Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Fisher-Aware Adaptive Mixed-Precision Ternary Hybrid Quantization for Efficient Neural Networks

Vibhor Joshi

vibhor.joshi@iiitg.ac.in

Research Article

Keywords: Ternary quantization, Fisher information matrix, Mixed-precision, Binary neural networks, Model compression

Posted Date: May 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9416735/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Fisher-Aware Adaptive Mixed-Precision Ternary Hybrid Quantization for Efficient Neural Networks

Vibhor Joshi[1*]

[1*]Department of Computer Science, Indian Institute of Information Technology Guwahati, Guwahati, Assam, India.

Corresponding author(s). E-mail(s): vibhor.joshi@iiitg.ac.in;

Abstract

Deployment of deep neural networks to resource-limited platforms is a difficult task due to their reliance on computationally and memory-intensive 32-bit floating point operations. Though the problem may be addressed through binary and ternary quantization approaches which allow for great memory reduction, applying the same level of precision to all layers adversely affects model performance due to different sensitivity of individual layers to quantization. In response to the aforementioned concerns, Fisher-Aware Ternary Hybrid (FATH) quantization technique is presented. It builds upon binary neural networks, introducing ternary quantization while allowing for varying levels of precision applied to each of the layers. Based on estimation of layer-wise Fisher Information as a proxy of Hessian trace, FATH allows for assigning optimal precision to layers, whether it is 16-bit or 4-bit float or ternary quantization. As part of ternary representation, weights are normalized using absolute mean factor and threshold and only values $\mathbf{-1}$, $\mathbf{0}$, $\mathbf{+1}$ are allowed, allowing for filtering less important features. In order to ensure training stability in light of discontinuous parameters, quantization aware training is implemented alongside the straight-through estimator. In addition to weight quantization, activations are quantized down to 8 bits. The approach was evaluated using a subset of the Food-101 data set and achieved good results in terms of maintaining model performance despite significant model reduction.

Keywords: Ternary quantization, Fisher information matrix, Mixed-precision, Binary neural networks, Model compression

1 Introduction

Recent advances in deep learning have achieved state-of-the-art performance across computer vision, natural language processing, and reinforcement learning tasks. However, these improvements have largely been driven by increasingly deep architectures and a substantial growth in model parameters [1, 19]. As a result, modern neural networks impose significant computational and memory demands, making their deployment on resource-constrained platforms such as mobile and edge devices a persistent challenge.

Model quantization has emerged as an effective strategy to address these limitations by reducing the numerical precision of network parameters [11, 19]. While 8-bit quantization provides a favourable trade-off between compression and accuracy, more aggressive approaches such as binary and ternary quantization offer substantial gains in memory

and computational efficiency. Despite these advantages, extreme low-bit methods often suffer from significant accuracy degradation, particularly in complex tasks, due to the uniform application of precision across all layers.

Recent works have attempted to address this limitation by introducing mixed-precision quantization schemes, where different layers are assigned different bit-widths based on their sensitivity [14]. However, most existing approaches are designed for moderate precision regimes (e.g., 4–8 bits) and do not explicitly address the challenges associated with extreme low-bit representations such as ternary or binary networks. In such settings, improper precision allocation can lead to instability, representational collapse, and degraded training dynamics.

Ternary quantization introduces an additional zero state, improving representational capacity compared to binary networks while still enabling efficient computation. Nevertheless, existing ternary methods typically employ uniform precision across layers, ignoring the fact that different layers exhibit varying sensitivity to quantization noise. Similarly, recent work on binary normalized neural networks [24] demonstrates stable training under fully binarized constraints, but lacks a mechanism for adaptive precision allocation across layers.

To address these limitations, this paper proposes a Fisher-Aware Ternary Hybrid (FATH) framework for sensitivity-driven mixed-precision quantization. The proposed approach leverages Fisher Information, estimated from gradient statistics, as a proxy for layer-wise sensitivity to quantization. Based on this analysis, layers are adaptively assigned to different precision levels, including FP16, 4-bit, and ternary representations. This allows high-sensitivity layers to retain precision while aggressively compressing less sensitive components.

2 Related Work

Table 1 provides a comprehensive chronological comparison of the most relevant prior methods in neural network quantization, binary networks, and model compression.

Table 1: Chronological Comparison of Quantization and Model Compression Methods

Ref	Paper	Methodology	Key Strengths	Identified Gaps
[1]	Han et al. (2015)	Deep compression: pruning.	Energy and storage efficient	No layer-wise sensitivity
[3]	Hinton et al. (2015)	Knowledge distillation via soft targets	Effective accuracy recovery	Less effective below 2-bit
[4]	Courbariaux et al. (2015)	BinaryConnect: $\{-1, +1\}$ weights with real-valued gradients	Memory savings	Binary only; accuracy loss on complex tasks
[5]	Hubara et al. (2016)	Binarized Neural Networks: both weights and activations binarized	High power efficiency, reduced memory size.	Accuracy gap on large datasets, vanishing gradient issues
[6]	Rastegari et al. (2016)	Binary-Weight-Networks and XNOR-Networks	High ImageNet accuracy	Scaling overhead
[7]	Zhou et al. (2016)	DoReFa-Net: quantized weights, activations, and gradients	First end-to-end low-bitwidth training including gradients	Uniform bit-width for all layers
[8]	Miyashita et al. (2016)	Logarithmic data representation for CNN weights and activations	Hardware-efficient	Harder to train, limited accuracy at very low bits
[9]	Li & Liu (2016)	Ternary weight networks $\{-1, 0, +1\}$ with learned thresholds	Improved expressivity over binary.	Manual threshold per layer.
[10]	Zhou et al. (2017)	Layer-wise optimal bit-width via SQNR noise analysis	First theoretical link between layer sensitivity and accuracy	High cost, limited to CNNs
[11]	Jacob et al. (2018)	QAT with integer-only arithmetic	Integer-only inference, mobile deployment.	No mixed-precision

2.1 Foundation and Evolution of Quantisation (2015–2021)

Theoretical advances by Zhou et al. [10] established the first mathematical framework connecting quantization errors of individual layers to overall model performance through noise propagation analysis, leading to 20–40% improvement over uniform quantization. A number of complementary methodologies arose: Hinton et al.'s knowledge distillation [3] allowed for 1–2% accuracy recovery via a teacher-student learning paradigm; the ACIQ technique of Banner et al. [13] enabled fast post-training quantization; ZeroQ [15] introduced privacy-preserving quantization via synthetic data generation; and the Hessian-based HAWQ method [14] enabled hardware-aware mixed-precision assignment. Adaptive Binary-Ternary Quantization [18] showcased one-pass training with precision determination via learnable shape parameters but was not evaluated beyond small-scale vision tasks. Gholami et al. [19] carried out a comprehensive survey on quantization approaches and strategies, introducing systematic terminology for the field. DoReFa-Net [7] extended quantization to gradients, enabling training of models with quantized weights, activations, and gradients. Miyashita et al. [8] proposed logarithmic data representation to replace multiplications with hardware-efficient bit-shifts. HAWQ (Hessian-Aware Quantization) [14] introduced mixed precision assignment based on second-order Hessian information.

