Complex & Intelligent Systems (2026) 12:267
https://doi.org/10.1007/s40747-026-02415-z

ORIGINAL ARTICLE

A lightweight hybrid 2D–1D depthwise-separable CNN with frequency-only pooling for efficient small-footprint keyword spotting

Jaewon Lee[1] · Sangbeom Lee[1] · Bumhwi Kim[1] · Doosik Kim[1] · Kwang-yong Kim[1]

Received: 22 January 2026 / Accepted: 9 July 2026
© The Author(s) 2026

Abstract

Small-footprint keyword spotting (KWS) requires an effective balance between local time–frequency modeling and efficient temporal processing under tight deployment constraints. Rather than introducing a new primitive module, this work investigates a lightweight hybrid design that combines a shallow 2D time–frequency front-end with frequency-only pooling, a 1D depthwise-separable temporal backbone, and statistics pooling for segment-level aggregation. The design is motivated by a compute-conscious allocation of modeling roles: early local spectro-temporal feature extraction is handled by a shallow 2D block, while low-cost temporal context modeling is performed by 1D depthwise-separable convolutions. Using log-Mel spectrograms as input, the proposed model preserves temporal resolution through frequency-only pooling and summarizes frame-level representations with statistics pooling. On the Google Speech Commands v2 12-class task, the model achieves 97.5% top-1 accuracy with 54 K parameters and 3.8 M MACs. To characterize this operating point more rigorously, we report ablation and sensitivity analyses, reproduced baseline comparisons under a shared preprocessing and training protocol, on-device inference latency on Raspberry Pi 5 and LattePanda V1, and controlled white-noise evaluation across multiple SNR levels. The results show that the proposed configuration provides a competitive low-MAC operating point within the tested GSC v2 setting. In particular, the combination of a shallow 2D front-end, frequency-only pooling, and statistics pooling yields modest but consistent gains under repeated runs, while the lower computational cost also translates into favorable CPU inference latency. Under the tested white-noise protocol, the proposed model additionally shows smaller relative degradation than the reproduced baselines. These findings suggest that the studied hybrid design is a practical option for efficient on-device KWS when computational cost is a primary constraint.

Keywords Lightweight convolutional neural networks · Hybrid 2D-1D architecture · On-device · Edge intelligence · Keyword spotting

Introduction

Keyword spotting (KWS) aims to detect a predefined keyword or a small set of command words in an audio stream. It is widely used in wake-word detection and low-latency voice interfaces on smart speakers, mobile devices, wearables, and other embedded platforms [1–4]. In these settings, a practical KWS model must operate with limited memory, low computational budget, and short response time, while still maintaining reliable recognition accuracy.

Traditional KWS systems were commonly based on hidden Markov models (HMMs), dynamic time warping (DTW), and related statistical pipelines [5, 6]. Although such methods can operate under constrained resources, they typically rely on hand-crafted features, explicit decoding procedures, and threshold tuning, and they become increasingly cumbersome as the vocabulary or deployment conditions change. Deep learning substantially improved KWS performance by learning discriminative representations directly from speech features [7, 8]. Recent KWS architectures now

✉ Jaewon Lee
jwl0127@etri.re.kr

[1] Present Address: Medical IT Convergence Research Section, Daegu-Gyeongbuk Research Division, Electronics and Telecommunications Research Institute (ETRI), 1 Techno Sunhwan-Ro 10-gil, Yuga-eup, Dalseong-gun, Daegu 42994, Republic of Korea

Published online: 25 July 2026

span convolutional, residual, temporal-convolutional, multilayer perceptron, and transformer-based designs [9–14], but the accuracy-efficiency trade-off remains a central challenge for small-footprint deployment [15, 16].

A common design choice in KWS is whether to emphasize 2D time–frequency processing or 1D temporal modeling. While 2D convolutions are well suited to capturing local spectro-temporal patterns from time–frequency inputs, 1D temporal convolutions can model longer-range context more efficiently after feature extraction. Hybrid 2D/1D designs themselves are not new, and the building blocks used in this study are all established in the literature [17]. Therefore, this work investigates whether a specific compute-conscious allocation of these components forms a competitive operating point when MAC budget is the primary constraint. More specifically, the studied configuration uses only a shallow 2D front-end for early local time–frequency feature extraction, applies frequency-only pooling to reduce spectral redundancy while preserving temporal resolution, and then relies on a 1D depthwise-separable backbone for efficient temporal modeling. Under strict compute constraints, such allocation of modeling roles can matter as much as the selection of the backbone itself.

Segment-level aggregation is another important yet relatively under-discussed choice in small-footprint KWS. Statistics pooling, originally popularized in speaker embedding systems such as x-vectors, summarizes a frame-level sequence with its mean and standard deviation [18–21]. This yields a fixed-dimensional embedding that captures not only average activation patterns but also feature variability across time. While statistics pooling has been studied extensively in speaker-related tasks, its role within lightweight hybrid KWS architectures has been less systematically examined under a unified training and evaluation pipeline [22].

In this paper, we study a lightweight hybrid KWS design built from a shallow 2D time–frequency front-end, frequency-only pooling, a 1D depthwise-separable temporal backbone, and statistics pooling. The goal is not to introduce new basic operators, but to understand whether this particular combination constitutes a competitive operating point when computational cost is a primary constraint. Our evaluation focuses on the Google Speech Commands v2 12-class setting, where we complement the main clean-condition results with controlled ablation, sensitivity analysis, reproduced same-protocol baselines, device-level latency measurements, and additional noisy-condition evaluation under varying SNR levels.

The main contributions of this paper are summarized as follows:

- We examine a lightweight hybrid KWS design that combines a shallow 2D time–frequency front-end with frequency-only pooling, a 1D depthwise-separable temporal backbone, and statistics pooling for segment-level embedding.
- We show, through controlled ablation and sensitivity analyses, that frequency-only pooling and statistics pooling provide a favorable accuracy-compute trade-off within this hybrid family under a small-footprint design environment.
- We further characterize this operating point using repeated-run evaluation, same-protocol reproduced baselines, on-device latency measurements, and controlled white-noise testing under a shared experimental setup.

