Speaker-Dependent Keyword Spotting with Tiny Data: Design, Training, and On-Device Evaluation

Mohamed Ahmed
Email: bazeet298@gmail.com

***Abstract*—We present a compact, speaker-dependent keyword spotting (KWS) pipeline that achieves strong performance with very limited data. The system uses Mel-frequency cepstral coefficients (MFCCs) computed from 1 s, 16 kHz audio and a small-footprint convolutional neural network trained with a balanced mixture of positives and broad non-speech negatives. We describe the data strategy, feature pipeline and windowing, model choices suitable for mobile/embedded devices, augmentation ablations, a benchmarking methodology, and practical personalization tactics. On a held-out test set at a fixed operating point (threshold 0.2), the model achieves 97.50% accuracy, 100.00% precision, 95.00% recall, and a 97.44% F1-score with a compact ∼1.2 MB footprint (313,857 parameters), demonstrating the viability of personalized KWS with tiny data.**

***Index Terms*—Keyword spotting, TinyML, MFCC, TensorFlow Lite, personalization, few-shot learning**

I. Introduction

Keyword spotting (KWS) enables hands-free interaction by detecting predefined words in continuous audio. Production-grade KWS typically relies on large datasets and heavily-optimized inference stacks. In contrast, we target *speaker-dependent* KWS in a *tiny-data* regime (tens of positive utterances), optimized for on-device use without a network connection. Our contribution is a complete, reproducible pipeline—from data collection to Android deployment—that remains accurate with minimal data and runs in real time on mobile hardware.

II. Related Work

Small-footprint KWS with convolutional models was popularized by [1]. Resource-constrained deployment on microcontrollers is surveyed in [2]. The Speech Commands dataset [3] standardizes evaluation for limited-vocabulary tasks. Our work differs by emphasizing speaker dependence under severe data scarcity and an end-to-end path to mobile inference.

III. Problem Framing

We formulate binary detection (keyword vs. non-keyword) under four constraints:

1) **Speaker-dependent:** tuned to a single user's voice.
2) **Tiny training data:** $\leq$30 positive 1 s utterances.
3) **On-device inference:** no cloud; sub-second latency.
4) **Small footprint:** model $\leq$1.5 MB.

The core challenge is maintaining high recall while keeping false accepts (FAs) low.

IV. Data Strategy & Negatives

Positives. We collect the user's keyword at 16 kHz mono in short sessions to capture natural variation (pace, emphasis, distance). **Negatives.** To curb FAs, we mix environmental noise and unrelated speech. Hard negatives (acoustically similar phrases) are valuable and can be mined from early false triggers. We maintain approximate class balance per mini-batch. Session-aware splits avoid temporal leakage.

V. Feature Pipeline & Windowing

Audio is standardized to 1 s at 16 kHz. We compute 40-D MFCCs using a 25 ms window and ≈23 ms hop, yielding a 44×40 (time×cepstra) tensor. For streaming, a 1 s window slides with a 250 ms hop, providing 4 predictions/s while keeping CPU usage modest. If normalization is used, mean/std are estimated only on the training split and reused at test and runtime.

VI. Model Choices for a Small Footprint

We employ a compact 2D CNN with ∼313,857 parameters (all trainable), followed by a sigmoid output for binary detection. Depthwise-separable alternatives (DS–CNN) can be swapped in to further reduce latency when needed. Table I summarizes size characteristics.

TABLE I
Model size summary

Total parameters	313,857
Trainable parameters	313,857
Non-trainable parameters	0
Approx. disk footprint	∼1.20 MB

VII. Training Setup & Logs

We train with cross-entropy and Adam, batch size 32, for 5 epochs, which sufficed for convergence. Representative training logs are in Table II.

