

Article

SSKD: Stepwise Self-Knowledge Distillation for Binary Neural Networks in Keyword Spotting

Hailong Zou ^{1,2}, Jionghao Zhang ^{1,2}, Jun Li ^{1,2}, Hang Ran ¹, Wulve Yang ^{1,2}, Rui Zhou ^{1,2}, Zenghui Yu ¹,
Yi Zhan ^{1,2,*} and Shushan Qiao ^{1,2,*}

¹ Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029, China; zouhailong24@ime.ac.cn (H.Z.); zhangjionghao@ime.ac.cn (J.Z.); lijun24@ime.ac.cn (J.L.); ranhang@ime.ac.cn (H.R.); yangwulve24@ime.ac.cn (W.Y.); zhourui25@ime.ac.cn (R.Z.); yuzenghui@ime.ac.cn (Z.Y.)

² University of Chinese Academy of Sciences, Beijing 101408, China

* Correspondence: yizhan@ime.ac.cn (Y.Z.); qiaoshushan@ime.ac.cn (S.Q.)

Abstract

The hardware power-aware keyword spotting (KWS) implementation requires small memory footprint, low-complex computation, and high accuracy performances. Binary neural networks (BNNs) naturally satisfy these constraints. They quantize both weights and activations to 1-bit. This reduces storage and replaces most multiply–accumulate operations with bitwise operations. However, such extreme quantization incurs substantial information loss and leaves a noticeable accuracy gap relative to full-precision models. Optimization is also more difficult because the sign function is non-differentiable, and surrogate-gradient updates introduce gradient mismatch. To preserve the hardware benefits of BNNs while alleviating the accuracy degeneration induced by 1-bit quantization, this article addresses the problem from two complementary aspects: Firstly, a Stepwise Self-Knowledge Distillation (SSKD) training approach is proposed to effectively improve the student BNN’s accuracy performance. The SSKD training framework achieves effective supervision for student BNNs. A Stepwise Training Strategy is proposed to optimize the training stability and accuracy. Weight Scaling Factor improves the student’s representational capability. Secondly, an extremely lightweight Binary Temporal Convolutional ResNet (BTC-ResNet) is also proposed. The parameters and calculations inside the network are greatly reduced for the inference. Experiments on the GSCD v1 and GSCD v2 benchmarks demonstrate the effectiveness of our methods for low-power keyword spotting. For the 12-class task, BTC-ResNet14 achieves 97.23% accuracy on GSCD v1 and 97.31% on GSCD v2 with 0.75 Mb parameters and 1.35 M FLOPs. For the 35-class task on GSCD v2, it reaches 95.56% accuracy with 0.76 Mb parameters and 1.35 M FLOPs. These results indicate that our method achieves a competitive accuracy–efficiency balance relative to recent distillation-based BNN KWS baselines reported in the comparative experiments. All these studies are helpful and promising for future KWS deployment on low-power hardware devices.

Keywords: keyword spotting; binary neural network; self-knowledge distillation; binary temporal convolutional ResNet; low-power design



Received: 20 January 2026

Revised: 13 February 2026

Accepted: 16 February 2026

Published: 18 February 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 \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Driven by the demand for always-on voice interfaces, keyword spotting (KWS) plays a vital “trigger” role for low-power edge devices, such as AIoT and wearables, etc. [1–6]. For In these power-aware hardware platforms, a critical challenge is to balance their accuracy

with model size and computational cost [7–16]. The BNN (Binary Neural Network) has only 1-bit weights/activations and performs simple bitwise operations instead of expensive floating-point multiplications. Therefore, it obviously downscales the model’s parameter size and computational cost. This advantage makes it naturally suitable to realize the KWS function inside edge terminals. However, binarization inevitably results in a significant information loss, and further deteriorates the accuracy. Improving the accuracy without inference increments is thereby essential for its deployment on the resource-constrained devices [14].

Among strategies to improve the accuracy of lightweight models, knowledge distillation (KD) is a powerful method [2–4,13,17–20]. Based on this approach, self-knowledge distillation (SKD) further simplifies the process by learning from itself instead of requiring a pre-trained teacher model in the KD. Nevertheless, the SKD approach for BNN network suffers from inaccurate predictions during early training, which deteriorates final accuracy performance [17]. Further, regarding the KD method being applied to KWS jobs, only a few previous studies have touched this topic [2,3]. Therefore, in this article, our study solves the previously unexplored challenging problem of using the SKD method for BNNs applied in KWS tasks.

Figure 1 illustrates the architecture of our BNN-based KWS system. A Stepwise Self-Knowledge Distillation (SSKD) training approach for the KWS function is proposed with four innovations.

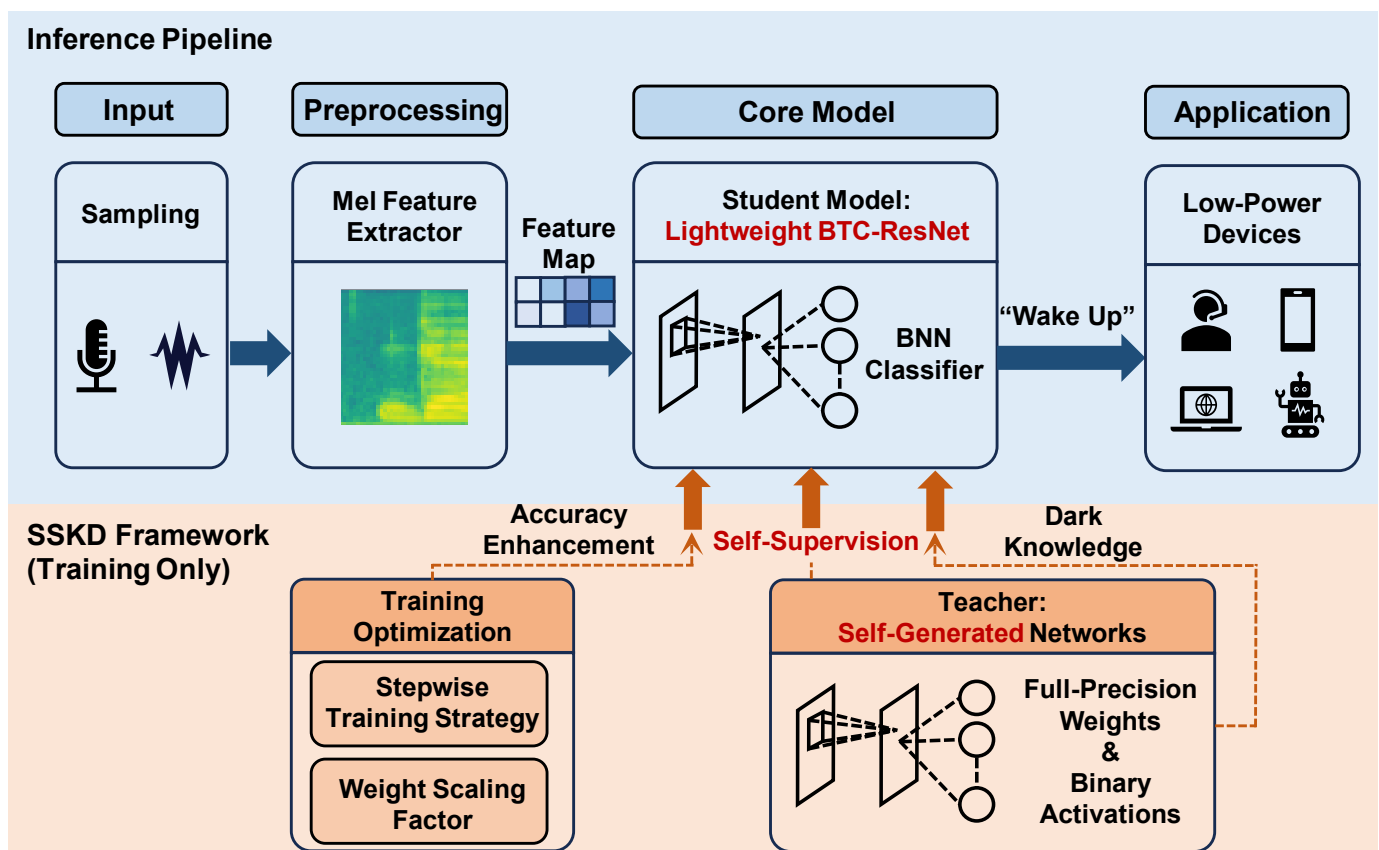


Figure 1. The overall architecture of the BNN-based KWS system with our proposed Stepwise Self-Knowledge Distillation (SSKD) training approach and lightweight BTC-ResNet.

- Inspired by the previous studies [21,22], a new SSKD training framework with an improved self-generated teacher branch is proposed. This differs from employing a deep teacher network to guide its shallow student [21]. Our teacher adopts full-

precision weights and binary activations from the student. Through this teacher's self-supervision, the knowledge transfer suffers no information loss.

- A two-step Stepwise Training Strategy is also devised to optimize the soft label bank and enhance the BNN stability, respectively. Step1's coarse-grained training helps the student learn high quality dark knowledge. Step2's fine-grained tuning benefits a stable clustering around the “−1, +1”.
- A Weight Scaling Factor is used to increase the network's characterization capability, and further improve the accuracy consequently.
- BTC-ResNet, a binary-friendly TC-ResNet [16], is ameliorated to realize an extremely low parameter amount and computations. With these advantages, our BTC-ResNet can not only achieve a satisfying classification accuracy, but also greatly lessen the parameter and computation scales.

2. Related Work

2.1. Evolution of Keyword Spotting Models

Keyword spotting (KWS) has become a core component of always-on voice interfaces. Practical deployment requires a small memory footprint, low computational complexity, and low latency under strict power budgets. Figure 2 summarizes a deployment-driven trajectory in KWS backbones: from small-footprint DNNs to CNNs, through residual and dilated architectures, and culminating in temporal-convolution designs, under tight memory, computation, latency, and power budgets. Early small-footprint KWS systems used deep neural networks (DNNs) to map compact acoustic features to keyword class probabilities [5,6,23]. The results confirmed that compact neural classifiers can deliver competitive accuracy under tight resource constraints. Subsequent work showed that Convolutional Neural Networks (CNNs) can further improve accuracy with fewer parameters by exploiting local time–frequency structures, making CNN-based backbones a mainstream choice for embedded KWS [23]. To improve accuracy–efficiency trade-offs, later studies explored architectural refinements such as residual learning, dilation, and temporal modeling. Residual connections enable deeper trainable networks for KWS, while dilated convolutions expand the temporal receptive field without proportional increases in computation [5,16,24].

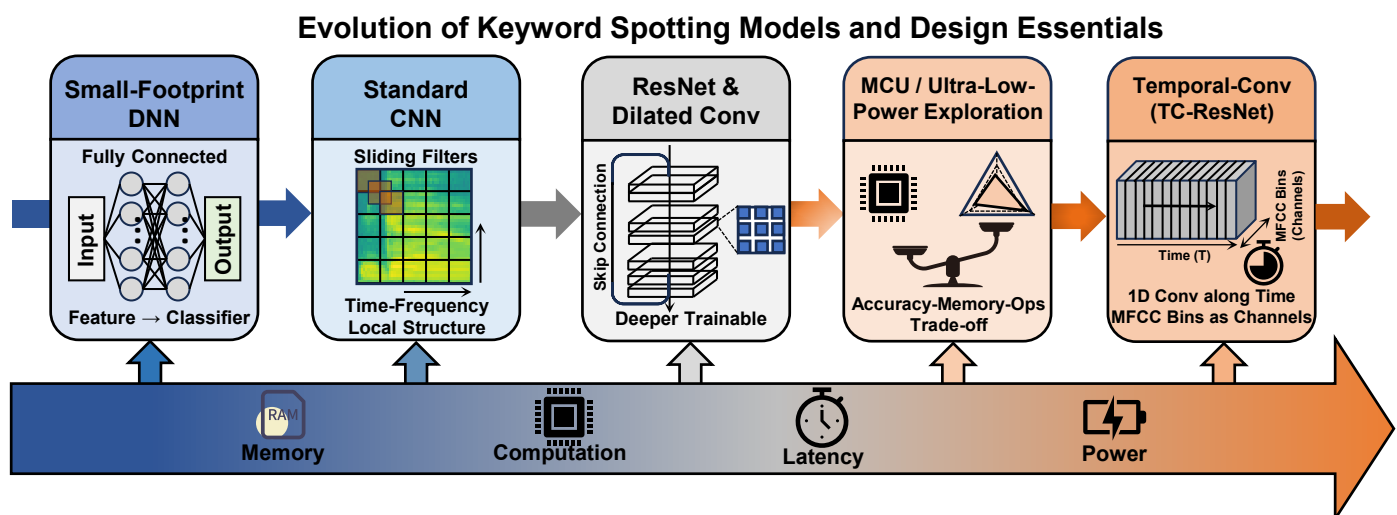


Figure 2. Evolution of keyword spotting (KWS) models and deployment-driven design essentials. The paradigm progresses from small-footprint DNN classifiers to 2D CNN backbones, then to residual and dilated-convolution architectures for improved accuracy–efficiency trade-offs, followed by MCU/ultra-low-power exploration under tight resource budgets, and finally to temporal-convolution designs that apply 1D convolutions along time while treating MFCC bins as channels. The bottom arrow highlights key on-device constraints: memory, computation, latency, and power.

