

Article

A Collaborative Multi-Compression Acceleration Mechanism for Neural Networks in Keyword Spotting

Junbang Jiang ¹, Rui Pu ¹, Jin Li ^{1,*} and Man Zhu ^{2,3,*}
¹ National “111 Research Center” Microelectronics and Integrated Circuits, School of Science, Hubei University of Technology, Wuhan 430068, China; jiangjunbang@hbut.edu.cn (J.J.); purui@hbut.edu.cn (R.P.)

² Intelligent Transportation Systems Research Center, Wuhan University of Technology, Wuhan 430063, China

³ State Key Laboratory of Maritime Technology and Safety, Wuhan University of Technology, Wuhan 430063, China

* Correspondence: lijn@hbut.edu.cn (J.L.); man.zhu.393@whut.edu.cn (M.Z.)

Abstract

To address the large model size, high computational cost, and limited deployment resources of keyword spotting models on edge platforms, this study proposes a collaborative multi-compression framework for lightweight deployment. Built on LiteKWS-Net, an attention-enhanced 2-D convolutional backbone, the framework combines adaptive importance-aware structured pruning, mixed-precision quantization, and quantization-aware multi-stage knowledge distillation. The retrained teacher reaches 97.90% (mean, 100,813 parameters, 0.385 MiB). MPDQ reaches $95.53 \pm 1.16\%$ at $8.27\times$ theoretical weight compression. AIASP reaches 97.59% at a 30% target and 43.9% realized sparsity. The final joint model reaches 96.82% and, under ideal packed sparse mixed-precision storage, has a $51.55\times$ theoretical weight-compression factor relative to the FP32 teacher; sparse-index overhead is excluded. On a Jetson Nano, the TensorRT FP16 network-body benchmark reports 2.86 ms latency and 0.69 mJ per inference.

Keywords: keyword spotting; neural network; model compression; mixed-precision quantization; structured pruning; knowledge distillation

1. Introduction

With the development of intelligent speech interaction technologies, keyword spotting (KWS) has become a common front-end component in smart homes, in-vehicle terminals, wearable devices, and edge speech nodes [1–3]. Compared with large-vocabulary continuous speech recognition tasks, KWS is typically oriented toward limited-vocabulary and always-listening scenarios and therefore places emphasis on low latency, low power consumption, and reliable operation. Consequently, efficient deployment on resource-constrained platforms remains an active research topic [1–3]. The introduction of the Speech Commands dataset provided a unified public benchmark for limited-vocabulary speech recognition and keyword spotting, supporting the development of compact speech models and embedded KWS systems [1].

Early research on keyword spotting model design mainly focused on lightweight convolutional neural networks. Representative models such as MatchboxNet reduce parameter count and computational complexity while maintaining recognition performance through depthwise separable convolutions, compact topologies, and edge-oriented design strategies [2]. However, the emergence of self-supervised speech representation models such as wav2vec 2.0 and HuBERT has improved feature representation capability while



Academic Editor: Jie Yang

Received: 3 July 2026

Revised: 9 August 2026

Accepted: 12 August 2026

Published: 18 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

also increasing model size, storage demand, and inference cost. This trend introduces additional challenges for direct deployment on resource-constrained platforms such as FPGAs, MCUs, and mobile devices [4–6]. Consequently, how to reduce model size and computational cost while preserving KWS accuracy and maintaining hardware-friendly deployment characteristics remains an important issue in current research.

To address these challenges, recent studies have mainly explored low-bit quantization, structured pruning, and knowledge distillation. In quantization, binarization and ultra-low-bit modeling have been reported for compact KWS networks, and related work has shown that low-bit representations can reduce storage overhead and multiply–accumulate complexity [7–10]. In distillation, teacher–student learning frameworks have been adopted for performance compensation in compact models and have been studied in device-constrained scenarios, self-supervised representation transfer, and in-memory computing settings [9–13]. These studies provide a basis for lightweight deployment of KWS models; however, most of them focus on a single compression strategy, and the interaction among multiple compression mechanisms remains insufficiently studied.

In broader speech-model compression research, joint optimization of pruning, distillation, and quantization has become an active direction. Existing studies indicate that the combination of distillation and pruning can be used to compress self-supervised speech models [12]. Structured pruning and related self-distillation methods have also been extended to end-to-end automatic speech recognition and wake-word models [13–15]. At the deployment level, hardware-aware co-design has jointly considered KWS network structures, quantization precision, and accelerator architectures to improve throughput and energy efficiency on FPGA platforms [10,16,17]. Variable-scale pruning has further been explored for transformer-based speech recognition [18]. In addition, integrated compression, one-pass multi-model compression, and ultra-low-bit mixed-precision quantization suggest that model compression is evolving from conventional staged optimization toward unified modeling and joint search [19–23]. Overall, quantization, pruning, and distillation have each been associated with improved deployment efficiency to some extent, yet their coordination under hardware-oriented deployment constraints remains insufficiently explored.

Current studies still exhibit several limitations. First, many methods focus on a single compression strategy, such as quantization alone or pruning alone, and therefore lack systematic analysis of the interaction among multiple compression mechanisms [7–15]. Second, although some methods achieve high compression ratios, they pay insufficient attention to practical deployment constraints such as on-chip storage, DSP utilization, and memory bandwidth, which limits the translation of algorithmic gains into real acceleration benefits [10,16,17]. Third, the combination of low-bit quantization and structured pruning often introduces non-negligible performance degradation, while existing recovery mechanisms still leave room for improvement in training stability, task-specific knowledge transfer, and deployment consistency [9–15]. It is therefore of both theoretical and practical interest to develop a collaborative multi-compression method that jointly balances model accuracy, compression efficiency, and deployment friendliness.

Motivated by these issues, this study proposes a collaborative multi-compression acceleration method for efficient deployment of KWS models on resource-constrained platforms. The proposed framework integrates hardware-friendly mixed-precision dynamic quantization, adaptive structured pruning, and quantization-aware multi-stage knowledge distillation into a unified optimization pipeline. Through coordinated design across parameter representation, network structure, and knowledge transfer, the framework jointly balances model-size reduction, computational-cost reduction, and accuracy preservation. The main contributions are as follows. First, a hardware-friendly mixed-precision dynamic

quantization method is proposed for KWS deployment, where Fisher sensitivity, activation outliers, and hardware cost are jointly used for bit-width allocation. Second, an adaptive structured pruning method is introduced, where channel gating, importance evaluation, and local structure preservation are combined to remove redundant channels in a regularized manner. Third, a quantization-aware multi-stage knowledge distillation method is proposed to improve accuracy recovery of compressed models. Fourth, these components are integrated into a unified collaborative optimization pipeline and evaluated on the KWS task. The reported results indicate that the proposed method reduces model complexity while retaining the observed recognition accuracy, providing an empirical reference for lightweight deployment of KWS models.

The remainder of this paper is organized as follows. Section 2 (Materials and Methods) introduces LiteKWS-Net (Section 2.1), the attention-enhanced lightweight convolutional backbone that serves as both the compression target and the source of the pruning prior. Section 2.2 details the collaborative multi-compression strategy, comprising mixed-precision dynamic quantization (MPDQ), adaptive importance-aware structured pruning (AIASP) with its attention-guided variant (AGP), and quantization-aware progressive multi-stage knowledge distillation (PMKD), together with their joint coordination. Section 3 reports the experimental results, including on-device hardware benchmarking, compression-ordering and component ablations, and hyper-parameter sweeps. Section 4 concludes with limitations and future work.

2. Materials and Methods

Software and implementation details. Model training and compression experiments were implemented in Python 3.10.15 using PyTorch 2.4.1, torchaudio 2.4.1, NumPy 2.2.6, pandas 2.2.3, SciPy 1.14.1, and Matplotlib 3.10.0 with CUDA 12.1 on an NVIDIA GeForce RTX 2080 Ti. For model export and on-device evaluation, ONNX 1.17.0 and ONNX Runtime 1.20.1 were used, and the exported network bodies were executed with TensorRT 8.2.1 (FP16) under JetPack R32.7.1 on an NVIDIA Jetson Nano Developer Kit. The TensorRT benchmarks used `trtexec` with `-warmUp = 200`, `-iterations = 200`, and `-avgRuns = 100`.

2.1. LiteKWS-Net: An Attention-Enhanced Lightweight Backbone

Keyword spotting on edge platforms such as FPGAs, microcontrollers, and always-listening wearable nodes often involves practical constraints such as limited on-chip memory, low per-inference latency, and low standby power. Although fully self-attentional models such as the Keyword Transformer and self-supervised encoders such as wav2vec 2.0 and HuBERT report high accuracy, their quadratic attention cost, large parameter footprint, and irregular memory access can make them less convenient for these constraints. We therefore retain a convolutional backbone and incorporate broadcasted residual learning, sub-spectral normalization, and squeeze-and-excitation (SE) channel attention to retain spectral capacity while keeping computational cost moderate. We name the resulting network LiteKWS-Net.

As shown in Figure 1, LiteKWS-Net takes a one-second, 16 kHz monaural waveform, converts it to a 40-band log-Mel spectrogram, and processes it through a 2-D convolutional stem, four broadcasted-residual stages with progressive frequency down-sampling and channel expansion, a point-wise head, and a temporal-attention pooling layer followed by a linear classifier, consistent with the standard setting of Speech Commands Dataset v2. Within each stage, most computation is carried out by 1-D temporal depth-wise convolutions while a small 2-D frequency-domain convolution captures spectral structure; following the broadcasted-residual principle, the temporal branch operates on the frequency-averaged feature and is broadcast back across the frequency axis before being

summed with the 2-D branch, keeping the multiply–accumulate cost close to a 1-D network while preserving 2-D spectral capacity. Each stage applies sub-spectral normalization, which splits the frequency axis into sub-bands and normalizes them independently to match the band-dependent statistics of speech, and ends with an SE module that produces a per-channel gating vector; these learned channel-importance scores are later reused by our Attention-Guided Pruning, turning attention into a bridge between architecture and compression. A learned temporal-attention pooling aggregates the most discriminative frames instead of a uniform average. The width is controlled by a single base-channel parameter, so the same topology serves as both the wider teacher and the narrower student in distillation. With a base width of 32 channels, the teacher contains 100,813 parameters (0.385 MB in FP32) and reaches 97.90% on the 8-keyword task, while the 16-channel student contains roughly 27 K parameters—both more than an order of magnitude smaller than the original 1.6 M-parameter (6.1 MB) 1-D baseline, providing a unified compression target for subsequent pruning, quantization, and distillation.