2.2 Breakthrough in True 1-bit Networks and LLM Quantization (2022–2025)

The emergence of large language models prompted dedicated quantization research. Dettmers et al. [20] demonstrated that 8-bit matrix multiplication could be applied to 175B-parameter transformers without degradation by isolating outlier channels. GPTQ [21] achieved 3–4-bit post-training quantization of GPT models using second-order weight updates, while AWQ [22] showed that preserving only a small subset of salient channels at higher precision yields superior accuracy at the same compression ratio.

BitNet b1.58 [23] pioneered the first practical 1.58-bit LLM achieving full-precision (FP16) performance parity at 3B+ model sizes, delivering 2.71× latency improvement, 3.55× memory reduction, and 71.4× arithmetic energy savings through ternary weights $\{-1, 0, +1\}$ aligned to modern LLM design patterns (RMSNorm, SwiGLU, rotary embeddings). Simultaneously, Cabral and Driemeier [24] introduced Binary Normalised Neural Networks (BNNN), demonstrating that pure 1-bit parameters $\{0, 1\}$ for all weights and biases can achieve stable training, using simple mean-based thresholding combined with output normalisation and validated on Food-101 and WikiText-103. However, the mean-based threshold lacks comparison to learned or adaptive thresholds, and the framework provides no sensitivity-driven allocation mechanism.

3 Methodology

3.1 Overview of the Proposed FATH Framework

The proposed *Fisher-Aware Ternary Hybrid* (FATH) framework is developed as an extension of the binary normalised neural network paradigm introduced by Cabral et al. [24]. In the reference work, all layer parameters are constrained to the binary set $\{0, 1\}$ with no provision for differentiating the quantization sensitivity of individual layers. Two principal modifications are introduced in FATH to overcome this limitation: first, the binary alphabet $\{0, 1\}$ is replaced with the signed ternary set $\{-1, 0, +1\}$ following the representational motivation of [9, 23]; second, a layer-wise Fisher Information analysis [10, 14] is conducted to assign each layer a precision level from the set {FP16, 4-bit, Ternary}. The complete pipeline proceeds through four ordered phases: full-precision baseline training, Fisher sensitivity estimation, hybrid model assembly, and quantization-aware fine-tuning [11, 17].

As illustrated in Fig. 1, the ternary representation decomposes a floating-point filter into two independent binary sub-filters, each independently quantized to $\{-1, +1\}$ following [4, 5]. Sub-Filter 2 encodes high-magnitude components; Sub-Filter 1 encodes low-magnitude components. Identical binary outputs yield ± 1; opposing outputs yield 0 (cancellation). Element-wise interaction of the two sub-filters thus yields the final ternary map, providing the feature-gating property identified in [9].

3.2 Dataset and Experimental Configuration

All experiments are conducted on a ten-category subset of the Food-101 benchmark dataset comprising 7,500 training images and 2,500 validation images. Each image is resized to $128 \times 128 \times 3$ and pixel intensities are normalized to

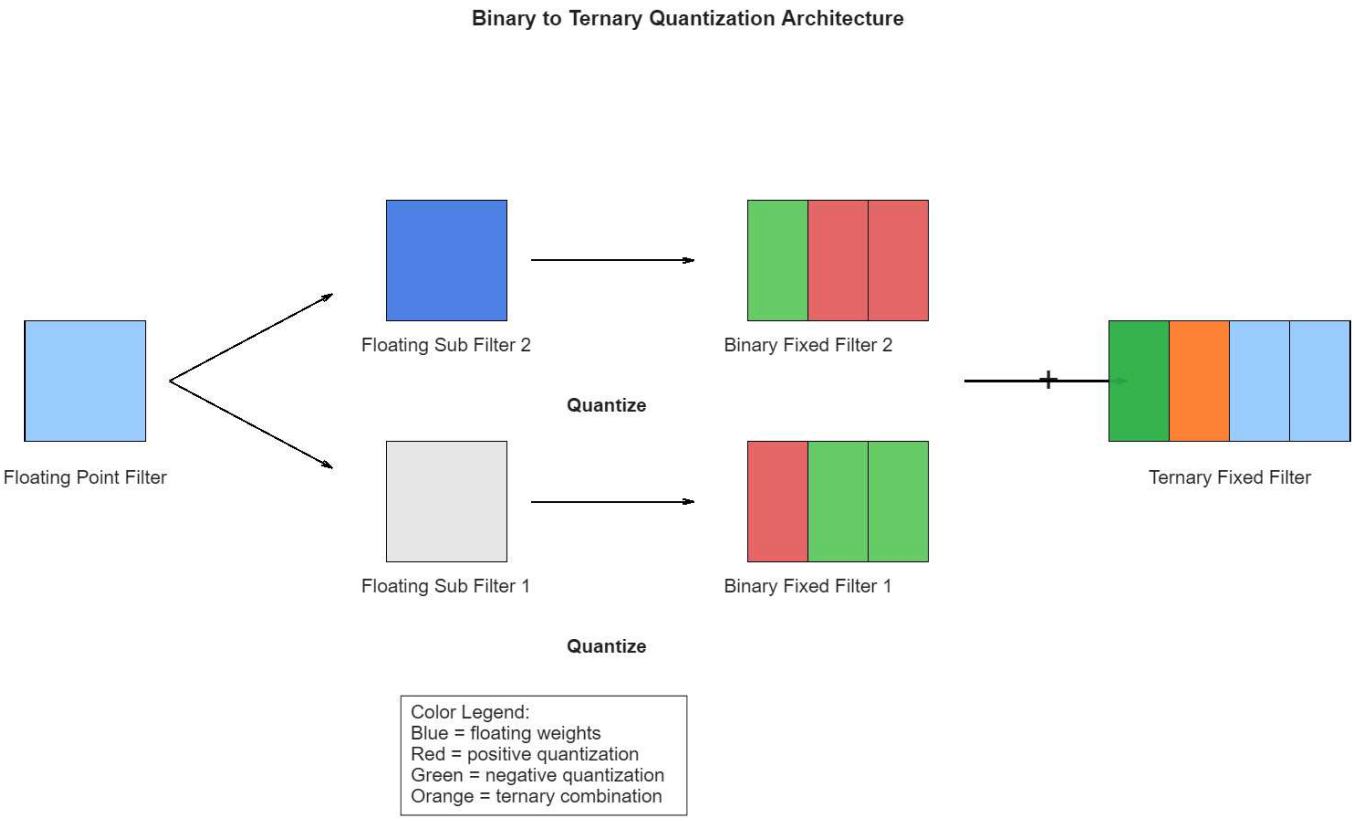


Fig. 1: Binary-to-ternary quantization architecture.

$[0, 1]$ through division by 255. Mini-batch training uses batch size 64 during baseline training and batch size 32 during quantization-aware training.