For ultra-low-power devices, systematic architecture exploration and deployment-oriented evaluation have further shaped KWS backbones, highlighting the importance of co-design between network structures and hardware constraints [15,25]. More recently, temporal-convolutional designs have been proposed to reduce latency by applying 1D convolutions along time and treating spectral bins as channels, achieving competitive recognition accuracy with practical speedups on mobile hardware [16].

2.2. Binary Neural Network-Based Keyword Spotting

BNN-based keyword spotting aims to retain the always-on efficiency of small-footprint KWS while further reducing computation and memory via 1-bit weights and activations. In practice, direct binarization often leads to a noticeable accuracy drop and unstable optimization, especially when low-level acoustic cues are heavily quantized. As a result, prior work typically combines binary-friendly architectures with dedicated training techniques to narrow the accuracy gap [2,3,26–29].

Early KWS-oriented BNN designs typically start from compact ResNet-style backbones. Then binary-friendly shortcut pathways and tailored gradient treatments are introduced to stabilize optimization [5,14,29,30]. For instance, B-ResNet applies binary quantization to downscale parameters and operations, and further improves training by optimizing activation distributions and mitigating gradient mismatch during back-propagation [14]. These ResNet-style BNNs establish practical baselines for KWS under strict resource budgets, but a visible accuracy gap to full-precision models still remains.

More recent work shows that achieving near-real-network performance usually requires KWS-specific binary architectures and binarization-aware distillation schemes. BiFSMN proposes a highly efficient binary FSMN-based model, together with binarization-aware distillation and an optimized bitwise kernel for ARM deployment, enabling substantial on-device speedups while maintaining competitive accuracy [3]. BiFSMNV2 further advances this direction by strengthening the end-to-end 1-bit architecture and adopting frequency-independent distillation to better align full-precision and binary representations [2]. It reports accuracy close to full-precision KWS models on the Google Speech Commands Dataset (GSCD) v1 and v2 [2]. Beyond network binarization, recent efforts also explore fully BNN-based KWS pipelines that remove the precision mismatch between input features and binary neural networks [31]. By combining input features with a BNN-based KWS model, these systems eliminate high-precision multiplications during inference and target extreme energy-efficient deployment [31–33].

Overall, the above progress indicates that BNN-based KWS has evolved from straightforward binarization of compact CNNs to KWS-tailored binary architectures and training strategies, and has further progressed toward end-to-end fully binary implementations.

2.3. Knowledge Distillation for Keyword Spotting

An overview of typical KD paradigms for keyword spotting is summarized in Figure 3. Knowledge distillation (KD) is widely used to compress a high-capacity teacher into a compact student by transferring predictive distributions and intermediate representations [34,35]. A common formulation of KD is logit-based distillation. It trains the student to match the teacher's soft targets (the temperature-scaled softmax distribution over classes). A higher temperature ($T > 1$) produces a softer distribution. Let $\mathbf{z}^t$ and $\mathbf{z}^s$ denote the teacher and student logits, respectively, and let z_i be the i -th element of $\mathbf{z}$.

$$p_i(\mathbf{z}, T) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}. \quad (1)$$

KD trains the student to match the teacher's softened class-probability distribution by minimizing the Kullback–Leibler (KL) divergence between their temperature-scaled softmax outputs.

$$\mathcal{L}_{\text{KD}} = D_{\text{KL}}(p(\mathbf{z}^t, T) \parallel p(\mathbf{z}^s, T)) = \sum_i p_i(\mathbf{z}^t, T) \log \frac{p_i(\mathbf{z}^t, T)}{p_i(\mathbf{z}^s, T)}. \quad (2)$$

The student is optimized with a weighted sum of the hard-label cross-entropy loss and a logit-based distillation loss, controlled by a mixing coefficient α and temperature T :

$$\mathcal{L} = (1 - \alpha) \mathcal{L}_{\text{CE}}(\mathbf{y}, p(\mathbf{z}^s, 1)) + \alpha T^2 \mathcal{L}_{\text{KD}}. \quad (3)$$

where $\alpha \in [0, 1]$ balances the two terms, $p(\mathbf{z}, \tau) = \text{softmax}(\mathbf{z}/\tau)$, and $\mathcal{L}_{\text{KD}} = \text{KL}(p(\mathbf{z}^t, T) \parallel p(\mathbf{z}^s, T))$.

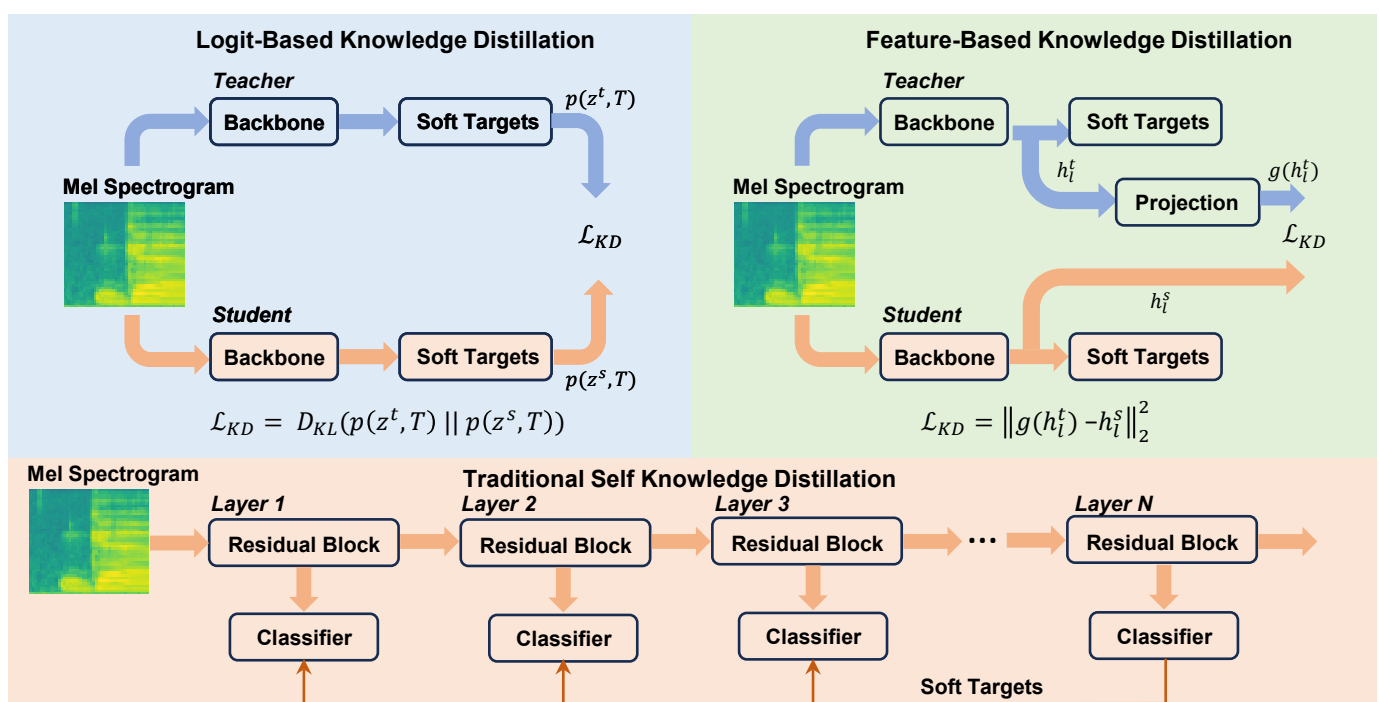


Figure 3. Overview of knowledge distillation paradigms for keyword spotting (KWS). Blue: logit-based distillation with temperature-scaled soft targets. Green: feature-based distillation aligning intermediate representations via a projection $g(\cdot)$. Orange: teacher-free self-distillation, where deeper blocks supervise shallower blocks.

Beyond logit-based distillation, feature-based KD transfers intermediate representations by aligning teacher and student feature-maps at selected layers. Given the teacher feature $\mathbf{h}_\ell^t$ and the student feature $\mathbf{h}_\ell^s$ at layer ℓ , a lightweight projection $g(\cdot)$ is often introduced to resolve potential dimensional mismatch. A common choice is to minimize an ℓ_2 regression loss between the aligned features [35]:

$$\mathcal{L}_{\text{feat}} = \|g(\mathbf{h}_\ell^t) - \mathbf{h}_\ell^s\|_2^2, \quad (4)$$

where $\|\cdot\|_2^2$ denotes the squared Frobenius norm for feature-maps. The overall training objective augments the logit-based loss with feature distillation,

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \beta \mathcal{L}_{\text{feat}}, \quad (5)$$

with β controlling the contribution of intermediate-feature matching.

This paradigm is particularly well-suited to KWS under on-device constraints. It preserves accuracy while operating under stringent budgets on parameters and operations. Hardware-driven KWS pipelines increasingly employ distillation to preserve accuracy under tight on-device budgets and restricted computing resources [36–39]. KD has also been combined with self-supervised speech representations to build on-device constrained KWS models [38,40,41]. The student is guided by rich teacher embeddings from a larger teacher network [38,40,41].

Overall, most KD methods for KWS rely on a pre-trained teacher to supervise a compact student [34]. When the teacher and student differ substantially in width or architecture, intermediate-feature distillation usually introduces an explicit projector to reconcile feature mismatch [35].

These observations motivate self-distillation as a teacher-free alternative for hardware deployment. Self-distillation uses an internal teacher. The deepest classifier provides soft targets for shallower blocks or early-exit heads [22,42]. Intermediate-feature self-distillation often still requires a projection module to align representations across depths [35,43,44]. Such overhead and mismatch handling remain less attractive for hardware-friendly KWS, since practical deployments may restrict the feature front-end and the available computing resources [36,45]. These limitations motivate our Stepwise Self Knowledge Distillation framework tailored to resource-constrained KWS deployment.

3. Proposed Method

3.1. SSKD Framework

To mitigate the accuracy degradation induced by 1-bit quantization in BNN-based keyword spotting, we propose a Stepwise Self-Knowledge Distillation (SSKD) framework. Different from existing Self-Knowledge Distillation paradigms that typically distill a shallow student from a deeper teacher [21], an internal teacher branch is self-generated during training, as illustrated in Figure 4. Specifically, the teacher keeps its weights in full precision and shares the student's binary activations. This design provides higher-capacity supervision while remaining quantization-consistent. Consequently, the teacher captures richer training information and transfers it to the student during training stage. For deployment, the teacher branch is discarded and retains only the binarized student to ensure efficient inference.