The remainder of this paper is organized as follows. Section "Related works" reviews related work on conventional and neural KWS, lightweight architectures, and segment-level aggregation. Section "Methods" describes the proposed architecture and implementation details. Section "Results and discussion" presents the experimental results and discussion. Section "Conclusion" concludes the paper and outlines future directions.

Related works

Conventional keyword spotting

Early keyword spotting systems were largely based on statistical sequence models and template-matching techniques. HMM-based systems typically modeled the target keyword together with filler or background speech models, and then used decoding scores and thresholds to make keyword decisions [5, 23]. DTW-based and segmental matching approaches were also studied for spoken term detection and unsupervised keyword discovery [6]. These methods can be effective under limited computational resources, but their performance is often sensitive to feature design, decoding configuration, and threshold selection. In addition, extending the vocabulary or adapting the system to new operating conditions generally requires nontrivial redesign.

Deep learning-based keyword spotting

Deep learning shifted KWS from manually designed pipelines toward direct discriminative modeling. Early neural KWS systems used deep neural networks to estimate keyword posterior probabilities from acoustic features [7]. Convolutional models operating on MFCC or log-Mel inputs then became a standard approach because they provided strong recognition accuracy with relatively compact architectures [8]. Subsequent work explored deeper residual models, temporal convolution, and attention-based sequence modeling [9, 11–13]. More recent studies and surveys have

broadened the design space further to include transformer-based and state-space-inspired architectures [9, 22, 24]. These developments substantially improved KWS performance, but they also made the trade-off between accuracy, parameter count, MACs, and deployment cost more central—especially for on-device scenarios.

Small-footprint KWS

A major branch of KWS research focuses on small-footprint models that can run efficiently on embedded hardware. Depthwise-separable CNN (DS-CNN) architectures showed that depthwise-separable convolutions can significantly reduce computation while retaining competitive accuracy [10, 25]. Lightweight residual architectures further improved this trade-off by combining residual learning with compact convolutional blocks [11, 26]. Temporal convolutional models such as TC-ResNet emphasized efficient 1D temporal modeling and low inference cost on mobile devices [12]. More recently, MatchboxNet [27] and BC-ResNet [28] demonstrated strong lightweight performance using time-channel separable convolutions and broadcasted residual learning, respectively.

A useful observation across these models is that lightweight KWS architectures often emphasize one of two representational biases. Some models rely more strongly on 2D time–frequency processing to capture local spectro-temporal structure, whereas others favor efficient 1D temporal modeling after feature extraction. Both directions are valid, but they lead to different compute allocations and efficiency characteristics. This makes it meaningful to study how a limited amount of 2D processing can be combined with a lightweight 1D temporal backbone under a fixed resource budget.

Segment-level aggregation and statistics pooling

Segment-level aggregation has been extensively studied in speaker recognition and related sequence classification tasks. In the x-vector framework, frame-level representations are aggregated using statistics pooling, which computes the mean and standard deviation across time to produce an utterance-level embedding [18]. Later work extended this idea through attentive statistics pooling and related variants that more flexibly weight frame-level information [19–21]. The key advantage of statistics pooling is that it produces a fixed-dimensional segment representation while also preserving information about temporal variability.

In KWS, frame-level aggregation is often simplified to global average pooling or related reductions. Although such choices are computationally attractive, they may discard useful variability information. For lightweight KWS in particular, the effect of using statistics pooling instead of simpler aggregation methods has not been examined as systematically under a unified training and evaluation pipeline. This is especially relevant when the front-end and temporal backbone are both constrained by a low-MAC design target.

Positioning of the present study

The present study is positioned as an empirical design investigation rather than a proposal of a new basic modeling primitive. Hybrid 2D/1D processing, depthwise-separable convolution, and statistics pooling are all established ideas in the literature. Our contribution lies in examining a specific low-cost combination of these components under a unified experimental setting.

More specifically, the design studied here differs from many prior small-footprint KWS models in three respects. First, 2D processing is intentionally restricted to a shallow front-end rather than being used throughout the network, so that early local spectro-temporal modeling can be retained without incurring the cost of a deeper 2D stack. Second, the front-end uses frequency-only pooling, which reduces spectral redundancy while preserving temporal resolution for the subsequent temporal backbone. Third, the model uses statistics pooling as the segment-level aggregation mechanism and evaluates its effect systematically against simpler alternatives under the same training pipeline.

Accordingly, we do not present the proposed model as a uniformly superior architecture across all KWS settings or efficiency axes. Instead, we study whether this particular configuration forms a competitive low-MAC operating point under the tested GSC v2 12-class setting, and we support that analysis through repeated-run evaluation, sensitivity analysis, same-protocol reproduced baselines, on-device latency measurements, and controlled noisy-condition testing.

Methods

This section describes the input preprocessing and the lightweight hybrid architecture studied in this work. The model combines a shallow 2D time–frequency front-end with frequency-only pooling, a 1D depthwise-separable temporal backbone for efficient sequence modeling, statistics pooling for segment-level aggregation, and a compact classifier.

Input preprocessing

Each input waveform sampled at 16 kHz is converted into a log-Mel spectrogram $\boldsymbol{S} \in \mathbb{R}^{T \times M}$. We compute the short-time Fourier transform using a Hann window with a 20 ms frame length and a 10 ms frame shift. Mel filters are applied to the magnitude spectrum from 60 to 7600 Hz with $\boldsymbol{M} =$ 40 bins, followed by a logarithmic transformation. Finally,