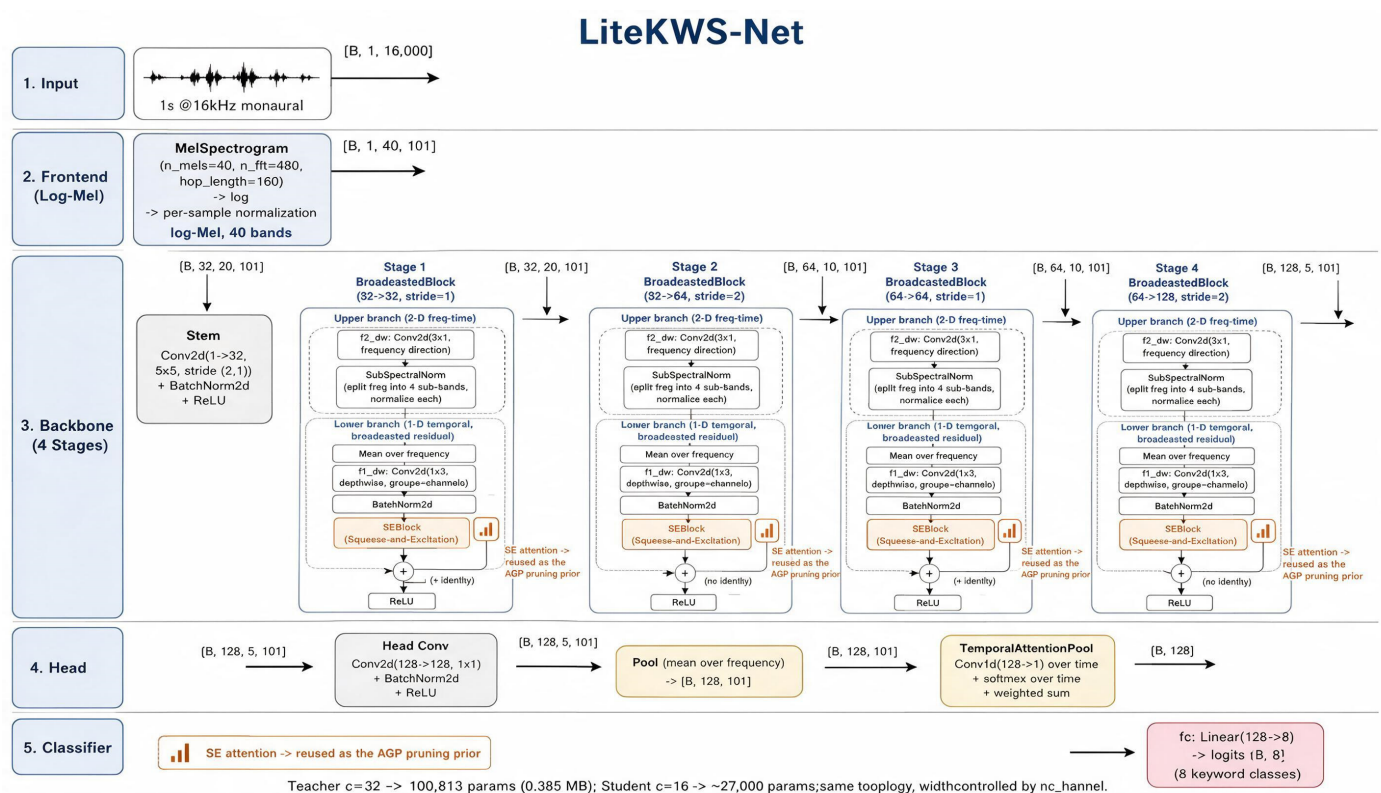


Figure 1. LiteKWS-Net architecture for keyword spotting.

2.2. Model Optimization with a Multi-Compression Strategy

To address limited storage resources, constrained computational capability, and imbalanced hardware resource allocation in edge deployment, this study proposes a collaborative multi-compression optimization method for lightweight keyword spotting models. The method is built on three technical components: mixed-precision dynamic quantization (MPDQ), adaptive importance-aware structured pruning (AIASP), and progressive multi-stage knowledge distillation (PMKD). In addition, hardware-cost constraints, quantization-noise modeling, and joint distillation mechanisms are introduced to perform systematic compression optimization for an end-to-end CNN-based KWS model. Figure 2 illustrates the proposed multi-compression framework.

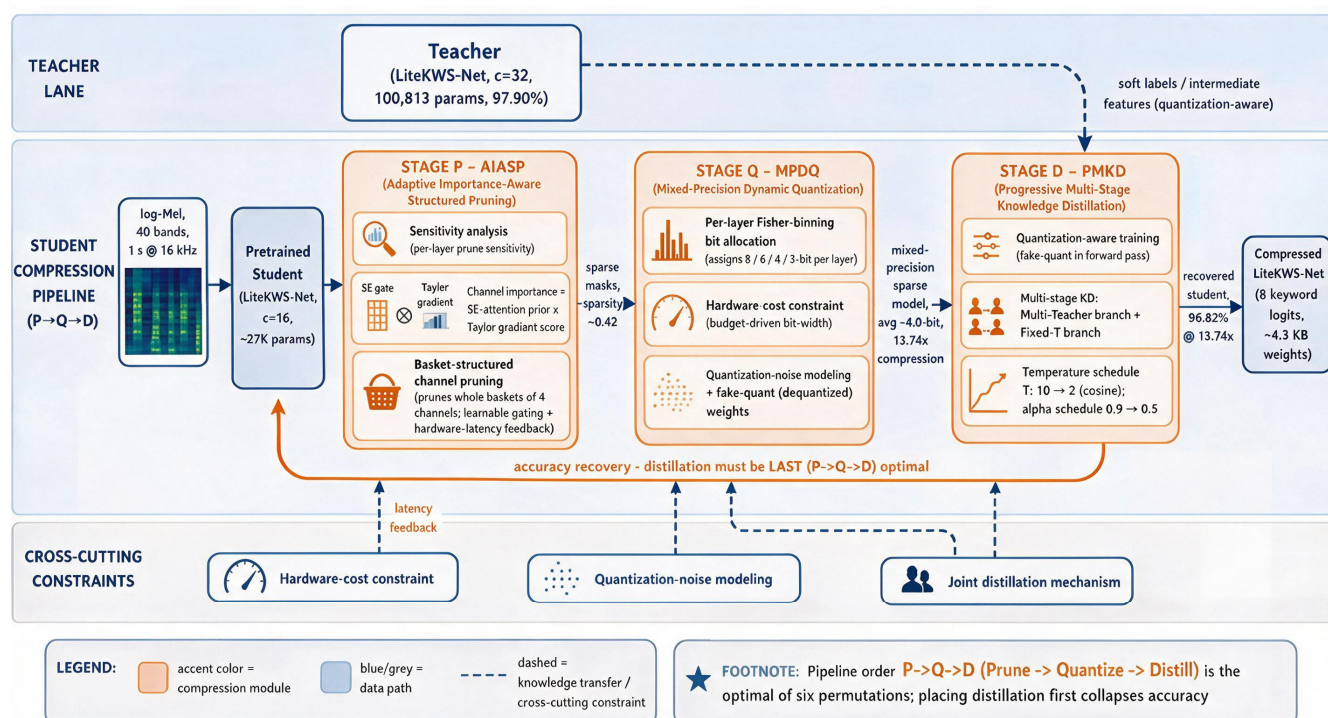


Figure 2. Conceptual framework of the collaborative multi-compression strategy; numerical results are reported separately in the tables.

The overall optimization procedure can be divided into three stages, although these stages are not isolated serial modules. First, AIASP performs regularized structured pruning through learnable channel gating, Taylor-based importance modeling, and local structure preservation, while using a latency proxy as feedback to constrain pruning intensity. Second, MPDQ applies hardware-friendly mixed-precision bit-width allocation to the compact network after pruning by jointly considering Fisher information, activation outliers, and a hardware-cost proxy. Third, PMKD introduces quantization-aware distillation and dynamic scheduling under low-bit and sparse structural conditions to recover the accuracy of the compressed model. This process forms a closed-loop compression pipeline involving structural pruning, low-bit adaptation, distillation-based compensation, and hardware feedback, enabling compression ratio, recognition accuracy, and deployment friendliness to be optimized within a unified framework.

2.2.1. Mixed-Precision Dynamic Quantization

Quantization maps high-precision floating-point parameters into low-bit integer representations and is a key technique for reducing model storage and computational overhead. As KWS networks are increasingly oriented toward edge deployment and lightweight implementation, the role of quantization extends beyond parameter compression and must also account for operator throughput, on-chip storage consumption, and tolerance to low-bit noise. Conventional uniform-precision quantization applies the same bit width to all layers and therefore cannot simultaneously protect critical layers and reduce storage for less sensitive ones. In addition, bit-width allocation methods based only on weight distributions tend to ignore activation outliers and differences in hardware mapping. To address these issues, this study proposes a hardware-friendly mixed-precision dynamic quantization method that jointly considers Fisher sensitivity, activation outlier strength, and deployment cost to obtain bit-width allocation intended to account for practical acceleration requirements.

Sensitivity Modeling

For a network parameterized by θ , we adopt the diagonal approximation of the Fisher Information Matrix (FIM) to estimate parameter importance. The diagonal element for the i -th parameter is defined as:

$$F_i = \frac{1}{N} \sum_{n=1}^N \left(\frac{\partial L_n}{\partial \theta_i} \right)^2 \quad (1)$$

where N denotes the number of calibration samples and L represents the loss for the n -th sample. A larger Fisher value indicates greater sensitivity. To account for activation outliers, we further introduce a layer-level activation statistic and combine it with the average Fisher sensitivity. The joint score for the l -th layer is defined as:

$$S_l = \alpha \cdot \text{Norm} \left(\frac{1}{|\Theta_l|} \sum_{i \in \Theta_l} F_i \right) + \beta \cdot \text{Norm}(O_l) \quad (2)$$

where Θ_l denotes the parameter set of the l -th layer, $\text{Norm}(\cdot)$ represents the normalization operation, and α and β are balancing coefficients. This score concurrently reflects the impact of the parameters on the loss function and the amplification effect of low bit-widths on peak responses. At the implementation level, forward and backward propagation are performed batch-by-batch on the calibration set to record the squared parameter gradients, layer output peaks, and activation dynamic ranges. Quantile truncation and exponential moving average (EMA) are then employed to smooth these estimates. Compared to methods that rely solely on weight magnitudes or single-pass statistics for quantization, the proposed strategy combines multiple layer-level signals for subsequent bit-width allocation.

Bit-Width Allocation and Quantization Implementation

Upon obtaining the layer-wise joint sensitivity, we map continuous scores to a discrete candidate bit-width set subject to a hardware-cost proxy. The optimization balances estimated quantization error with the proxy cost and is expressed as:

$$\min_{\{b_l\}} \sum_{l=1}^L (\lambda_1 S_l E_q(b_l) + \lambda_2 C_{hw}(b_l)) \quad (3)$$

where b_l denotes the quantization bit-width of the l -th layer, $E_q(b_l)$ represents the quantization error term under the corresponding bit-width, $C_{hw}(b_l)$ denotes the hardware cost function, and λ_1 and λ_2 are balancing coefficients.