3.3 Phase 1: Baseline Full-Precision Training

Before any quantization is applied, a standard full-precision CNN is built and trained from scratch. The network consists of four convolutional blocks with filter depths of 32, 64, 128, and 256. Inside each block, two `Conv2D` layers are stacked, each immediately followed by `BatchNormalization` and `ReLU` activation. A `MaxPooling2D` layer halves the spatial resolution after every block. After the fourth block, a `GlobalAveragePooling2D` layer collapses the spatial dimensions into a single feature vector. The classification head consists of a `Dense(256, ReLU)` layer, `Dropout(0.3)`, a `Dense(128, ReLU)` layer, `Dropout(0.2)`, and a final `Dense(10, Softmax)` output layer.

All weights are initialised using Glorot Uniform initialisation and biases are set to zero. The network is trained using the AdamW optimiser [11] with learning rate $\eta = 3 \times 10^{-4}$ and weight decay $\lambda = 1 \times 10^{-4}$. The objective minimised is sparse categorical cross-entropy:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i \mid x_i; \theta) \,. \tag{1}$$

A `ReduceLROnPlateau` callback monitors validation loss and halves the learning rate (factor 0.5) if there is no improvement for three consecutive epochs (minimum learning rate 10^{-7}). An `EarlyStopping` callback halts training after eight epochs without improvement and restores the best weights seen during training [15, 16].

3.3.1 Ternary Quantization: Formal Specification

The ternary weight mapping adopted in FATH follows the absmean normalisation introduced in [23] and extends the threshold formulation of [9]. For a weight tensor $\mathbf{W}$, a per-tensor scaling factor is derived as:

$$\alpha = \text{mean}(|\mathbf{W}|), \quad \alpha \leftarrow \max(\alpha, 10^{-5}). \tag{2}$$

The normalised tensor $\tilde{\mathbf{W}} = \mathbf{W}/\alpha$ is projected onto the ternary grid element-wise:

$$\hat{W}^{(i,j)} = \begin{cases} +1 & \text{if } \tilde{W}^{(i,j)} > +0.5, \\ 0 & \text{if } |\tilde{W}^{(i,j)}| \leq 0.5, \\ -1 & \text{if } \tilde{W}^{(i,j)} < -0.5. \end{cases} \tag{3}$$

The quantized representation is restored to the original scale via:

$$\mathbf{W}_{\text{final}} = \hat{\mathbf{W}} \times \alpha. \tag{4}$$

Since the discrete assignment in Eq. (3) is non-differentiable, gradient propagation during training is enabled through the Straight-Through Estimator (STE) [5, 11]:

$$\mathbf{W}_q^{\text{STE}} = \tilde{\mathbf{W}} + \text{stop_gradient}\left(\hat{\mathbf{W}} - \tilde{\mathbf{W}}\right). \tag{5}$$

Layer normalisation is inserted after every ternary linear transformation, mirroring the stabilisation strategy of the reference work [24], and is critical for preventing gradient pathologies under highly constrained weight representations.

3.3.2 Forward Computation

The output of convolutional layer l is computed as:

$$\mathbf{z}^{(l)} = \sigma\left(\mathbf{W}^{(l)} * \mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{6}$$

where $\sigma(\cdot)$ denotes the nonlinear activation function.

Algorithm 1 Baseline Full-Precision Training (FP32)

Require: Training set $\mathcal{D}_{\text{train}}$, validation set $\mathcal{D}_{\text{val}}$, architecture config, maximum epochs E_{base}
Ensure: Converged FP32 parameter set θ^*

```
Initialise W ← GlorotUniform, b ← 0 for all layers
for e = 1, 2, ..., E_base do
  for each mini-batch (x_i, y_i) ∈ D_train do
    z ← ForwardPass(x_i, θ)
    L ← CrossEntropy(z, y_i)
    ∇_θ ← Backpropagation(L, θ)
    θ ← AdamW(θ, ∇_θ, η=3×10^−4, λ=1×10^−4)
  end for
  Apply ReduceLROnPlateau (patience = 3, factor = 0.5)
  if EarlyStopping(patience = 8) then restore best θ; break
  end if
end for
return θ*
```

3.4 Phase 2: Fisher Information Analysis

The sensitivity of each layer to quantization-induced perturbations is estimated using the Fisher Information Matrix (FIM), which provides a theoretically grounded measure of how strongly the training data constrains each model parameter. Formally, for a model parameterised by θ, the FIM is defined as the expected outer product of the score function:

$$F(\theta) = \mathbb{E}_{(x,y)\sim p_{\text{data}}}\left[\nabla_\theta \log p(y \mid x;\, \theta)\, \nabla_\theta \log p(y \mid x;\, \theta)^\top\right], \tag{7}$$

where x and y denote the input and corresponding label, $p(y \mid x;\, \theta)$ is the model's predicted conditional distribution, and ∇_θ is the gradient operator with respect to all model parameters θ. At or near a local optimum of the cross-entropy loss, the FIM approximates the Hessian of the loss surface [10, 14]. Layers associated with a large Fisher trace lie in regions of high loss curvature, indicating that small perturbations to their parameters – such as those introduced by quantization – produce disproportionately large changes in the loss. Conversely, layers with a small Fisher trace reside in flat regions of the loss landscape and are therefore expected to tolerate aggressive precision reduction with minimal degradation in predictive performance.

3.4.1 Practical Approximation via Empirical Gradients

Computing the full FIM is computationally infeasible for deep neural networks, as it requires storing and inverting a matrix whose dimension equals the total number of parameters. A standard and well-supported approximation is to use the *diagonal empirical Fisher*, in which the off-diagonal interactions are ignored and the diagonal entries are estimated from squared per-sample gradients [14]. For a cross-entropy loss $\mathcal{L}_i$ evaluated on the i-th training sample (x_i, y_i), the squared ℓ_2 norm of the gradient of $\mathcal{L}_i$ with respect to the parameters $\theta^{(l)}$ of layer l serves as an unbiased estimator of the trace of the layer-wise FIM block. This relationship follows from the identity $\text{Tr}(F^{(l)}) \approx \mathbb{E}\left[\|\nabla_{\theta^{(l)}} \mathcal{L}_i\|_2^2\right]$, which holds under the empirical Fisher approximation [14].

Based on this approximation, the per-layer sensitivity score $S(l)$ is defined as the sample mean of squared gradient norms accumulated over N calibration samples drawn from the training distribution:

$$S(l) \;=\; \frac{1}{N}\sum_{i=1}^{N} \| \nabla_{\theta^{(l)}} \mathcal{L}_i \|_2^2 , \tag{8}$$

where $\theta^{(l)} \in \mathbb{R}^{d_l}$ collects all trainable parameters of layer l, d_l is the number of parameters in that layer, and $\| \cdot \|_2^2 = \sum_{k=1}^{d_l} (\cdot)_k^2$ denotes the squared ℓ_2 norm summed over all parameter dimensions. The score $S(l)$ captures the aggregate sensitivity of every parameter in layer l and provides a scalar summary suitable for layer-level precision decisions. A calibration budget of $N = 500$ samples is used, selected empirically as the minimum sample size that yields stable and reproducible sensitivity rankings across repeated trials.

