PatchCore-Q: Robust On-Device Anomaly Detection via Quantized Feature Compensation

Hyunyup Kwak
Department of Semiconductor and Display Engineering,
Sungkyunkwan University
Suwon 16419, Republic of Korea
Samsung Institute of Technology
Samsung-ro 1, Yongin-si, Gyeonggi-do 17113, Republic of Korea
skku.lets.go@g.skku.edu

Jitae Shin[†]
Department of Electrical and Computer Engineering,
Sungkyunkwan University
Suwon 16419, Republic of Korea
jtshin@skku.edu

***Abstract*—Anomaly detection is increasingly executed directly on edge devices, where compute and memory are constrained, latency budgets are tight, and data residency is paramount. Running train and inference on the device yields low-latency and confidentiality benefits. PatchCore builds a memory bank of patch-level features from defect-free images and uses distance-based kNN scoring at test time, achieving anomaly detection performance. However, its large FP32 (Floating-Point 32 bit) backbone and feature storage limit deployment on resource-constrained edge devices. As representative quantization methodologies, PTQ (post-training quantization) and QAT (quantization-aware training) provide practical routes to faster inference and smaller models, but naively quantizing PatchCore with either scheme can severely degrade performance by distorting the feature-space geometry underlying its distance-based scoring. We introduce PatchCore-Q, a family of quantized PatchCore variants to restore low-bit model fidelity. PatchCore-QC (Quantization Compensation) applies lightweight per-channel affine correction for efficiency, while PatchCore-QA (Quantization Advanced Alignment) uses geometry-aware alignment losses to match the quantized backbone to an original backbone. On MVTec AD, our best variant improves image-level AUROC from 79% (pure PTQ) to 92%, while achieving $5\times$ model compression and more than $2\times$ throughput compared to FP32, highlighting post-quantization feature manipulation as a key ingredient for accurate and efficient on-device anomaly detection.**

***Index Terms*—Anomaly Detection, Quantization Compensation, Knowledge Distillation, On-Device AI**

I. Introduction

Visual inspection is a cornerstone of modern manufacturing, underlying quality control and process efficiency and serving as a primary use case for unsupervised anomaly detection [8] in datasets such as MVTec AD [3]. Within this domain, memory–bank anomaly detectors have set accuracy baselines [8], and PatchCore has emerged as a leading approach [1]. PatchCore constructs a memory bank of patch–level feature embeddings from defect–free images and identifies anomalies at test time by measuring Euclidean distances between new patch embeddings and this bank in feature space [1]. Despite its strong performance, PatchCore's reliance on a large, FP32–precision backbone and an FP32 memory bank creates a substantial barrier to deployment on resource–constrained edge devices where real-time processing and low power are essential.

Quantization offers a path to lighter backbones and has been extensively studied as a model compression technique for efficient inference [4]–[7]. Among quantization methods, PTQ (Post-Training Quantization) is particularly practical [5]: it converts FP32 weights and activations to low-bit integers (typically INT8), yielding sizable reductions in model size, memory bandwidth, and inference latency. PTQ further requires only a small, unlabeled calibration set—avoiding costly retraining and the often-unavailable access to the original large-scale training data. In contrast, QAT (Quantization-Aware Training) explicitly simulates quantization effects during training or fine-tuning [4], [6], [7], thereby improving robustness to quantization noise at the cost of additional optimization.

However, this efficiency comes with a trade-off. While accuracy losses in conventional classifiers are often manageable under quantization, naively applying quantization to PatchCore can be fatal: performance degrades so severely that the model becomes unusable. We hypothesize that this is not a generic accuracy drop but a specific phenomenon we term *geometric fragility*. Unlike supervised classifiers that can learn robust decision boundaries, PatchCore fundamentally depends on preserving the fine-grained geometry of the feature space. Its k-nearest-neighbor scoring discriminates subtle normal variations from true anomalies based on relative distances. Quantization introduces non-uniform scaling and shifting errors that disrupt this delicate geometry, collapsing the manifold of normal features and destabilizing distance-based scoring.