Motivated by the empirical observation that knowledge distillation tends to be more effective when the capacity gap between the teacher and the student is modest [46], we introduce two techniques to (i) stabilize the parallel training in SSKD and (ii) narrow the teacher–student gap. Let t and s denote the self-generated teacher and the student, respectively. Here, “modest gap” refers to our self-generated teacher–student design with a topology-matched backbone (layer-wise matched feature dimensions), where the remaining mismatch is explicitly reduced by BN parameter sharing and magnitude transfer (Equations (6) and (7)). For an L -layer network where the corresponding layers of the two branches have identical feature dimensions, we denote the output features at the l -th layer by F_l^t and F_l^s , defined as:

$$\begin{aligned} U_l^t &= W_l^t \otimes F_{l-1}^t, & F_l^t &= \Phi(\text{BN}_{\text{copy}}(U_l^t)), \\ U_l^s &= W_l^s \oplus F_{l-1}^s, & F_l^s &= \Phi(\text{BN}^s(U_l^s)), \end{aligned} \quad (6)$$

where $W_l^{(\cdot)}$ denotes the learnable weights at layer l , and $\Phi(\cdot)$ denotes the activation function. $\otimes$ denotes the standard convolution operator. In contrast, $\oplus$ denotes the binary convolution operator. The intermediate variables U_l^t and U_l^s are defined as the branch-specific feature representations before batch normalization. Subsequently, the corresponding normalized

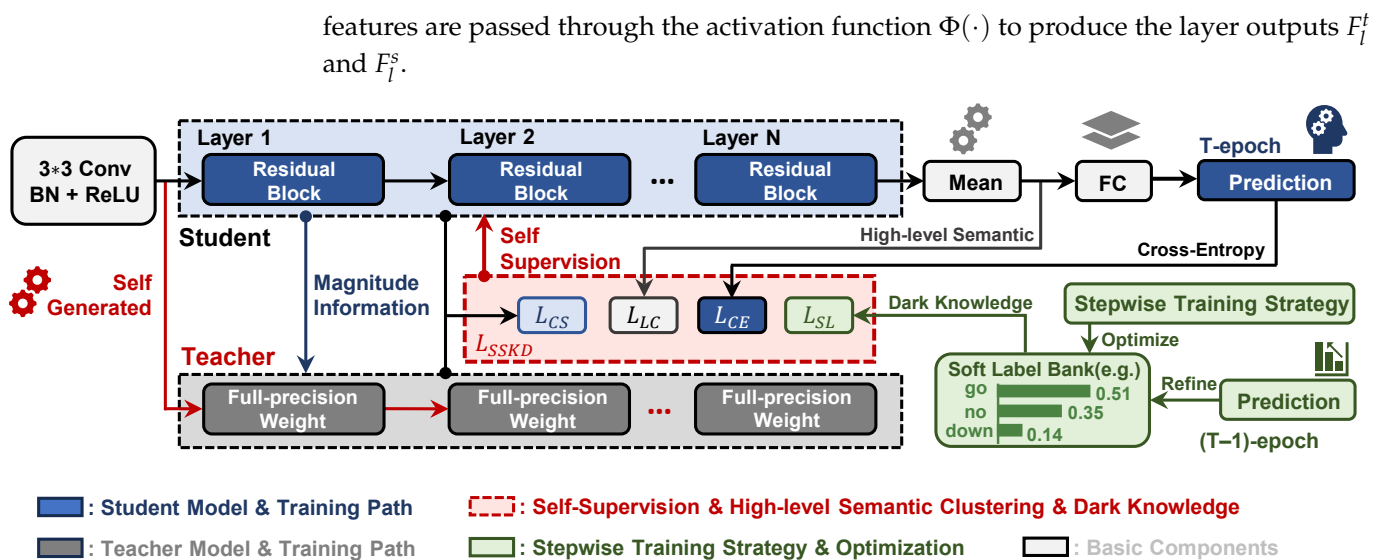


Figure 4. Overall SSKD training framework. The teacher network is self-generated and runs in parallel with the binary student. Four loss functions constitute the self-supervision L_{SSKD} . L_{CS} uses the same feature-maps and channels so that the knowledge transfer introduces no additional information loss. L_{LC} preserves high-level semantics in label clusters. L_{CE} captures hard-label knowledge. L_{SL} uses the soft labels and conveys the dark knowledge.

Method 1 (BN parameter sharing).

To narrow the distribution mismatch between the teacher and student branches, we constrain the learnable affine parameters (γ, β) in each BN_{copy} layer to be identical to those in the corresponding BN layer. The shared BN parameters (γ, β) keep the post-normalization scaling and bias consistent across the two branches. Consistent (γ, β) -based scaling and bias across branches reduces the teacher–student mismatch and stabilizes parallel SSKD training.

Method 2 (Magnitude transfer).

The missing magnitude introduces a scale mismatch between the teacher and student features. Such a scale mismatch weakens distillation effectiveness. To restore amplitude information while keeping the teacher quantization-consistent, we transfer the student-side magnitude to the teacher branch, as formulated in Equation (7). Injecting the student magnitude into the teacher keeps the teacher features and logits on a similar scale to the student. Comparable magnitudes of intermediate features and logits across the two branches reduce the teacher–student gap and stabilize parallel distillation.

$$\tilde{F}_l^t = \Phi(\text{BN}_{\text{copy}}(\text{sign}(U_l^t) \odot |U_l^s|)), \quad (7)$$

$$\text{sign}(x) = \begin{cases} +1, & x \geq 0, x \in \mathbb{R}, \\ -1, & x < 0, x \in \mathbb{R}. \end{cases} \quad (8)$$

Feature-level distillation transfers intermediate representations and provides dense supervision. However, direct feature matching typically requires matched channel dimensions and feature-map sizes between the teacher and the student. A mismatch in these dimensions necessitates an explicit channel and feature-map projection module. The additional projection module can attenuate the transferred knowledge. With this motivation, we design a channel–spatial differential distillation loss L_{CS} for feature distillation. In Figure 4, the teacher network is self-generated with the same number of channels and the same feature-map size as the student, so a channel projection module is not required. Thus,

knowledge transfer across all $L - 1$ intermediate layers does not suffer from alignment-induced information loss during distillation.

$$L_{CS} = \frac{1}{L-1} \sum_{l=2}^L \sum_{d \in \{\text{ch}, \text{sp}\}} \frac{1}{N_d} \|\hat{\mathcal{A}}_d(\tilde{F}_l^t) - \hat{\mathcal{A}}_d(F_l^s)\|_2, \quad N_{\text{ch}} = C, \quad N_{\text{sp}} = WH. \quad (9)$$

$$\hat{\mathcal{A}}_d(X) \stackrel{\text{def}}{=} \frac{\mathcal{A}_d(X)}{\|\mathcal{A}_d(X)\|_2}, \quad d \in \{\text{ch}, \text{sp}\}. \quad (10)$$

Here, $d \in \{\text{ch}, \text{sp}\}$ indexes the descriptor type, and $N_{\text{ch}} = C$ and $N_{\text{sp}} = WH$ are the corresponding normalization factors. C , H , and W denote the channel number, height, and width of a feature tensor, respectively. The subscripts ch and sp indicate channel-wise and spatial-wise descriptors. Given a feature-map $X \in \mathbb{R}^{C \times H \times W}$, $\mathcal{A}_{\text{ch}}(X) \in \mathbb{R}^C$ computes a channel descriptor by averaging X over the spatial dimensions, so each element measures the response strength of one channel. In contrast, $\mathcal{A}_{\text{sp}}(X) \in \mathbb{R}^{H \times W}$ computes a spatial descriptor by averaging X over the channel dimension, so each location reflects the saliency pattern on the feature map. When computing $\|\cdot\|_2$, $\mathcal{A}_{\text{sp}}(X) \in \mathbb{R}^{H \times W}$ is implicitly vectorized. $\hat{\mathcal{A}}_d(\cdot)$ denotes the ℓ_2 -normalized descriptor, and the normalization removes absolute magnitude so that the loss focuses on relative activation patterns.

Consequently, L_{CS} matches the teacher and student in two complementary aspects: channel utilization (via $\hat{\mathcal{A}}_{\text{ch}}$) and spatial saliency (via $\hat{\mathcal{A}}_{\text{sp}}$) across all intermediate layers. This dual matching reduces the teacher–student accuracy gap without introducing an extra projection module, and it improves the stability of parallel feature distillation under binarization.

The final classifier takes the penultimate-layer feature vector as input to compute logits for label prediction. The penultimate-layer feature vector encodes high-level, class-discriminative semantics. The geometry of this penultimate-layer feature space strongly affects the decision boundary [47,48]. The previous distillation losses align teacher and student features, but they do not explicitly enforce intra-class compactness in the student’s penultimate-layer feature space. In the BNNs student model, the penultimate-layer feature distribution becomes noisier and less clustered. The noisier and less clustered feature distribution reduces classification robustness [49].

To address this issue, we introduce a label-cluster loss L_{LC} on the student penultimate features. Let $\mathbf{f}_i \in \mathbb{R}^d$ denote the penultimate-layer feature of the i -th sample in a mini-batch, and let $y_i \in \{1, \dots\}$ be its class label. For each class k appearing in the mini-batch, we form the index set $\mathcal{I}_k = \{i \mid y_i = k\}$ with $N_k = |\mathcal{I}_k|$ samples, and compute the class centroid as the mean feature vector:

$$\mathbf{c}_k = \frac{1}{N_k} \sum_{i \in \mathcal{I}_k} \mathbf{f}_i. \quad (11)$$

We then minimize the cosine distance between each sample feature and the centroid of its class:

$$L_{LC} = \frac{1}{B} \sum_{k=1}^M \sum_{i \in \mathcal{I}_k} \frac{1 - \cos(\mathbf{f}_i, \mathbf{c}_k)}{2}, \quad \cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2}. \quad (12)$$

Here, M is the number of classes present in the mini-batch and $B = \sum_{k=1}^M N_k$ is the mini-batch size. By contracting same-class features around their centroid in the penultimate feature space, L_{LC} encourages semantic clustering and improves intra-class compactness. More compact high-level semantic clusters improve class separability and yield a more stable decision boundary. This improved stability benefits the binarized student during

parallel training distillation. The cosine-to-centroid form makes the regularization insensitive to feature magnitude and focuses on directional semantic alignment. Triplet loss typically relies on triplet mining and can be sensitive to hard negatives, which may add optimization instability [50]. Center loss introduces extra class-center states that must be updated during training [51,52]. Our batch-centroid design avoids this additional state while retaining the desired intra-class compactness.

The soft-label bank SL^t stores the self-generated teacher's softened predictive distributions in a class-wise manner. Temperature scaling exposes dark knowledge in the teacher outputs. Dark knowledge refers to the relative probability mass over non-ground-truth classes, and it encodes inter-class similarity that is absent from one-hot labels. Such similarity information provides dense supervision and often improves calibration during distillation [34].

At the end of epoch $T-1$, SL^t is updated as an $M \times M$ matrix. Each row of $SL^t \in \mathbb{R}^{M \times M}$ corresponds to one ground-truth class label. The row indexed by $y \in \{1, \dots, M\}$ stores the averaged teacher distribution over the M classes for label y . In epoch T , the student matches its softened output distribution to the row selected from SL^t via KL divergence:

$$L_{SL} = \text{KL} \left(\text{softmax} \left(\frac{P^s}{T_d} \right), \text{softmax} \left(\frac{SL^t}{T_d} \right) \right), \quad (13)$$

where P^s denotes the student logits, and T_d denotes the distillation temperature. The soft-label bank is designed to smooth the mini-batch-level noise in teacher predictions by aggregating class-wise soft targets across an epoch, which is conceptually related to using temporally averaged predictions as more stable targets [34]. This soft-label-bank supervision (carrying dark knowledge via temperature-scaled soft targets) complements hard-label training and supports stable parallel stepwise self-knowledge distillation in the BNN student.

Finally, SSKD framework optimizes a composite objective with four complementary losses. L_{CE} provides hard-label supervision for the binarized student. L_{CS} transfers intermediate representations and aligns channel-wise and spatial-wise response patterns. L_{SL} conveys dark knowledge through softened class-probability distributions and strengthens inter-class similarity supervision. L_{LC} promotes high-level semantic clustering in the penultimate-layer feature space and improves class separability.

$$L_{SSKD} = L_{CE} + \mu_{CS} L_{CS} + \mu_{SL} L_{SL} + \mu_{LC} L_{LC}, \quad (14)$$

In our implementation, we set $\mu_{CS} = 0.30$, $\mu_{SL} = 0.25$, and $\mu_{LC} = 0.25$. These factors are chosen to balance the typical scales of the auxiliary loss functions during training and to prevent any single term from dominating the optimization.