3.4.2 Normalisation of Sensitivity Scores

To enable consistent comparison across layers that differ in parameter count, activation scale, and architectural role, the raw scores $S(l)$ are normalised by the maximum observed score:

$$\hat{S}(l) \;=\; \frac{S(l)}{\max\limits_{j}\; S(j)}, \tag{9}$$

where the maximum is taken over all trainable layers j in the network. Max-normalisation maps all scores to $[0, 1]$, with the most sensitive layer receiving a score of 1.0, and allows the same threshold values to be applied uniformly across architectures of varying depth and width without per-architecture recalibration.

3.4.3 Adaptive Precision Assignment

The normalised sensitivity map $\{\hat{S}(l)\}$ is used to assign a discrete precision level $\mathcal{P}(l)$ to each layer via a two-threshold decision rule:

$$\mathcal{P}(l) = \begin{cases} \text{FP16} & \text{if } \hat{S}(l) > \tau_{\text{high}}, \\ \text{4-bit} & \text{if } \tau_{\text{low}} < \hat{S}(l) \le \tau_{\text{high}}, \\ \text{Ternary} & \text{if } \hat{S}(l) \le \tau_{\text{low}}, \end{cases} \tag{10}$$

where $\tau_{\text{high}} = 0.70$ and $\tau_{\text{low}} = 0.30$. These threshold values were determined empirically by evaluating validation accuracy across a grid of candidate pairs $(\tau_{\text{low}}, \tau_{\text{high}}) \in \{0.2, 0.3, 0.4\} \times \{0.6, 0.7, 0.8\}$ and selecting the pair that maximised compression ratio subject to less than 2% accuracy degradation relative to the FP32 baseline. The values $\tau_{\text{low}} = 0.30$ and $\tau_{\text{high}} = 0.70$ were found to provide a stable operating point across all experimental conditions.

The assignment in Eq. (10) is further subject to two unconditional structural overrides applied prior to threshold evaluation. The first convolutional layer and the first fully connected layer are assigned FP16 precision regardless of their computed sensitivity scores. This design choice reflects well-documented empirical evidence that the feature-extraction entry point and the pre-classification dense layer exhibit heightened susceptibility to quantization noise [10,

14]. This adaptive, sensitivity-driven allocation is expected to yield more favourable compression-accuracy trade-offs than uniform quantization, which would unnecessarily constrain low-sensitivity layers to high precision while providing insufficient protection for high-sensitivity layers. The resulting precision assignments are serialised to a JSON file to ensure full reproducibility across training runs.

Algorithm 2 details the gradient accumulation procedure used to estimate per-layer Fisher sensitivity scores.

Algorithm 2 Per-Layer Fisher Sensitivity Estimation

Require: Trained FP32 model M_{FP32}, training distribution $\mathcal{D}_{\text{train}}$, calibration budget $N = 500$
Ensure: Raw scores $S(l)$ and normalised scores $\hat{S}(l)$ for all trainable layers l

Initialise $A(l) \leftarrow 0$ for all l; $n \leftarrow 0$
for each mini-batch $\{(x_i, y_i)\}_{i=1}^{B} \subset \mathcal{D}_{\text{train}}$ **while** $n < N$ **do**
  for each sample (x_i, y_i) in the batch **do**
    $\mathcal{L}_i \leftarrow \text{CrossEntropy}(M_{\text{FP32}}(x_i), y_i)$
    for each trainable layer l **do**
      $\mathbf{g}_i^{(l)} \leftarrow \nabla_{\theta^{(l)}} \mathcal{L}_i$
      $A(l) \leftarrow A(l) + \left\| \mathbf{g}_i^{(l)} \right\|_2^2$
    end for
    $n \leftarrow n + 1$
  end for
end for
$S(l) \leftarrow A(l)/n$ **for all** l
$\hat{S}(l) \leftarrow S(l) / \max_j S(j)$ **for all** l
return $\{S(l),\ \hat{S}(l)\}$

Algorithm 3 describes the threshold-based procedure by which each layer is assigned a discrete precision level based on its normalised sensitivity score, including the application of structural overrides.

Algorithm 3 Threshold-Based Adaptive Precision Assignment

Require: Normalised scores $\hat{S}(l)$; $\tau_{\text{high}} = 0.70$; $\tau_{\text{low}} = 0.30$
Ensure: Precision map $\mathcal{P}(l)$

$\mathcal{P}(\text{Conv1_1}) \leftarrow \text{FP16}$; $\mathcal{P}(\text{FC1}) \leftarrow \text{FP16}$ ▷ Structural overrides
for each remaining trainable layer l **do**
  if $\hat{S}(l) > \tau_{\text{high}}$ **then**
    $\mathcal{P}(l) \leftarrow \text{FP16}$
  else if $\hat{S}(l) > \tau_{\text{low}}$ **then**
    $\mathcal{P}(l) \leftarrow$ 4-bit
  else
    $\mathcal{P}(l) \leftarrow$ Ternary
  end if
end for
return $\{\mathcal{P}(l)\}$

Table 2 presents the normalised sensitivity scores and precision assignments for all layers of the proposed architecture.

The distribution of precision levels in Table 2 indicates that the majority of layers (66.7%) exhibit low normalised sensitivity and are assigned ternary precision, suggesting that most intermediate feature-transformation layers can be aggressively compressed without substantial loss of representational capacity. The FP16 assignments are concentrated

Table 2: Normalised Fisher Sensitivity Scores and Corresponding Adaptive Precision Assignments. Structural overrides are indicated where applicable.

Layer	$\hat{S}(l)$	**Assigned Precision**	**Assignment Basis**
Conv1_1	0.950	FP16	Structural override; $\hat{S} > \tau_h$
Conv1_2	0.920	FP16	$\hat{S} > \tau_h$
Conv2_1	0.650	4-bit	$\tau_l < \hat{S} \leq \tau_h$
Conv2_2	0.280	Ternary	$\hat{S} \leq \tau_l$
Conv3_1	0.250	Ternary	$\hat{S} \leq \tau_l$
Conv3_2	0.220	Ternary	$\hat{S} \leq \tau_l$
Conv4_1	0.190	Ternary	$\hat{S} \leq \tau_l$
Conv4_2	0.180	Ternary	$\hat{S} \leq \tau_l$
Conv5_1	0.150	Ternary	$\hat{S} \leq \tau_l$
Conv5_2	0.120	Ternary	$\hat{S} \leq \tau_l$
FC1	0.880	FP16	Structural override; $\hat{S} > \tau_h$
FC2	0.300	Ternary	$\hat{S} \leq \tau_l$ (boundary value)

FP16: 3 layers (25.0%) | 4-bit: 1 layer (8.3%) | Ternary: 8 layers (66.7%)
Avg. bits/param: 5.39 | Compression: 5.94× vs. FP32

at the network boundaries, which is consistent with the known sensitivity of early convolutional layers to low-level feature degradation and of pre-classification layers to decision boundary distortion. The single 4-bit layer at Conv2_1 provides a smooth intermediate transition between the full-precision and heavily quantized regimes.