Specifically, high-sensitivity layers are assigned 8-bit or 6-bit representations, moderately sensitive layers preferentially use 4-bit representations, and low-sensitivity layers may use lower precision. We construct the hardware-cost proxy from on-chip storage, multiply-accumulate, and parallelism terms; it is used only for the bit-allocation study and is not a direct Jetson Nano measurement (Table 1).

Table 1. Mixed-precision bit-width allocation rules.

Fisher Normalization Range	Sensitivity Level	Assigned Bit-Width
[0.75, 1.00]	High Sensitivity	8-bit
[0.50, 0.75)	Medium-High Sensitivity	6-bit
[0.25, 0.50)	Medium-Low Sensitivity	4-bit
[0.00, 0.25)	Low Sensitivity	2-bit

For the quantization implementation, we employ a per-channel asymmetric integer mapping scheme. The quantization process is expressed as:

$$q = \text{clip}\left(\text{round}\left(\frac{x}{s}\right) + z, q_{\min}, q_{\max}\right) \quad (4)$$

where x is the input tensor, s is the scaling factor, z is the zero-point offset, and q is the quantized integer value. We further combine per-channel scaling with outlier clipping so that training-time error estimation better approximates the intended low-bit representation.

Overall, the proposed mixed-precision dynamic quantization method extends bit-width partitioning based solely on statistical thresholds by incorporating Fisher sensitivity, outlier awareness, and hardware-cost feedback. The method reduces the weight-storage estimate and provides a low-bit basis for subsequent structured pruning and distillation compensation.

2.2.2. Adaptive Structured Pruning

After baseline training and sensitivity analysis, the model may retain structural redundancy. To further reduce parameter count and computational complexity, this study extends the original structured pruning framework by introducing learnable gating and hardware-latency feedback, allowing redundant channels and filters to be pruned in a regularized manner. Unlike traditional pruning approaches that rely solely on static ranking, the proposed method combines importance evaluation, local structure preservation, and deployment-cost constraints, with the aim of producing pruning patterns suitable for resource-constrained platforms.

Channel Importance Evaluation

The crux of pruning lies in evaluating the importance of each channel to the model output. Traditional weight magnitude-based methods solely focus on parameter size, which fails to accurately reflect the true impact on the loss function. To address this, the Taylor expansion is retained as the theoretical foundation, while the object of importance modeling is extended from individual weights to channel-level gating variables. Let $g_{l,c}$ denote the gating variable corresponding to the c -th output channel of the l -th layer; the change in loss induced by the attenuation of this channel can then be approximated as:

$$\Delta L_{l,c} \approx \left| \frac{\partial L}{\partial g_{l,c}} g_{l,c} \right| \quad (5)$$

where L is the loss function for the keyword spotting (KWS) task. This metric approximates the instantaneous impact of channel pruning on model performance. Because gradient fluctuations can arise in single-batch samples, channel activation energy and layer-level sensitivity information are further incorporated to construct a joint importance score:

$$I_{l,c} = \alpha \left| \frac{\partial L}{\partial g_{l,c}} g_{l,c} \right| + \beta \text{Norm}(A_{l,c}) + \gamma \text{Norm}(F_l) \quad (6)$$

where $A_{l,c}$ represents the average activation energy of the c -th channel in the l -th layer, F_l denotes the Fisher sensitivity statistic of that layer, and α , β , and γ are balancing coefficients. This score combines gradient sensitivity, channel response strength, and layer vulnerability to identify high-contribution, redundant, and noise-sensitive channels. At the implementation level, forward and backward propagation are performed batch-by-batch on the calibration set, the gating sensitivity is averaged across multiple samples, and the results are integrated with activation statistics through normalized fusion. Compared to methods that rank channels based solely on instantaneous gradients or weight norms, this

strategy uses aggregated signals as a structural prior for subsequent local pruning and recovery training.

Local Gated Pruning

Conventional channel pruning typically employs a global uniform ranking scheme, wherein all channels are sorted by their importance scores and the lowest-scoring channels are directly pruned. Although simple to implement, this method is prone to disrupting local functional structures, particularly causing concentrated damage to consecutive convolutional channels responsible for short-term pattern extraction in KWS networks. To mitigate this, the local partitioning philosophy of Spinal Basket is retained, and a gating competition mechanism is introduced within each local unit, allowing channels to undergo a two-stage “soft selection—hard pruning” filtering process inside local functional blocks. Specifically, the output channels of each layer are partitioned into several local Baskets. Within each Basket, channels are ranked based on their joint importance scores, while the retention ratio is dynamically adjusted in conjunction with layer-wise hardware latency. The local retention ratio for the k -th Basket is defined as:

$$r_{l,k} = r_0 - \eta_1 \text{Norm}(T_l) + \eta_2 \text{Norm}(\bar{T}_{l,k}) \quad (7)$$

where r_0 is the baseline retention rate, T_l denotes the latency cost proxy for the l -th layer, $\bar{T}_{l,k}$ represents the average importance of the k -th Basket, and η_1 and η_2 are regulation coefficients. For latency-sensitive layers with low importance, the local pruning ratio is increased; conversely, for layers that are important to final KWS accuracy, the pruning intensity is reduced. This approach is intended to limit concentrated removal of consecutive functional channels and to align the number of retained channels with parallel-array and cache-organization schemes in resource-constrained deployment scenarios. Furthermore, a progressive gated annealing strategy is adopted to execute structured pruning. Specifically, prunable channels are first searched via soft gating, and the gating variables are then progressively forced towards a binary form, deleting the corresponding channels. The gating mechanism can be expressed as:

$$\hat{g}_{l,c}^{(t)} = \sigma\left(\frac{u_{l,c}}{\tau_t}\right) \quad (8)$$

where $u_{l,c}$ is the learnable gating parameter, $\sigma(\cdot)$ is the Sigmoid function, and τ_t is a temperature parameter that decays progressively with training epochs. As τ_t decreases, the gating output gradually approaches a 0–1 form, thereby realizing a smooth transition from soft searching to hard pruning. Compared to one-shot hard pruning, this approach can mitigate accuracy regression induced by structural abruptness and provides a smoother optimization trajectory for subsequent recovery training.

Stable Recovery Training

The model architecture changes after pruning, and direct fine-tuning can over-modify important, retained parameters. We therefore use Elastic Weight Consolidation (EWC) as a stabilization constraint during compression recovery. Equation (9) defines the recovery loss:

$$L_{\text{total}} = L_{\text{CE}}(p^S, y) + \lambda_{\text{ewc}} \sum_i F_i(\theta_i - \theta_i^*)^2 + \lambda_{\text{kd}} L_{\text{KD}}(p^S, p^T) \quad (9)$$

where the first term, cross-entropy loss, maintains task supervision; the second term, the EWC regularization term, restricts high-importance parameters from deviating from the

original solution; and the third term, the distillation auxiliary term L_{KD} , helps the student model inherit the discriminative boundaries of the pre-pruning teacher network. λ_{ewc} and λ_{kd} denote the weight coefficients for the stabilization and distillation constraints, respectively. It is worth noting that EWC acts exclusively on the retained parameters, whereas the channels eliminated during the gating annealing phase are exempt from this constraint. Through this joint strategy of “gated pruning + EWC protection + distillation recovery,” the upgraded AIASP can reduce model parameters and computational complexity while retaining the reported accuracy (Table 2).

Table 2. Complete pruning procedure of AIASP.

Step	Operation	Output
Step 1	Taylor importance calculation	Channel importance scores
Step 2	Sensitivity normalization	Normalized importance metrics
Step 3	Adaptive pruning rate calculation	Pruning rate for each layer
Step 4	Spinal Basket pruning	Generation of structural mask
Step 5	EWC weight registration	Saving Fisher information and parameter snapshots
Step 6	Fine-tuning training	Model accuracy recovery via EWC + distillation fine-tuning

Attention-Guided Pruning (AGP)

Conventional structured pruning estimates channel importance from external signals—weight magnitude, a Taylor expansion of the loss, or Fisher sensitivity—computed separately from the network’s own representation. We observe that the SE modules already embedded in LiteKWS-Net learn, during normal training, a per-channel importance signal in the form of the excitation gate. Attention-Guided Pruning (AGP) reuses this signal as a prior for structured channel pruning, so that the importance estimate is partly produced by the architecture itself. AGP is therefore not a replacement for the adaptive structured pruning (AIASP) introduced above but an attention-guided refinement of its channel-importance step: AIASP still provides the basket-wise local gating, hardware-cost-aware sparsity allocation, and stable distillation-based recovery, while AGP modulates the importance scores with the learned attention prior.

For the c -th output channel of a prunable layer l , let $T(l,c)$ denote normalized Taylor importance and $a(l,c)$ the normalized SE excitation score collected on a calibration set. AGP refines the existing Taylor score as $I(l,c) = T(l,c) \times ((1 - \lambda) + \lambda \times a(l,c))$, with $\lambda = 0.25$. The SE prior is obtained with a forward pass; AGP therefore adds no additional backward pass beyond the Taylor computation already required by the baseline criterion.

The refined scores feed the same basket-wise pruning and recovery procedure used by AIASP. The available extreme-sparsity comparison is based on a limited evaluation, so AGP is presented as an architecture-derived refinement rather than as a uniformly superior accuracy method.

2.2.3. Quantization-Aware Multi-Stage Knowledge Distillation

After quantization and pruning, the model is compressed; however, discretization error, structural sparsification, and weakened representation capacity may jointly reduce KWS performance. To recover the performance of the compressed model, this study extends the conventional knowledge distillation framework by introducing quantization-aware training and stage-wise knowledge transfer, thereby constructing a multi-stage distillation method for low-bit sparse models. The method considers not only the output-distribution discrepancy between teacher and student models but also the effects of intermediate feature

structure and quantization perturbation on the distillation process, with the aim of aligning knowledge transfer with the deployment setting.

Response Distillation

The core objective of traditional logits distillation is to make the output distribution of the student model approximate that of the teacher model. However, for low-bit student models, performing distillation solely on the full-precision training trajectory often underestimates the performance shift induced by quantization noise. To address this, quantization-aware response distillation is introduced in the first stage. Specifically, quantization operators or pseudo-quantization processes are explicitly retained during the forward propagation of the student network, allowing the distillation loss to act directly on the low-bit response space rather than an idealized floating-point space. The distillation objective of this stage is defined as:

$$L_{\text{resp}} = \lambda_{\text{ce}} L_{\text{ce}} + \lambda_{\text{kd}} T^2 D_{\text{KL}}(p_i^{(T)} \parallel p_s^{(T)}) \quad (10)$$

where L_{ce} is the cross-entropy loss for the keyword spotting (KWS) task, $D_{\text{KL}}(\cdot)$ denotes the Kullback–Leibler (KL) divergence, $p_i^{(T)}$ and $p_s^{(T)}$ represent the soft output distributions of the teacher and student models at temperature T , respectively, and λ_{ce} and λ_{kd} are balancing coefficients. This loss enables the low-bit student model to learn the inter-class relational information from the teacher model while maintaining task discrimination capability, thereby mitigating the destruction of decision boundaries caused by quantization errors.

