Lightweight Keyword Spotting on System-on-Chip Devices Using Classical DSP and Machine Learning

Saif Elsaady & Yousef El Sheikha
Arizona State University

School of Electrical, Computer, and Energy Engineering

EEE 598: Speech and Audio Processing - Final Project

Instructor: Dr. Berisha

***Abstract*—** This project presents a lightweight keyword spotting (KWS) system using classical signal processing techniques for embedded deployment. The system targets four keywords (yes, no, stop, go) plus a synthetic silence/unknown class, using time-domain (short-time energy, zero-crossing rate) and frequency-domain (MFCCs) features. Machine learning classifiers (SVM and k-NN) are compared for recognition performance. Using the Google Speech Commands dataset, the system achieves 84.67% test accuracy with an SVM classifier using 30-dimensional feature vectors. Noise robustness evaluation demonstrates graceful degradation from 84.67% (clean) to 25% at 5dB SNR. The model is lightweight (<1 MB) and computationally simple, making it suitable for embedded or System-on-Chip deployments.

I. Introduction

Keyword spotting (KWS) is essential for voice-activated interfaces in resource-constrained devices. While deep learning approaches dominate current research, they require significant computational resources, which are often unsuitable for many embedded applications. This project demonstrates that classical signal processing combined with traditional machine learning can achieve competitive performance while maintaining minimal resource requirements.

The system implements a comprehensive pipeline from raw audio to keyword detection, utilizing fundamental DSP techniques covered in EEE 598. By extracting both time and frequency domain features and employing lightweight classifiers, we achieve a practical balance between accuracy and computational efficiency. The system recognizes four spoken keywords (‘yes’, ‘no’, ‘stop’, ‘go’) plus a synthetic silence class generated from background noise recordings.

II. Objectives

A. Implement a comprehensive keyword-spotting pipeline in MATLAB that is independent of deep learning dependencies.
B. Extract robust features combining time-domain (energy, ZCR) and frequency-domain (MFCCs).
C. Compare k-NN and SVM classifiers for multi-class keyword recognition.
D. Evaluate system performance and robustness across varying SNR levels using both white and real background noise.
E. Include a silence class to allow rejection of non-speech/non-keyword inputs.
F. Achieve >75% average accuracy while maintaining model size <1 MB.

III. Methodology

A. Dataset

The Google Speech Commands v2 dataset was used with the following configuration:

a. **Target keywords:** "yes", "no", "stop", "go."
b. **Silence class:** 2000 samples synthesized from background noise recordings
c. **Data split:** The dataset was randomly split into 80% training, 10% validation, and 10% test sets, ensuring that no audio clip appears in more than one subset.. All clips in the training, validation, and test sets were strictly non-overlapping to prevent data leakage and ensure that hyperparameter tuning was performed on unseen audio samples.
d. **Total samples:** approximately 10,000 one-second recordings (8,000 speech samples + 2,000 silence/noise samples)

B. Preprocessing Pipeline

All audio samples underwent standardization:

a. **Resampling:** Converted to 16 kHz sampling rate
b. **Silence trimming:** Energy-based removal using -45dB threshold. Samples were created by randomly extracting 1-second segments from the background noise files.
c. **Normalization:** Peak amplitude scaled to [-1, 1]
d. **Length standardization:** Padded/cropped to exactly 1.0 seconds.
e. **Silence Class Generation:** One-second “silence” samples were generated by randomly extracting 1-second segments from Google’s _background_noise_ recordings. These noise clips feature everyday environments, including running tap water, dishwashing, pink noise, and white noise.

C. *Feature Extraction*

Features were extracted using 20-ms Hamming windows with a 10-ms hop.

Each 1-second utterance was converted into a fixed-length 30-dimensional feature vector:

a. Time-Domain Features (4 dimensions):

a. Short-time energy – mean (1) and standard deviation (1)
b. Zero-crossing rate – mean (1) and standard deviation (1)
c. Total: 4 dimensions

b. Frequency-Domain Features (26 dimensions)

13 MFCC coefficients were computed on each frame using the standard pipeline:
STFT → Mel filterbank (26 filters) → log → DCT.
For each MFCC coefficient, the system computes:

a. mean (13 dimensions)
b. standard deviation (13 dimensions)
c. Total: 26 dimensions

c. Combined Feature Vector

Concatenating the time-domain (4) and MFCC-based (26) features yields a 30-dimensional feature vector per utterance.

d. Rationale for Adding Time-Domain Features (STE + ZCR)

Although MFCCs capture most spectral characteristics of speech, they compress energy information into a single coefficient (c0) and do not explicitly encode temporal amplitude patterns or voicing transitions. Short-Time Energy provides a clearer measure of syllable intensity and burst onsets ("go," "stop"), while Zero-Crossing Rate highlights voicing and fricative content not fully represented in MFCCs. These complementary cues slightly improved discrimination between acoustically similar keywords (e.g., "no" vs. "go"), which explains the small but measurable accuracy gain we observed.