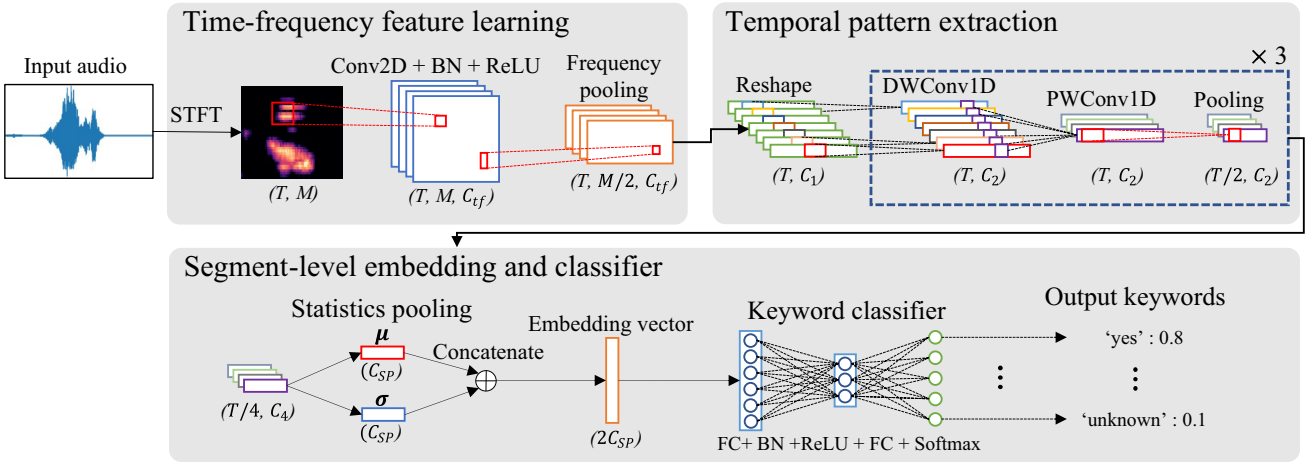

Fig. 1 Overall architecture of the studied hybrid design. The pipeline combines a shallow 2D time–frequency front-end with frequency-only pooling, a 1D depthwise separable temporal backbone, and statistics pooling for embedding. *FC* fully connected, *BN* batch normalization, *Conv2D* 2D convolution, *DWConv1D* 1D depthwise convolution, *PWConv1D* 1D pointwise convolution)

to normalize the input spectrogram, we apply z-score normalization by computing the mean and variance of $\boldsymbol{S}$. In our experiments, each input segment has a log-Mel spectrogram of shape 98 × 40.

Architecture overview

Given a log-Mel spectrogram $\boldsymbol{S}$, the model first uses a shallow 2D convolutional front-end to extract local time–frequency patterns. Then, frequency-only pooling is applied to reduce the frequency resolution while preserving the frame-level temporal resolution. Next, the resulting feature map is reshaped into a frame-level sequence and processed by a stack of 1D depthwise-separable convolution blocks to model temporal patterns along the time axis. Finally, statistics pooling aggregates the frame-level features into a fixed-dimensional segment-level embedding. This embedding vector is then fed to a compact classifier that predicts the target keyword. Figure 1 illustrates the overall pipeline.

Shallow 2D front-end and frequency-only pooling

The input spectrogram $\mathbf{S} \in R^{T \times M}$ is first processed by a shallow 2D convolution block to capture local patterns in the time–frequency domain. Specifically, a 2D convolution with a 3 × 3 kernel is applied, followed by batch normalization and ReLU activation.

We then apply frequency-only pooling instead of standard 2D pooling. In short-command KWS, pooling along the time axis may remove temporally localized phonetic cues before the temporal backbone processes them. By contrast, pooling only along the frequency axis reduces spectral redundancy while preserving frame-level temporal resolution. This design is particularly compatible with the subsequent 1D temporal backbone, whose role is to model temporal dependencies after local spectro-temporal features have been extracted. In the present configuration, average pooling with a size of 1 × 2 is used, and the number of front-end output channels is set to 16 based on the sensitivity analysis in Sect. "Ablation study and sensitivity analysis", which showed that increasing the width beyond 16 produced only marginal accuracy gains relative to the additional computation. In our implementation, we set the number of front-end channels to C_{tf}=16. After this stage, the feature map has the shape of $T \times M_p \times C_{tf}$, where M_p is the dimension after frequency pooling (20 in our case).

Temporal modeling with 1D depthwise-separable convolution blocks

After the 2D front-end, the feature map is reshaped into a frame-level feature sequence $\boldsymbol{X} \in \mathbb{R}^{T \times C_1}$, where $C_1 = M_p \times C_{tf}$. The temporal network is implemented as a stack of 1D depthwise separable convolution blocks. Each block consists of a 1D depthwise convolution (DWConv1D) for temporal filtering across channels, followed by a 1D pointwise convolution (PWConv1D) for mixing across channels. This is then followed by BN and ReLU. Compared to a standard CNN, DS convolution decomposes temporal feature learning and channel mixing, thereby reducing the number of parameters and multiply-accumulate operations (MACs). To avoid a loss of temporal resolution, 1D average pooling is applied only in the first two blocks. Specifically, Block 1 and 2 use DWConv1D–PWConv1D–BN–ReLU–AvgPooling1D, whereas the remaining block uses

Table 1 Architecture of the proposed hybrid design: shallow 2D time–frequency front-end with frequency-only pooling, 1D depthwise-separable temporal backbone, and statistics pooling

Stage	Layer	Kernel/STRIDE	Output shape
2D front-end with frequency pooling	Conv2D + BN + ReLU + AvgPooling2D	Conv: 3 × 3, Pool: 1 × 2	(98, 20, 16)
Reshape for DS-Conv	Reshape	–	(98, 320)
1D DS-Conv block 1	DWConv1D + PWConv1D + BN + ReLU + AvgPooling1D	DW: 3, PW: 1, Pool: 2	(49, 64)
1D DS-Conv block 2	DWConv1D + PWConv1D + BN + ReLU + AvgPooling1D	DW: 3, PW: 1, Pool: 2	(24, 96)
1D DS-Conv block 3	DWConv1D + PWConv1D + BN + ReLU	DW: 3, PW: 1	(24, 96)
Statistics pooling	StatsPooling + Concatenate ([$\boldsymbol{\mu}$, $\boldsymbol{\sigma}$])	–	(192)
Classifier	FC + BN + ReLU + Dropout + FC + Softmax	–	(12)

FC fully connected, *BN* batch normalization

DWConv1D–PWConv1D–BN–ReLU without pooling. The detailed layer and the output shapes of the stages are summarized in Table 1.