Compared to traditional single-stage output distillation, quantization-aware response distillation places greater emphasis on the “consistency between the distillation process and the deployment state.” Instead of first learning in the floating-point space and subsequently adapting to quantization errors, knowledge transfer is accomplished directly within the compressed environment. Consequently, this approach is more suitable for the optimization of KWS models targeted at resource-constrained deployments.

Feature Transfer

Relying solely on output-layer distillation remains insufficient for fully recovering the internal representation capability of the compressed model. To further enhance the student model’s performance, intermediate-layer feature transfer and relational distillation are incorporated in the second stage, enabling the student model to simultaneously approximate the teacher model in terms of temporal feature expression and channel correlation. First, projection layers are employed to align the intermediate feature dimensions between the teacher and the student, and a feature reconstruction loss is applied to constrain local representation consistency:

$$L_{\text{feat}} = \sum_{l=1}^{L_f} \| P_l(F_s^l) - F_t^l \|_2^2 \quad (11)$$

where F_s^l and F_t^l denote the feature representations of the student and teacher models at the l -th distillation layer, respectively, $P_l(\cdot)$ represents the projection mapping, and L_f is the number of layers participating in distillation. Second, to alleviate the disruption of feature dependencies following structured pruning, relational distillation is further introduced. A Gram matrix is constructed for the features of each layer, and the correlation discrepancy between the teacher and the student is minimized:

$$L_{\text{rel}} = \sum_{l=1}^{L_f} \| G(P_l(F_s^l)) - G(F_t^l) \|_F^2 \quad (12)$$

where $G(\cdot)$ denotes the Gram matrix mapping, and $\| \cdot \|_F$ represents the Frobenius norm. This term preserves the relative structural relationships among channels in the teacher model, which is more suitable for knowledge transfer in compression scenarios than mere point-wise alignment. By integrating the constraints of the above two parts, the distillation objective of the second stage can be formulated as:

$$L_{\text{stage 2}} = L_{\text{resp}} + \lambda_{\text{feat}} L_{\text{feat}} + \lambda_{\text{rel}} L_{\text{rel}} \quad (13)$$

where λ_{feat} and λ_{rel} denote the weights for feature distillation and relational distillation, respectively. Through the optimization of this stage, the student model not only learns the final output of the teacher model but also inherits its intermediate-layer representation patterns and class structural information, thereby improving the generalization capability of the compressed model on complex keyword samples.

Dynamic Scheduling

Fixed temperatures and fixed distillation weights often struggle to accommodate the requirements of different training phases. In the early stage of training, the student model has not yet converged, necessitating a higher temperature to obtain smoother inter-class relational information. In the middle stage, feature distillation and relational distillation need to be strengthened to help the model establish stable internal representations. In the late stage, the temperature should be lowered and auxiliary constraints weakened, allowing the model to focus on converging towards the ultimate KWS objective. Based on this, a dynamic temperature and distillation intensity collaborative scheduling strategy is designed. The temperature parameter varies dynamically with the training epochs, expressed as:

$$T(t) = T_{\min} + (T_{\max} - T_{\min})\exp(-t/\tau) \quad (14)$$

where t denotes the current training epoch, $T_{\max}$ and $T_{\min}$ represent the initial and final temperatures, respectively, and τ is the decay constant. As training progresses, the temperature gradually decreases, enabling the distillation process to transition smoothly from “high-temperature soft supervision” to “low-temperature discriminative convergence.”

Simultaneously, stage-wise collaborative adjustment is adopted for λ_{feat} , λ_{rel} , and λ_{kd} : the early stage focuses on response distillation, the middle stage enhances feature transfer and relation preservation, and the late stage progressively attenuates the weights of the auxiliary terms. This strategy helps to reduce the sensitivity of fixed hyperparameter schemes to experimental environments, cuts down the cost of manual parameter tuning, and improves the stability and reproducibility of the joint compression training.

In summary, the proposed quantization-aware multi-stage knowledge distillation method enhances the compressed model’s capability to inherit teacher knowledge through three mechanisms: response distillation, feature transfer, and dynamic scheduling. This forms a tighter synergy with the aforementioned mixed-precision quantization and structured pruning. Compared to traditional cascaded compression frameworks, this approach better aligns with the current developmental demands for the integrated optimization of lightweight and edge-deployed KWS models.

2.2.4. Joint Method Coordination

Overall, the model initially performs sensitivity analysis and gated pruning, followed by hardware-constrained bit-width allocation, and subsequently executes quantization-

aware multi-stage distillation within the low-bit sparse structure. Compared to the traditional cascaded paradigm of “compress first, remediate later,” this pipeline places greater emphasis on the information feedback loop between compression decisions and recovery training.

The synergistic effects among the three components are primarily reflected in three aspects: First, gated structured pruning initially removes redundant channels, thereby reducing the subsequent search space for mixed-precision and enhancing the stability of bit-width allocation. Second, hardware-constrained quantization concentrates resources on critical layers, yielding higher resource utilization efficiency for low-bit deployment. Third, quantization-aware distillation recovers decision boundaries and internal representations under real compression conditions, thereby compensating for the performance degradation caused by the superimposition of pruning and quantization. Ultimately, the joint compression pipeline forms a closed-loop mechanism of “structural compression—bit-width optimization—representation recovery—hardware feedback” (Tables 3 and 4).

Table 3. Multi-stage PMKD strategy.

Stage	Knowledge Type	Transfer Form	Loss Function
Stage 1	Output distribution knowledge	Teacher/Student soft logits	$L_{CE} + \lambda_1 \cdot T^2 \cdot KL(p_t p_s)$
Stage 2	Intermediate layer feature knowledge	Intermediate features & Gram relation matrix	$\lambda_2 \cdot L_{feat} + \lambda_3 \cdot L_{rel}$
Stage 3	Dynamic scheduling	Temperature and loss-weight schedule	Scheduled response and feature losses

Table 4. Parameter configuration of the joint compression pipeline.

Stage	Method	Key Parameters	Expected Effect
Stage 1	AIASP	Base sparsity 10–60%, Basket = 4, progressive annealing	Reduce parameters, generate structural mask
Stage 2	MPDQ	Candidate bit-widths {2, 4, 6, 8}, asymmetric, per-layer mixed allocation T annealed 10→2, 3-stage $\alpha = 0.9/0.7/0.5$,	Average bit-width ~3.2-bit
Stage 3	PMKD	feature-KD weight $\beta = 0/0.5/0.2$ at block3 & block5, AdamW + cosine LR	Recover compressed model accuracy

In general, through quantization-aware response distillation, relational-level feature transfer, and dynamic collaborative scheduling, the joint pipeline enhances the compressed model’s capacity to inherit knowledge from the teacher, establishing a tighter synergy with the previously described hardware-friendly quantization and gated structured pruning. Compared to traditional cascaded compression frameworks, this pipeline better aligns with the developmental trend of integrated optimization for lightweight speech models and edge deployment.

Distinction from prior joint compression. Unlike architecture-search pipelines, the proposed pipeline does not add an architecture-search module. It uses the existing SE attention as a pruning prior, Fisher sensitivity for bit assignment, and quantization-aware recovery with fixed pruning masks.

Coefficient selection. The reported schedules are fixed within an experiment. The temperature and alpha sweeps reported here are sensitivity analyses; they do not establish $\alpha = 0.5$ as a preferred setting. The highest alpha-sweep mean is observed at $\alpha = 0.3$. The 3-bit assignments are applied to low-sensitivity layers; high-sensitivity layers are

assigned higher precision. The base pruning target is 30%, with the realized sparsity determined by the resulting masks.

3. Experimental Results and Analysis

Experiments are conducted on the Google Speech Commands Dataset v2. Baseline and uniform-quantization branches use the same data split, optimizer, and learning-rate setting. Results are reported as mean $\pm$ standard deviation over three independent runs.

The distillation branch is further evaluated under a unified teacher (32 channels, 100,813 params) and student (16 channels, 26,861 params) with a consistent training budget of 30 epochs. Fixed-T KD, Multi-Teacher KD, and PMKD are all evaluated on the same student-side architecture, so the performance differences primarily reflect the knowledge-transfer mechanism itself rather than differences in model size or training protocol. Results are reported as mean $\pm$ standard deviation over three independent runs for all distillation methods under this unified teacher–student setting.

3.1. Quantization Results and Ablation Analysis

3.1.1. Main Quantization Results

As shown in Figure 3a, MPDQ achieves $95.53 \pm 1.16\%$ with an $8.27\times$ theoretical weight-compression factor. Among the compared quantization methods, it has the largest reported compression factor, while its mean accuracy is lower than the 8-bit and RTN-4-bit baselines. Figure 3b shows a representative allocation in which the stem, head, and classifier are retained at higher precision while lower-sensitivity convolutional layers use lower bit-widths.

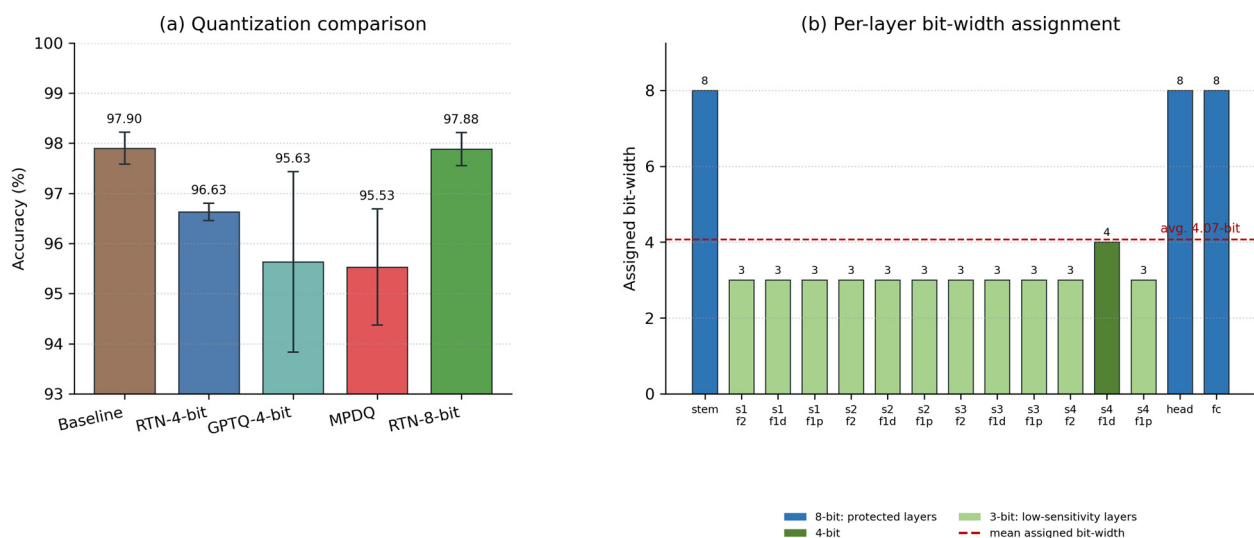


Figure 3. Quantization results: (a) Mean accuracy with standard deviation; (b) representative per-layer bit-width assignment learned by MPDQ. The assignment is shown for a representative run and is interpreted separately from the aggregate accuracy statistics.