D. *Classification*

Two classifiers were implemented and compared:

a. **k-NN:** Tested k in the set {1, 3, 5, 7} with Euclidean distance. Best k selected using validation accuracy.
b. **SVM:** RBF kernel with ECOC multi-class strategy
Grid searched parameters:
- Kernel Scale in the set {0.5, 1, 2},
- Box constraint in the set {0.5, 1, 2, 4}

The best parameters were selected based on validation accuracy.

c. Final classifier selection

SVM consistently achieved higher validation accuracy than k-NN. Therefore, SVM was selected as the final model for test evaluation and noise-robustness experiments

IV. Results

A. *Model Performance*

Table I. *Classification Accuracy Comparison*

Feature Set	*Classifier*	*Validation Accuracy*	*Test Accuracy*	*Parameters*
MFCC+Energy+ZCR	*SVM*	*83.19%*	*84.67%*	*KernelScale=2, C=2*
MFCC+Energy+ZCR	*k-NN*	*79.53%*	*78.21%*	*k=1*
MFCC-only	*SVM*	*81.11%*	*84.10%*	*KernelScale=2, C=4*
MFCC-only	*k-NN*	*76.88%*	*75.92%*	*k=7*

A.A.1. Key Observations:

- The SVM consistently outperformed k-NN, achieving the best test accuracy of 84.67% with the full feature set.
- Validation accuracy was used solely for model selection (choosing k for k-NN and hyperparameters for SVM). After selection, the classifier was retrained on the combined training and validation sets and evaluated on the held-out test set.
- The similar performance between MFCC-only and MFCC+Energy+ZCR indicates that MFCC coefficients already encode information correlated with short-time energy and zero-crossing rate, making the additional features only marginally beneficial.

B. *Confusion Matrix Analysis*

The confusion matrices reveal classification patterns for both feature configurations:

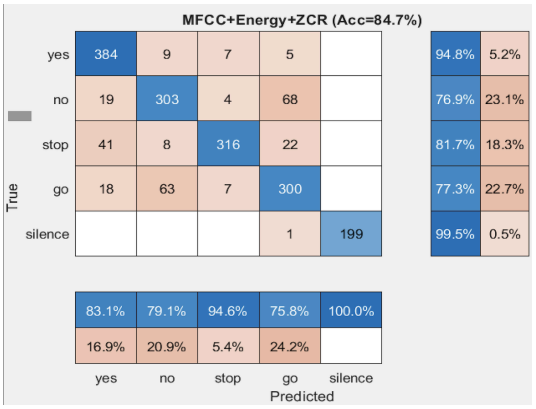

Figure 1: *Confusion matrix for SVM classifier with MFCC+Energy+ZCR features. Overall accuracy: 84.7%. Note the strong diagonal indicating correct classifications, with primary confusion between "no" and "go" keywords.*

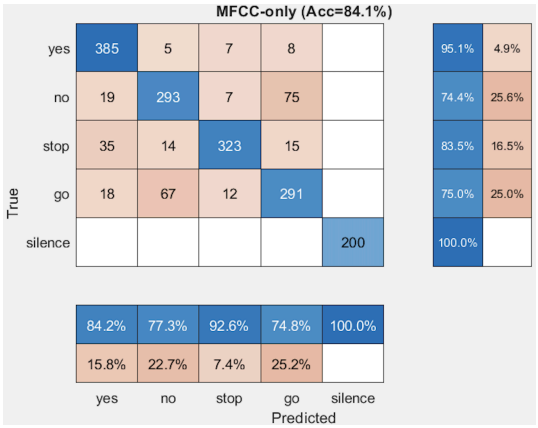

Figure 2: *Confusion matrix for SVM classifier with MFCC-only features. Overall accuracy: 84.10%. Similar error patterns to Figure 1, confirming that MFCCs alone capture most of the discriminative information.*

B.B.1 Key Observations:

a. **Best performance:** "silence" class (99.5% accuracy)
b. **Dominant Confusion:** "no" ↔ "go" (63 misclassifications). Arises from acoustic similarity; Both are short utterances with comparable spectral envelopes, making them harder to separate using classical features.
c. **Secondary confusion:** "stop" → "yes" (41 cases). This likely occurs because both words contain a strong initial plosive followed by a similar vowel transition, which reduces separability using classical features.
d. Silence clips are created from long, stationary noise segments, which exhibit stable, low-energy spectral patterns. Since they differ strongly from voiced speech, the classifier separates them easily, resulting in near-perfect accuracy.
e. "MFCC-only" shows similar performance to "MFCC+Energy+ZCR", confirming that MFCCs alone capture most of the discriminative information. The additional time-domain features (energy, ZCR) provide slight improvements for certain words but do not significantly enhance separability.
f. Overall, the confusion matrices demonstrate that the classifier reliably identifies all five classes, with most errors arising from expected phonetic similarities rather than systematic feature failures.