Segment-level aggregation via statistics pooling

Let $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{T/4}] \in \mathbb{R}^{T/4 \times C_4}$, denote the frame-level output sequence from the final DS block. We use statistics pooling to produce a segment-level embedding with fixed-dimension by computing the mean and standard deviation across the time axis. This layer calculates the mean and standard deviation of the channels across the time axis, summarizing the variable-length sequence into an embedding vector. Specifically, the mean vector $\boldsymbol{\mu}$ and standard deviation vector $\boldsymbol{\sigma}$ are computed as expressed in Eqs. (1) and (2).

$$\boldsymbol{\mu} = \frac{1}{T/4} \sum_{t=1}^{T/4} \boldsymbol{h}_t, \tag{1}$$

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{T/4} \sum_{t=1}^{T/4} (\boldsymbol{h}_t - \boldsymbol{\mu})^2}. \tag{2}$$

These two statistical vectors are concatenated to form the embedding vector $\boldsymbol{z} = [\boldsymbol{\mu}; \boldsymbol{\sigma}] \in \mathbb{R}^{2 \bullet C_{SP}}$. In our setting, $C_{SP} = 96$, hence $\boldsymbol{z} \in \mathbb{R}^{192}$. This aggregation introduces no additional learnable parameters and provides a fixed-dimensional representation for classification.

Keyword classifier

The embedding vector is fed into a compact classifier consisting of a fully connected layer, batch normalization, ReLU, dropout, and a final fully connected layer followed by softmax. The dropout probability is set to 0.2, as the sensitivity analysis in Sect. "Ablation study and sensitivity analysis" showed that this value provided the most favorable balance among the tested settings, whereas both smaller and larger values led to lower average accuracy.

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log(p_{\boldsymbol{y}_i}) \tag{3}$$

where N is the mini-batch size, and $p_{\boldsymbol{y}_i}$ is the predicted probability for the ground-truth label $\boldsymbol{y}_i$ of the i-th sample. Sect. "Results and discussion" provides optimization and other training details.

Results and discussion

Datasets and experimental setup

We evaluate the studied hybrid design using the Google Speech Commands v2 (GSC v2) dataset [29], which contains 105,829 audio 1-s clips sampled at 16 kHz. We follow the widely used 12-class setting [13, 30]. The target classes include 10 keywords ('yes', 'down', 'no', 'stop', 'go', 'on', 'left', 'right', 'off', 'up'), 'silence' (background noise) and 'unknown' (all remaining words). Following previous studies implementation, we generate 'silence' samples by randomly sampling and clipping in 'background noise' file as instructed by Google [29]. 'unknown' samples are all the remaining samples from other 25 keywords.

We apply random sampling to the 'silence' and 'unknown' samples, averaging the number of samples from the other keywords, as is common in related works [27, 28, 30], to rebalance the dataset. We use the 'validation list' and testing lists' provided in the dataset [29], yielding an approximately 80/10/10 split for training, validation, and test. The same preprocessing, training, and evaluation protocol was applied

Table 2 Ablation study of the hybrid design on the GSC v2 12-class test set

Ablation factor	Setting	#Param (K)	MACs (M)	Accuracy (%)
Front-end block	No-Conv	68	3.9	97.0 ± 0.05
	1D-Conv	62	3.4	96.7 ± 0.27
	2D-Conv (proposed)	54	3.8	97.5 ± 0.07
TF pooling stride (t, f)	(2, 1)	76	3.6	96.9 ± 0.09
	(2, 2)	54	2.2	96.8 ± 0.31
	(1, 2) (proposed)	54	3.8	97.5 ± 0.07
Segment embedding	GMP	48	3.8	97.1 ± 0.08
	GAP	48	3.8	96.7 ± 0.19
	ASP	66	4.1	97.2 ± 0.15
	SP (proposed)	54	3.8	97.5 ± 0.07

Accuracy is reported as mean ± standard deviation over 5 random seeds

to the reproduced baselines reported in Sect. "Comparison with reproduced baselines and prior work".

We train models using Adam with an initial learning rate of 0.001, a batch size of 64, and 50 epochs. We reduce the learning rate by a factor of 0.1 when the validation accuracy does not improve for 10 consecutive epochs. Top-1 accuracy on the test set is reported for all experiments. All models are implemented in TensorFlow 2.10 and Keras 2.10, and are trained on a single NVIDIA RTX 3070 GPU.

To characterize computational cost, we report multiply-accumulate (MAC) for a single forward pass with an input size of 98 × 40. MACs were counted for the convolutional and fully connected layers under the same input configuration across models. In addition, to complement the complexity comparison with a practical deployment-oriented metric, we measured on-device inference latency on two CPU-based edge platforms: Raspberry Pi 5 and LattePanda V1. Latency was measured using TensorFlow Lite inference with batch size 1.

Ablation study and sensitivity analysis

The ablation study was conducted to identify which components of the hybrid design contribute most to the observed accuracy-efficiency trade-off. Three design factors were examined while keeping the remaining components fixed: the front-end block, the pooling stride in the 2D front-end, and the segment-level embedding method. The results are summarized in Table 2.

First, we compared three front-end variants: No-Conv, 1D-Conv, and the proposed shallow 2D front-end. As shown in Table 2, the 2D front-end achieved the highest accuracy at 97.5 ± 0.07%, compared with 97.0 ± 0.05% for No-Conv and 96.7 ± 0.27% for 1D-Conv. Although the absolute gain is modest, it was consistent across five random seeds, suggesting that even a single lightweight 2D front-end block provides useful local spectro-temporal feature extraction before temporal modeling.

Next, we examined the pooling stride in the 2D front-end. Among the tested settings, frequency-only pooling with stride (1, 2) achieved the best result. By contrast, temporal pooling reduced MACs but also reduced accuracy, indicating that preserving frame-level temporal resolution is beneficial in this hybrid architecture. This trend is consistent with the role separation adopted in the present design: the 2D front-end extracts local time–frequency patterns, whereas temporal dependency modeling is deferred to the 1D backbone.

