; mean $\pm$ standard deviation (SD) over 50 inferences after one warm-up invocation. Both models share the identical MFCC, which adds (100.79 $\pm$ 0.10) ms per 1 s window.

maintains a 200-ms preroll buffer to ensure complete keyword capture and continues recording until 400 ms of silence is detected or 1 s of audio is collected. This approach minimizes unnecessary processing while ensuring reliable keyword detection across varying acoustic environments.

2.4 | Real-Time Audio Processing

The embedded audio processing pipeline adapts the established MFCC extraction methodology for real-time operation under ESP32-S3 computational constraints. Following the VAD trigger, Core 1 executes the audio processing task to transform captured audio into MFCC features through the following pipeline:

Resampling: The 16 kHz input is downsampled to 8000 Hz using a 5-tap finite impulse response antialiasing filter. While the training pipeline employed SciPy's FFT-based resampling with Kaiser-windowed sinc interpolation, computational constraints necessitated this simplified approach. The symmetric filter coefficients (0.0625, 0.25, 0.375, 0.25, 0.0625) provide acceptable antialiasing while maintaining real-time performance.

Frame extraction: The resampled signal is segmented into overlapping frames of 25 ms with a 20 ms stride, yielding 50 frames from 1 s of audio.

Windowing: Each frame is multiplied by a Hanning window (Equation (1)) to reduce spectral leakage.

$$w[n] = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right), \tag{1}$$

where $N = 200$.

Spectral analysis: A 256-point FFT transforms each windowed frame to the frequency domain. The magnitude spectrum is computed from the complex FFT output, with only the first 128 bins retained due to symmetry.

Mel filtering: The power spectrum (squared magnitude) is filtered through 20 triangular Mel-scale filter banks spanning 300–4000 Hz. Each filter is normalized by its width to ensure consistent energy measurement across frequencies.

Logarithmic compression: Natural logarithm is applied to each Mel filter output using Equation (2) to mimic human auditory perception.

$$y = \log(\text{energy} + \epsilon), \tag{2}$$

where $\epsilon = 10^{-10}$ prevents numerical instability.

DCT transform: A Type-II discrete cosine transform extracts 13 cepstral coefficients from the log–Mel spectrum. The first coefficient is replaced with the frame's log energy, providing overall loudness information.

Feature organization: The 50 frames × 13 coefficients are transposed to produce a 13×50 feature matrix matching the model's expected input shape.

Quantization: Features are quantized to 8-bit integers by using the following equation:

$$q = \text{round}\left(\frac{x}{\text{scale}} + \text{zero}_{\text{point}}\right), \tag{3}$$

where $\text{scale} = 0.359$ and $\text{zero}_{\text{point}} = 42$.

2.5 | Model Deployment and Memory Management

Model deployment utilizes TFLM runtime integrated with Espressif's official IoT Development Framework (ESP-IDF) V5.0 for hardware abstraction. The quantized optimal Model 7 [11] is converted to C array format and embedded directly into firmware, requiring 0.67 MB of storage. The interpreter operates with a 1000 kB tensor arena allocated in PSRAM, accommodating the model execution while maintaining headroom for runtime operations.

The memory allocation strategy optimizes performance by placing computationally intensive buffers (FFT arrays, Mel filter bank coefficients, and MFCC features) in external PSRAM while keeping timing-critical structures in internal SRAM for fast access. The inference pipeline processes quantized feature matrices through integer-only arithmetic operations, dramatically reducing computational requirements compared to floating-point alternatives. This optimization enables inference completion within acceptable latency bounds for real-time KWS applications.

3 | Results

While previous work [11] established baseline computational performance metrics for model optimization, this evaluation emphasizes the transition from simulated performance to actual embedded deployment characteristics under practical operating conditions. Real-world performance evaluation examined the system performance across two complementary experimental conditions. The first experiment evaluated human speech recognition using 13 participants to assess usability, while the second experiment utilized 10 American text-to-speech AI voices to provide controlled testing conditions with consistent audio quality and pronunciation patterns.