To reconcile efficiency with geometric fidelity, we introduce *PatchCore-Q*, a framework that manipulates quantized feature streams to counteract geometry distortions. The framework comprises two complementary strategies, each striking a distinct balance between fidelity and overhead:

1) **PatchCore-QC** (Quantization Compensation): a high-efficiency, per-channel affine correction that restores first-order statistics with negligible overhead.
2) **PatchCore-QA** (Quantization Advanced Alignment): a stronger, cross-channel alignment strategy that directly optimizes the student [9], [10] (quantized) backbone to match the FP32 teacher's feature geometry using a light, geometry-aware objective.

Motivated by the fact that QAT is a widely used alternative to PTQ, we also study PatchCore under the QAT regime, where quantization is simulated during fine-tuning and our geometry-aware objectives are applied to the resulting low-precision features. In this way, the proposed compensation framework is shown to be compatible with both PTQ and QAT, yielding a family of quantized PatchCore models that range from pure PTQ deployments to QAT-trained variants.

Contributions. (i) We identify and analyze geometric fragility as the primary way in which quantization breaks memory–bank detectors such as PatchCore, clarifying why naive quantization can make them unusable. (ii) We propose PatchCore-Q, a unified quantization compensation framework with two new strategies (QC and QA) that directly manipulate feature representations to mitigate quantization-induced degradation, and we relate them to existing quantization and compensation techniques [2], [5]. (iii) Through extensive experiments on MVTec AD [3], including PTQ, QAT, and compensation-enhanced baselines, we show that the proposed methods retain the efficiency of INT8 / INT4 inference while recovering—and in some cases surpassing—FP32 accuracy.

II. Preliminary

A. PatchCore Recap

PatchCore is built on a simple but powerful idea: normal images can be represented by a set of patch-level features, and anomalies can be detected by measuring how far a new image's patches lie from this "memory" of normal patterns. Specifically, features are extracted from intermediate layers of a pretrained encoder (e.g., Wide-ResNet50-2 at layer2 and layer3), then upsampled and concatenated so that each spatial location corresponds to a feature vector (or patch embedding). A coreset selection algorithm is used to store a representative subset of these embeddings as the memory bank $\mathcal{M}$.

During testing, each patch embedding z, the feature-space representation of a local image patch, is scored by its distance to the closest memory item, and the image-level score is taken as the maximum among its patch scores:

$$S(x) = \max_{z \in \mathcal{P}(x)} \big(\min_{m \in \mathcal{M}} \|z - m\|_2 \big). \tag{1}$$

Here, x denotes the test image and $\mathcal{P}(x)$ the set of patch embeddings extracted from x. Each embedding $z \in \mathcal{P}(x)$ is scored by its distance to the closest memory item $m \in \mathcal{M}$, where $\mathcal{M}$ is the memory bank constructed from normal data. The image-level score $S(x)$ is then defined as the maximum patch score, indicating the most anomalous patch within the image. This definition highlights the critical role of geometry: the relative distances between embeddings directly determine whether an image is classified as normal or anomalous. Even small shifts in feature space can change nearest-neighbor assignments and thereby degrade AUROC.

B. Quantization Methodology

Post-Training Quantization (PTQ). The backbone encoder is frozen and quantized for efficient on-device inference using a standard uniform affine quantizer [5]. Formally, let $y_\ell = f_\ell(x)$ denote the activation of the encoder at layer ℓ. A uniform quantizer $Q(\cdot)$ maps real-valued numbers into a fixed set of integer levels. For a scalar u, we use

$$q = \mathrm{clip}\big(\lfloor \tfrac{u}{s} \rceil + z,\ q_{\min}, q_{\max}\big), \qquad \widetilde{u} = s\,(q - z), \tag{2}$$

where s is the scale, z is the zero-point, q is the integer code, and $\widetilde{u}$ is the dequantized approximation of u. In practice, this operator is applied element-wise to y_ℓ, yielding quantized features $\widetilde{y}_\ell = Q_\ell(y_\ell; s_\ell, z_\ell)$ for each layer. PTQ, the primary quantization method in our study, estimates the quantization parameters (s_ℓ, z_ℓ) from a small calibration set without any gradient updates [5]. This makes PTQ highly practical, but the resulting quantization error can introduce structured scaling and shifting of intermediate features, which in turn distorts the relative distances between patch embeddings and motivates an explicit compensation mechanism.