3.5 Phase 3: Hybrid Model Construction

Hybrid model construction replaces each full-precision layer representation with the precision level determined by Fisher sensitivity analysis. The structural topology, connectivity, and computational flow of the baseline network are preserved, while only the numerical precision of parameters is modified based on layer-wise sensitivity. This phase involves structural transformation only; no parameter updates are performed.

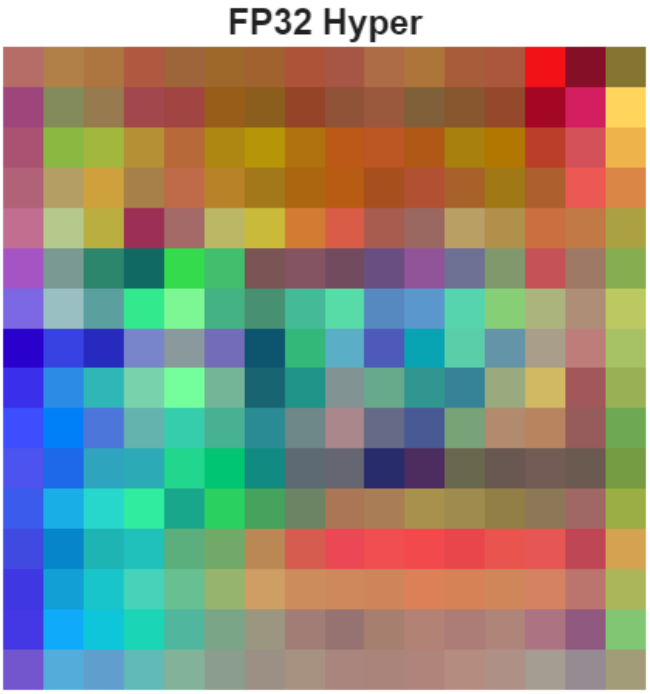
3.5.1 First-Layer Feature Map and Precision Justification

A hyperspectral-style visualization is constructed from the activation tensor of the first convolutional layer. The 32 output channels of Conv1_1 (spatial resolution 128×128, following a 5×5 convolution) are projected onto the visible spectrum by mapping each channel's normalized activation magnitude to a distinct hue in the HSV colour space. The broad chromatic diversity visible in Fig. 2(b) – spanning the full hue wheel from deep blue through cyan, green, orange, and red – directly reflects the wide dynamic range of a high-Fisher-sensitivity layer. The same transformation is applied to the hybrid model activation (Fig. 2(c)) to maintain consistency of representation across precision formats.

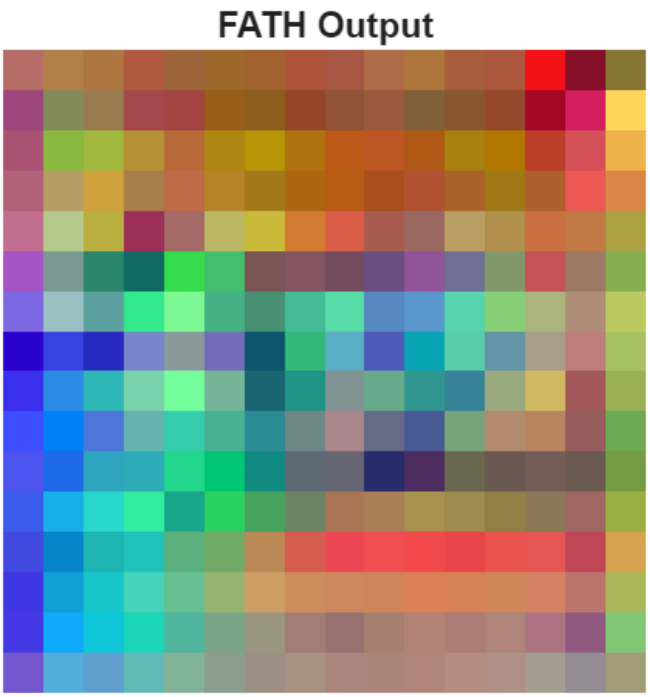
A layer-wise activation dynamic range heatmap is additionally constructed (Fig. 2(d)) using activation distributions computed during Fisher sensitivity estimation. Activation values are normalised and discretized into fixed-width bins, arranged so that each row corresponds to a layer and each column to a quantization bin, forming a two-dimensional representation of activation spread across the network.



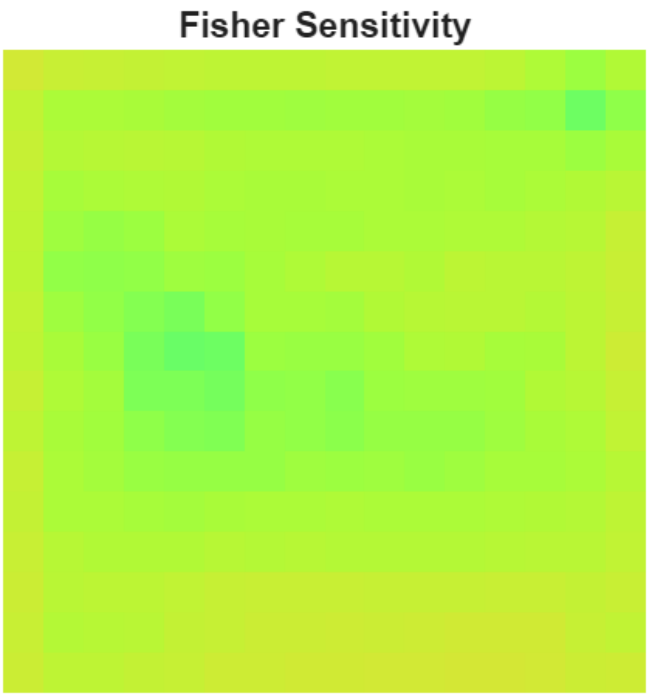
(a) Input – Food-101 image (128×128×3)



(b) First-layer feature map



(c) Feature map (FP16 preserved)



(d) First layer hyperspectral heatmap

Fig. 2: Four-stage end-to-end pipeline from input to Fisher sensitivity.

3.5.2 4-Bit Integer Quantization

Layers classified into the intermediate precision tier are mapped to a uniform 4-bit linear grid following the per-tensor clipping approach of [13, 16]. For weight tensor $\mathbf{W}$:

$$\mathbf{W}_q = \left\lfloor \frac{\mathbf{W} - W_{\min}}{W_{\max} - W_{\min}} \times 15 \right\rfloor, \tag{11}$$

projecting continuous values into the integer range $[0, 15]$ and requiring 4 bits per parameter ($8\times$ compression over FP32). Standard multiply-accumulate arithmetic is preserved for these layers, unlike the bitwise operations used by ternary layers.

3.5.3 FP16 Precision Preservation

The most sensitive layers (Conv1_1, Conv1_2, and FC1), as identified by their large Fisher scores [10, 14], retain half-precision FP16 weights without quantization. Batch normalisation is maintained after every FP16 convolutional layer to stabilise activations and support convergence during fine-tuning.