Evaluation was conducted primarily with non-native English speakers. This reflects a deliberate design objective rather than an incidental mismatch. In the target deployment region, native English speakers are rare, and the realistic user population for an embedded voice interface consists overwhelmingly of non-native English speakers. Testing under this distribution therefore measures the system's performance for its intended users and quantifies the robustness of a model trained on native-speaker data when exposed to the accent variation it will encounter in practice. The native trained non-native tested setting is thus treated as an explicit robustness stress test and the resulting performance gap relative to controlled synthetic speech.

3.1 | Human Speech Recognition Experiment

The human speech recognition experiment assessed real-world performance of the system using 13 non-native English-speaking participants aged 18–33 years with a gender distribution of 8 male and 5 female volunteers. The standardized testing protocol involved 10 repetitions of each command word per participant, enabling evaluation of recognition accuracy across diverse speech patterns and pronunciations. In this experiment, the

deployed Model 7 [11] achieved an overall accuracy of 81.82% with an F1-score of 0.8229 across 1430 test instances, demonstrating performance of sufficient quality for real-world deployment scenarios. However, this represents a notable decrease from the theoretical accuracy of 90.04% achieved during laboratory testing with the Google Speech Commands dataset, highlighting the significant impact of real-world deployment conditions and speaker diversity on system performance.

Analysis of prediction confidence levels revealed an average confidence of 87.62% across all predictions, with correctly classified words showing a higher confidence of 91.52% compared to misclassified instances at 70.05%. The confusion matrix in Figure 2 reveals systematic misclassification patterns that provide insights into the acoustic challenges faced by the deployed system. The most frequent error involves the word "no" being misclassified as "go," which occurred 48 times, representing 18.4% of all errors, and "off" being misclassified as "on" 18 times, highlighting the difficulty in distinguishing between phonetically similar words, suggesting that the system struggles particularly with words containing similar vowel patterns. This is better illustrated in Figure 3.

This misclassification pattern is particularly pronounced among non-native English speakers, where accent-caused vowel changes create auditory signatures that deviate from the training data distribution [17]. The systematic nature of these errors suggests that targeted data augmentation strategies, incorporating accent-specific variations or phonetic transformations, could improve recognition accuracy for challenging word pairs.

To further evaluate system reliability in real-world deployment scenarios, we conducted a confidence threshold analysis using a 0.7 threshold value, as illustrated in Figure 4. When implementing a strict confidence requirement where predictions are only considered correct if both the word matches and confidence exceeds 0.7, the system's effective accuracy drops from 81.82% to 80.01%. This reveals that 122 predictions, or 8.5%, were correctly recognized, but with confidence below the threshold.

Performance analysis among individual participants revealed accuracy ranges from 74.55% to 94.55%, as seen in Figure 5, indicating substantial variation in recognition effectiveness based on individual speech characteristics. Participants with a stronger consistency of English pronunciation achieved recognition rates that resembled laboratory conditions, while those with more pronounced accent features experienced greater difficulty. This variation underscores the importance of robust training data diversity and the potential benefits of speaker adaptation techniques for deployment in multilingual environments.

Real-world deployment validation demonstrates that while the embedded KWS system achieves functional performance levels,