Figure 3a reports the quantization results. MPDQ has $95.53 \pm 1.16\%$ mean accuracy and an $8.27\times$ theoretical weight-compression factor. Figure 3b uses the same mean values to display the accuracy–compression trade-off.

Table 5 summarizes the accuracy, compression factor, and theoretical model size of each quantization method.

Table 5. Quantization main results. All rows report mean $\pm$ standard deviation over three independent runs.

Method	Configuration	Accuracy (Mean $\pm$ Std, %)	Compression Ratio ($\times$)	Model Size (MB)
Baseline	Independently retrained baseline	97.90 $\pm$ 0.32	1.00	0.385
RTN-4-bit	4-bit uniform	96.63 $\pm$ 0.17	7.11	0.054
GPTQ-4-bit	4-bit GPTQ-corrected	95.63 $\pm$ 1.80	7.11	0.054
MPDQ	Mixed precision	95.53 $\pm$ 1.16	8.27	0.047
RTN-8-bit	8-bit uniform	97.88 $\pm$ 0.33	4.00	0.096

3.1.2. Quantization Ablation Analysis

Figure 4a demonstrates that the quantization ablation results exhibit a distinct stratification with variations in bit-width. Among them, 8-bit quantization achieves $97.88 \pm 0.33\%$, decreasing to $97.75 \pm 0.33\%$ at 6-bit and $96.63 \pm 0.17\%$ at 4-bit as the bit-width reduces. This indicates that under the current model and data conditions, bit-width remains the dominant factor affecting quantization accuracy, and the quantization errors introduced by lower bit-widths gradually accumulate in high compression regimes.

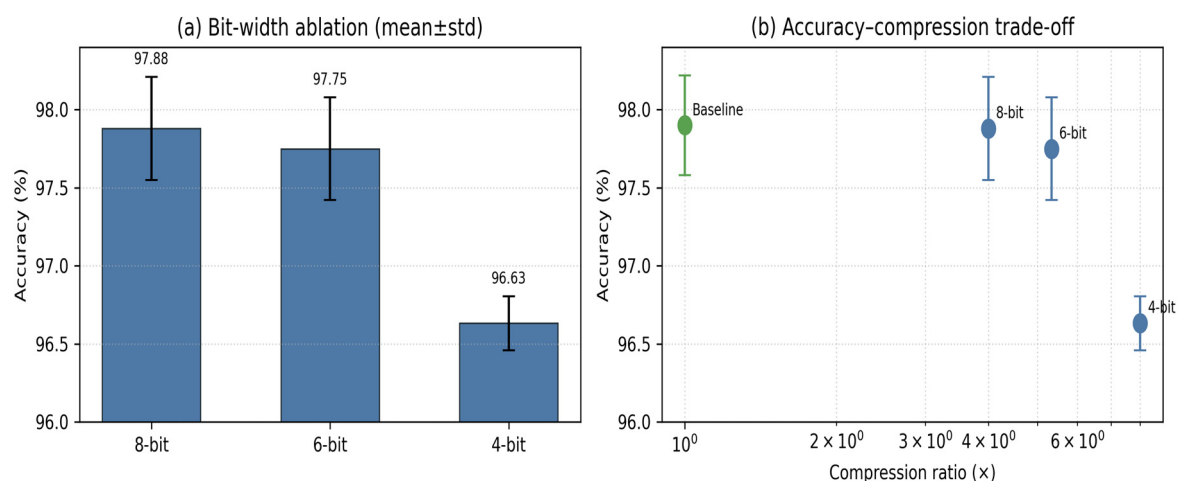
**Figure 4.** Unified benchmark quantization ablation: Bit-width and accuracy-compression trade-off (mean $\pm$ std over three independent runs).

Table 6 and Figure 4b show the nominal bit-width trade-off: 8-bit, 6-bit, and 4-bit correspond to nominal $4.00\times$, $5.33\times$, and $8.00\times$ weight-bit reductions. Table 5 reports effective factors after accounting for layers retained above the nominal weight precision.

Table 6. Unified quantization ablation results.

Configuration	Accuracy (Mean $\pm$ Std, %)
8-bit	97.88 $\pm$ 0.33
6-bit	97.75 $\pm$ 0.33
4-bit	96.63 $\pm$ 0.17

3.2. Pruning Results and Ablation Analysis

3.2.1. Main Pruning Results

Figure 5a shows that AIASP reaches $97.59 \pm 0.13\%$ at a 30% target sparsity and 43.9% realized sparsity, with a $1.78\times$ theoretical compression factor. Magnitude, gradient, and random pruning use 25% sparsity and therefore provide lower-compression comparison points. Random pruning has the largest observed mean accuracy in this table and should not be described as a lower-bound accuracy result.

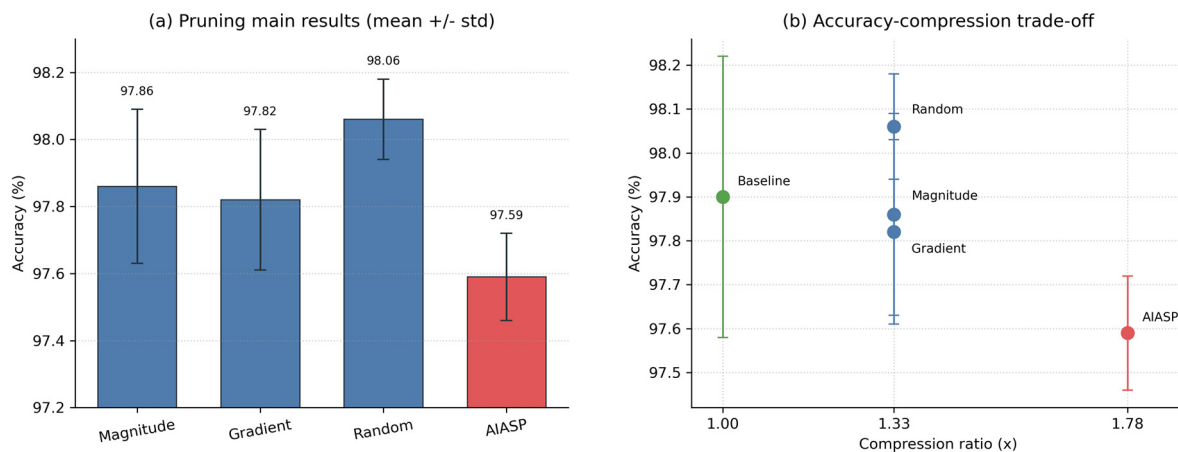


Figure 5. Main results of unified benchmark pruning: Method comparison and accuracy–compression ratio trade-off.

Table 7 separates targets from realized sparsity. AIASP trades a modest mean-accuracy reduction relative to the teacher for a higher pruning compression factor among the reported pruning methods.

Table 7. Unified pruning main results.

Method	Final Accuracy (Mean $\pm$ Std, %)	Target/Realized Sparsity	Compression Ratio ($\times$)	Model Size (MB)	Conclusion
Magnitude	97.86 ± 0.23	25%	1.33	0.288	Magnitude criterion provides a comparison result with limited retention capability
Gradient	97.82 ± 0.21	25%	1.33	0.288	Gradient criterion exhibits larger fluctuations
Random	98.06 ± 0.12	25%	1.33	0.288	Observed mean accuracy at the lower 25% sparsity setting
AIASP	97.59 ± 0.13	30%/43.9%	1.78	0.216	$1.78\times$ pruning compression factor; modest mean-accuracy reduction

3.2.2. Pruning Ablation Analysis

Figure 6a shows the AIASP sweep: the 10%, 20%, 30%, 40%, 50%, and 60% target settings reach $97.80 \pm 0.13\%$, $98.12 \pm 0.13\%$, $97.82 \pm 0.32\%$, $97.67 \pm 0.28\%$, $97.24 \pm 0.09\%$, and $97.20 \pm 0.19\%$, respectively. Figure 6b visualizes a representative stage-4 distribution: the AGP score combines the already computed Taylor importance with the SE-attention prior. It is not a forward-only replacement for Taylor scoring.

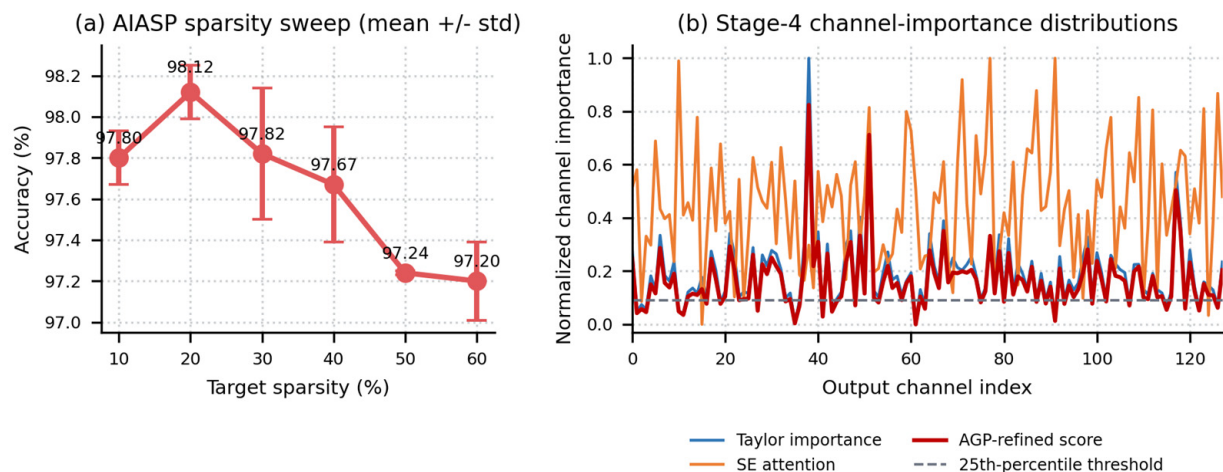


Figure 6. Pruning: (a) AIASP sparsity sweep (mean $\pm$ std over three independent runs); (b) representative stage-4 channel-importance distributions for Taylor importance, SE attention, and the AGP-refined score.

Table 8 reports the same values shown in Figure 6a. The 20% setting has the largest mean accuracy in this sweep, while the 30% setting is used in the main pruning configuration because it provides greater realized sparsity with a smaller accuracy reduction than the more aggressive settings.

Table 8. Unified pruning ablation results.