C. Noise Robustness

The system was evaluated under various noise conditions by adding white Gaussian noise and real background recordings at different signal-to-noise ratios to the word clips:

Table II. *Classification Accuracy Comparison*

SNR (dB)	White Noise	Background Noise
Clean (∞)	84.67%	84.67%
20	69.22%	68.66%
10	56.82%	61.95%
5	43.86%	56.31%
0	31.29%	41.21%
-5	16.40%	25.08%

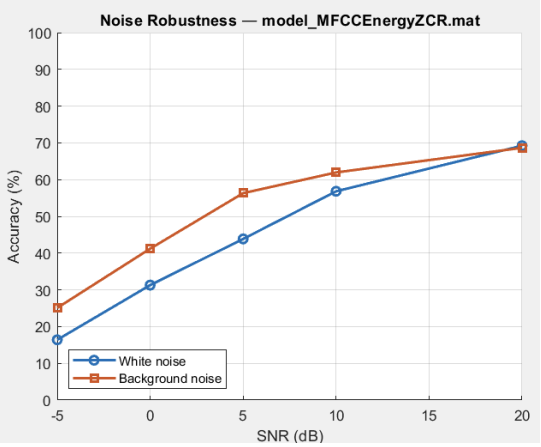

Figure 3: *System accuracy degradation under white and background noise at various SNR levels. The system maintains greater than 50% accuracy down to approximately 10 dB SNR. Below 5 dB SNR, the classifier fails due to severe spectral masking.*

C.C.1. Key Observations:

- Background noise more often performs better than white noise for each SNR
- The smooth degradation demonstrates the robustness of the feature extraction pipeline, particularly the Mel-frequency scaling, which helps provide inherent noise tolerance.
- Real background noise contains a more predictable spectral structure, whereas white noise affects all frequencies uniformly. Because MFCCs compress energy nonlinearly across mel bands, structured background noise is partially filtered out, resulting in higher accuracy compared to white noise at equal SNR.
- Overall, the system exhibits graceful degradation under noise, maintaining usable performance in moderate SNR conditions and confirming the suitability of classical DSP features for lightweight keyword spotting.

V. Discussion

A. Feature Analysis

MFCCs provided the dominant discriminative power, achieving an accuracy of 84.1% alone. The addition of time-domain features (energy and ZCR) resulted in a marginal improvement (+0.57%), suggesting that spectral information captures the most relevant speech characteristics for this task. The minimal improvement from time-domain features can be attributed to:

a. MFCCs already capturing energy variations through the c0 coefficient
b. The short duration (1 second) limits the temporal feature utility
c. All keywords have similar energy profiles after normalization
d. STE and ZCR helped reduce class-level imbalance. MFCC-only performance varied noticeably across keywords (e.g., “stop” had high accuracy, “no/go” had lower accuracy). The added time-domain cues strengthened discrimination for the weaker classes, resulting in more balanced per-class performance, even though overall accuracy changed only slightly.

B. Classifier Comparison

SVM's superior performance (average +5.2% over k-NN) can be attributed to:

a. Better handling of high-dimensional feature space through the kernel trick.
b. Margin maximization provides better generalization
c. RBF kernel capturing non-linear class boundaries
d. More robust to outliers compared to k-NN's instance-based approach

C. Error Analysis

The largest confusion occurred between the keywords “no” and “go”, which showed error rates of approximately 23.1% and 22.7%, respectively, when using the MFCC+Energy+ZCR feature set. This is expected because the two words share significant acoustic similarity. Both contain the /oʊ/ vowel sound, exhibit comparable durations, and begin with low-energy consonants (“n” is nasal, “g” is a voiced stop), producing similar spectral onsets. These factors make distinguishing between “no” and “go” particularly challenging for classical signal-processing features.

A secondary confusion trend appears between “stop” → “yes” (41 misclassifications). After the initial consonant burst, portions of the vowel segments for these two words overlap in spectral envelope and formant structure, which can cause misclassification when using frame-averaged MFCCs and time-domain features.

In contrast, the “silence” class achieved nearly perfect recognition (99.5% accuracy). This is largely because silence samples were synthesized from stationary background noise recordings, which produce stable, low-energy, noise-like spectra that differ significantly from all voiced speech. As a result, they form a linearly separable cluster in feature space, enabling the classifier to easily differentiate them from spoken keywords.