All auxiliary losses and the teacher-branch computations are enabled only during training. At inference time, the deployed model uses the BNNs student model only. The hardware-deployed model introduces no extra parameters and no additional FLOPs beyond the BNNs student backbone [34].

3.2. Stepwise Training Strategy

According to [30], a major source of instability in BNN training is the frequent back-and-forth sign changes of weights around zero. These sign reversals induce non-negligible prediction fluctuations across successive iterations. This issue is particularly problematic for SSKD: the soft-label bank is constructed from the model's own predictions. Unstable predictions can accumulate incorrect pseudo-labels in the soft-label bank. As a result, the soft-label bank accumulates misleading teacher logits over iterations. These

erroneous teacher signals tend to reinforce mistakes during Stepwise Self-Knowledge Distillation (SSKD).

Therefore, we explicitly characterize the training dynamics from two complementary perspectives. First, we quantify iteration-to-iteration sign flipping using the Optimization Stability Ratio (OSR). OSR is defined as the fraction of weights whose signs change between two consecutive updates. OSR serves as the sign-flip statistic in our stability analysis, since it directly measures iteration-to-iteration changes of binarized weight signs in BNN optimization. Second, we quantify initial sign dependency using the initialization dependency ratio (IDR). IDR is defined as the sign correlation between the weight signs at the end of training and those at initialization.

Let W_l be the set of weights in layer l and $N_{\text{total}} = \sum_{l=1}^L |W_l|$ be the total number of weights across L layers. Then OSR (evaluated at iteration t) and IDR (evaluated at the end of training) are defined as:

$$\text{OSR} = \sum_{l=1}^L \sum_{w \in W_l} \frac{|\text{sign}(w_{t+1}) - \text{sign}(w_t)|}{2N_{\text{total}}}, \quad (15)$$

$$\text{IDR} = 1 - \frac{1}{2} \sum_{l=1}^L \sum_{w \in W_l} \frac{|\text{sign}(w_{\text{final}}) - \text{sign}(w_{\text{init}})|}{2N_{\text{total}}}, \quad (16)$$

Here N_{total} is the total number of weights and t is the t -th iteration. OSR measures the proportion of weight sign flips between iterations. In contrast, IDR evaluates the dependency between final and initial weight signs.

As reported in Table 1, single step training yields a higher OSR and a lower IDR. This indicates more iteration-to-iteration weight sign flip-flops and weaker correlation between the final weight signs and their initialization. These observations are consistent with unstable optimization in the early stage. Such instability often arises from a weight distribution concentrated around zero. In this case, weight signs change easily, and the training process tends to exhibit unstable optimization and loss oscillations.

Table 1. OSR, IDR, and accuracy of BTC-ResNet8/14 trained with SSKD and Weight Scaling Factor (SCALE) on GSCDv1 and GSCDv2. Results are compared with and without the Stepwise Training Strategy (STS).

Model	Dataset	STS	OSR ↓	IDR ↑	Accuracy(%) ↑
BTC-ResNet8	GSCDv1-12	No	7.65×10^{-4}	0.3283	95.69
		Yes	8.33×10^{-5}	0.5139	96.04
	GSCDv2-12	No	7.59×10^{-4}	0.3334	95.86
		Yes	8.38×10^{-5}	0.5035	96.06
	GSCDv2-35	No	7.93×10^{-4}	0.2947	92.09
		Yes	8.41×10^{-5}	0.4881	92.40
BTC-ResNet14	GSCDv1-12	No	5.82×10^{-4}	0.5283	96.75
		Yes	1.63×10^{-5}	0.6179	97.23
	GSCDv2-12	No	5.73×10^{-4}	0.5272	96.80
		Yes	1.65×10^{-5}	0.6029	97.31
	GSCDv2-35	No	5.91×10^{-4}	0.4263	95.20
		Yes	1.78×10^{-5}	0.4999	95.56

↑/↓ indicate that higher/lower values are better, respectively. Bold values denote the better results under the same model and dataset setting (STS: Yes vs. No).

Under SSKD, this instability degrades the teacher logit distribution used for self-supervision. It produces unreliable softened targets and weakens the effectiveness of distillation.

Formally, let the binary weights be $b_t = \text{sign}(w_t)$, and the teacher logits at iteration t be

$$\mathbf{z}_t(\mathbf{x}) = f(\mathbf{x}; b_t). \quad (17)$$

The softened teacher logits used for distillation are computed by temperature scaling:

$$\mathbf{q}_t(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{z}_t(\mathbf{x})}{T}\right), \quad (18)$$

which is the standard formulation in knowledge distillation.

In SSKD, the soft-label bank SL^t stores the self-generated teacher's class-conditional predictive statistics across epochs. Let $P^t(\mathbf{x})$ and $P^s(\mathbf{x})$ denote the teacher and student logits respectively, and let T_d be the distillation temperature. For each class $y \in \{1, \dots, M\}$, the soft-label bank is updated at the end of epoch $E - 1$ by averaging teacher outputs over samples from class y :

$$\text{SL}_{E-1}^t(y, :) = \frac{1}{N_y} \sum_{i: y_i=y} P^t(\mathbf{x}_i), \quad N_y = |\{i : y_i = y\}|. \quad (19)$$

In epoch E , the student matches its softened predictive distribution to the row indexed by the ground-truth label y through KL divergence:

$$L_{\text{SL}} = \text{KL}\left(\text{softmax}\left(\frac{P^s(\mathbf{x})}{T_d}\right) \parallel \text{softmax}\left(\frac{\text{SL}_{E-1}^t(y, :)}{T_d}\right)\right), \quad (20)$$

which follows the standard temperature-scaled distillation formulation.

Therefore, when frequent sign flip-flops (high OSR and low IDR) cause the teacher outputs to fluctuate, the class-conditional teacher distributions become less reliable within an epoch. The resulting $\text{SL}_{E-1}^t(y, :)$ can deviate from the true class-conditional teacher behavior and provide misleading supervision in L_{SL} . The student then repeatedly fits self-generated logits that contain systematic errors, and these errors tend to be reinforced across epochs. This effect is closely related to confirmation bias in pseudo-label learning [53].

This instability arises from the discontinuity introduced by the sign operation in BNN optimization. The sign function is discontinuous and non-differentiable, so the true gradient with respect to binary weights is not available [54]. Many latent weights stay close to zero in the early stage, and small parameter updates can switch their signs back and forth. These sign flip-flops perturb the model outputs and make the teacher logits fluctuate across iterations. SSKD further amplifies this issue because the soft-label bank is updated with teacher logits from previous iterations. Early-stage fluctuations therefore update the soft-label bank with inconsistent and incorrect teacher targets. Subsequent SSKD trains against these logits, and the resulting errors tend to be reinforced over iterations. To address this instability, we propose the Stepwise Training Strategy shown in Figure 5. In Step 1, we conduct coarse-grained training with full-precision weights, and the activations remain binarized. This stage bypasses the discontinuity of the sign operator and pushes latent weights away from the zero boundary. It suppresses early sign flip-flops and leads to lower OSR and higher IDR. Meanwhile, the self-generated teacher produces more consistent logits, and the soft-label bank is updated with reliable class-conditional targets that preserve meaningful inter-class relations. Step 1 therefore serves as a dedicated stabilization stage

within our SSKD framework, and it secures reliable self-supervision before Step 2 enables weight binarization and performs fine-grained tuning.

In Step 2, we activate weight binarization and proceed with fine-grained tuning after Step 1. Step 1 coarse-grained training pushes the latent full-precision weights away from the sign threshold around zero. Successive updates trigger fewer sign switches, so OSR decreases and the optimized training becomes more stable. The self-generated teacher therefore produces more consistent logits within each epoch, and the soft-label bank is updated with more reliable class-conditional targets. The distillation term L_{SL} can thus transfer dark knowledge. Consequently, compared with single-step training, our proposed Stepwise Training Strategy improves both training stability and the final accuracy. The corresponding analysis and experimental results of OSR and IDR are reported in Table 1 and Figure 7.

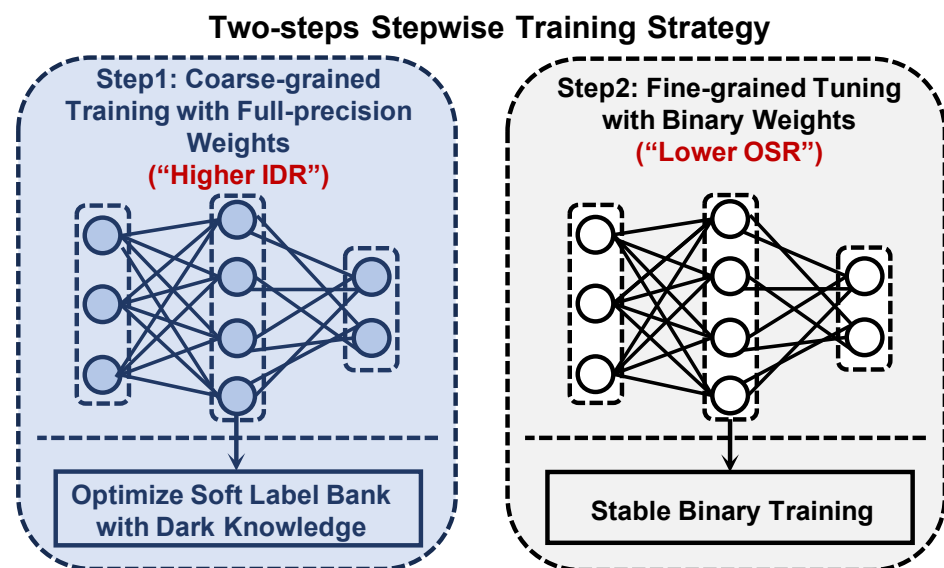


Figure 5. Stepwise Training Strategy. Step 1: coarse-grained training with full-precision weights. Step 2: fine-grained tuning to achieve stable binary training

3.3. Weight Scaling Factor

Binary weights preserve only sign information and discard magnitude. Binarization therefore introduces an intrinsic scale mismatch between full-precision weights and their binarized weights. To compensate for this mismatch and improve the student's accuracy performance in approximating full-precision teacher's accuracy, we introduce a positive scaling factor α and seek the best scaled binary approximation in the least-squares sense. Given a full-precision weight filter $W \in \mathbb{R}^{C \times H \times W}$ and a binary filter $B \in \{+1, -1\}^{C \times H \times W}$, we solve:

$$\min_{\alpha > 0, B} \|W - \alpha B\|_2^2 = \min_{\alpha > 0, B} \left[\|W\|^2 - 2\alpha W^T B + \alpha^2 (B^T B) \right]. \quad (21)$$

Let $w = \text{vec}(W) \in \mathbb{R}^n$ and $b = \text{vec}(B) \in \{+1, -1\}^n$, where n is the number of elements. For a fixed b , expanding $\|w - \alpha b\|_2^2$ and setting $\partial/\partial\alpha = 0$ yields

$$\alpha^*(b) = \frac{w^T b}{b^T b}. \quad (22)$$

Since $b_i \in \{+1, -1\}$, we have $b^T b = n$. Substituting this into the objective shows that minimizing $\|w - \alpha b\|_2^2$ is equivalent to maximizing $w^T b$ over $b \in \{+1, -1\}^n$. The optimum is attained at

$$b^* = \text{sign}(w), \quad \alpha^* = \frac{w^T b^*}{n} = \frac{1}{n} \sum_{i=1}^n |w_i| = \frac{1}{n} \|w\|_1. \quad (23)$$

This optimal Weight Scaling Factor α^* is an important component of our SSKD framework. α^* minimizes the binary approximation error $\|W - \alpha B\|_2^2$ and compensates for the magnitude loss introduced by binarization. The scaled binary weights $\alpha^* B$ yield feature responses and logits that are closer in scale to the full-precision teacher and narrow the teacher–student mismatch at both the feature and logit levels. It reduces distortion caused by amplitude mismatch and strengthens feature-level distillation, which benefits L_{CS} and L_{LC} . It also yields more reliable softened output distributions under temperature scaling, so dark knowledge can be transferred more effectively. As a result, the soft-label bank is updated with more reliable class-conditional teacher targets, and L_{SL} provides stronger self-supervision.