We also compared several segment-level aggregation strategies, including global max pooling (GMP), global average pooling (GAP), attentive statistics pooling (ASP), and statistics pooling (SP). SP achieved the highest accuracy while maintaining the same MACs as the proposed configuration and substantially lower cost than ASP. Relative to GAP and GMP, SP can encode not only average activation patterns but also frame-level variability through the standard deviation term. In the tested setting, this additional variability information appears to provide a small but consistent benefit for keyword classification. After two temporal pooling stages, the sequence length is reduced from 98 to about 24 frames; within this range, statistics pooling remained stable across five random seeds and consistently outperformed GAP and GMP in our experiments.

Overall, the ablation results support the proposed configuration as a well-balanced design choice within the tested search space, rather than as evidence that any single component is universally superior across all KWS settings.

To further examine the stability of this design, we additionally performed a sensitivity analysis on several key hyperparameters, including dropout rate, front-end channel width, and the depths of the front-end and 1D temporal blocks. The results are summarized in Table 3. The sensitivity analysis in Table 3 further clarifies why the proposed

Table 3 Sensitivity analysis of key hyperparameters

Key hyperparameter	Setting	#Param (K)	MACs (M)	Accuracy (%)
Dropout (%)	0.1	54	3.8	96.2 ± 0.51
	0.2	54	3.8	**97.5 ± 0.07**
	0.3	54	3.8	96.9 ± 0.27
Front-end channel width	8	42	2.1	96.9 ± 0.09
	16	54	3.8	**97.5 ± 0.07**
	32	77	8.2	97.4 ± 0.06
Depth of the front-end blocks	1	54	3.8	**97.5 ± 0.07**
	2	56	13.1	97.3 ± 0.16
	3	58	22.3	97.5 ± 0.12
Depth of the 1D blocks	3	54	3.8	**97.5 ± 0.07**
	5	89	4.6	97.2 ± 0.26
	7	143	5.8	96.9 ± 0.31

Accuracy is reported as mean ± standard deviation over 5 random seeds

configuration was selected. For the dropout rate, 0.2 produced the highest mean accuracy, whereas both 0.1 and 0.3 resulted in lower performance, suggesting that moderate regularization is preferable in this setting. For the front-end channel width, increasing the width from 8 to 16 improved accuracy, but a further increase to 32 produced only a marginal change while more than doubling the MACs. This indicates diminishing returns beyond the proposed width of 16.

We also examined the depth of the front-end and the 1D temporal stack. Increasing the front-end depth beyond a single block did not yield a meaningful gain despite a substantial increase in computation, suggesting that a shallow front-end is sufficient for early local feature extraction in the present hybrid design. Likewise, increasing the number of 1D blocks from 3 to 7 increased complexity without improving accuracy, and deeper temporal stacks slightly degraded performance in our experiments. Taken together, these results suggest that the proposed setting is not only competitive in clean-condition accuracy, but also close to an efficient local optimum within the tested low-MAC design range.

Comparison with reproduced baselines and prior work

We compared the proposed model with representative prior KWS architectures in terms of accuracy, parameter count, MACs, and on-device inference latency. Because model performance in KWS is known to depend on the preprocessing, augmentation, optimization, and evaluation protocol, we additionally reproduced several lightweight baselines under the same pipeline used for the proposed model. We selected MatchboxNet and BC-ResNet because they are widely referenced lightweight KWS architectures that span different capacity-compute trade-offs. For BC-ResNet, three width settings ($\tau = 1$, 3, and 6) were considered to cover a broader range of operating points.

Table 4 reported the reproduced results obtained under the shared protocol of this study. These reproduced results are intended as controlled same-protocol reference points rather than as replacements for the architecture-specific results originally reported in prior work. Some baseline architectures may require architecture-specific tuning to recover their originally reported peak performance, so the reproduced numbers should be interpreted primarily as under-one-pipeline comparisons.

Under this shared protocol, the proposed model achieved 97.5 ± 0.07% accuracy with 54 K parameters and 3.8 M MACs, providing the most favorable overall accuracy-compute-latency balance among the reproduced models. In particular, it outperformed the reproduced MatchboxNet-3 × 2 × 64 while requiring substantially fewer MACs, and it also achieved higher accuracy than the reproduced BC-ResNet-3 at a much lower computational cost. Although the reproduced BC-ResNet-6 approached the proposed model in accuracy, it did so with considerably larger parameter count, MACs, and latency. Importantly, these findings should not be interpreted as claiming uniform superiority over all prior architectures or all reported operating points in the literature. Rather, they indicate that within the shared experimental pipeline used in this study, the proposed hybrid design forms a particularly competitive operating point when computational cost and practical CPU inference latency are treated as primary constraints. This interpretation is also consistent with the literature-reported comparison in Table 5, where some larger or more compute-intensive models achieve higher peak accuracy but at substantially higher cost.

The latency measurements provide an additional deployment-oriented perspective beyond parameter count

Table 4 Comparison of the proposed model and reproduced baselines under the same protocol (accuracy reported as mean ± standard deviation over five random seeds)

Model	#Param (K)	MACs (M)	Accuracy (%)	Latency (ms, RP5)	Latency (ms, LP1)
MatchboxNet-3 × 2 × 64 [27]	101	9.7	96.5 ± 0.31	0.89 ± 0.01	10.19 ± 0.20
BC-ResNet-1 [28]	12	3.6	90.6 ± 0.67	0.82 ± 0.01	6.31 ± 0.50
BC-ResNet-3 [28]	61	16.9	96.3 ± 0.26	2.78 ± 0.02	22.80 ± 1.74
BC-ResNet-6 [28]	202	54.8	97.0 ± 0.57	6.19 ± 0.02	63.03 ± 2.46
Proposed model	54	3.8	97.5 ± 0.07	0.41 ± 0.01	4.26 ± 0.19

RP5 Raspberry Pi 5, *LP1* LattePanda V1

Table 5 Literature-reported comparison with representative KWS models on the GSC v2 12-class task, including recent compact and higher-capacity architectures