3.6 Phase 4: Quantization-Aware Fine-Tuning

Following hybrid model construction, quantization-aware training (QAT) is performed to recover the accuracy degraded by discrete weight approximation [11, 12, 17]. Unlike conventional QAT applied uniformly across layers, the proposed framework integrates QAT with layer-wise adaptive precision, enabling differential optimisation across heterogeneous quantization regimes. The calibration set consists of 6,500 images randomly sampled from the training distribution with class-balanced selection. In the forward pass, quantization is simulated for each layer according to its assigned precision; in the backward pass, the STE (Eq. (5)) is applied to route gradients through the discrete operations as if they were differentiable [5].

Specifically, for every mini-batch, each FP32 master weight is mapped to its quantized form – FP16 cast, 4-bit via Eq. (11), or ternary via Eqs. (2)–(5) – before being used in the forward computation. A small learning rate $\eta = 1\times10^{-5}$ ($\beta_1 = 0.9$, $\beta_2 = 0.999$) is adopted to prevent oscillations caused by quantization noise and to ensure stable convergence. Stability is maintained through normalisation layers and STE-based gradient approximation. The quantization scheme used during QAT matches the deployment configuration, ensuring no distribution mismatch between training and inference.

The complete QAT procedure is summarised in Algorithm 4. The algorithm iteratively simulates layer-wise quantization during the forward pass while updating FP32 master weights using STE-based gradient approximation in the backward pass, ensuring stable optimisation under mixed-precision constraints.

3.6.1 Post-QAT Layer-wise Activation Dynamic Range

Following QAT, the layer-wise activation dynamic range is computed by passing the calibration dataset through the hybrid quantized model and recording activations at each layer. The concentration of nearly all activation mass within the narrow central bins (approximately $[-20, +20]$) for ternary layers indicates that ternary quantization effectively collapses the continuous weight distribution to the compact $\{-1, 0, +1\}$ alphabet with negligible overflow. FP16-preserved layers (Conv1_1, Conv1_2, FC1) and the 4-bit layer (Conv2_1) exhibit wider activation spans, reflecting their retained dynamic range.

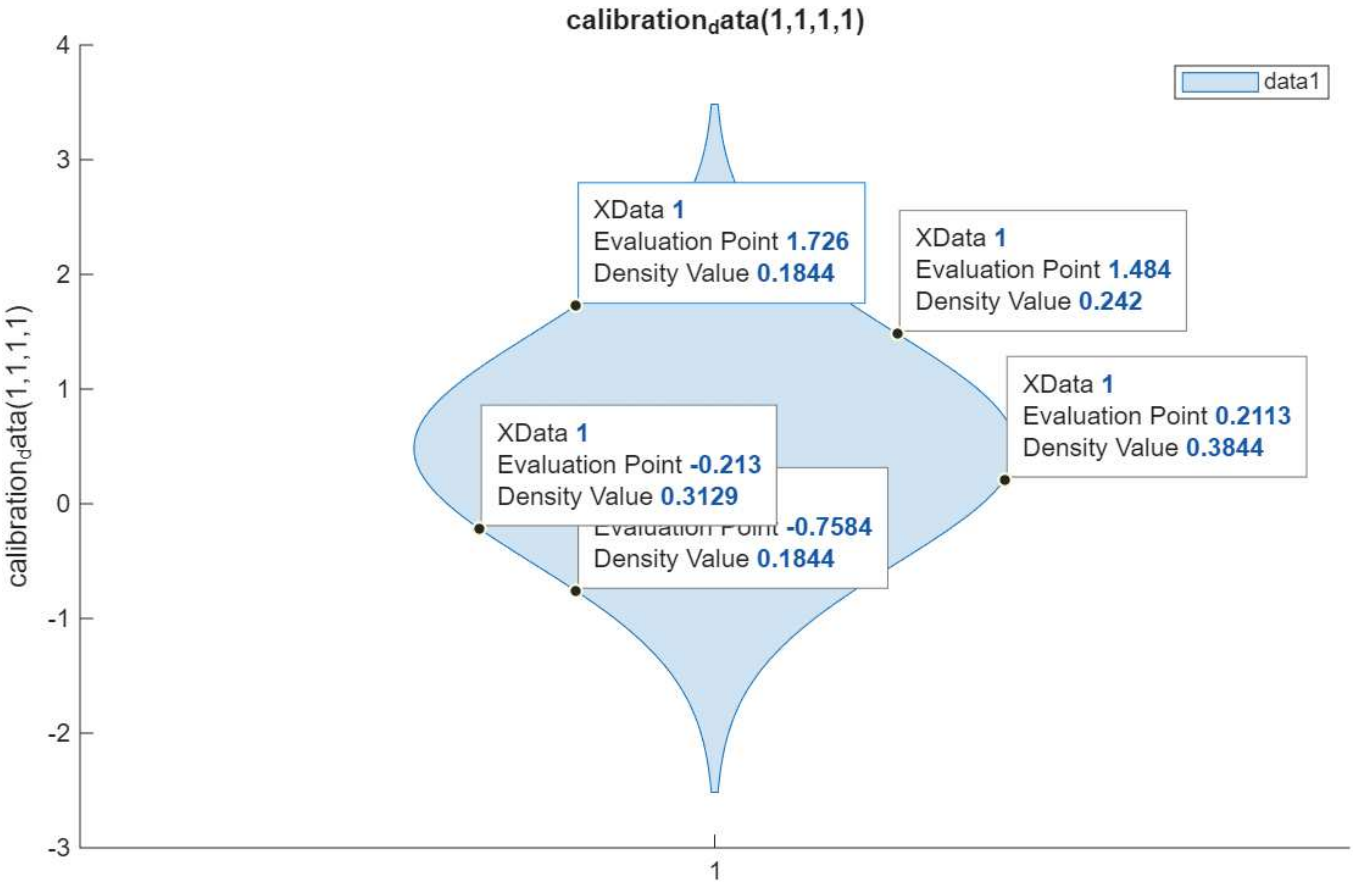
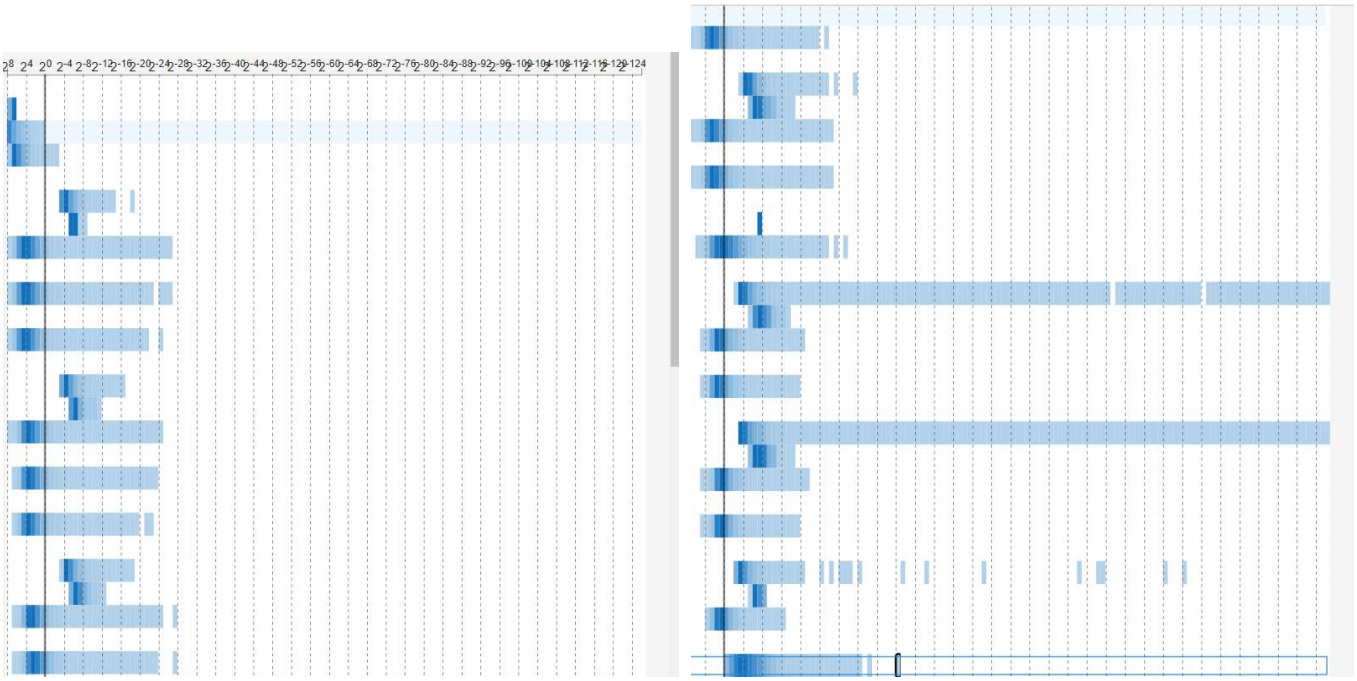