3.4. Model Architecture

Always-on keyword spotting requires low latency and tight computing budgets on edge devices. Given a $T \times F$ time–frequency Mel spectrogram feature map, we treat the frequency bins as channels and apply 1-D temporal convolutions along the time axis to obtain an efficient backbone with a large temporal receptive field. Following this design principle, we instantiate our skeleton binary neural network BTC-ResNet in Figure 6 by adopting the compact residual topology of TC-ResNet [16] and adapting it to the binary setting. In particular, BTC-ResNet8 stacks lightweight residual blocks with temporal downsampling, and it uses global average pooling followed by a linear classifier to produce keyword logits.

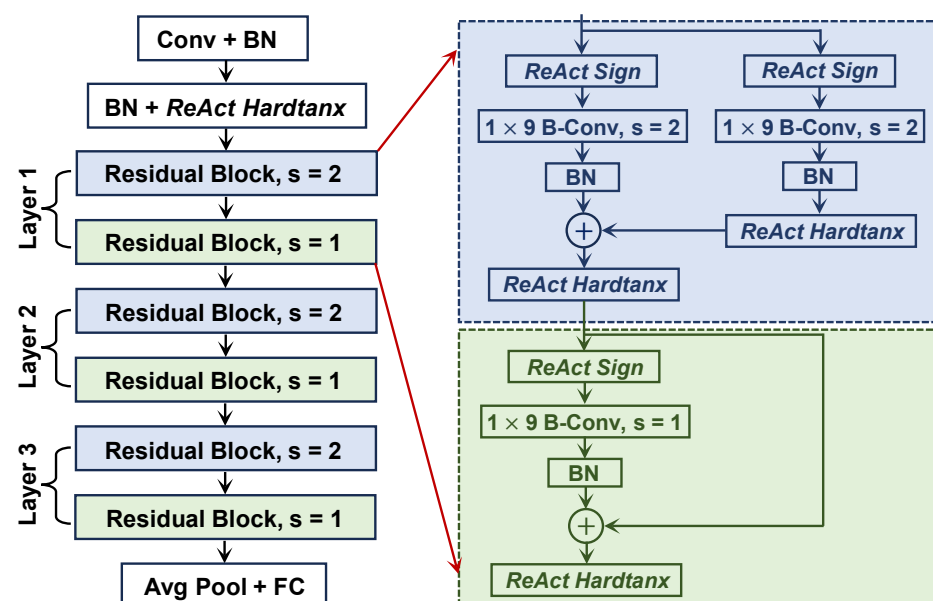


Figure 6. Architecture of our BTC-ResNet8 and its residual blocks.

Inspired by Bi-Real Net [29], our proposed BTC-ResNet introduces a real-valued shortcut pathway to compensate for the representational loss caused by 1-bit binarization. The shortcut propagates full-precision feature information across residual blocks and provides an additional high-fidelity information route alongside the binarized path.

For our proposed BTC-ResNet8, a compact configuration is adopted as shown in Figure 6. The channel widths of the convolutional layer and the three subsequent residual blocks are set to 16, 24, 32, and 48, respectively. For BTC-ResNet14, we increase depth by doubling the number of residual blocks per layer. We scale the output channels proportionally to maintain sufficient capacity under binarization. The additional blocks use stride 1 to preserve the temporal resolution and to enlarge the effective receptive field without additional temporal downsampling.

To alleviate quantization-induced degradation, the first convolutional layer and the final fully connected layer remain in full precision. These two layers are particularly sensitive to binarization because the first layer directly transforms real-valued acoustic features and the last layer determines the class-separating logits [26,29,54,55]. Keeping them in full precision is a common and effective practice in strong BNN designs. For activations, we employ ReAct-Sign to explicitly reshape and shift activation distributions. ReActNet shows that binary networks are highly sensitive to near-zero distribution shifts, and learnable distribution control improves semantic feature learning under 1-bit constraints [30]. In addition, we apply HardTanh-style clipping to bound the pre-activation range. HardTanh clips the pre-activations to $[-1, 1]$. This clipping suppresses extreme values and stabilizes gradient propagation under the STE with binarized activations. The bounded range also prevents degenerate all-one feature maps at inference and improves the reliability of the binarized representations.

Together, the real-valued shortcut, the full-precision boundary layers, and the activation distribution control via ReAct-Sign and HardTanh clipping form our proposed hardware-friendly BTC-ResNet backbone. These components target complementary failure modes of BNNs, including limited representational capacity and unstable optimization caused by the discontinuous sign operation. The architecture preserves the core efficiency advantage of BNN inference because the intermediate binarized-convolutions remain bitwise-friendly. The TC-ResNet-style 1-D temporal convolution further reduces computation compared with 2-D time–frequency kernels and fits the latency constraints of always-on KWS.

Consequently, our proposed BTC-ResNet delivers high accuracy under extremely tight parameter and computational budgets.

4. Experiment

4.1. Experiment Setup

Experiments are conducted on two versions of the Google Speech Commands Datasets (GSCD) [56]. GSCDv1 contains 64,727 audio files, and GSCDv2 contains 105,829 audio files. Each audio clip is a single-channel WAV file with a duration of at most 1 s, sampled at 16 kHz. For GSCDv1, we follow the standard 12-class KWS protocol with ten command keywords (“yes”, “no”, “up”, “down”, “left”, “right”, “on”, “off”, “stop”, “go”), plus two auxiliary labels: “unknown” and “silence”. For GSCDv2, we evaluate the full 35-word classification setting with the following keywords: “yes”, “no”, “up”, “down”, “left”, “right”, “on”, “off”, “stop”, “go”, “zero”, “one”, “two”, “three”, “four”, “five”, “six”, “seven”, “eight”, “nine”, “bed”, “bird”, “cat”, “dog”, “happy”, “house”, “marvin”, “sheila”, “tree”, “wow”, “backward”, “forward”, “follow”, “learn”, “visual”. We follow the standard Google Speech Commands (GSCD) split protocol, where the validation and test sets are predefined by `validation_list.txt` and `testing_list.txt`, and the training set consists of the remaining samples (approximately 80%/10%/10%). The split assignment is deterministic and is computed from a hash of the file name. The hashing rule first canonicalizes the file identifier by removing the trailing variation index in the file name. This keeps related recordings in the same partition and avoids cross-contamination across splits. Standard speech augmentations are applied including random time shifting and background-noise mixing. The augmentation protocol used in the official Speech Commands training setup is adopted. Specifically, background noise is sampled from the official `_background_noise_` recordings and mixed into training utterances with probability 0.8 using a bounded mixing scale $\alpha \in [0, 0.1]$. Specifically, each training waveform is randomly shifted in time (up to 100 ms), and background noise is mixed into a portion of the training samples with a controlled mixing volume. Feature extraction is performed according to prior work [5].

Mel spectrogram features are computed at a 16 kHz sampling rate using a 30 ms analysis window and a 10 ms frame shift. The feature dimension is set to 40. Moreover, the Periodic Window (PW) is adopted for Backward Gradient Correction (BGC) to reduce STE-induced gradient mismatch and stabilize binary training [14].

All models are implemented in PyTorch (v1.12.0, CUDA 10.2) and trained for 300 epochs using the Adam optimizer. The learning rate is initialized to 2×10^{-3} and is adjusted by a ReduceLROnPlateau scheduler based on the validation metric. During the proposed two-step Stepwise Training Strategy, a weight decay of 5×10^{-7} is applied in Step 1 and is disabled in Step 2. For fair comparisons with other BNN-based KWS studies, the total computation is reported as

$$\text{FLOPs}^* = \text{FLOPs} + \frac{\text{BOPs}}{64}, \quad (24)$$

where FLOPs counts floating-point operations in full-precision layers and BOPs counts bitwise operations in binarized layers [30]. The model size and computational cost are reported as Para. (Mb) and FLOPs* (Mil.) in Table 4.

4.2. Experimental Results and Discussion

Training Stability

Training stability is evaluated using the Optimization Stability Ratio (OSR) and the Initialization Dependency Ratio (IDR). Iteration-to-iteration sign flipping is quantified by OSR. OSR is defined as the fraction of weights with sign changes between two consecutive iterations. Initial sign dependency is quantified by IDR. IDR is defined as the sign correlation between the weight signs at the end of training and those at initialization. We use OSR as the primary quantitative indicator of training stability. A lower OSR indicates fewer iteration-to-iteration sign flips, which reflects a more stable optimization process. As summarized in Table 1, OSR exhibits a clear negative correlation with IDR, and the achieved accuracy is consistent with these stability indicators. This behavior is expected in binary neural network optimization. The sign function introduces a discontinuity at the binarization threshold ($w = 0$). This pattern is also observed in Figure 7a. A pronounced concentration around $w \approx 0$ is shown under single-step training. A large fraction of latent real-valued weights stays close to this threshold under standard single-step training. The signs of these near-zero weights are highly sensitive to mini-batch SGD updates. Sign flips are therefore frequently observed across iterations. Backpropagation relies on the straight-through estimator (STE). A forward-backward mismatch is introduced by this surrogate gradient. Persistent sign flips increase the variance of the effective mini-batch SGD updates and reduce convergence stability.

The contribution of our proposed Stepwise Training Strategy (STS) is quantified in Table 1. All results are obtained with SSKD and the Weight Scaling Factor (SCALE) enabled. Consistent stability gains are observed with STS for both BTC-ResNet8/14 and for both 12/35 class keyword spotting tasks. For BTC-ResNet8, OSR is reduced from $(7.59\text{--}7.93) \times 10^{-4}$ to $(8.33\text{--}8.41) \times 10^{-5}$, corresponding to an $\sim 9\times$ reduction. IDR increases from 0.29–0.33 to 0.49–0.51. The recognition accuracy is improved by 0.20–0.35 points. For BTC-ResNet14, a larger reduction in OSR is obtained. OSR decreases from $(5.73\text{--}5.91) \times 10^{-4}$ to $(1.63\text{--}1.78) \times 10^{-5}$, corresponding to a 33–36 $\times$ reduction. IDR increases from 0.43–0.53 to 0.50–0.62. The recognition accuracy is improved by 0.36–0.51 points. Overall, the effect of STS is reflected in both stability indicators: iteration-to-iteration sign flips are suppressed (lower OSR), and the final sign pattern becomes more consistent with initialization (higher IDR).

In addition to the STS-induced improvements, Table 1 provides two additional observations that support the interpretability of OSR and IDR. (i) Model scaling. BTC-ResNet14 exhibits lower OSR and higher IDR than BTC-ResNet8 under the same setting. The accuracy advantage is preserved with and without STS. This trend indicates stronger optimization robustness for the larger backbone. (ii) Task complexity. The 35-class task is more complex than the 12-class task. Higher OSR and lower IDR are consistently observed in the 35-class setting for both backbones, accompanied by a consistent drop in accuracy. This pattern suggests that more weights remain close to the binarization threshold under the more complex task. Sign flips are therefore more frequent for these near-zero weights.

Figure 7 illustrates the weight distribution of BTC-ResNet8 as a representative example. Under single-step training, a pronounced concentration is observed around the binarization threshold ($w \approx 0$) in Figure 7a. Such near-threshold weights are highly sensitive to mini-batch SGD update fluctuations, and frequent sign flips are consequently observed across iterations. With STS, the pronounced concentration near the binarization threshold is substantially attenuated, and the latent-weight distribution becomes more strongly bimodal toward $\{-1, +1\}$ in Figure 7b. A non-negligible density may still remain around $w \approx 0$, since the latent weights are real-valued and are not explicitly driven away from the binarization threshold. This shift is consistent with the reduced OSR and increased IDR reported in Table 1, and it aligns with the observed accuracy improvements.