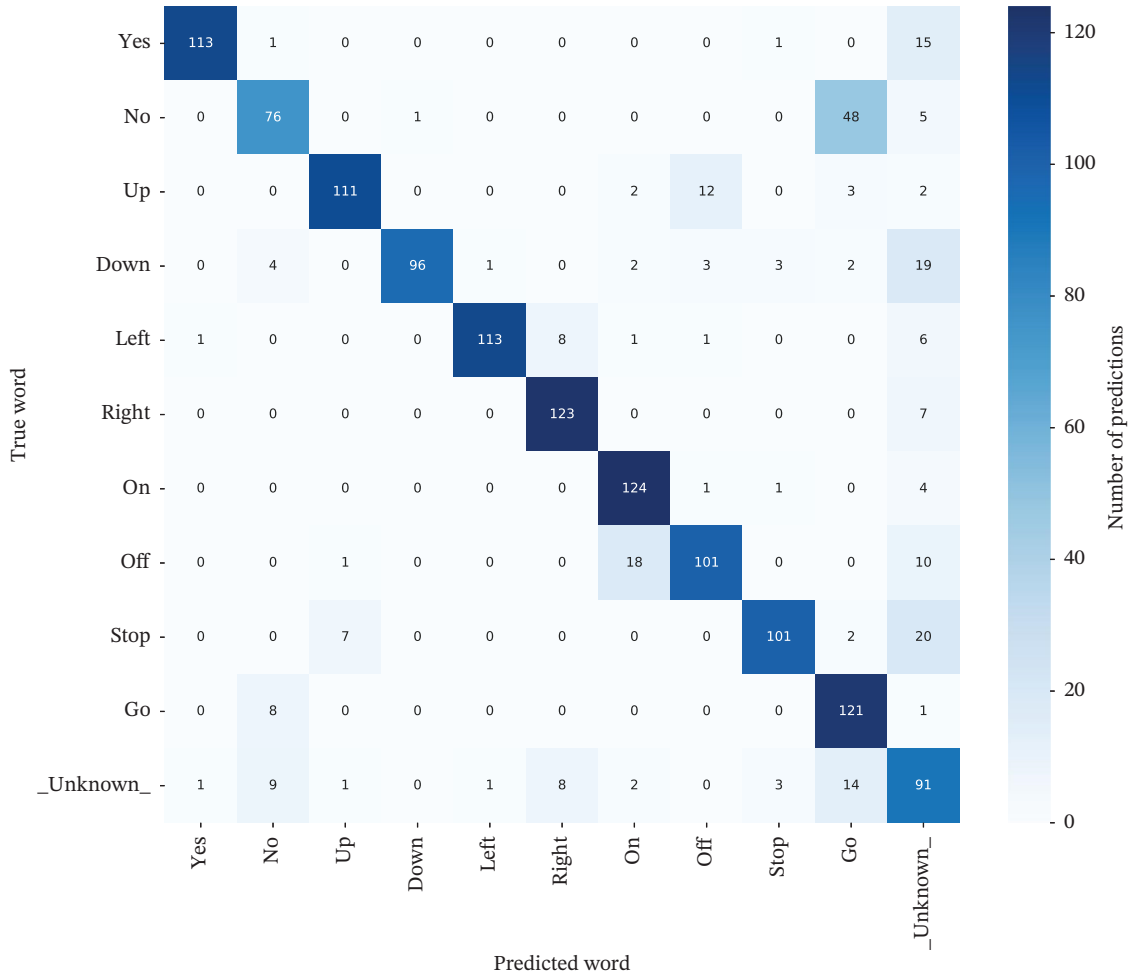

FIGURE 2 | Human speech recognition confusion matrix across 10 words and 1 unknown from 13 non-native English-speaking participants.

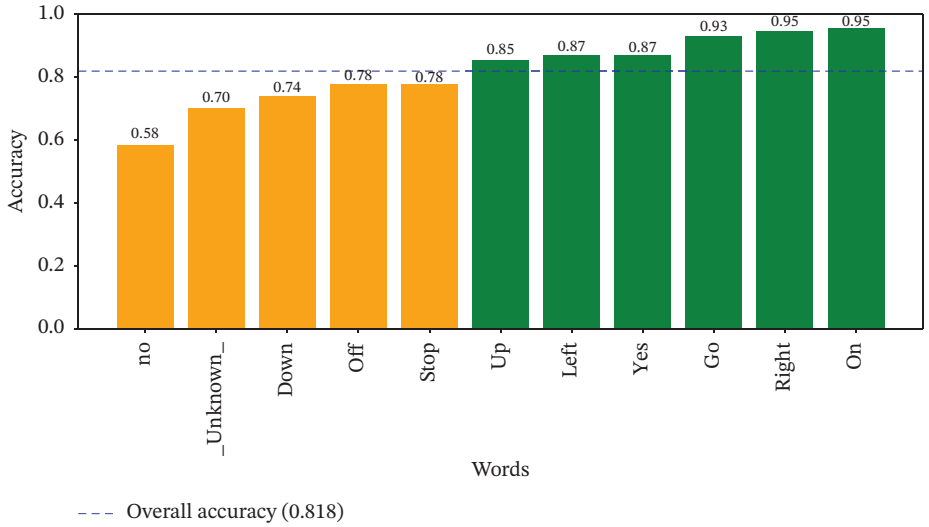

FIGURE 3 | Word-specific classification accuracy for human speech recognition, ordered by increasing accuracy, with the dashed line indicating overall accuracy of 81.82%.