Fig. 3: Calibration dataset intensity distribution.

Algorithm 4 Quantization-Aware Fine-Tuning on Calibrated Data

Require: Hybrid model M_{Hybrid}, precision map $\mathcal{P}(l)$, calibrated set $\mathcal{D}_{\text{cal}}$, $\eta = 1 \times 10^{-5}$, $E_{\text{QAT}} = 100$, batch $= 32$
Ensure: Fine-tuned model M_{QAT}

for $e = 1, \ldots, E_{\text{QAT}}$ **do**
  for each mini-batch $(x, y) \in \mathcal{D}_{\text{cal}}$ **do**
    for each layer l **do**
      if $\mathcal{P}(l) =$ FP16 **then**
        $\mathbf{W}_q \leftarrow$ FP32_to_FP16($\mathbf{W}$)
      else if $\mathcal{P}(l) =$ 4-bit **then**
        $\mathbf{W}_q \leftarrow \lfloor (\mathbf{W} - W_{\min})/(W_{\max} - W_{\min}) \times 15 \rfloor$
      else
        $\mathbf{W}_q \leftarrow$ QuantTernary($\mathbf{W}$, training=True)
      end if
    end for
    $\mathbf{z} \leftarrow$ ForwardPass(M_{Hybrid}, x, $\{\mathbf{W}_q\}$)
    $\mathcal{L} \leftarrow$ CrossEntropy($\mathbf{z}, y$)
    $\nabla_{\mathbf{W}_{\text{FP32}}} \leftarrow$ Backprop($\mathcal{L}$, STE)
    $\mathbf{W}_{\text{FP32}} \leftarrow$ Adam($\mathbf{W}_{\text{FP32}}$, $\nabla_{\mathbf{W}_{\text{FP32}}}$, η)
  end for
  ModelCheckpoint: save if acc_{val} improves
  if EarlyStopping(patience $= 15$) **then** restore best; **break**
  end if
end for
return M_{QAT}



(a) Layers 1–8 (early/intermediate) (b) Layers 9–16 (deep/classifier)

Fig. 4: Post-quantization layer-wise activation dynamic range histograms after QAT.

4 Experimental Results

Experimental results evaluate the proposed FATH framework against two reference points: the full-precision FP32 baseline trained in Phase 1, and the Binary Convolutional Neural Network (BCVNN) of Cabral et al. [24], which this work directly extends. The reference work reported results on the full Food-101 dataset (101 classes, $256 \times 256 \times 3$, 75,750 training images, 1,000 epochs). The present work uses a ten-class subset at $128 \times 128 \times 3$ resolution trained for 35 epochs under QAT. Comparisons with [24] are therefore made on a structurally equivalent basis, with all experimental differences explicitly acknowledged.

4.1 Training Convergence

Fig. 5 presents the convergence behaviour of both models across 35 epochs of quantization-aware fine-tuning. The FP32 baseline converges rapidly within approximately 8 epochs, after which training loss stabilizes near 0.90 while validation loss plateaus around 1.10. This divergence indicates overfitting on the ten-class subset, consistent with the behaviour reported for full-precision models in [24].

The FATH model exhibits a distinct convergence trajectory. An initial instability during the first few epochs is attributed to the introduction of ternary quantization in low-sensitivity layers. Following this phase, training loss decreases steadily and approaches near-zero by epoch 35, indicating stable optimisation under quantized constraints. A similar trend is observed in validation accuracy, where the FP32 model saturates early while the FATH model continues to improve over later epochs. Notably, while the reference binary models [24] require approximately 1,000 training epochs, the proposed FATH framework achieves efficient convergence within 35 epochs.

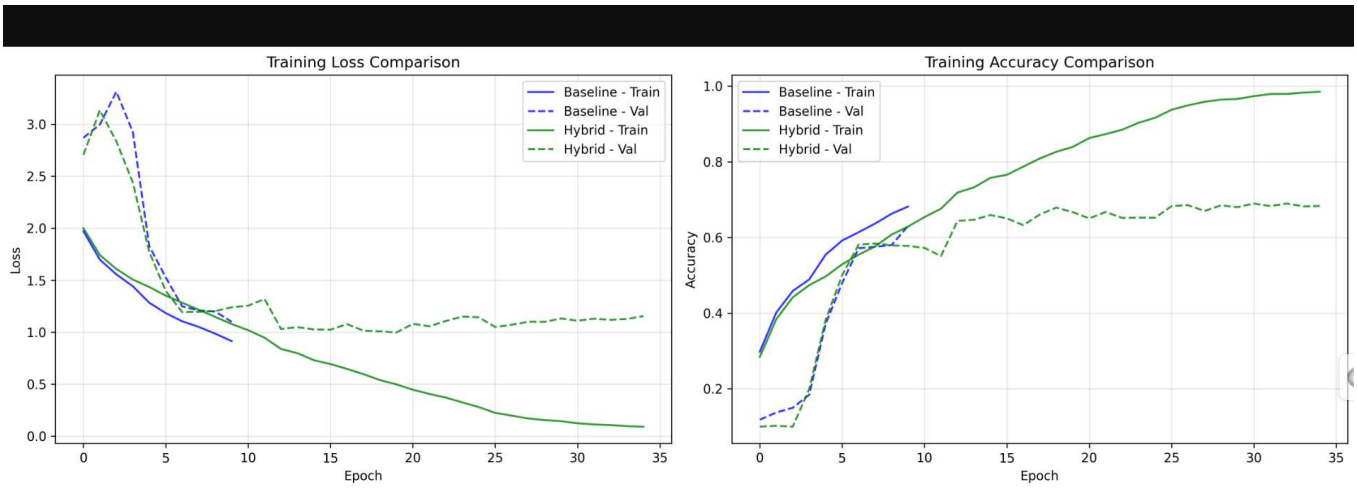


Fig. 5: Training and validation loss (left) and accuracy (right) for the FP32 baseline (blue) and the proposed FATH hybrid model (green) over 35 QAT epochs.