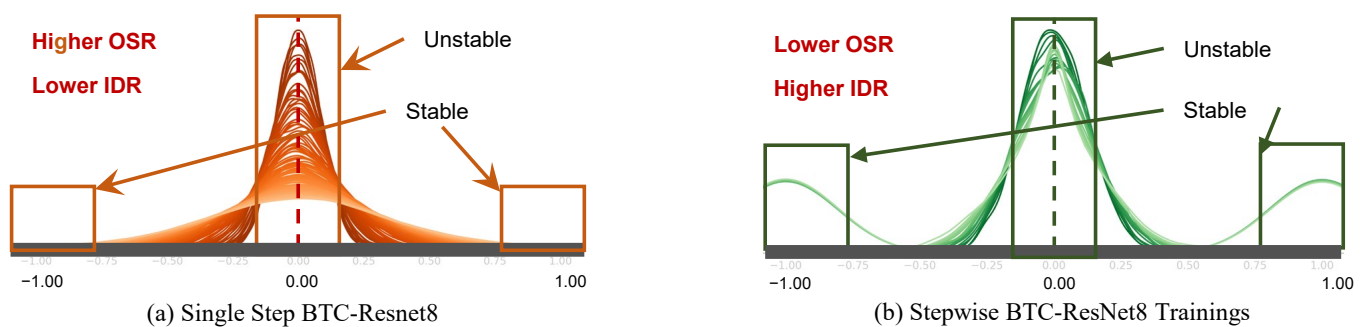


Figure 7. Distributions of latent (pre-binarization) weights in BTC-ResNet8 trained with (a) single-step training and (b) our two-step stepwise training. The Stepwise Training Strategy (STS) substantially suppresses the concentration near $w \approx 0$ and produces a more polarized distribution with peaks near -1 and $+1$. Improved weight stability is evidenced by lower OSR and higher IDR.

4.3. Ablation Study

4.3.1. Ablation Study I: Contribution of Components in Our Proposed SSKD Framework

The component-wise ablation study of our proposed SSKD framework is reported in Table 2 under identical backbones and training settings.

Compared with the hard-label baseline trained with cross-entropy only, the full SSKD framework yields substantial and consistent improvements across all datasets and backbones. For CLASS12, BTC-ResNet8 improves from 93.06% to 96.04% on GSCDv1 and from 92.98% to 96.06% on GSCDv2. BTC-ResNet14 improves from 94.86% to 97.23% on GSCDv1 and from 94.72% to 97.31% on GSCDv2. For the more complex CLASS35 task on GSCDv2, the gains are larger. In particular, BTC-ResNet8 improves from 89.30% to 92.40%, and BTC-ResNet14 improves from 90.32% to 95.56%. These results verify the effectiveness of the self-generated teacher supervision and our proposed auxiliary components in bridging the accuracy gap caused by binarization.

The largest accuracy drops are observed when BN parameter sharing or magnitude transfer is removed. This indicates that reducing the teacher–student mismatch is important for effective parallel distillation. On CLASS12 with BTC-ResNet14, disabling BN parameter sharing reduces accuracy from 97.23% to 95.65% on GSCDv1 and from 97.31% to 95.65% on

GSCDv2. Disabling magnitude transfer leads to a similar degradation. Accuracy decreases from 97.23% to 95.48% on GSCDv1 and from 97.31% to 95.53% on GSCDv2. A larger accuracy degradation is observed for the CLASS35 task with BTC-ResNet14. Removing BN parameter sharing reduces the GSCDv2 accuracy from 95.56% to 92.73%, and removing magnitude transfer reduces it from 95.56% to 92.53%. This is consistent with our design motivation. BN parameter sharing enforces identical learnable scale and shift parameters (γ, β) in the batch-normalization layers of the teacher and student branches. Magnitude transfer in Equation (7) transfers the student-side magnitude to the teacher branch and maintains comparable feature and logit scales across branches. Together, these two methods reduce the teacher–student gap and improve the stability of our proposed SSKD framework.

Table 2. Component-wise ablation of our proposed SSKD framework. BN denotes BN parameter sharing; Mag denotes magnitude transfer in Equation (7). L_{CS} , L_{LC} , and L_{SL} correspond to the channel–spatial distillation loss, label-cluster loss, and soft-label bank loss, respectively.

Model	Components					Accuracy (%)	
	BN	Mag	L_{CS}	L_{LC}	L_{SL}	GSCDv1	GSCDv2
CLASS12	–	–	–	–	–	93.06	92.98
	✓	✓	✓	✓	✓	96.04	96.06
	×	✓	✓	✓	✓	94.92	95.01
	✓	×	✓	✓	✓	94.88	94.79
	✓	✓	×	✓	✓	95.43	95.38
	✓	✓	✓	×	✓	95.77	95.81
	✓	✓	✓	✓	×	95.62	95.57
	–	–	–	–	–	94.86	94.72
	✓	✓	✓	✓	✓	97.23	97.31
	×	✓	✓	✓	✓	95.65	95.65
	✓	×	✓	✓	✓	95.48	95.53
	✓	✓	×	✓	✓	96.03	95.99
	✓	✓	✓	×	✓	96.75	96.82
	✓	✓	✓	✓	×	96.86	96.91
CLASS35	–	–	–	–	–	–	89.30
	✓	✓	✓	✓	✓	–	92.40
	×	✓	✓	✓	✓	–	90.25
	✓	×	✓	✓	✓	–	89.97
	✓	✓	×	✓	✓	–	90.76
	✓	✓	✓	×	✓	–	91.52
	✓	✓	✓	✓	×	–	91.54
	–	–	–	–	–	–	90.32
	✓	✓	✓	✓	✓	–	95.56
	×	✓	✓	✓	✓	–	92.73
	✓	×	✓	✓	✓	–	92.53
	✓	✓	×	✓	✓	–	93.78
	✓	✓	✓	×	✓	–	94.81
	✓	✓	✓	✓	×	–	94.62

✓ indicates that the corresponding component is enabled; × indicates that the component is disabled; “–” indicates not applicable (N/A) for that configuration.

Among the auxiliary losses, L_{CS} yields the main additional improvement. Dense intermediate feature alignment is achieved without introducing extra projection modules. Removing L_{CS} reduces BTC-ResNet14 accuracy on CLASS12 from 97.23% to 96.03% on GSCDv1 and from 97.31% to 95.99% on GSCDv2. On CLASS35, the accuracy decreases from 95.56% to 93.78%. The accuracy gains contributed by L_{LC} and L_{SL} are smaller but remain consistent across all settings. On CLASS35 with BTC-ResNet14, removing L_{LC} reduces the GSCDv2 accuracy from 95.56% to 94.81%. Removing L_{SL} reduces the GSCDv2 accuracy from 95.56% to 94.62%. This pattern is consistent with the effects targeted by the

two components. L_{LC} encourages intra-class compactness in the student's penultimate feature space and strengthens high-level semantic separability. L_{SL} matches softened class-probability distributions stored in the soft-label bank, thereby conveying dark knowledge beyond hard-label supervision.

Overall, the highest accuracy is achieved when all components are enabled. The results in Table 2 indicate that BN parameter sharing and magnitude transfer are critical for our proposed SSKD framework. The auxiliary components L_{CS} , L_{LC} , and L_{SL} contribute additional gains via representation alignment and semantic supervision.

4.3.2. Ablation Study II: Effect of SSKD Framework, STS, and SCALE

Table 3 presents ablations of three components in our experimental setting: the SSKD framework, Stepwise Training Strategy (STS), and Weight Scaling Factor (SCALE). The SSKD framework is proposed to provide self-supervised distilled knowledge for compact BNN students, STS is proposed to enhance the stability of weight-sign, and SCALE is proposed to compensate for the magnitude mismatch introduced by binarization.

Table 3. Ablation study of SSKD framework, Stepwise Training Strategy (STS), and Weight Scaling Factor (SCALE) using BTC-ResNet8/14 on GSCDv1 and GSCDv2.

	Model	Components			Accuracy (%)	
		SSKD	STS	SCALE	GSCDv1	GSCDv2
CLASS12	BTC-ResNet8	×	×	×	93.06	92.98
		✓	×	×	95.57	95.69
		✓	✓	×	95.71	95.88
		✓	×	✓	95.69	95.86
		✓	✓	✓	96.04	96.06
	BTC-ResNet14	×	×	×	94.86	94.72
		✓	×	×	96.52	96.40
		✓	✓	×	96.90	96.80
		✓	×	✓	96.75	96.80
		✓	✓	✓	97.23	97.31
CLASS35	BTC-ResNet8	×	×	×	-	89.30
		✓	×	×	-	91.85
		✓	✓	×	-	92.28
		✓	×	✓	-	92.08
		✓	✓	✓	-	92.40
	BTC-ResNet14	×	×	×	-	90.32
		✓	×	×	-	94.97
		✓	✓	×	-	95.27
		✓	×	✓	-	95.20
		✓	✓	✓	-	95.56

✓ indicates that the corresponding component is enabled; × indicates that the component is disabled; “-” indicates not applicable (N/A). Bold values denote the best accuracy within the same setting.

The SSKD framework contributes the dominant accuracy gain in all settings. For CLASS12, BTC-ResNet8 improves from 93.06%/92.98% to 95.57%/95.69% on GSCDv1/v2, and BTC-ResNet14 improves from 94.86%/94.72% to 96.52%/96.40%. For the more complex CLASS35 task on GSCDv2, the gain becomes larger, especially for BTC-ResNet14 (from 90.32% to 94.97%, +4.65 percentage points). This behavior is consistent with the proposed SSKD mechanism. A self-generated full-precision teacher is trained in parallel with the binary student under quantization-consistent activations, and the teacher–student gap is further reduced by BN parameter sharing and magnitude transfer. The resulting supervision becomes richer than hard labels alone. The gain is more pronounced on the

35-class task, since softened targets convey inter-class similarity (dark knowledge) and provide denser guidance under a more complex label space.

Relative to the SSKD-only setting, Stepwise Training Strategy (STS) provides consistent additional accuracy gains. For BTC-ResNet8, STS improves accuracy by about 0.14%~0.19% on CLASS12 and by 0.43% on CLASS35. For BTC-ResNet14, STS improves accuracy by about 0.38%~0.40% on CLASS12 and by 0.30% on CLASS35. These gains are attributed to improved training stability in BNN optimization. Binary training relies on surrogate gradients for the non-differentiable sign function, and frequent sign flips can destabilize optimization. STS mitigates training instability by suppressing iteration-to-iteration sign flips.

Weight Scaling Factor (SCALE) consistently improves accuracy across the evaluated settings. The accuracy gain is 0.12–0.17% on BTC-ResNet8 for CLASS12, 0.23% on BTC-ResNet8 for CLASS35, and 0.23% on BTC-ResNet14 for CLASS35. On BTC-ResNet14 for CLASS12 (GSCDv2), SCALE matches STS. This behavior is consistent with scaling factors compensating for the scale mismatch introduced by binarization.

These improvements reflect complementary contributions of all three components. SSKD provides a self-generated teacher and self-supervised distillation targets without requiring an external pretrained teacher. STS suppresses iteration-to-iteration sign flips and improves optimization stability under surrogate-gradient training for binarization. SCALE compensates for the magnitude loss introduced by binarization and improves scale consistency for forward activations and backward updates. The three components jointly strengthen supervision, stabilize training, and reduce scale mismatch. The best accuracy is therefore obtained.

4.4. Comparative Experiments

Table 4 compares our BTC-ResNet models with representative BNN-based KWS models, lightweight full-precision KWS models, and BNN+knowledge distillation-based KWS models on GSCDv1 and GSCDv2. All reported accuracy, parameter counts, and FLOPs* are measured for the BNNs student model during inference on the test dataset. The self-generated teacher branch and all distillation-related components are used only during training and are discarded at inference. No additional parameters or inference FLOPs are required for hardware deployment. The parameter count and FLOPs* follow the definitions in Section 4.1. The Para. and FLOPs* of our BTC-ResNet8/14 differ slightly between the 12-class and 35-class configurations. The classifier head accounts for this difference. The output dimension of the final linear layer equals the number of target classes, and the head parameters and operations scale accordingly.