Model	#Param (K)	MACs (M)	Accuracy (%)
Attention RNN [31]	202	22.3	96.9
TC-ResNet14-1.5 [12]	305	6.7	97.4
MHAtt-RNN [32]	743	22.7	98.0
TENet6-narrow [30]	17	0.6	96.0
TENet12 [30]	100	2.9	96.6
MatchboxNet-3 × 2 × 64 [27]	93	11.3	97.2
KWT-1 [9]	607	–	97.7
KWT-3 [9]	5361	–	98.5
LG-Net6 [33]	313	–	96.8
BC-ResNet-1 [28]	9	3.1	96.9
BC-ResNet-3 [28]	54	16.2	98.2
BC-ResNet-6 [28]	188	53.1	98.6
KW-MLP [13]	424	–	97.6
PhonMatchNet [34]	653	–	96.8
KWM-T-12 [24]	700	–	98.6
LeTR-128t [35]	404	37.2	97.8
LeTR-128s [35]	454	40.4	98.1
DART [36]	93	4.9	96.6
FairDART [36]	188	10.6	96.7
NoisyDART [36]	109	6.3	96.8
DynTempConvNet[37]	62	6.1	97.5
Proposed (2D + 1D DS + SP)	54	3.8	97.5

All results are taken from the corresponding papers

and MACs. On both Raspberry Pi 5 and LattePanda V1, the proposed model showed the lowest mean inference latency among the reproduced models. This result is consistent with its low-MAC design and suggests that the lower computational burden also translates into practical runtime efficiency on CPU-based edge platforms. While latency will still depend on implementation details and hardware characteristics, the observed trend supports the suitability of the proposed configuration for small-footprint on-device KWS.

Table 5 summarizes literature-reported results for representative KWS models on the GSC v2 12-class task, including both earlier small-footprint baselines and more recent architectures. Because these values were obtained under different preprocessing, augmentation, and optimization settings, they should be interpreted as indicative operating points rather than strictly controlled head-to-head comparisons. Within this broader landscape, the proposed model remains competitive in the low-compute regime: it achieves the same or higher reported accuracy than several compact CNN-based and dynamically efficient baselines at lower or comparable MAC budgets, while some larger attention-based, transformer-based, and state-space-based models achieve stronger peak accuracy only at substantially larger parameter scales or computational costs. These observations reinforce the positioning of the present work as a compute-conscious design study rather than as a claim of universal state-of-the-art superiority.

With 54 K parameters and 3.8 M MACs, the proposed hybrid design achieved 97.5% top-1 accuracy. Figure 2 plots accuracy against MACs for models that report computational cost. As shown in the figure, the proposed design achieves higher accuracy than Attention RNN while requiring substantially fewer computations (22.3 M vs. 3.8 M MACs). Relative to compact baselines such as MatchboxNet-3 × 2 × 64 and DART-style models, the proposed design also occupies a favorable operating point, achieving higher accuracy with lower or comparable computational cost. In addition, the proposed model matches the reported accuracy of DynTempConvNet while using fewer MACs (3.8 M vs. 6.1 M). These comparisons suggest that the studied hybrid design remains competitive among recent low-compute KWS architectures without requiring a larger computational budget.

Figure 3 summarizes accuracy with respect to the number of parameters. The proposed model remains competitive with only 54 K parameters, outperforming several compact baselines such as TENet12, TC-ResNet, and DART-style variants

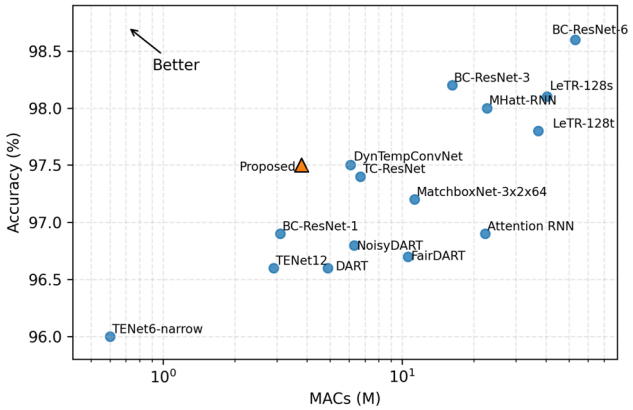

Fig. 2 Accuracy versus MACs for representative KWS models on the GSC v2 12-class task. The proposed hybrid design (orange triangle) achieves 97.5% accuracy with 3.8 M MACs, indicating a favorable accuracy-computation operating point

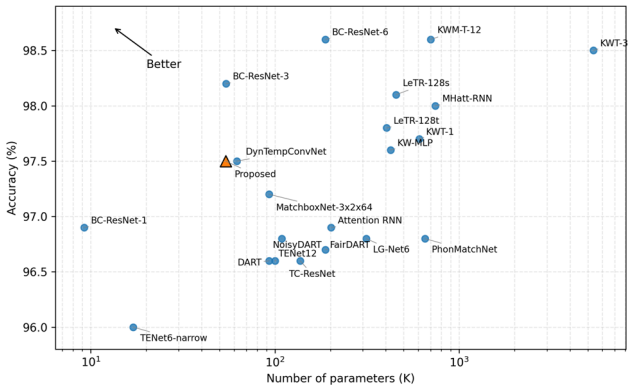

Fig. 3 Accuracy versus model size (number of parameters) for representative KWS models on the GSC v2 12-class task. Although several larger models achieve higher accuracy, the proposed model remains competitive with fewer than 100 K parameters

while using fewer or comparable parameters. At the same time, parameter count alone does not fully capture efficiency. For example, BC-ResNet-3 uses the same number of parameters (54 K) but requires a much higher computational cost (16.2 M MACs), whereas BC-ResNet-1 operates at a similar MAC budget (3.1 M) but achieves lower accuracy (96.9%). Likewise, some recent models with similar or moderately larger parameter counts, such as LeTR-128 t/s and DynTempConvNet, provide strong reported accuracy but generally at higher computational cost than the proposed design. These observations indicate that the proposed design is particularly attractive when computational cost is a stricter constraint than parameter count alone.