Quantization-Aware Training (QAT). To further improve robustness against quantization noise, we also consider a QAT variant [4], [6], [7]. Instead of quantizing a pretrained network only once after training, QAT inserts the same quantizer $Q(\cdot)$ into the forward path during fine-tuning so that the model parameters can adapt to the quantized representation. Let $f_{\text{qat}}(x;\theta)$ denote the network where selected weights and activations are replaced by their quantized counterparts, e.g.,

$$\widetilde{y}_\ell = Q_\ell\big(f_\ell(\widetilde{y}_{\ell-1}; \theta_\ell)\big), \tag{3}$$

and the overall objective is

$$\theta^\star = \arg\min_\theta \mathbb{E}_x \big[\mathcal{L}_{\text{task}}(f_{\text{qat}}(x;\theta))\big], \tag{4}$$

where $\mathcal{L}_{\text{task}}$ is the task loss (in our case, derived from normal samples for anomaly detection). During backpropagation, gradients through $Q(\cdot)$ are through $Q(\cdot)$ are typically approximated using straight-through estimators, allowing the network to learn parameters θ that are explicitly optimized for low-bit inference. In our framework, QAT serves as a strong quantized training scheme, and our geometry-aware compensation objectives are applied on top of PTQ- or QAT-trained backbones to further stabilize their low-bit feature geometry.

C. Affine Compensation

This work aims to directly compensate quantization error through lightweight *affine* mappings applied on top of quantized networks. Notably, the Quantization-aware Weight Transformation (QwT) framework [2] proposes to approximate the discrepancy between quantized and full-precision representations using a closed-form, layer-wise affine correction. The central idea is that quantization error, while structured, can often be captured by a low-complexity affine transformation applied post quantization:

$$\hat{y} \approx W\tilde{y} + b, \tag{5}$$

where $\tilde{y}$ is the quantized feature, and (W, b) are lightweight compensation parameters derived without end-to-end retraining. This philosophy emphasizes preserving feature fidelity through