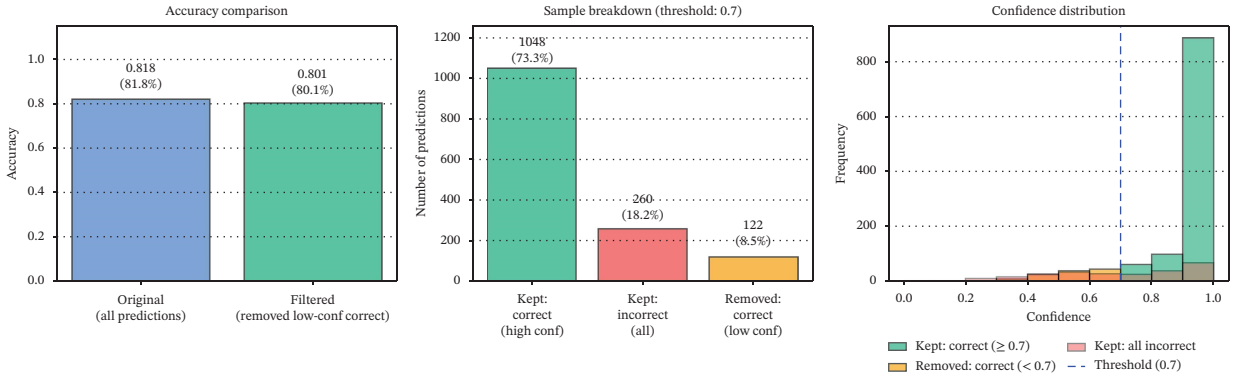

FIGURE 4 | Confidence threshold analysis for human speech recognition at 0.7 threshold, which includes the comparison of accuracy between all predictions and filtered predictions, sample breakdown by correctness and confidence level, and confidence distribution histogram.

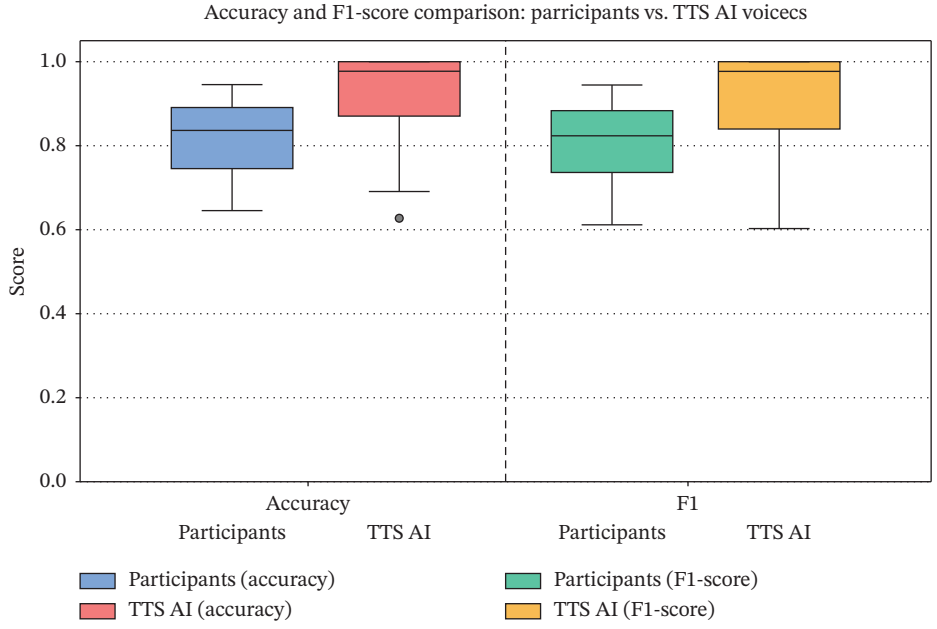

FIGURE 5 | Comparison of recognition performance between human participants and TTS AI voices in terms of accuracy and F1-score.