Several attention-based, transformer-based, and recent state-space-style models achieve higher peak accuracy, but at substantially larger scales. For instance, KWT-1 and KWT-3 report 97.7% and 98.5% accuracy with 607 K and 5.36 M parameters, respectively, while MHAtt-RNN and KWM-T-12 also exceed the parameter budget of the proposed design by a wide margin. More recent models such as LeTR-128s and Keyword Mamba-style architectures further illustrate that stronger peak accuracy can be achieved, but typically with larger parameter counts, higher MACs, or both. In contrast, the proposed model is designed for a stricter small-footprint regime, where low computational cost and practical on-device efficiency are treated as primary constraints.

Noise-condition evaluation under varying SNR conditions

To provide additional evidence beyond the clean-condition test set, we evaluated the models under varying SNR conditions using controlled white-noise corruption. Each clean test utterance was mixed with additive Gaussian white noise scaled to a predefined waveform-level SNR, and no additional noisy-data retraining was performed. The same procedure was applied to both the proposed model and the reproduced baselines so that relative degradation trends could be compared under a shared noisy-test setting.

As shown in Table 6, performance decreased for all models as the SNR was reduced. Within this shared protocol, the proposed model consistently achieved higher accuracy than the reproduced baselines across all tested SNR levels and showed smaller relative degradation in the lower-SNR regime. This suggests that the studied hybrid configuration preserved discriminative keyword information more effectively than the reproduced baselines under the tested white-noise conditions.

These results should be interpreted as controlled noisy-condition stress tests rather than as a comprehensive robustness benchmark. In particular, the present experiments do not cover other important real-world factors such as nonstationary environmental noise, reverberation, far-field effects,

Table 6 Classification accuracy (%) of the proposed model and the reproduced baselines under varying white-noise SNR conditions (average over 5 seeds)

Model	SNR (in dB)					
	Clean	20	10	5	0	− 5
MatchboxNet-3 × 2 × 64 [27]	96.5	91.7	83.4	74.3	52.5	11.9
BC-ResNet-3 [28]	96.3	91.1	84.4	73.1	40.8	12.8
Proposed model	97.5	94.3	89.7	83.9	66.4	23.5

speaker variation, accent variation, or streaming false-accept/false-reject behavior. Nevertheless, the observed SNR trends provide useful supplementary evidence that the proposed low-MAC design remains competitive beyond the clean GSC v2 test condition.

Conclusion

This paper investigated a lightweight hybrid design for small-footprint keyword spotting that combines a shallow 2D time–frequency front-end, frequency-only pooling, a 1D depthwise-separable temporal backbone, and statistics pooling. Rather than introducing a new primitive module, the study focused on whether this particular allocation of established components forms a competitive operating point when computational cost is a primary constraint.

On the GSC v2 12-class task, the proposed model achieved 97.5% accuracy with 54 K parameters and 3.8 M MACs. The ablation and sensitivity analyses showed that a shallow 2D front-end, frequency-only pooling, and statistics pooling together provide modest but consistent benefits within the tested low-MAC design range. Under the same evaluation pipeline, the reproduced baseline comparisons further indicated that the proposed model offers a favorable balance of accuracy, computational cost, and CPU inference latency. In addition, controlled white-noise experiments showed smaller relative degradation than the reproduced baselines across the tested SNR levels.

These findings should be interpreted within the scope of the GSC v2 12-class setting and the shared protocols adopted in this study. The present work therefore does not claim universal superiority across all KWS architectures or deployment scenarios, but instead identifies a practical low-MAC operating point for efficient on-device KWS. Future work will extend this analysis to broader benchmark settings, various noise conditions, streaming evaluation metrics, quantized deployment, and energy-aware validation on lower-power edge hardware, as well as to multilingual and larger-vocabulary KWS settings.

Author contributions All authors contributed to the study conception and design. Jaewon Lee and Sangbeom Lee performed the conceptualization of the research. Methodology and software development were conducted by Jaewon Lee and Bumhwi Kim. Investigation and data collection were carried out by Jaewon Lee and Doosik Kim. Formal analysis, validation, and visualization were performed by Jaewon Lee. Project administration was managed by Sangbeom Lee. The first draft of the manuscript was written by Jaewon Lee. Funding acquisition were provided by Kwang-yong Kim. All authors read and approved the final manuscript.

Funding This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [26ZD1140, Advancement and Commercialization of Daegu-Gyeongbuk Regional Strategic Industries (Robots, Mobility, AI, Medical, etc.)].

Data availability The datasets (Google Speech Commands dataset) analyzed during the current study are available in the public domain.

Code availability The code will be made available upon reasonable request to the corresponding author.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethics approval Not applicable.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

References

1. Sigtia S, Haynes R, Richards H, Marchi E, Bridle J (2018) Efficient voice trigger detection for low resource hardware. In: Interspeech 2018. ISCA, pp 2092–2096
2. López-Espejo I, Tan Z-H, Hansen JHL, Jensen J (2022) Deep spoken keyword spotting: an overview. IEEE Access 10:4169–4199. https://doi.org/10.1109/ACCESS.2021.3139508
3. Barovic A, Moin A (2025) TinyML for speech recognition. In: 2025 IEEE 49th annual computers, software, and applications conference (COMPSAC), pp 1631–1636
4. Bartoli P, Bondini T, Veronesi C, Giudici A, Antonello N, Zappa F (2025) End-to-end efficiency in keyword spotting: a system-level approach for embedded microcontrollers. In: 2025 IEEE SENSORS, pp 1–4
5. Rose RC (1995) Keyword detection in conversational speech utterances using hidden Markov model based continuous speech recognition. Comput Speech Lang 9:309–333. https://doi.org/10.1006/csla.1995.0015
6. Zhang Y, Glass JR (2009) Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. In: 2009 IEEE workshop on automatic speech recognition & understanding, pp 398–403
7. Chen G, Parada C, Heigold G (2014) Small-footprint keyword spotting using deep neural networks. In: 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 4087–4091
8. Sainath TN, Parada C (2015) Convolutional neural networks for small-footprint keyword spotting. In: Interspeech 2015. ISCA, pp 1478–1482