Configuration	Final Accuracy (Mean $\pm$ Std, %)
AIASP-10%	97.80 $\pm$ 0.13
AIASP-20%	98.12 $\pm$ 0.13
AIASP-30%	97.82 $\pm$ 0.32
AIASP-40%	97.67 $\pm$ 0.28
AIASP-50%	97.24 $\pm$ 0.09
AIASP-60%	97.20 $\pm$ 0.19

The sparsity sweep is not a direct ranking of the final joint pipeline: it evaluates standalone AIASP recovery. The main pruning configuration is therefore reported separately in Table 7 with its target and realized sparsity.

3.3. Distillation Results and Ablation Analysis

3.3.1. Main Distillation Results

Figure 7a shows that under the current teacher model and unified student model configurations, the Teacher achieves 97.90%, while the Student-Standalone reaches $96.53 \pm 0.39\%$. This indicates that reducing the student scale is associated with a performance gap of approximately 1.37 percentage points between the student and teacher models. Under this setting, Fixed-T KD and PMKD reach $97.16 \pm 0.31\%$ and $97.08 \pm 0.20\%$, respectively, both above the Student-Standalone. These results indicate that the distillation schemes reduce the observed teacher–student gap. Multi-Teacher KD achieves $97.06 \pm 0.05\%$, with PMKD ($97.08 \pm 0.20\%$) overlapping Fixed-T KD ($97.16 \pm 0.31\%$) within the reported variation.

Table 9 further places the teacher model, the standalone student model, and various distillation strategies under the same benchmark for comparison. In conjunction with Figure 7, it can be observed that PMKD attains $97.08 \pm 0.20\%$, with reported variation overlapping Fixed-T KD ($97.16 \pm 0.31\%$) and Multi-Teacher KD ($97.06 \pm 0.05\%$). Under the current student capacity and training budget, all three transfer schemes are above

the $96.53 \pm 0.39\%$ standalone-student result. Multi-Teacher KD reaches $97.06 \pm 0.05\%$; its difference from the simpler Fixed-T KD is small under the reported protocol and comes with higher training costs. Overall, PMKD, Fixed-T KD, and Multi-Teacher KD all yield higher mean accuracy than the standalone student, with PMKD overlapping Fixed-T KD within the reported variation.

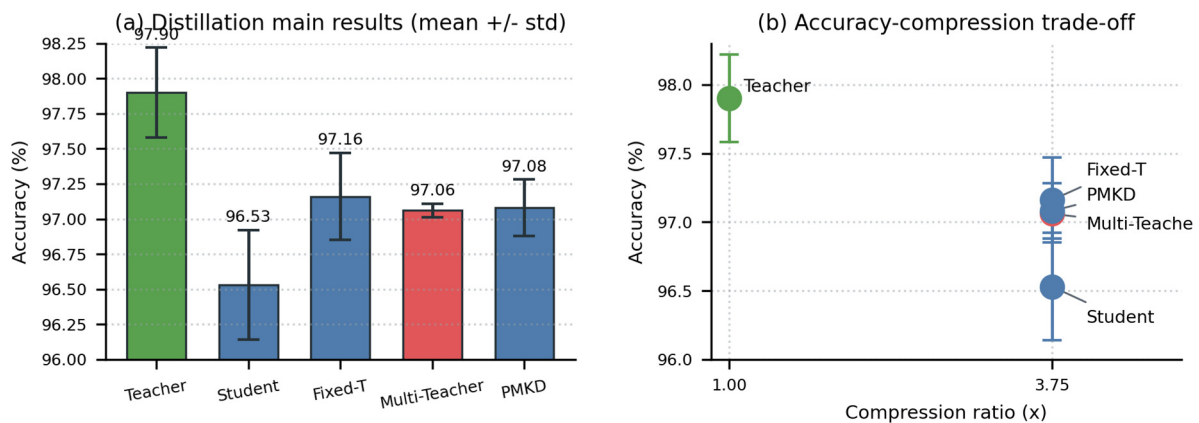


Figure 7. Unified benchmark distillation main results (mean $\pm$ std); labels are positioned to distinguish the clustered student configurations.

Table 9. Unified main distillation results (mean $\pm$ std).

Distillation Method	Accuracy (Mean $\pm$ Std, %)
Teacher (32 ch, 30 ep)	97.90 ± 0.32
Student-Standalone (16 ch, 30 ep)	96.53 ± 0.39
Fixed-T KD	97.16 ± 0.31
Multi-Teacher KD	97.06 ± 0.05
PMKD	97.08 ± 0.20

3.3.2. Distillation Ablation Analysis

Figure 8a shows that the distillation temperature has a non-monotonic association with the results. In the temperature scan, $T = 6.0$ has the largest mean accuracy of $97.39 \pm 0.08\%$, indicating that the temperature setting should be considered together with the current student model capacity, training stage, and scheduling approach rather than interpreted as “the higher the temperature, the better.” Figure 8b shows that $\alpha = 0.3$ yields the largest mean accuracy in this sweep ($97.47 \pm 0.16\%$); as α increases to 0.9, the result decreases to $96.96 \pm 0.16\%$, indicating that soft-target supervision contributes to the current distillation task.

The itemized results in Tables 9 and 10 indicate that distillation performance is sensitive to hyperparameters. In the temperature scan, the largest mean appears at $T = 6.0$ ($97.39 \pm 0.08\%$), suggesting that this setting is compatible with the current student model, training stage, and schedule. Increasing α from 0.3 to 0.9 corresponds to a gradual performance decline ($97.47\% \rightarrow 96.96\%$), indicating that the balance between distillation loss and hard-label supervision affects the reported result. Combining the mean and standard deviation, the region around $T = 6.0$ has a higher mean with a limited fluctuation range under the current training budget. Figure 8 therefore supports interpreting PMKD across the tested schedule rather than from a single hyperparameter setting.

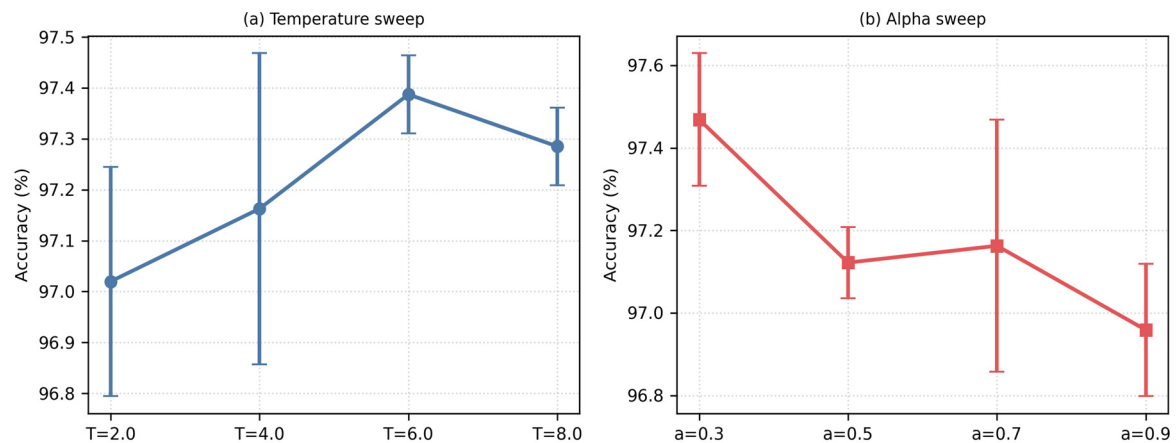


Figure 8. Unified benchmark distillation ablation (mean $\pm$ std): Temperature and distillation weight scanning.

Table 10. Unified distillation ablation results (mean $\pm$ std over three independent runs).

Configuration	Accuracy (Mean $\pm$ Std, %)
Temp T = 2.0 (alpha = 0.5)	97.02 $\pm$ 0.23
Temp T = 4.0 (alpha = 0.5)	97.16 $\pm$ 0.31
Temp T = 6.0 (alpha = 0.5)	97.39 $\pm$ 0.08
Temp T = 8.0 (alpha = 0.5)	97.29 $\pm$ 0.08
alpha = 0.3 (T = 4.0)	97.47 $\pm$ 0.16
alpha = 0.5 (T = 4.0)	97.12 $\pm$ 0.09
alpha = 0.7 (T = 4.0)	97.16 $\pm$ 0.31
alpha = 0.9 (T = 4.0)	96.96 $\pm$ 0.16

3.4. Joint Compression Results Analysis

For a transparent end-to-end comparison, Figure 9 and Table 11 use theoretical packed-weight compression relative to the FP32 teacher. The Final-Joint factor is $51.51\times$, calculated from the 16-channel student, 41.9% masked weights, and 4.01-bit average precision. This is an ideal storage estimate and excludes sparse-index overhead; it is not the FP16 TensorRT engine footprint. The corresponding trade-off score is 6.61.

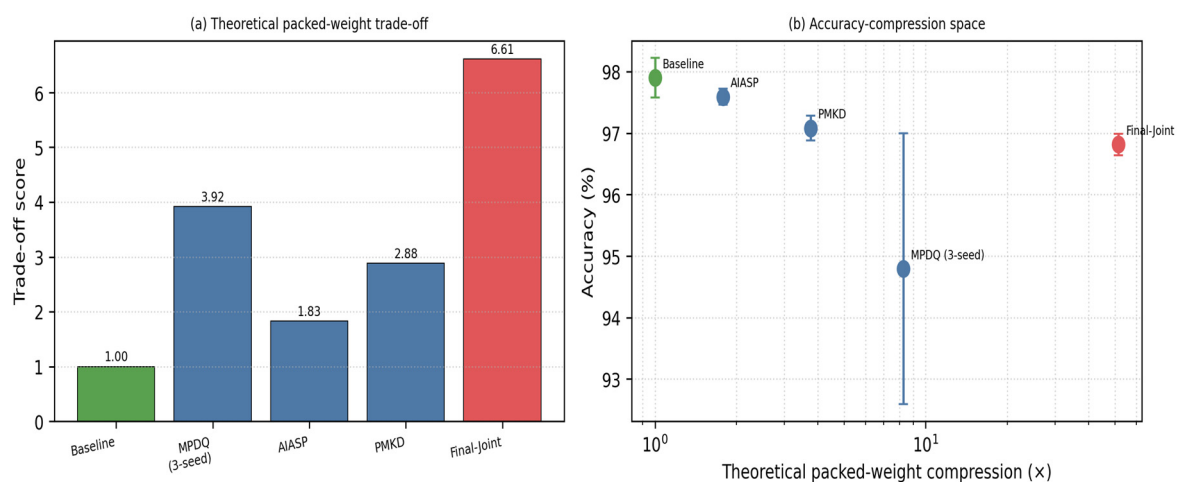


Figure 9. Theoretical packed-weight trade-off relative to the FP32 teacher; sparse-index overhead is excluded.