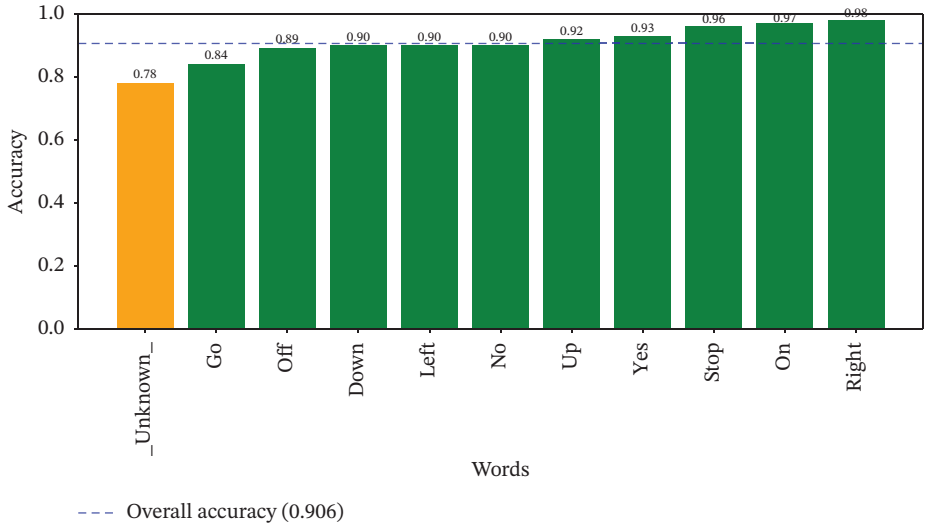

FIGURE 6 | The word-specific classification accuracy of synthetic speech, ordered by increasing accuracy, with the dashed line indicating an overall accuracy of 90.06%.

the gap between laboratory and practical conditions remains substantial. These findings emphasize the critical importance of comprehensive testing with diverse speaker populations and suggest specific optimization strategies to improve deployment reliability in real-world applications.

3.2 | Synthetic Speech Evaluation

To investigate whether the performance gap observed with human speakers stems from pronunciation variability, the system was evaluated using 10 distinct American English text-to-