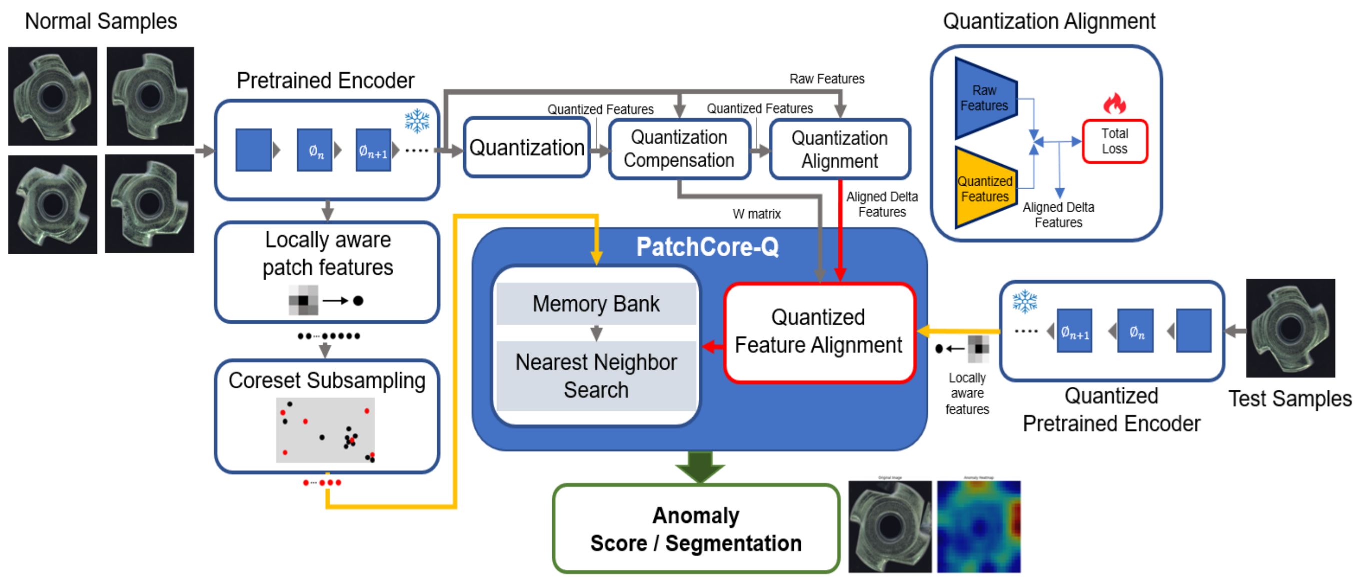

Fig. 1: Overall architecture of PatchCore-Q. Compensation modules are refine the locally-aware patch features extracted from the quantized encoder to match them against the coreset. Memory bank construction and nearest-neighbor scoring follow the standard PatchCore pipeline.

simple, analytic corrections rather than re-optimizing the full network. Our PatchCore-QA variant inherits this spirit by extending such linear compensation to geometry-sensitive anomaly detection, where the relative arrangement of embeddings in feature space is critical.

III. PROPOSED METHOD

In this section, we describe PatchCore-Q, a quantization compensation framework designed to preserve the feature space geometry that underlies PatchCore's kNN-based anomaly scoring. Our design follows a teacher–student paradigm: an FP32 PatchCore model serves as the teacher, while a quantized PatchCore model acts as the student. PatchCore-Q introduces two complementary strategies, PatchCore-QC and PatchCore-QA, which operate on top of the quantized backbone to restore the distance structure required for reliable anomaly detection.

A. Overall Framework

We first train a standard FP32 PatchCore model and construct the memory bank using defect-free training images. This model defines the target feature geometry and the reference distance distribution. Next, we obtain a quantized backbone using either PTQ or QAT. PatchCore-Q compensates for this degradation by explicitly aligning the quantized features with the FP32 features. During the compensation stage, normal training images are fed to both the teacher and the quantized student. PatchCore-QC estimates per-channel affine parameters that approximate the teacher's activations from the student's activations, while PatchCore-QA further refines the student backbone itself through geometry-aware objectives. At inference time, only the compensated quantized backbone and the FP32 memory bank are used: test images are processed by the quantized backbone, optionally transformed by stored compensation parameters, converted into patch embeddings, and finally scored via kNN distance to the memory bank. Thus, the compensation is learned once on normal data and then reused for all test samples without additional overhead.

B. PatchCore-QC: Per-Channel Affine Compensation

PatchCore-QC aims to correct quantization error with minimal cost by learning lightweight affine transformations on top of the quantized backbone. Let $y_\ell^{(q)} \in \mathbb{R}^{C_\ell \times H_\ell \times W_\ell}$ denote the activation tensor at layer ℓ of the quantized student, and $y_\ell^{(t)}$ the corresponding activation of the FP32 teacher. PatchCore-QC introduces a small affine "compensation head" parameterized by (W_ℓ, b_ℓ) and applies

$$\widehat{y}_\ell = f_\ell(y_\ell^{(q)}; W_\ell, b_\ell), \tag{6}$$

where f_ℓ is implemented as a 1×1 convolution (or linear layer) followed by bias addition. Conceptually, one can interpret $\Delta y_\ell = \widehat{y}_\ell - y_\ell^{(q)}$ as a compensation delta feature, but in practice we do not store Δy_ℓ explicitly; only the affine parameters (W_ℓ, b_ℓ) are learned and reused at inference.

During the QC stage, we freeze both the FP32 teacher and the quantized backbone and estimate (W_ℓ, b_ℓ) on normal images so that $\widehat{y}_\ell$ regresses toward $y_\ell^{(t)}$ (using the closed-form QwT-style estimator described in Section II-C). Intuitively, the compensation head learns how the quantized activations must be shifted and mixed across channels to mimic the teacher's features. At inference time, the teacher network is discarded and only the quantized backbone plus the stored (W_ℓ, b_ℓ) are used: for each test image, the quantized activation $y_\ell^{(q)}$ is passed through $f_\ell(\cdot; W_\ell, b_\ell)$ to obtain $\widehat{y}_\ell$, and all subsequent PatchCore operations (patch extraction, coreset,

kNN scoring) operate on these compensated features. As a result, QC incurs minimal runtime overhead while providing a coarse but effective correction of quantization-induced shifts and distortions.