Table 11. End-to-end theoretical packed-weight comparison relative to the FP32 teacher. Sparse-index overhead is excluded; this table does not report FP16 TensorRT engine memory.

Scheme (Configuration)	Accuracy (Mean $\pm$ Std, %)	Theoretical Compression ($\times$)	Comprehensive Score
MPDQ	95.53 $\pm$ 1.16	8.27	3.88
AIASP (30%/43.9%)	97.59 $\pm$ 0.13	1.78	1.83
PMKD	97.08 $\pm$ 0.20	3.75	2.88
Final-Joint	96.82 $\pm$ 0.17	51.51	6.61

Table 11 reports mean values with standard deviations for all rows. The final row reports the end-to-end theoretical packed-weight factor relative to the FP32 teacher; its storage interpretation is explicitly separated from the FP16 engine benchmark.

3.5. Ablation on Compression Ordering

To examine ordering effects under a common evaluation protocol, Table 12 reports $P \rightarrow Q \rightarrow D$, $Q \rightarrow P \rightarrow D$, and $P \rightarrow D \rightarrow Q$. $P \rightarrow Q \rightarrow D$ and $Q \rightarrow P \rightarrow D$ have overlapping reported variation, whereas applying quantization after distillation ($P \rightarrow D \rightarrow Q$) is associated with markedly lower accuracy. Exploratory permutations are omitted from the main claim because they are not comparable to the main rows.

Table 12. Compression-order ablation under the common evaluation protocol. The reported factor is the stage-local pruning-and-quantization factor within the 16-channel student and is not the teacher-relative packed-weight factor in Table 11.

Ordering	Accuracy (Mean $\pm$ Std, %)	Stage-Local Compression ($\times$)
$P \rightarrow Q \rightarrow D$ (ours)	96.82 $\pm$ 0.17	13.74
$Q \rightarrow P \rightarrow D$	96.90 $\pm$ 0.32	13.74
$P \rightarrow D \rightarrow Q$	68.05 $\pm$ 16.24	13.74

3.6. Component Ablation of the Joint Pipeline

Table 13 summarizes the contribution of each component when one operator is removed from the full $P \rightarrow Q \rightarrow D$ pipeline. Removing distillation is associated with the largest observed degradation. The final model reaches 96.82 $\pm$ 0.17%, which is 1.08 percentage points below the 97.90% teacher baseline; it should not be described as baseline-level accuracy.

Table 13. Component ablation of the joint compression pipeline (mean $\pm$ standard deviation over three independent runs). Factors are stage-local pruning-and-quantization factors within the 16-channel student, not teacher-relative packed-weight factors.

Configuration	Accuracy (Mean $\pm$ Std, %)	Stage-Local Compression ($\times$)
Full pipeline (ours)	96.82 $\pm$ 0.17	13.74
w/o Distillation	83.28 $\pm$ 10.69	13.74
w/o Quantization	96.94 $\pm$ 0.23	1.72
w/o Pruning	97.65 $\pm$ 0.28	7.98
w/o HW-cost term	96.94 $\pm$ 0.28	11.68

3.7. Comparison with Selected KWS Models

To position LiteKWS-Net within the keyword-spotting literature, Table 14 compares representative architectures by their reported task settings, parameter counts, and input

features. The accuracy figures come from different protocols and are not used as a direct ranking. Final-Joint is shown with both dense parameter count and theoretical packed storage; the latter is not an ONNX/TensorRT FP16 engine size.

Table 14. Representative KWS models. Accuracy values are taken from different task protocols and are not directly comparable. BC-ResNet and KWT rows are reported from the original model papers [24,25]. Final-Joint storage is an ideal packed estimate, not an FP16 engine footprint.

Model	Accuracy (%)	Parameters/Storage	Input/Protocol
MatchboxNet ($3 \times 1 \times 64$)	97.2	93 K	MFCC
BC-ResNet-1	96.9	9.2 K	log-Mel
BC-ResNet-3	98.2	54.2 K	log-Mel
BC-ResNet-8	98.7	321 K	log-Mel
KWT-1 (Transformer)	97.7	559 K	MFCC
KWT-3 (Transformer)	98.5	5366 K	MFCC
LiteKWS-Net (ours, teacher)	97.90 ± 0.32	100.8 K	log-Mel (8-keyword protocol)
Final-Joint (ours, compressed)	96.82 ± 0.17 ; $51.51 \times$ ideal	26.9 K dense; 15.6 K active *	log-Mel (8-keyword protocol)

Note: * Active parameter-equivalents denote the weights structurally retained after pruning; they are not a dense parameter count or an FP16 TensorRT engine footprint.

LiteKWS-Net has 100.8 K dense teacher parameters, while the Final-Joint export retains 26.9 K dense parameters. Under ideal sparse mixed-precision packing, the final configuration has approximately 15.6 K active parameter-equivalents and about 7.8 kB of weight values before sparse-index overhead. Its 96.82% accuracy is 1.08 percentage points below the teacher on the 8-keyword task. Direct accuracy comparisons with the literature rows are not warranted because their protocols differ.

3.8. On-Device Hardware Benchmarking

To characterize deployment behavior on edge hardware, we benchmarked the exported network bodies on an NVIDIA Jetson Nano Developer Kit using TensorRT 8.2.1 with FP16 precision. The log-Mel front end is excluded from the timed engine; reported latency is therefore network-body latency, not end-to-end KWS-system latency. Each engine was benchmarked after warm-up with timed TensorRT iterations. Dynamic power was measured from the on-board INA3221 sensor; energy is dynamic power multiplied by the reported network-body latency. The test evaluates FP16 engine execution, not native mixed-bit inference kernels.

As shown in Table 15, the Final-Joint FP16 engine reaches 2.86 ms network-body latency, while the uncompressed LiteKWS-Net engine is 2.85 ms. The compression result therefore does not show a single-inference latency reduction over the teacher; the measured comparison shows comparable network-body latency against the larger BC-ResNet-8 and KWT-3 exports. Table 16 reports energy and FP16 engine weight footprints. The theoretical packed mixed-precision storage estimate is reported separately and must not be interpreted as an observed TensorRT FP16 memory read.

3.9. Additional Sensitivity and Ordering Checks

Channel-width sensitivity was evaluated under a fixed 30-epoch protocol. For the teacher topology, base widths of 8, 16, 24, 32, and 48 produced 95.47%, 96.39%, 97.73%, 97.73%, and 97.43% accuracy with 7681, 26,861, 57,949, 100,813, and 222,063 parameters, respectively. For the student topology, widths of 8, 16, and 24 produced 95.47%, 96.39%, and 97.73% accuracy with 7681, 26,861, and 57,949 parameters. The 32-channel teacher is therefore retained as a wider reference configuration, while the 16-channel student is used as the compact student configuration; these settings are not presented as globally optimal.

Table 15. Jetson Nano TensorRT FP16 network-body benchmark (mean over three engine runs). Throughput is TensorRT-reported throughput and is not the reciprocal of host-side end-to-end latency.

Model	#Params	Latency Mean (ms)	Median (ms)	P99 (ms)	Throughput (qps)
BC-ResNet-1	9232	6.05	6.00	6.74	165
BC-ResNet-3	54,168	7.20	7.14	8.15	139
BC-ResNet-8	321,068	7.95	7.75	9.20	126
KWT-1	559,011	4.43	4.41	5.06	226
KWT-3	5,365,667	17.67	16.40	41.00	51.9
LiteKWS-Net (ours)	100,813	2.85	2.81	3.21	347
Final-Joint (ours)	26,861	2.86	2.82	3.41	313

Table 16. Dynamic energy and FP16 TensorRT engine weight footprint. The engine footprint is distinct from the ideal packed sparse mixed-precision estimate.

Model	#Params	Dynamic Power (W)	Energy/Inference (mJ)	FP16 Engine Weight Footprint	vs. Final-Joint (Energy)
BC-ResNet-1	9232	0.699	4.23	18.5 KB	6.1×
BC-ResNet-3	54,168	0.498	3.59	108.3 KB	5.2×
BC-ResNet-8	321,068	1.643	13.06	642.1 KB	18.9×
KWT-3	5,365,667	1.07	18.90	10.73 MB	27.4×
Final-Joint (ours)	26,861	0.242	0.69	53.7 kB	baseline

For the Fisher-binning sensitivity check, all strategies used a shared 40.5% pruning sparsity. Fisher bins yielded 97.24% accuracy at 3.92 average bits and a 13.74× theoretical compression factor. Uniform-quantile allocation yielded 96.82% at 5.71 bits and 9.43×, uniform 4-bit yielded 97.24% at 4.50 bits and 11.96×, and uniform 6-bit yielded 97.43% at 6.25 bits and 8.61×. Thus, the Fisher-bin setting does not maximize accuracy alone, but it provides the largest compression factor among these tested settings at the same reported 97.24% accuracy as the uniform 4-bit setting.

A supplementary six-order check used a common 42.9% sparsity, 4.17-bit average precision, and 13.45× theoretical compression factor. The resulting accuracies for $P \rightarrow Q \rightarrow D$, $Q \rightarrow P \rightarrow D$, $P \rightarrow D \rightarrow Q$, $D \rightarrow P \rightarrow Q$, $D \rightarrow Q \rightarrow P$, and $Q \rightarrow D \rightarrow P$ were 96.94%, 97.06%, 81.81%, 88.49%, 68.77%, and 96.57%, respectively. This single-run check complements the three-run results in Table 12 and supports the observation that quantization after distillation is associated with lower accuracy. The two protocols are reported separately and are not pooled into one aggregate comparison.

4. Conclusions

This study investigated a collaborative multi-compression acceleration mechanism for deploying neural network models in keyword spotting tasks on resource-constrained platforms and established a unified optimization framework consisting of mixed-precision dynamic quantization, importance-aware structured pruning, and progressive multi-stage knowledge distillation. To address the difficulty faced by conventional single compression methods in balancing compression ratio, recognition accuracy, and hardware friendliness, coordinated design was carried out from three perspectives: parameter-representation compression, network-structure pruning, and knowledge-transfer compensation. In the final stage, unified data partitions, unified training budgets, and repeated-run statistics were introduced to re-examine the actual performance differences among compression branches.

The unified evaluation results show that the teacher reaches 97.90% mean accuracy. MPDQ reaches $95.53 \pm 1.16\%$, with an $8.27\times$ theoretical weight-compression factor. AIASP reaches $97.59 \pm 0.13\%$ at a 30% target and 43.9% realized sparsity. PMKD reaches $97.08 \pm 0.20\%$. Final-Joint reaches $96.82 \pm 0.17\%$ and has a $51.51\times$ ideal packed-weight factor relative to the FP32 teacher. The Jetson Nano result is a TensorRT FP16 network-body benchmark; it should be interpreted as deployment evidence for the exported FP16 engine rather than as direct mixed-bit kernel acceleration.