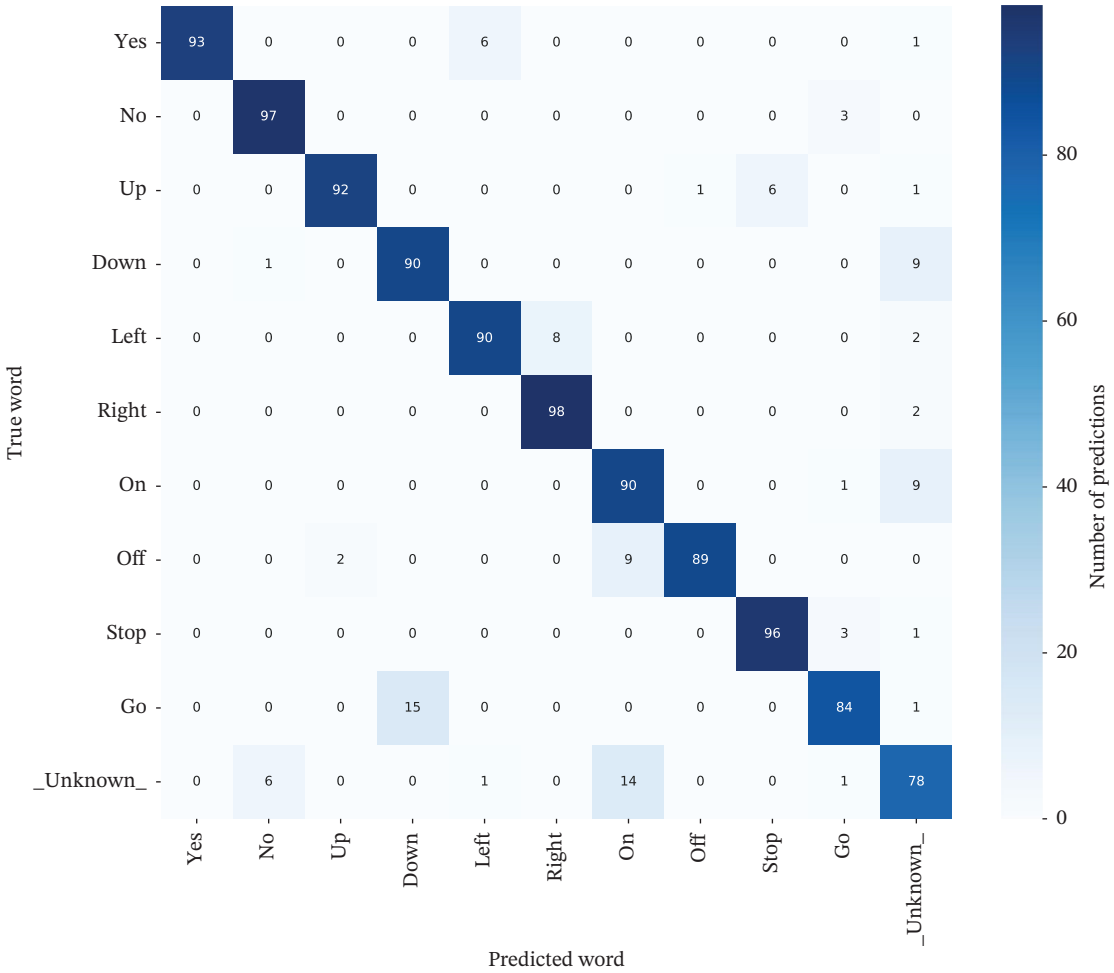

FIGURE 7 | Synthetic speech recognition confusion matrix across 11 words and 1 unknown from 10 American English TTS AI voices.

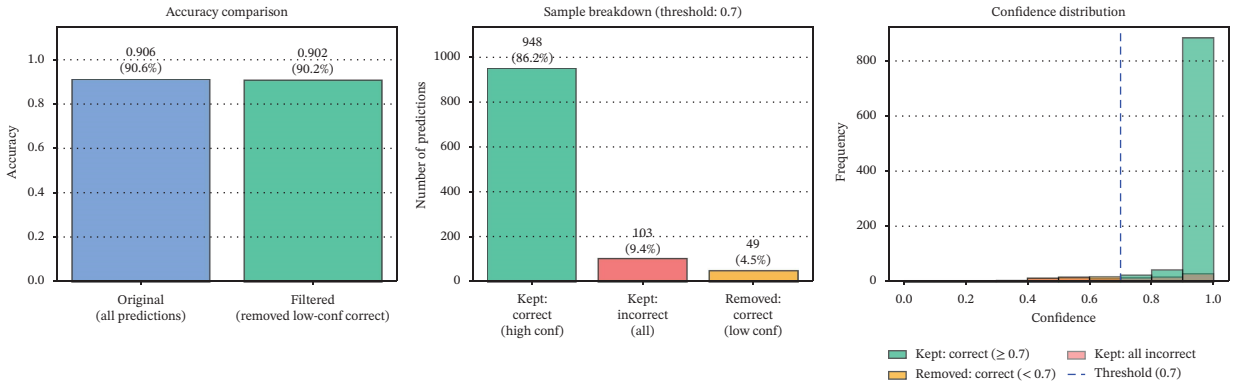

FIGURE 8 | Confidence threshold analysis for synthetic speech recognition at a 0.7 threshold, which includes the comparison of accuracy between all predictions and filtered predictions, sample breakdown by correctness and confidence level, and confidence distribution histogram.

speech AI voices to assess robustness against artificial speech patterns. The controlled testing approach eliminates variables such as accent variation and pronunciation inconsistencies, providing a precise appraisal of the model's core recognition capabilities under idealized acoustic conditions.

The synthetic speech evaluation demonstrated significantly improved performance, achieving an overall accuracy of 90.06% and an F1-score of 0.9071. This represents a substantial 8.24% point improvement over human speech recognition and even exceeds the theoretical offline performance of 89.58% as seen in Figure 5, validating that the embedded implementation maintains its recognition capabilities when presented with a high-quality audio input. As shown in Figure 6, the vast majority of vocabulary achieved recognition rates above 80%. Individual AI voices also show performance ranging from 62.73% to 100.00%, demonstrating high variation like human testing but with a better ceiling.

Analysis of misclassification patterns revealed significantly reduced systematic errors compared to human speech recognition. The confusion matrix in Figure 7 shows notably fewer systematic error patterns compared to human speech. The most common errors involved "go" misclassified as "down" (15 occurrences), "off" confused with "on" (9 occurrences), and "unknown" words being misclassified as "on" (14 occurrences), representing substantially fewer systematic errors than observed with human speakers.