C. PatchCore-QA: Geometry-Aware Alignment

While PatchCore-QC provides a lightweight, layer-wise correction, its per-channel affine model is limited in how precisely it can reconstruct the FP32 feature geometry. PatchCore-QA therefore adds a geometry-aware fine-tuning stage that learns *delta features* on normal images so that compensated quantized embeddings better match the FP32 ones.

Let z denote FP32 patch embeddings and $\tilde{z}$ the corresponding embeddings obtained from the quantized backbone. A QwT-style compensation head takes the input image and produces aligned deltas Δz, and the final embeddings used by PatchCore-QA are

$$\hat{z} = \tilde{z} + \Delta z. \tag{7}$$

QA minimizes a multi-term loss on $(\hat{z}, z)$. The feature alignment term combines a robust regression loss with a directional penalty:

$$\mathcal{L}_{\text{feat}} = \big\|\hat{z} - z\big\|_{\text{smooth-}L_1} + \tfrac{1}{2}\big(1 - \cos(\hat{z}, z)\big), \tag{8}$$

where $\|\cdot\|_{\text{smooth-}L_1}$ is a Huber-style smooth L_1 loss and $\cos(\hat{z}, z)$ is the cosine similarity between compensated and FP32 embeddings. This term encourages QA to match both the magnitude and direction of FP32 patch features.

To preserve the anomaly-sensitive distance structure, we introduce a distance-distribution matching term with respect to a memory-bank coreset $\mathcal{C}$:

$$d_s(i) = \min_{c \in \mathcal{C}} \big(1 - \cos(\hat{z}_i, c)\big), \tag{9}$$

$$d_t(i) = \min_{c \in \mathcal{C}} \big(1 - \cos(z_i, c)\big), \tag{10}$$

$$\mathcal{L}_{\text{dist}} = \frac{1}{N} \sum_{i=1}^{N} \big(d_s(i) - d_t(i)\big)^2. \tag{11}$$

Here $d_s(i)$ and $d_t(i)$ are the nearest-neighbor cosine distances from compensated and FP32 embeddings to the coreset. Minimizing $\mathcal{L}_{\text{dist}}$ aligns the local distance distributions that drive PatchCore's kNN-based anomaly scores.

In addition, we apply an ℓ_2 regularizer to the compensation parameters W,

$$\mathcal{L}_{\text{reg}} = \|W\|_F^2, \tag{12}$$

and define the overall objective as

$$\mathcal{L} = \alpha\,\mathcal{L}_{\text{feat}} + \beta\,\mathcal{L}_{\text{dist}} + \lambda\,\mathcal{L}_{\text{reg}}, \tag{13}$$

with α, β, and λ controlling the trade-off among feature alignment, distance matching, and regularization.

At inference time, the FP32 model is discarded. The quantized backbone produces $\tilde{z}$, the compensation head outputs Δz for each test sample, and the element-wise sum $\hat{z} = \tilde{z} + \Delta z$ is fed into the existing PatchCore pipeline (patch extraction, coreset, kNN scoring). Conceptually, PatchCore-QC performs a coarse, per-channel shift using affine parameters, while PatchCore-QA supplies finer, geometry-aware deltas in feature space, making the two mechanisms complementary.

IV. EXPERIMENT

A. Datasets and Evaluation Metrics

We evaluate PatchCore-Q on the MVTec AD dataset [3], a standard industrial visual anomaly benchmark consisting of 15 object and texture categories with normal-only training images and mixed normal/defective test images. Following common practice, we report three performance metrics: image-level AUROC for classifying each image as normal or anomalous, pixel-level AUROC for anomaly localization, and AUPRO (area under the per-region overlap curve) to summarize region-wise segmentation quality. In addition, we report two efficiency metrics, model size and per-image latency, to quantify the impact of quantization on compactness and inference speed.

B. Environmental Setup

TABLE I: Experimental setup (hardware, training schedule, and calibration details).