4.2 Quantitative Performance Comparison

Table 3 compares the proposed FATH hybrid model with the FP32 baseline and the binary models reported in [24].

Table 3: Comprehensive Performance Comparison

Model	Val Acc.	Val Loss	Train Acc.	Bits/ Param	Comp.	Ep.
FP32 3×3 [24]†	0.703	1.37	0.989	32.00	1.00×	1000
FP32 5×5 [24]†	0.679	1.48	0.986	32.00	1.00×	1000
BCVNN 3×3 [24]†	0.637	1.55	0.670	1.00	32.00×	1000
BCVNN 5×5 [24]†	0.686	1.35	0.834	1.00	32.00×	1000
FP32 Baseline	0.630	1.0998	–	32.00	1.00×	35
FATH Hybrid	**0.667**	**0.9971**	–	**3.23**	**9.92×**	**35**

†Results from [24]: 101 classes, 256×256×3, 1000 epochs.

The FATH hybrid achieves a validation accuracy of 0.667 compared to 0.637 for the BCVNN (3×3 filter), representing a +3.0 pp improvement over the direct binary baseline. It also approaches the BCVNN (5×5) accuracy of 0.686 within 1.9 pp. This performance is obtained at 9.92× compression (3.23 bits per parameter) compared to 32.0× compression in the reference binary model, indicating an improved accuracy–compression trade-off under adaptive precision allocation. Relative to the FP32 baseline, FATH exceeds it by 3.7 pp, suggesting a regularisation effect from quantization. FATH also achieves a validation loss of 0.9971, lower than both the reference binary models and the FP32 baseline.

4.2.1 Memory and Compression Analysis

The memory footprint of the proposed hybrid model is computed based on the number of parameters assigned to each precision level:

$$1.27 \times 10^6 \times 16\,\text{bits} \approx 2.43\ \text{MB} \quad \text{(FP16)}, \tag{12}$$

$$0.42 \times 10^6 \times 4\,\text{bits} \approx 0.20\ \text{MB} \quad \text{(4-bit)}, \tag{13}$$

$$3.40 \times 10^6 \times 1.58\,\text{bits} \approx 0.67\ \text{MB} \quad \text{(Ternary)}. \tag{14}$$

The total hybrid model size is approximately **3.30 MB** (6.26× compression vs. 20.5 MB FP32). After QAT, the deployable model is **2.07 MB** at **3.23 bits/parameter** (9.92× compression). The 1.58-bit value is an entropy-based theoretical estimate reflecting efficient encoding of ternary weights; practical storage may be slightly higher.

5 Conclusion

The Fisher-Aware Ternary Hybrid (FATH) framework extends the binary normalised neural network paradigm by introducing a signed ternary representation and layer-wise adaptive precision directed by Fisher Information. The binary constraint is relaxed to a ternary set that permits suppression of non-contributive features, while sensitivity-driven precision assignment allows crucial layers to maintain higher fidelity and less sensitive layers to be aggressively compressed. Experiments on the Food-101 subset demonstrate that the proposed framework requires significantly fewer training epochs and produces better accuracy than uniform binary quantization while maintaining significant compression. The decrease in validation loss and stable post-QAT activation distributions indicate that the quantization process is well-calibrated. These results establish Fisher-driven ternary hybrid quantization as an effective strategy for efficient neural network deployment on resource-constrained hardware.

Declarations

Funding

No funding was received to assist with the preparation of this manuscript.

Competing Interests

The author has no relevant financial or non-financial interests to disclose.

Data Availability

The experiments in this study use the publicly available Food-101 benchmark dataset which can be downloaded from https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/. No additional datasets were generated during this study.

Ethics Approval

Not applicable. This study does not involve human participants, human data, or animals.

References

[1] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning Both Weights and Connections for Efficient Neural Networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2015.

[2] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2016.

[3] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531*, 2015.

[4] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training Deep Neural Networks with Binary Weights During Propagations," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015.

[5] I. Hubara et al., "Binarized Neural Networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2016.

[6] M. Rastegari et al., "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016.

[7] S. Zhou et al., "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[8] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional Neural Networks using Logarithmic Data Representation," *arXiv preprint arXiv:1603.01025*, 2016.

[9] F. Li and B. Liu, "Ternary Weight Networks," *arXiv preprint arXiv:1605.04711*, 2016.

[10] Y. Zhou et al., "Adaptive Quantization for Deep Neural Network," *arXiv preprint arXiv:1712.01048*, 2017.

[11] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018.

[12] J. Choi et al., "PACT: Parameterized Clipping Activation for Quantized Neural Networks," *arXiv preprint arXiv:1805.06085*, 2018.

[13] R. Banner et al., "Post-training 4-bit Quantization of Convolution Networks for Rapid Deployment," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019.

[14] Z. Dong et al., "HAWQ: Hessian Aware Quantization of Neural Networks with Mixed-Precision," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019.

[15] Y. Cai et al., "ZeroQ: A Novel Zero Shot Quantization Framework," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[16] M. Nagel et al., "Up or Down? Adaptive Rounding for Post-Training Quantization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020.

[17] S. K. Esser et al., "Learned Step Size Quantization," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2020.

[18] R. Razani et al., "Adaptive Binary-Ternary Quantization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021.

[19] A. Gholami et al., "A Survey of Quantization Methods for Efficient Neural Network Inference," *arXiv preprint arXiv:2103.13630*, 2021.

[20] T. Dettmers et al., "LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale," *arXiv preprint arXiv:2208.07339*, 2022.

[21] E. Frantar et al., "GPTQ: Accurate Post-Training Quantization for Generative Transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[22] J. Lin et al., "AWQ: Activation-Aware Weight Quantization for LLM Compression," *arXiv preprint arXiv:2306.00978*, 2023.

[23] S. Ma et al., "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits," *arXiv preprint arXiv:2402.17764*, 2024.

[24] E. L. L. Cabral, P. Pirozelli, and L. Driemeier, "1 Bit Is All We Need: Binary Normalized Neural Networks," *arXiv preprint arXiv:2509.07025*, 2025.