To further evaluate system reliability with synthetic speech, we applied a confidence threshold analysis using a 0.7 threshold value, as illustrated in Figure 8. With TTS voices, applying this confidence threshold reduced accuracy from 90.06% to 90.02%, a decrease of only 0.04 percentage points. The analysis revealed that 948 predictions were both correct and had high confidence, while 49 predictions were correctly recognized, but with confidence below the threshold.

The results validate that the embedded KWS system performs nearly identical to the theoretical expectations under controlled conditions. The performance gap of 8.5% points between synthetic and human speech quantifies the impact of acoustic variability on real-world deployment, revealing the limitations of training on standardized datasets for diverse real-world applications. The findings demonstrate that while the system architecture is fundamentally sound, the primary deployment challenge lies in handling natural speech diversity, providing clear direction for future optimization through enhanced training data diversity and speaker adaptation techniques.

4 | Conclusion

The embedded implementation of CNN-based KWS on the ESP32-S3-EYE platform demonstrates the successful deployment of tiny machine learning models on a resource-constrained edge device. Model 7 [11], which was identified as the optimal candidate due to its superior balance of performance and resource constraints, achieves practical real-time performance with 0.68 MB memory and 89.58% accuracy under laboratory conditions. The proposed representation uses a 13-coefficient MFCC format at 8 kHz, which results in a smaller input tensor, a lower memory footprint of the model, and a reduced number of operations per inference when compared against higher-dimensional log Mel spectrograms.

Beyond model optimization, strategic system design proved equally critical for successful deployment. The dual-core architecture with FreeRTOS effectively distributes processing tasks, with dedicated cores for audio capture and neural network inference, eliminating bottlenecks that would compromise real-time performance on single-core platforms. The memory allocation strategy places critical timed structures in internal SRAM while utilizing external PSRAM for larger buffers such as the tensor arena, enabling efficient operation within the platform's memory hierarchy. These implementation strategies, combined with VAD-triggered processing to minimize unnecessary computation, provide a framework for deploying neural network inference on similar microcontroller platforms.

Real-world validation reveals significant performance challenges, with human speech recognition achieving 81.82% accuracy compared to 90.06% for synthetic speech evaluation. The performance gap quantifies the impact of pronunciation variability and accent diversity on embedded deployment. Systematic misclassifications between phonetically similar words, particularly "no" misclassified as "go" in 48 instances, highlight specific acoustic challenges faced by non-native speakers. The synthetic speech results confirm that the embedded system maintains near-theoretical performance when presented with consistent audio input. These findings suggest that practical KWS applications should select phonetically distinct keywords to maximize

recognition reliability. Furthermore, while fine-tuning with native speaker data showed no meaningful improvement in our previous work [11], incorporating accent-diverse training data may yield significant benefits for non-native speaker populations.

These findings demonstrate that CNN-based architectures provide an effective balance between recognition accuracy and resource efficiency for embedded KWS. A direct comparison with BC-ResNet, retrained under the same constrained MFCC format, supports this conclusion. Although BC-ResNet attains a smaller quantized footprint, the proposed model proved substantially more robust to quantization, retaining 89.58% accuracy against BC-ResNet's 85.35%, and was therefore retained for deployment. In addition, despite its smaller footprint, BC-ResNet required 2.5 times longer inference time due to operators lacking optimized embedded kernels, confirming that measured execution time and quantization robustness, rather than model size alone, are the decisive criteria for architecture selection. Transformer-based approaches [18], by contrast, remain infeasible on microcontroller devices.

Author Contributions

Patrik Medur: conceptualization (supporting), data curation (lead), formal analysis (lead), investigation (lead), methodology (equal), software (lead), validation (lead), visualization (lead), writing–original draft (lead), and writing–review and editing (supporting).

Mart Lubbers: conceptualization (supporting), methodology (equal), supervision (supporting), visualization (supporting), and writing–review and editing (supporting).

Goran Mauša: conceptualization (lead), formal analysis (supporting), funding acquisition (lead), methodology (equal), project administration (lead), resources (lead), supervision (lead), visualization (supporting), and writing–review and editing (lead).