Category	**Value**
GPU	A5000
CUDA Version	12.2
PyTorch Version	2.3.1
Batch size	32
PTQ Calibration Iterations	500
QAT Epochs	10
QAT learning rate	2e-5
PTQ Calibration Iterations	500
QwT γ (initial)	0.9
QwT β (initial)	0.001
Alignment Epochs	80
Alignment learning rate	2e-5

All experiments are conducted on a single NVIDIA RTX A5000 GPU with a standard PyTorch/CUDA stack, using the training schedules and hyper-parameters summarized in Table I. This configuration is used for the FP32 baseline, pure PTQ/QAT models, and all PatchCore-Q variants.

C. Implementation Details

We adopt Wide-ResNet-50-2 pretrained on ImageNet as the backbone and extract feature maps from the second and third residual stages. An FP32 PatchCore baseline is first constructed by converting these features into locally aware patch embeddings, applying coreset subsampling, and building a memory bank on normal training images; this FP32 memory bank is reused for all quantized variants.

For the quantized models, we consider both PTQ and QAT backbones. For the QAT baseline, our goal is to isolate the incremental effect of the proposed compensation module on top of representative quantization pipelines(PTQ and QAT), rather than to maximize the absolute QAT performance by extensive hyperparameter tuning. Therefore, we adopt a single commonly-used QAT recipe with a fixed training budget. Although more aggressive tuning (e.g., longer training or per-category learning rates) may further improve QAT, such optimization is beyond the scope of our contribution.

PatchCore-QC estimates lightweight affine compensation parameters on normal images using the QwT-style procedure described in Section II, while PatchCore-QA trains a geometry-aware compensation head with the loss defined in Section III to produce aligned delta features. At test time, each method (FP32, pure PTQ/QAT, QC, or QA) produces patch embeddings that are scored against the FP32 memory bank via kNN distance, and anomaly metrics are computed from the resulting scores.

D. Quantitative Results

Table II summarizes the results for the FP32 baseline, pure PTQ/QAT models, and our quantization-compensated variants. Several observations can be drawn:

- **Accuracy restoration.** In the most aggressive INT4 setting, pure quantization degrades performance substantially: image AUROC drops to about 78–80% and AUPRO to roughly 47% (PTQ INT4), showing that low-bit PatchCore is highly sensitive to geometric distortion. With our compensation, the best INT4 PatchCore-QA variant lifts image AUROC to 92.7% and pixel AUROC to above 93%, while AUPRO exceeds 69%. These numbers approach the FP32 baseline (96.4% / 96.9% / 78.3%), indicating that most of the accuracy lost by pure INT4 quantization is recovered.
- **Efficiency gains.** Compared to FP32 PatchCore (197 MB, 18.5 ms/image), the best INT4 PatchCore-QA model is about 5$\times$ smaller ($\approx$ 38 MB) and more than 2$\times$ faster ($\approx$ 8–9 ms/image). Thus, even our most accurate compensated variant remains markedly more compact and efficient than the FP32 baseline.

Overall, the results show that PatchCore-Q achieves near-FP32 accuracy in the INT8/INT4 regime, while preserving the compression and latency advantages of aggressive low-bit quantization.

TABLE II: Quantization performance comparison.

Model	**Quantization**	AUROC image (%) ↑	AUROC pixel (%) ↑	AUPRO (%) ↑	Model Size (MB) ↓	Latency (ms/img) ↓
Patchcore	None (FP32)	96.40	96.88	78.29	197.16	18.49
Patchcore	PTQ INT8	92.26	95.38	76.37	65.69	8.25
Patchcore	QAT INT8	84.03	92.93	68.01	23.71	8.46
Patchcore-QC	PTQ INT8	96.75	97.44	78.63	70.93	11.12
Patchcore-QC	QAT INT8	86.63	94.42	65.61	28.95	13.36
Patchcore-QA (Ours)	**PTQ INT8**	**96.76**	**97.45**	**78.88**	**75.57**	**13.42**
Patchcore-QA (Ours)	**QAT INT8**	**87.92**	**94.56**	**70.54**	**34.65**	**14.97**
Patchcore	PTQ INT4	79.64	79.96	46.97	25.46	5.02
Patchcore	QAT INT4	77.93	89.08	61.09	25.12	6.23
Patchcore-QC	PTQ INT4	84.81	88.79	61.28	30.59	6.37
Patchcore-QC	QAT INT4	88.61	93.34	71.77	30.47	7.01
Patchcore-QA (Ours)	**PTQ INT4**	**92.74**	**93.66**	**69.18**	**37.96**	**8.33**
Patchcore-QA (Ours)	**QAT INT4**	**89.39**	**94.46**	**72.29**	**37.62**	**8.85**