CLASS12 (GSCD v1-12/v2-12): Our BTC-ResNet8 achieves 96.04% on GSCD v1-12 and 96.06% on GSCD v2-12 with only 0.26 Mb parameters and 0.39 Mil. FLOPs*. Compared with our previous BNN-based KWS model, B-ResNet14 (S-FC) [14], BTC-ResNet8 improves accuracy by 4.56 percentage points on GSCD v1-12. The parameter size is reduced from 0.43 to 0.26 Mb, and FLOPs* is reduced from 3.23 to 0.39 Mil. BiFSMNv2 [2] is the latest knowledge distillation-assisted BNN-based KWS model. For BiFSMNv2 (1.0 $\times$, $\delta = 1$), FLOPs* is 35.01 Mil. with 96.42% (GSCD v1-12) and 96.45% (GSCD v2-12). BTC-ResNet8 reduces FLOPs* by 89.77 $\times$; the accuracy differences are 0.38 and 0.39 percentage points on GSCD v1-12 and GSCD v2-12, respectively. For BiFSMNv2 (0.5 $\times$, $\delta = 2$), FLOPs* is 27.96 Mil. with 96.23% (GSCD v1-12) and 95.87% (GSCD v2-12). BTC-ResNet8 reduces FLOPs* by 71.69 $\times$; the accuracy differences are 0.19 percentage points on GSCD v1-12 and 0.19 percentage points on GSCD v2-12. For BiFSMNv2 (0.25 $\times$, $\delta = 4$), FLOPs* is 24.43 Mil. with 94.65% (GSCD v1-12) and 94.25% (GSCD v2-12). BTC-ResNet8 reduces FLOPs* by 62.64 $\times$; the accuracy improvements are 1.39 and 1.81 percentage points on v1-12 and

v2-12, respectively. BTC-ResNet14 achieves 97.23% (GSCD v1-12) and 97.31% (GSCD v2-12) with 0.75 Mb parameters and 1.35 Mil. FLOPs*. It exceeds BiFSMNv2 ($1.0\times, \delta = 1$) [2] by 0.81 and 0.86 percentage points on GSCD v1-12 and GSCD v2-12, respectively. FLOPs* is reduced by $25.93\times$.

CLASS35 (GSCD v2-35): BTC-ResNet8 achieves 92.40% with 0.30 Mb parameters and 0.40 Mil. FLOPs*. Table 4 also includes BNN-based KWS baselines (BNN, XNOR++, Bi-Real, and ReActNet). BTC-ResNet8 improves accuracy by 2.17 percentage points (vs. XNOR++) to 10.54 percentage points (vs. Bi-Real). FLOPs* is reduced by $97.20\times$ (vs. BNN) to $122.45\times$ (vs. ReActNet). Under ($1.0\times, \delta = 1$), BiFSMNv2 reports 93.05% with 38.02 Mil. FLOPs*. FLOPs* is reduced by $95.05\times$, and the accuracy difference is 0.65 percentage points. Under ($0.5\times, \delta = 2$) and ($0.25\times, \delta = 4$), BiFSMNv2 reports 92.04% and 90.04% with 30.97 and 27.45 Mil. FLOPs*, respectively. FLOPs* is reduced by $77.43\times$ and $68.63\times$. BTC-ResNet8 is higher by 0.36 and 2.36 percentage points, respectively. BTC-ResNet14 achieves 95.56% with 0.76 Mb parameters and 1.35 Mil. FLOPs*. It improves accuracy by 5.33 percentage points (vs. XNOR++) to 13.70 percentage points (vs. Bi-Real). FLOPs* is reduced by $28.80\times$ (vs. BNN) to $36.28\times$ (vs. ReActNet). It exceeds BiFSMNv2 under ($1.0\times, \delta = 1$), ($0.5\times, \delta = 2$), and ($0.25\times, \delta = 4$) by 2.51, 3.52, and 5.52 percentage points, respectively. FLOPs* is reduced by $28.16\times$, $22.94\times$, and $20.33\times$, respectively.

Table 4. Comparison of our BTC-ResNet with other models on GSCDv1/GSCDv2. For BiFSMNv2 [2], ($s\times, \delta$) denotes the thinnable setting with block interval δ ; only blocks in $\{i\delta\}$ are executed (thus $s \in \{1.0, 0.5, 0.25\}$ for $\delta \in \{1, 2, 4\}$). Para. is fixed, while FLOPs* changes with block skipping.

	Model	Para. (Mb)	FLOPs* (Mil.)	GSCD v1(%)	GSCD v2(%)
CLASS 12	BNN [54]	0.60	35.87	68.84	70.87
	XNOR++ [28]	0.61	45.04	91.71	82.24
	Bi-Real [29]	0.60	40.46	85.87	87.93
	ReActNet [30]	0.60	45.96	93.96	94.02
	TC-ResNet8 [16]	2.11	3.00	96.10	-
	TC-ResNet14 [16]	4.39	6.10	96.20	-
	B-ResNet8 [14]	1.24	1.32	88.32	-
	B-ResNet14 (S-FC) [14]	0.43	3.23	91.48	-
	BiFSMNv2 ($1.0\times, \delta = 1$) [2]	0.30	35.01	96.42	96.45
	BiFSMNv2 ($0.5\times, \delta = 2$) [2]	0.30	27.96	96.23	95.87
	BiFSMNv2 ($0.25\times, \delta = 4$) [2]	0.30	24.43	94.65	94.25
	our BTC-ResNet8	0.26	0.39	96.04	96.06
	our BTC-ResNet14	0.75	1.35	97.23	97.31
CLASS 35	BNN [54]	0.60	38.88	-	84.13
	XNOR++ [28]	0.61	48.06	-	90.23
	Bi-Real [29]	0.60	43.47	-	81.86
	ReActNet [30]	0.60	48.98	-	88.96
	BiFSMNv2 ($1.0\times, \delta = 1$) [2]	0.30	38.02	-	93.05
	BiFSMNv2 ($0.5\times, \delta = 2$) [2]	0.30	30.97	-	92.04
	BiFSMNv2 ($0.25\times, \delta = 4$) [2]	0.30	27.45	-	90.04
	our BTC-ResNet8	0.30	0.40	-	92.40
	our BTC-ResNet14	0.76	1.35	-	95.56

Bold values indicate the best results in the corresponding column among the compared methods.

Overall, Table 4 confirms the synergy of the four proposed components. SSKD framework strengthens supervision through a self-generated teacher, and it removes the reliance on an external pretrained teacher for hardware deployment. Stepwise Training Strategy improves optimization stability for binarization under surrogate-gradient training and

suppresses iteration-to-iteration sign flips. Weight Scaling Factor compensates for the magnitude loss introduced by binarization. BTC-ResNet provides a compact residual backbone for small-footprint KWS; it reduces parameter and computation costs while maintaining strong representational capacity. The four components are integrated within a consistent framework. Overall, the BNNs student model achieves a better accuracy–efficiency trade-off. Table 4 shows that our models attain competitive or higher accuracy with lower parameter count and FLOPs* than other BNN-based KWS baselines and distillation-assisted BNN-based KWS models.

5. Discussion and Limitations

5.1. Broader Applicability to Other Tasks

Three contributions are proposed in our work: the Stepwise Self-Knowledge Distillation (SSKD) training framework, the Stepwise Training Strategy, and the Weight Scaling Factor. We develop these designs on our proposed BTC-ResNet backbone and target optimization issues common in binary neural networks (BNNs). These designs operate on intermediate representations, batch-normalization behavior, and the magnitude gap introduced by binarization. No keyword-spotting-specific label structure or KWS-specific objective is required. Therefore, our proposed training framework is applicable to other BNN-based tasks after adapting the task head and the task objective. The student–self-generated-teacher distillation structure and the intermediate-layer constraints remain unchanged. This framework is well suited for transfer to other classification and recognition tasks with BNN backbones, such as speech recognition, computer vision, and biomedical signal analysis. Cross-task evaluation in these domains is part of our planned future work.

5.2. Interaction with Other Compression Techniques

This paper focuses on improving the training accuracy and optimization stability of binary neural networks (BNNs) through a Stepwise Self-Knowledge Distillation (SSKD) framework with a Stepwise Training Strategy and Weight Scaling Factor. The deployed model uses binarized weights and activations in $\{-1, +1\}$, and the teacher branch is enabled only during training and discarded at inference. Therefore, our evaluation is centered on the 1-bit setting and does not include additional compression pipelines such as pruning or multi-bit quantization in this submission.

Pruning typically introduces structured or unstructured sparsity by masking parameters, which effectively yields zeros and leads to a $\{-1, 0, +1\}$ -like weight pattern [57]. This setting is closely related to the ternary-weight formulations studied in prior work, where zero values represent disabled connections [58]. In principle, SSKD is largely orthogonal to pruning and could be combined with a pruning pipeline. However, a careful co-design is required because some objectives in SSKD assume matched feature dimensions between the teacher and the student for layer-wise feature alignment. If pruning changes channel widths or layer shapes, additional projection modules may be needed, which would materially modify the training pipeline and introduce new design choices. For these reasons, we do not include joint pruning–SSKD experiments in this submission, and we leave a systematic study of their interaction for future work.

Although our experiments focus on 1-bit BNNs, the distillation structure of SSKD and logit-level supervision for a low-precision student may be extendable to multi-bit quantization settings. Such an extension would require replacing sign-based operators and the magnitude-transfer design with counterparts compatible with k -bit quantizers, and re-validating the stability and accuracy behavior under the new quantization operator. We view this as a promising research direction, but it is beyond the experimental scope of this submission.

5.3. Real-World Deployment and Hardware Limitations

This paper focuses on algorithmic and training improvements. Therefore, all experiments are conducted in a controlled software setting. The inference-time parameter count and computational cost are provided. Both quantities are obtained via analytical derivation from the network architecture. We do not include a real-world KWS deployment on physical devices in this submission, since a full deployment requires hardware–software co-design, including a verified binarized compute unit, runtime integration, and end-to-end measurements of latency, power, and memory. Accordingly, in practical KWS deployments, real-world performance is often interpreted through the trade-off between false alarms and false rejects at calibrated operating points, rather than clip-level accuracy alone. A strict “physical-environment” comparison across mechanisms therefore typically requires a unified end-to-end streaming and device-level evaluation pipeline. As a result, cross-paper device-level results are often not directly comparable due to operating-point and implementation differences. These components require substantial engineering effort and a complete measurement pipeline on real devices. This paper emphasizes hardware-friendly efficiency from the algorithmic and network-structure perspectives. The parameter count and computational cost during inference are derived analytically from the network structure, and they can be verified with standard structure-based profilers. As future work, we plan to implement and validate a real binarized processing unit and an ASIC-level system to enable end-to-end evaluation under practical hardware constraints.

6. Conclusions

KWS implementation on resource-constrained device requires low storage and computational overhead while maintaining high accuracy. To solve this challenge, the Stepwise Self-Knowledge Distillation (SSKD) training framework, Stepwise Training Strategy, and Weight Scaling Factor are proposed. These training improvements enhance the student BNN’s optimization stability and representational capability, leading to higher accuracy. At the same time, our BTC-ResNet downsizes the student classifier’s parameters and computations efficiently. Experimental results on two GSCD(v1/v2) databases prove the effectiveness of our proposals. These studies are very helpful and promising for the future hardware KWS implementations on low-power platforms.

Author Contributions: Conceptualization, H.Z.; methodology, H.Z., J.Z. and H.R.; software, H.Z., J.Z. and H.R.; validation, H.Z., J.Z. and J.L.; formal analysis, H.Z., R.Z. and Z.Y.; investigation, H.Z., W.Y. and R.Z.; resources, Y.Z. and S.Q.; data curation, H.Z., J.Z. and W.Y.; writing—original draft preparation, H.Z.; writing—review and editing, J.Z., J.L., H.R., W.Y., R.Z., Z.Y., Y.Z. and S.Q.; visualization, H.Z.; supervision, Y.Z. and S.Q.; project administration, Y.Z. and S.Q.; funding acquisition, Y.Z. and S.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research and APC was funded by the Beijing Natural Science Foundation, grant number L244052.

Institutional Review Board Statement: Not applicable. The study did not involve humans or animals.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets analyzed in this study are publicly available. The Google Speech Commands Dataset (GSCD) Version 1 (speech_commands_v0.01) and Version 2 (speech_commands_v0.02) can be obtained from the official TensorFlow download server at http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz (accessed on 20 January 2026) and http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz (accessed on 20 January 2026), respectively. No new data were created in this study.