Overall, the collaborative multi-compression acceleration mechanism proposed in this work provides a feasible path for the lightweight design and efficient deployment of keyword-spotting neural networks, and may serve as a useful reference for real-time speech processing on edge intelligent hardware.

Several limitations should be noted. First, not every exploratory sub-study has the same replication level as the main results. Second, the hardware benchmark is limited to one platform and FP16 TensorRT execution; it does not measure a native mixed-bit sparse kernel. Third, the compression figures use ideal packed-weight estimates and exclude sparse-index overhead. Fourth, the task is an 8-keyword clean-speech setting, and protocol-matched comparisons with the literature models remain future work.

Future work will proceed along four directions: (i) cross-platform on-device benchmarking on target MCU/FPGA boards, reporting measured latency, energy, and peak memory; (ii) a larger repeated-run and 35-keyword evaluation to assess whether the observed ordering and component trends generalize; (iii) noise-robustness and cross-language studies, including agglutinative-language tasks, with re-calibrated pruning and quantization under acoustic shift; and (iv) extending AGP to attention-free or non-SE backbones by injecting a lightweight auxiliary attention module, to broaden the applicability of architecture-derived pruning priors.

Author Contributions: Conceptualization, J.J., R.P., J.L., and M.Z.; methodology, J.J., R.P., and J.L.; software, J.J., R.P., and J.L.; validation, J.J., R.P., and J.L.; formal analysis, J.J., R.P., and J.L.; investigation, J.J., R.P., and J.L.; resources, J.J., R.P., and J.L.; data curation, J.J., R.P., and J.L.; writing-original draft preparation, J.J., R.P., and J.L.; writing-review and editing, M.Z.; supervision, M.Z.; funding acquisition, M.Z. All authors have read and agreed to the published version of the manuscript.

Funding: PhD Research Foundation Project of Hubei University of Technology, Research on Sun Tracking Control and System State Monitoring of Butterfly Concentrating Photovoltaic Based on Deep Learning Image Analysis (No. 00185); Guiding project of Scientific Research Plan of Hubei Provincial Department of Education, Design of Application-Specific Integrated Circuit for Voice Command Recognition Based on Compute-in-Memory Architecture (No. B2020046); and Cooperation Agreement on the Project of Design and Process Manufacturing Technology of 3D Stacked Code Flash Memory Chip (No. 2021802).

Data Availability Statement: The Speech Commands Dataset v2 is open to the public and can be obtained from the TensorFlow official repository at https://storage.googleapis.com/download.tensorflow.org/data/speech_commands_v0.02.tar.gz, released by Google's TensorFlow and AIY teams for keyword spotting research (accessed on 8 August 2026).

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT (GPT-5.6-terra) only for Chinese-to-English translation and grammatical polishing. All scientific content—including methodology, experimental design, data analysis, and conclusions—was entirely authored and verified by the authors. The AI-assisted output was carefully reviewed, edited, and approved by all authors, who take full responsibility for the accuracy and integrity of the published content.

Conflicts of Interest: The authors declare that this study received funding from [Wuhan Xinxin Semiconductor Manufacturing Co., Ltd.]. The funder was not involved in the study design, collection,

analysis, interpretation of data, the writing of this article, or the decision to submit it for publication. The authors declare no conflicts of interest.

References

1. Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv* **2018**, arXiv:1804.03209. [CrossRef]
2. Majumdar, S.; Ginsburg, B. MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition. *arXiv* **2020**, arXiv:2004.08531. [CrossRef]
3. Baevski, A.; Zhou, H.; Mohamed, A.; Auli, M. Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *arXiv* **2020**, arXiv:2006.11477. [CrossRef]
4. Hsu, W.N.; Bolte, B.; Tsai, Y.H.H.; Lakhota, K.; Salakhutdinov, R.; Mohamed, A. HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *arXiv* **2021**, arXiv:2106.07447. [CrossRef]
5. Kim, D.H.; Lee, J.H.; Mo, J.H.; Chang, J.-H. W2V2-Light: A Lightweight Version of Wav2vec 2.0 for Automatic Speech Recognition. In Proceedings of the Interspeech 2022, Incheon, Republic of Korea, 18–22 September 2022; pp. 3038–3042. [CrossRef]
6. Qin, H.; Ma, X.; Ding, Y.; Li, X.; Zhang, Y.; Tian, Y.; Ma, Z.; Luo, J.; Liu, X. BiFSMN: Binary Neural Network for Keyword Spotting. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, Vienna, Austria, 23–29 July 2022; pp. 4346–4352. [CrossRef] [PubMed]
7. Wang, X.; Cheng, S.; Li, J.; Qiao, S.; Zhou, Y.; Zhan, Y. Low-complex and Highly-performed Binary Residual Neural Network for Small-footprint Keyword Spotting. In Proceedings of the Interspeech 2022, Incheon, Republic of Korea, 18–22 September 2022; pp. 3233–3237. [CrossRef]
8. Hard, A.; Partridge, K.; Chen, N.; Augenstein, S.; Shah, A.; Park, H.J.; Park, A.; Ng, S.; Nguyen, J.; Lopez-Moreno, I.; et al. Production Federated Keyword Spotting via Distillation, Filtering, and Joint Federated-Centralized Training. In Proceedings of the Interspeech 2022, Incheon, Republic of Korea, 18–22 September 2022; pp. 76–80. [CrossRef]
9. Song, Z.; Liu, Q.; Yang, Q.; Li, H. Knowledge Distillation for In-Memory Keyword Spotting Model. In Proceedings of the Interspeech 2022, Incheon, Republic of Korea, 18–22 September 2022; pp. 4128–4132. [CrossRef]
10. He, K.; Chen, D.; Su, T. A Configurable Accelerator for Keyword Spotting Based on Small-Footprint Temporal Efficient Neural Network. *Electronics* **2022**, *11*, 2571. [CrossRef]
11. Yang, G.P.; Gu, Y.; Tang, Q.; Du, D.; Liu, Y. On-Device Constrained Self-Supervised Speech Representation Learning for Keyword Spotting via Knowledge Distillation. In Proceedings of the Interspeech 2023, Dublin, Ireland, 20–24 August 2023. Available online: https://www.isca-archive.org/interspeech_2023/yang23y_interspeech.html (accessed on 8 August 2026).
12. Peng, Y.; Sudo, Y.; Muhammad, S.; Watanabe, S. DPHuBERT: Joint Distillation and Pruning of Self-Supervised Speech Models. In Proceedings of the Interspeech 2023, Dublin, Ireland, 20–24 August 2023; pp. 62–66. Available online: https://www.isca-archive.org/interspeech_2023/peng23c_interspeech.html (accessed on 8 August 2026).
13. Bekal, D.; Gopalakrishnan, K.; Mundnich, K.; Ronanki, S.; Bodapati, S.; Kirchhoff, K. A Metric-Driven Approach to Conformer Layer Pruning for Efficient ASR Inference. In Proceedings of the Interspeech 2023, Dublin, Ireland, 20–24 August 2023; pp. 4079–4083. [CrossRef]
14. Jiang, H.; Zhang, L.L.; Li, Y.; Wu, Y.; Cao, S.; Cao, T.; Yang, Y.; Li, J.; Yang, M.; Qiu, L. Accurate and Structured Pruning for Efficient Automatic Speech Recognition. In Proceedings of the Interspeech 2023, Dublin, Ireland, 20–24 August 2023. Available online: https://www.isca-archive.org/interspeech_2023/jiang23d_interspeech.html (accessed on 8 August 2026).
15. Wang, H.; Du, J.; Zhou, H.; Lee, C.-H.; Ren, Y.; Zhao, J. A Multiple-Teacher Pruning Based Self-Distillation (MT-PSD) Approach to Model Compression for Audio-Visual Wake Word Spotting. In Proceedings of the Interspeech 2023, Dublin, Ireland, 20–24 August 2023; pp. 2678–2682. [CrossRef]
16. Yoon, J.; Kim, N.; Lee, D.; Lee, S.-J.; Kwak, G.-H.; Kim, T.-H. A Resource-Efficient Keyword Spotting System Based on a One-Dimensional Binary Convolutional Neural Network. *Electronics* **2023**, *12*, 3964. [CrossRef]
17. Bae, S.; Kim, H.; Lee, S.; Jung, Y. FPGA Implementation of Keyword Spotting System Using Depthwise Separable Binarized and Ternarized Neural Networks. *Sensors* **2023**, *23*, 5701. [CrossRef] [PubMed]
18. Ben Letaifa, L.; Rouas, J.L. Variable Scale Pruning for Transformer Model Compression in End-to-End Speech Recognition. *Algorithms* **2023**, *16*, 398. [CrossRef]
19. Park, E.; Ahn, D.; Kim, H. RepTor: Re-parameterizable Temporal Convolution for Keyword Spotting via Differentiable Kernel Search. In Proceedings of the Interspeech 2024, Kos, Greece, 20–24 August 2023; pp. 4518–4522. [CrossRef]
20. Gu, T.; Liu, B.; Shao, H.; Qian, Y. SparseWAV: Fast and Accurate One-Shot Unstructured Pruning for Large Speech Foundation Models. In Proceedings of the Interspeech 2024, Kos, Greece, 1–5 September 2024; pp. 4498–4502. [CrossRef]
21. Li, Z.; Xu, H.; Wang, T.; Hu, S.; Jin, Z.; Hu, S.; Deng, J.; Cui, M.; Geng, M.; Liu, X. One-pass Multiple Conformer and Foundation Speech Systems Compression and Quantization Using an All-in-one Neural Model. In Proceedings of the Interspeech 2024, Kos, Greece, 1–5 September 2024; pp. 4503–4507. [CrossRef]

22. Gu, T.; Liu, B.; Wang, H.; Qian, Y. Ultra-Low Bit Post-Training Quantization of Large Speech Models via K-Means Clustering and Mixed Precision Allocation. In Proceedings of the Interspeech 2025, Rotterdam, The Netherlands, 17–21 August 2025; pp. 1988–1992. [[CrossRef](#)]
23. Li, Z.; Xu, H.; Jin, Z.; Meng, L.; Wang, T.; Wang, H.; Chen, Y.; Cui, M.; Hu, S.; Liu, X. Towards One-bit ASR: Extremely Low-bit Conformer Quantization Using Co-training and Stochastic Precision. In Proceedings of the Interspeech 2025, Rotterdam, The Netherlands, 17–21 August 2025; pp. 1973–1977. [[CrossRef](#)]
24. Kim, B.; Chang, S.; Lee, J.; Sung, D. Broadcasted Residual Learning for Efficient Keyword Spotting. In Proceedings of the Interspeech 2021, Brno, Czech Republic, 30 August–3 September 2021; pp. 4538–4542. [[CrossRef](#)]
25. Berg, A.; O’connor, M.; Cruz, M.T. Keyword Transformer: A Self-Attention Model for Keyword Spotting. In Proceedings of the Interspeech 2021, Brno, Czech Republic, 30 August–3 September 2021; pp. 4249–4253. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.