VIII. Augmentation Ablations

To combat overfitting in the tiny-data regime, we use waveform-domain augmentation: time-stretch ($0.9\times$–$1.1\times$), pitch shift (±2 semitones), additive noise (5 dB–30 dB SNR), light EQ, and shallow reverberation. In ablations we compare: (A) none, (B) noise only, (C) time/pitch only, (D)

TABLE II
TRAINING/VALIDATION PROGRESSION (5 EPOCHS)

Epoch	Train Acc	Train Loss	Val Acc	Val Loss
1	0.7372	2.2917	0.9231	0.1457
2	0.9615	0.0785	0.9846	0.0467
3	0.9922	0.0346	0.9692	0.0924
4	0.9958	0.0187	0.9923	0.0201
5	1.0000	0.0038	0.9846	0.0194

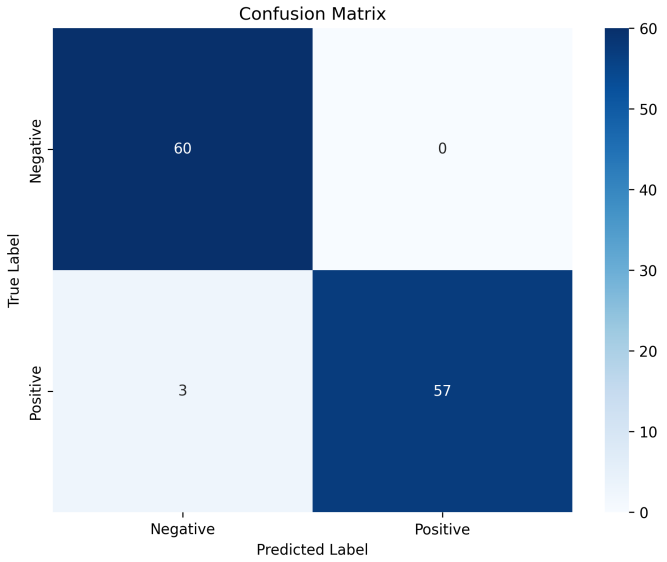

Fig. 1. Confusion matrix on the held-out test set at threshold 0.2 (TN=60, FP=0, FN=3, TP=57).

noise+time/pitch, (E) all transforms. (We omit numeric ablation results here to avoid overclaiming; the final model below includes moderate augmentation.)

IX. BENCHMARKING METHODOLOGY

A. Operating Point and Metrics

We fix a detection threshold on validation data and report standard metrics on a held-out test set. For streaming, metrics are computed per 1 s window.

B. Results

At threshold 0.2 on 120 test samples, the detector achieves:

- **Accuracy:** 97.50%
- **Precision:** 100.00%
- **Recall:** 95.00%
- **F1-score:** 97.44%

The confusion matrix is TN=60, FP=0, FN=3, TP=57 (Fig. 1). These results indicate a conservative operating point (zero false accepts in this test) with a small miss rate, appropriate for wake-word scenarios in quiet-to-moderate noise.

X. PERSONALIZATION & GENERALIZATION

Threshold calibration per user/environment reduces FAs without retraining. **Hard-negative mining** from real usage improves the next training round by adding look-alike acoustics the model confuses with the keyword. For broader generalization, a shared backbone trained on multi-speaker data with a tiny user-specific head fine-tuned on-device is a practical path.

TABLE III
AUTOMATIC ACCURACY TEST (THRESHOLD 0.2, N=120)

Metric	Value
Accuracy	97.50%
Precision	100.00%
Recall	95.00%
F1-score	97.44%

XI. ANDROID DEPLOYMENT

We provide a Flutter app that captures 16 kHz audio, computes 44×40 MFCCs in Dart, runs a TensorFlow Lite model, and triggers when the sigmoid score exceeds the chosen threshold. A UI slider lets the user tune the operating point quickly across environments.

XII. LIMITATIONS AND ETHICS

Speaker-dependent systems can degrade with illness or accent drift; lightweight on-device adaptation is a priority for future work. Always obtain consent in shared spaces and avoid recording or uploading raw audio; this work is designed for on-device operation.

XIII. CONCLUSION

With careful negatives, MFCC features, compact CNNs, and realistic augmentation, personalized KWS is feasible with tiny data. The resulting model exports cleanly to TFLite and runs in real time on Android. Future work includes quantization-aware training for MCUs, multi-keyword extension, and on-device continual learning.

REFERENCES

[1] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. Interspeech*, 2015.

[2] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello Edge: Keyword spotting on microcontrollers," *arXiv:1711.07128*, 2018.

[3] P. Warden, "Speech Commands: A dataset for limited-vocabulary speech recognition," *arXiv:1804.03209*, 2018.