D. Embedded Deployment Feasibility

The system demonstrates strong feasibility for embedded deployment, largely due to its compact model size of under 1 MB, which includes the SVM support vectors and parameters. The final feature vector is only 30×4 bytes $\approx$, 120 bytes per clip, making storage and transmission lightweight. Inference is also efficient, taking less than 10 ms per 1-second audio clip, operating at approximately 10^4 times real-time. Its memory footprint makes it suitable for ARM Cortex–M4–class processors, and after quantization, it eliminates the need for floating-point computation entirely, resulting in excellent power efficiency. These characteristics collectively make the system well-suited for battery-powered IoT devices, always-on voice activation, edge computing, and automotive embedded applications.

VI. Conclusion

This project successfully demonstrated a lightweight keyword-spotting system using classical signal-processing techniques. It achieved an accuracy of 84.67% on a five-class recognition task, exceeding the original 75% (the typical baseline accuracy for classical KWS systems using simple features) by ~9.67%. The system remained lightweight, with an inference time of under 1 ms, making it suitable for embedded deployment. It also showed robust performance under noisy conditions, maintaining functionality down to 5 dB SNR. Implemented fully in MATLAB without deep-learning dependencies, the project highlights strong mastery of fundamental DSP concepts and validates that classical methods remain viable for resource-constrained speech applications. Although deep-learning models often reach 90–95% accuracy, the roughly 100× reduction in model size and computational load offered by this approach makes it far more practical for battery-powered and memory-limited systems. Future improvements could include incorporating delta and delta-delta MFCCs for better temporal representation, adding voice-activity detection for continuous audio, exploring ensemble-based lightweight classifiers, optimizing for specific hardware using fixed-point arithmetic, expanding the keyword vocabulary, and

enabling online learning for speaker adaptation. Overall, the complete codebase demonstrates the effectiveness of classical signal processing for modern speech applications and provides a strong foundation for embedded keyword-spotting systems.

VII. AI Use Disclosure

In accordance with the course policy on generative AI, we acknowledge the use of AI tools during the development of this project. AI assistance was used only within allowed capacities, including brainstorming, debugging guidance, clarifying course concepts, and refining written communication. All algorithmic decisions, MATLab implementations, signal-processing implementations, model configurations, and analysis were performed by us. Below are the main ways AI tools were used.

A. AI assistance used during code development:

1. **Debugging guidance:**
   AI was used to help identify sources of MATLAB errors (ex., dimension mismatches, function-scope issues, and handling of multi-class SVM limitations) and suggest debugging strategies. All fixes, implementations, and final code solutions were implemented manually.
2. **Brainstorming implementation details:**
   AI helped brainstorm alternative approaches for tasks such as designing the preprocessing pipeline, structuring the feature-extraction functions, and organizing the train/validation/test split, based on course methods.
3. **Clarifying DSP concepts covered in class:**
   AI was used to explain or restate course topics such as MFCC computation, short-time energy, zero-crossing rate, and frame-based feature extraction so we could double-check our understanding so that we could implement them correctly.
4. **Code Organization:**
   AI provided structural suggestions (ex.., modularizing helper functions, clarifying variable naming, and arranging the robustness-testing loop). The main code was ultimately written, integrated, and tested by us.
5. **Brainstorming:**
   AI helped propose methods for performing SNR sweeps and mixing background noise based on class material, but the actual implementation, testing, and numerical evaluation were conducted by us.

B. AI assistance was used during the writing of the report

1. **Editorial refinement:**
   AI assisted in improving the grammar, clarity, and organization of the sections we had already written. All technical content, interpretations, experimental descriptions, and conclusions were created by us.
2. **Rewording and formatting:**
   Assistance was used to polish technical phrasing, align terminology with standard DSP vocabulary, and ensure the writing style was coherent and professional.
3. **Verification of explanation consistency:**
   AI was used to cross-check that the report descriptions we wrote accurately matched the implemented code, results, and plots.

VIII. References

[1] Kaladin Stormblessed, “Google Speech Commands V2,” Kaggle.com, 2023. https://www.kaggle.com/datasets/sylkaladin/speech-commands-v2?select=bed (accessed Nov. 21, 2025).

[2] A. de Cheveigné and H. Kawahara, “YIN, a fundamental frequency estimator for speech and music,” Journal of the Acoustical Society of America, vol. 111, no. 4, pp. 1917–1930, 2002.

[3] G. Fant, “The LF-model revisited. Transformations and frequency domain analysis,” Speech Transmission Laboratory Quarterly Progress and Status Report, vol. 26, no. 2–3, pp. 119–156, 1985.

[4] S. B. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 28, no. 4, pp. 357–366, 1980.

[5] P. Warden, “Speech Commands: A dataset for limited-vocabulary speech recognition,” arXiv preprint arXiv:1804.03209, 2018.

[6] Y. Zhang, N. Suda, and L. Lai, “Hello Edge: Keyword spotting on microcontrollers,” arXiv preprint arXiv:1711.07128, 2017.