9. Berg A, O'Connor M, Cruz MT (2021) Keyword transformer: a self-attention model for keyword spotting. Interspeech 2021:4249–4253
10. Zhang Y, Suda N, Lai L, Chandra V (2018) Hello edge: keyword spotting on microcontrollers. arXiv preprint arXiv:1711.07128, https://doi.org/10.48550/arXiv.1711.07128
11. Tang R, Lin J (2018) Deep residual learning for small-footprint keyword spotting. In: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 5484–5488
12. Choi S, Seo S, Shin B, Byun H, Kersner M, Kim B, Kim D, Ha S (2019) Temporal convolution for real-time keyword spotting on mobile devices. arXiv preprint arXiv:1904.03814, https://doi.org/10.48550/arXiv.1904.03814
13. Morshed MM, Ahsan AO (2022) Attention-free keyword spotting. arXiv preprint arXiv:2110.07749, https://doi.org/10.48550/arXiv.2110.07749
14. Dai W, Jiang Y, Liu Y, Chen J, Sun X, Tao J (2025) CAB-KWS : contrastive augmentation: an unsupervised learning approach for keyword spotting in speech technology. In: Antonacopoulos A, Chaudhuri S, Chellappa R, Liu C-L, Bhattacharya S, Pal U (eds) Pattern recognition. Springer Nature Switzerland, Cham, pp 98–112
15. Garai S, Samui S (2024) Exploring TinyML frameworks for small-footprint keyword spotting: a concise overview. In: 2024 international conference on signal processing and communications (SPCOM), pp 1–5
16. Cioflan C, Cavigelli L, Rusci M, de Prado M, Benini L (2024) On-device domain learning for keyword spotting on low-power extreme edge embedded systems. In: 2024 IEEE 6th international conference on AI circuits and systems (AICAS), pp 6–10
17. Syafalni I, Amadeus C, Sutisna N, Adiono T (2024) Efficient real-time smart keyword spotting using spectrogram-based hybrid CNN-LSTM for edge system. IEEE Access 12:43109–43125. https://doi.org/10.1109/ACCESS.2024.3380350
18. Snyder D, Garcia-Romero D, Sell G, Povey D, Khudanpur S (2018) X-vectors: robust DNN embeddings for speaker recognition. In: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 5329–5333
19. Okabe K, Koshinaka T, Shinoda K (2018) Attentive statistics pooling for deep speaker embedding. Interspeech 2018:2252–2256
20. Desplanques B, Thienpondt J, Demuynck K (2020) ECAPA-TDNN: emphasized channel attention, propagation and aggregation in TDNN based speaker verification. Interspeech 2020:3830–3834
21. Wang S, Yang Y, Qian Y, Yu K (2021) Revisiting the statistics pooling layer in deep speaker embedding learning. In: 2021 12th international symposium on Chinese spoken language processing (ISCSLP), pp 1–5
22. Garai S, Samui S (2025) Advances in small-footprint keyword spotting: a comprehensive review of efficient models and algorithms. arXiv:https://arxiv.org/abs/2506.11169v1. Accessed 20 Nov 2025
23. Rohlicek JR, Russell W, Roukos S, Gish H (1989) Continuous hidden Markov modeling for speaker-independent word spotting. In: International conference on acoustics, speech, and signal processing, vol 1, pp 627–630
24. Ding H, Dong W, Mao Q (2025) Keyword mamba: spoken keyword spotting with state space models. Comput Speech Lang 99:101909. https://doi.org/10.1016/j.csl.2025.101909
25. Sørensen PM, Epp B, May T (2020) A depthwise separable convolutional neural network for keyword spotting on an embedded system. J Audio Speech Music Proc 2020:10. https://doi.org/10.1186/s13636-020-00176-2
26. Xu M, Zhang X-L (2020) Depthwise separable convolutional ResNet with squeeze-and-excitation blocks for small-footprint keyword spotting. Interspeech 2020:2547–2551
27. Majumdar S, Ginsburg B (2020) MatchboxNet: 1D time-channel separable convolutional neural network architecture for speech commands recognition. Interspeech 2020:3356–3360
28. Kim B, Chang S, Lee J, Sung D (2023) Broadcasted residual learning for efficient keyword spotting. arXiv preprint arXiv:2106.04140, https://doi.org/10.48550/arXiv.2106.04140
29. Warden P (2018) Speech commands: a dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209, https://doi.org/10.48550/arXiv.1804.03209
30. Li X, Wei X, Qin X (2020) Small-footprint keyword spotting with multi-scale temporal convolution. arXiv preprint arXiv:2010.09960, https://doi.org/10.48550/arXiv.2010.09960
31. de Andrade DC, Leo S, Viana MLDS, Bernkopf C (2018) A neural attention model for speech command recognition. arXiv:https://arxiv.org/abs/1808.08929v1. Accessed 26 Nov 2025
32. Rybakov O, Kononenko N, Subrahmanya N, Visontai M, Laurenzo S (2020) Streaming keyword spotting on mobile devices. Interspeech 2020:2277–2281
33. Wang L, Gu R, Chen N, Zou Y (2021) Text anchor based metric learning for small-footprint keyword spotting. In: Interspeech 2021. ISCA, pp 4219–4223
34. Lee Y-H, Cho N (2023) PhonMatchNet: phoneme-guided zero-shot keyword spotting for user-defined keywords. Interspeech 2023:3964–3968
35. Ding K, Zong M, Li J, Li B (2022) LETR: a lightweight and efficient transformer for keyword spotting. In: ICASSP 2022—2022 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 7987–7991
36. Zhang B, Li W, Li Q, Zhuang W, Chu X, Wang Y (2021) AutoKWS: keyword spotting with differentiable architecture search. In: ICASSP 2021—2021 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 2830–2834
37. Liu R, Chen W, Zhang X (2025) Dynamic convolution models for cross-frontend keyword spotting. Sci Rep 15:16745. https://doi.org/10.1038/s41598-025-00304-y

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.