Acknowledgments: The authors would like to thank the Academic Editor and the anonymous reviewers for their careful evaluation and constructive comments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Peng, T.; Xiao, Y. Dark Experience for Incremental Keyword Spotting. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Hyderabad, India, 6–11 April 2025; pp. 1–5.
- Qin, H.; Ma, X.; Din, Y.; Li, X.; Zhang, Y.; Ma, Z.; Wang, J.; Luo, J.; Liu, X. BiFSMNv2: Pushing Binary Neural Networks for Keyword Spotting to Real-World Performance. *IEEE Trans. Neural Netw. Learn. Syst. (TNNLS)* **2024**, *35*, 10674–10686. [[CrossRef](#)] [[PubMed](#)]
- 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 (IJCAI), Vienna, Austria, 23–29 July 2022; pp. 4346–4352.
- Gao, T.; Zhou, Y.; Duan, S.; Hu, X. Memristive KDG-BNN: Memristive Binary Neural Networks Trained via Knowledge Distillation and Generative Adversarial Networks. *Knowl. Based Syst.* **2022**, *249*, 108962. [[CrossRef](#)]
- Tang, R.; Lin, J. Deep Residual Learning for Small-Footprint Keyword Spotting. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 5484–5488.
- Chen, G.; Parada, C.; Heigold, G. Small-Footprint Keyword Spotting Using Deep Neural Networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; pp. 4087–4091.
- Franco, B.L.; Mique, J.; Romdhane, A.; Prenat, G.; Anghel, L.; Novo, D.; Benoit, P. Trade-Offs in Neural Network Compression: Quantized and Binary Models for Keyword Spotting. In Proceedings of the 31st IEEE International Conference on Electronics, Circuits and Systems (ICECS), Nancy, France, 18–20 November 2024; pp. 1–4.
- Li, C.; Zhi, H.; Yang, K.; Qian, J.; Yan, Z.; Zhu, L.; Chen, C.; Wang, X.; Shan, W. A 0.61- μ W Fully Integrated Keyword-Spotting ASIC with Real-Point Serial FFT-Based MFCC and Temporal Depthwise Separable CNN. *IEEE J. Solid-State Circuits (JSSC)* **2024**, *59*, 867–877. [[CrossRef](#)]
- Yuan, C.; Agaian, S.S. A Comprehensive Review of Binary Neural Network. *Artif. Intell. Rev.* **2023**, *56*, 12949–13013. [[CrossRef](#)]
- Pouransari, H.; Tu, Z.; Tuzel, O. Least Squares Binary Quantization of Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; pp. 2986–2996.
- Zhang, B.; Xu, S.; Lin, M.; Wang, T.; Doermann, D. *Binary Neural Networks: Algorithms, Architectures, and Applications*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2023.
- Wang, Z.; Lu, J.; Tao, C.; Zhou, J.; Tian, Q. Learning Channel-Wise Interactions for Binary Convolutional Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 568–577.
- Guimaraes, H.R.; Pimentel, A.; Avila, A.; Falk, T.H. VIC-KD: Variance-Invariance-Covariance Knowledge Distillation to Make Keyword Spotting More Robust Against Adversarial Attacks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Republic of Korea, 14–19 April 2024; pp. 12196–12200.
- 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, Incheon, Republic of Korea, 18–22 September 2022; pp. 3233–3237.
- Zhang, Y.; Suda, N.; Lai, L.; Chandra, V. Hello Edge: Keyword Spotting on Microcontrollers. *arXiv* **2017**, arXiv:1711.07128.
- Choi, S.; Seo, S.; Shin, B.; Byun, H.; Kersner, M.; Kim, B.; Kim, D.; Ha, S. Temporal Convolution for Real-time Keyword Spotting on Mobile Devices. In Proceedings of the Interspeech, Graz, Austria, 15–19 September 2019.
- Zeng, K.; Wan, Z.; Gu, H.; Shen, T. Self-Knowledge Distillation Enhanced Binary Neural Networks Derived from Underutilized Information. *Appl. Intell.* **2024**, *54*, 4994–5014. [[CrossRef](#)]
- Ye, J.; Wang, J.; Zhang, S. Distillation-Guided Residual Learning for Binary Convolutional Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst. (TNNLS)* **2022**, *33*, 7765–7777. [[CrossRef](#)]
- Xie, Y.; Hou, X.; Guo, Y.; Wang, X.; Zheng, J. Joint-Guided Distillation Binary Neural Network via Dynamic Channel-Wise Diversity Enhancement for Object Detection. *IEEE Trans. Circuits Syst. Video Technol.* **2024**, *34*, 448–460. [[CrossRef](#)]
- Chen, X.; Liu, G.; Shi, J.; Xu, J.; Xu, B. Distilled Binary Neural Network for Monaural Speech Separation. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
- Zhang, L.; Bao, C.; Ma, K. Self-Distillation: Towards Efficient and Compact Neural Networks. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)* **2022**, *44*, 4388–4403. [[CrossRef](#)]
- Zhang, L.; Song, J.; Gao, A.; Chen, J.; Bao, C.; Ma, K. Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. *arXiv* **2019**, arXiv:1905.08094. [[CrossRef](#)]

23. Sainath, T.N.; Parada, C. Convolutional Neural Networks for Small-footprint Keyword Spotting. In Proceedings of the Interspeech, Dresden, Germany, 6–10 September 2015; pp. 1478–1482.
24. Coucke, A.; Chlieh, M.; Gisselbrecht, T.; Leroy, D.; Poumeyrol, M.; Lavril, T. Efficient keyword spotting using dilated convolutions and gating. *arXiv* **2019**, arXiv:1811.07684. [[CrossRef](#)]
25. Banbury, C.; Reddi, V.J.; Torelli, P.; Holleman, J.; Jeffries, N.; Kiraly, C.; Montino, P.; Kanter, D.; Ahmed, S.; Pau, D.; et al. MLPerf Tiny Benchmark. In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, Online, 6–14 December 2021.
26. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016.
27. Qin, H.; Zhang, X.; Gong, R.; Ding, Y.; Xu, Y.; Liu, X. Distribution-Sensitive Information Retention for Accurate Binary Neural Network. *Int. J. Comput. Vis.* **2022**, *131*, 26–47. [[CrossRef](#)]
28. Bulat, A.; Tzimiropoulos, G. XNOR-Net++: Improved Binary Neural Networks. In Proceedings of the 30th British Machine Vision Conference (BMVC), Cardiff, UK, 9–12 September 2019.
29. Liu, Z.; Wu, B.; Luo, W.; Yang, X.; Liu, W.; Cheng, K. Bi-Real Net: Enhancing the Performance of 1-Bit CNNs with Improved Representational Capability and Advanced Training Algorithm. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 722–727.
30. Liu, Z.; Shen, Z.; Savvides, M.; Cheng, K. ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020; pp. 143–159.
31. Wang, D.; Luo, M.; Li, L.; Wang, X.; Qiao, S.; Zhou, Y. A Binary Keyword Spotting System with Error-Diffusion Based Feature Binarization. In Proceedings of the Interspeech, Dublin, Ireland, 20–24 August 2023; pp. 1424–1428.
32. Riviello, A.; David, J.-P. Binary Speech Features for Keyword Spotting Tasks. In Proceedings of the Interspeech, Graz, Austria, 15–19 September 2019; pp. 3460–3464.
33. Cerutti, G.; Cavigelli, L.; Andri, R.; Magno, M.; Farella, E.; Benini, L. Sub-mW Keyword Spotting on an MCU: Analog Binary Feature Extraction and Binary Neural Networks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 2002–2012. [[CrossRef](#)]
34. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531. [[CrossRef](#)]
35. Romero, A.; Ballas, N.; Kahou, S.E.; Chassang, A.; Gatta, C.; Bengio, Y. FitNets: Hints for Thin Deep Nets. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
36. Song, Z.; Liu, Q.; Yang, Q.; Li, H. Knowledge distillation for In-memory keyword spotting model. In Proceedings of the Interspeech, Incheon, Republic of Korea, 18–22 September 2022; pp. 4128–4132.
37. Kim, J.; Chang, S.; Kwak, N. PQK: Model Compression via Pruning, Quantization, and Knowledge Distillation. In Proceedings of the Interspeech, Brno, Czech Republic, 30 August–3 September 2021; pp. 4568–4572.
38. 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, Dublin, Ireland, 20–24 August 2023; pp. 1623–1627.
39. Kim, D.; Kim, G.; Lee, B.; Ko, H. Prototypical Knowledge Distillation for Noise Robust Keyword Spotting. *IEEE Signal Process. Lett.* **2022**, *29*, 2298–2302. [[CrossRef](#)]
40. Yang, G.-P.; Gu, Y.; Macha, S.; Tang, Q.; Liu, Y. On-Device Constrained Self-Supervised Learning for Keyword Spotting via Quantization Aware Pre-Training and Fine-Tuning. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Republic of Korea, 14–19 April 2024; pp. 10951–10955.
41. Kerpici, M.; Nguyen, V.; Zhang, S.; Visser, E. Application of Knowledge Distillation to Multi-task Speech Representation Learning. *arXiv* **2022**, arXiv:2210.16611.
42. Phuong, M.; Lampert, C. Distillation-Based Training for Multi-Exit Architectures. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1355–1364.
43. Chen, Y.; Wang, S.; Liu, J.; Xu, X.; de Hoog, F.; Kusy, B.; Huang, Z. Understanding the Effects of Projectors in Knowledge Distillation. *arXiv* **2023**, arXiv:2310.17183. [[CrossRef](#)]
44. Miles, R.; Mikolajczyk, K. Understanding the Effects of Projectors in Knowledge Distillation. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 26–27 February 2024; pp. 4233–4241.
45. Ulkar, M.G.; Okman, O.E. Ultra-Low Power Keyword Spotting at the Edge. *arXiv* **2021**, arXiv:2111.04988. [[CrossRef](#)]
46. Jin, X.; Peng, B.; Wu, Y.; Liu, Y.; Liu, J.; Liang, D.; Yan, J.; Hu, X. Knowledge Distillation via Route Constrained Optimization. *arXiv* **2019**, arXiv:1904.09149. [[CrossRef](#)]
47. Rangamani, A.; Lindegaard, M.; Galanti, T.; Poggio, T.A. Feature Learning in Deep Classifiers through Intermediate Neural Collapse. In Proceedings of the International Conference on Machine Learning (ICML), Honolulu, HI, USA, 23–29 July 2023.
48. Chen, Z.; Zhang, M.; Cui, S.; Li, H.; Niu, G.; Gong, M.; Zhang, C.; Zhang, K. Neural Collapse Inspired Feature Alignment for Out-of-Distribution Generalization. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 9–15 December 2024.

49. Li, Q.; Meng, Y.; Tang, C.; Jiang, J.; Wang, Z. Investigating the Impact of Quantization on Adversarial Robustness. *arXiv* **2024**, arXiv:2404.05639. [[CrossRef](#)]
50. Schroff, F.; Kalenichenko, D.; Philbin, J. FaceNet: A Unified Embedding for Face Recognition and Clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 815–823.
51. Wen, Y.; Zhang, K.; Li, Z.; Qiao, Y. A Discriminative Feature Learning Approach for Deep Face Recognition. In Proceedings of the ECCV, Amsterdam, The Netherlands, 11–14 October 2016; pp. 499–515.
52. Qi, C.; Su, F. Contrastive-Center Loss for Deep Neural Networks. *arXiv* **2017**, arXiv:1707.07391. [[CrossRef](#)]
53. Chen, B.; Jiang, J.; Wang, X.; Wan, P.; Wang, J.; Long, M. Debiased Self-Training for Semi-Supervised Learning. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 28 November–9 December 2022.
54. Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or −1. *arXiv* **2016**, arXiv:1602.02830. [[CrossRef](#)]
55. Atabey, S.; Akagündüz, E. Binary SqueezeNet: Enhancing Parameter Efficiency in Binary Neural Networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Honolulu, HI, USA, 19–25 October 2025; pp. 4052–4061.
56. Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv* **2018**, arXiv:1804.03209. [[CrossRef](#)]
57. Zhu, C.; Han, S.; Mao, H.; Dally, W.J. Trained Ternary Quantization. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
58. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016.

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.