Funding

This research was supported in part by ERASMUS + Teaching Artificial Intelligence (TAI), 2024-1-SI01-KA220-HED-000252673 funded by the European Union.

Disclosure

Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available in Kaggle at https://www.kaggle.com/datasets/. These data were derived from the following resources available in the public domain: Google's Speech Commands Dataset V0.02, https://www.kaggle.com/datasets/yashdogra/speech-commands.

References

1. C. R. Banbury, V. J. Reddi, M. Lam, et al., "Benchmarking Tinyml Systems: Challenges and Direction" (2020).

2. C. Cioflan, L. Cavigelli, M. Rusci, M. De Prado, and L. Benini, "On-Device Domain Learning for Keyword Spotting on Low-Power Extreme Edge Embedded Systems," in *IEEE* (2024), 6–10.

3. J. Arthi E and M. Jagadeeswari, "Control of Electrical Appliances Through Voice Commands," *IOSR Journal of Electrical and Electronics Engineering* 9, no. 1 (2014): 13–18, https://doi.org/10.9790/1676-09151318.

4. J. Chen, T. H. Teo, C. L. Kok, and Y. Y. Koh, "A Novel Single-Word Speech Recognition on Embedded Systems Using a Convolution Neuron Network With Improved out-of-Distribution Detection," *Electronics* 13, no. 3 (2024): 530, https://doi.org/10.3390/electronics13030530.

5. I. Froiz-Míguez, P. Fraga-Lamas, and T. M. Fernández-CaraméS, "Design, Implementation, and Practical Evaluation of a Voice Recognition Based Iot Home Automation System for Low-Resource Languages and Resource-Constrained Edge Iot Devices: A System for Galician and Mobile Opportunistic Scenarios," *IEEE Access* 11 (2023): 63623–63649, https://doi.org/10.1109/ACCESS.2023.3286391.

6. Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello Edge: Keyword Spotting on Microcontrollers" (2017).

7. P. M. Sørensen, B. Epp, and T. May, "A Depthwise Separable Convolutional Neural Network for Keyword Spotting on an Embedded System," *EURASIP Journal on Audio Speech and Music Processing* 2020, no. 1 (2020): 10, https://doi.org/10.1186/s13636-020-00176-2.

8. J. Amoh and K. M. Odame, "An Optimized Recurrent Unit for Ultra-Low-Power Keyword Spotting," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, no. 2 (2019): 1–17, https://doi.org/10.1145/3328907.

9. J. Bushur and C. Chen, "Neural Network Exploration for Keyword Spotting on Edge Devices," *Electronics* 12, no. 12 (2023): 2706.

10. B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted Residual Learning for Efficient Keyword Spotting" (2023).

11. P. Medur, M. Lubbers, and G. Mauša, "Optimizing Keyword Spotting Classifier Based on Tiny Machine Learning for Low-Power Embedded Devices," in *2025 MIPRO 48th ICT and Electronics Convention* (IEEE, 2025), 186–191.

12. V. Tiwari, "MFCC and Its Applications in Speaker Recognition," *International Journal on Emerging Technologies* 1, no. 1 (2010): 19–22.

13. B. Jacob, S. Kligys, B. Chen, et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2018), 2704–2713.

14. R. David, J. Duke, A. Jain, et al., "Tensorflow Lite Micro: Embedded Machine Learning for Tinyml Systems," *Proceedings of Machine Learning and Systems* 3 (2021): 800–811.

15. P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition" (2018).

16. J. Wang and S. Li, "Keyword Spotting System and Evaluation of Pruning and Quantization Methods on Low-Power Edge Microcontrollers" (2022).

17. K. Radzikowski, L. Wang, O. Yoshie, and R. Nowak, "Accent Modification for Speech Recognition of Non-Native Speakers Using Neural Style Transfer," *EURASIP Journal on Audio Speech and Music Processing* 2021, no. 1 (2021): 11, https://doi.org/10.1186/s13636-021-00199-3.

18. S. Sarkar, M. F. Babar, M. M. Hassan, M. Hasan, and S. K. Karmaker Santu, "Processing Natural Language on Embedded Devices: How Well Do Transformer Models Perform?" in *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering* (2024), 211–222.