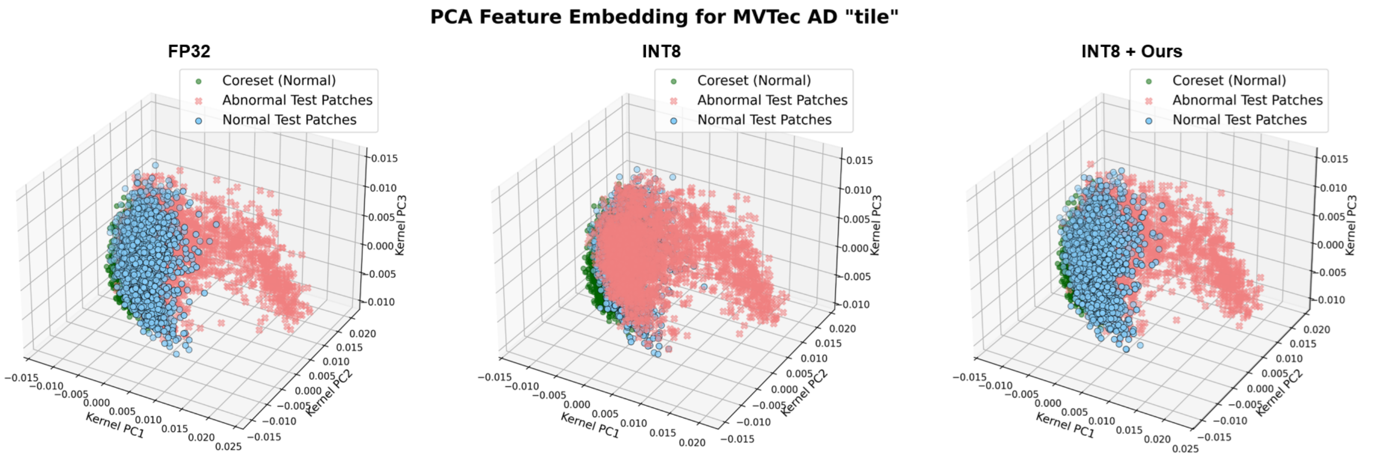

Fig. 2: 3D Kernel PCA visualization for the MVTec AD `tile` category. Normal features for (left) FP32, (center) INT8, and (right) INT8 + Ours are projected onto the first three Kernel PCA components (PC1, PC2, PC3).

E. Qualitative Results

Beyond aggregate metrics, we qualitatively examine how quantization and our compensation affect the underlying feature geometry. We focus on two complementary indicators: (i) Kernel PCA projections of patch embeddings for the `tile` category, and (ii) nearest-neighbor distance distributions between normal samples and the coreset. In both cases, we compare three models that appear in our legends: **FP32**, **INT8**, and **INT8 + Ours**. The following visual analyses show that INT8 quantization distorts the PatchCore feature manifold, whereas INT8 + Ours largely restores the FP32 structure.

F. Visual Analysis of the Quantization Compensation

1) Geometric Structure via Kernel PCA: Fig. 2 shows the feature-space distribution for the MVTec AD `tile` category, projected onto the first three Kernel PCA components (PC1, PC2, PC3) that capture the largest nonlinear variance.

- **FP32:** Normal patches form a smooth nonlinear manifold around the memory bank, while abnormal patches occupy a clearly separated region. This structure allows kNN scoring to cleanly distinguish defects.
- **INT8:** After pure INT8 quantization, the normal manifold partially collapses toward the memory bank and mixes with abnormal patches. The separation boundary becomes blurred, consistent with the accuracy drop of the INT8 rows in Table II.
- **INT8 + Ours:** With our compensation, the normal manifold and abnormal region recover a configuration close to FP32. Normal patches again wrap around the memory bank, and abnormal patches are pushed outward, indicating that the semantic geometry required by PatchCore is effectively restored.

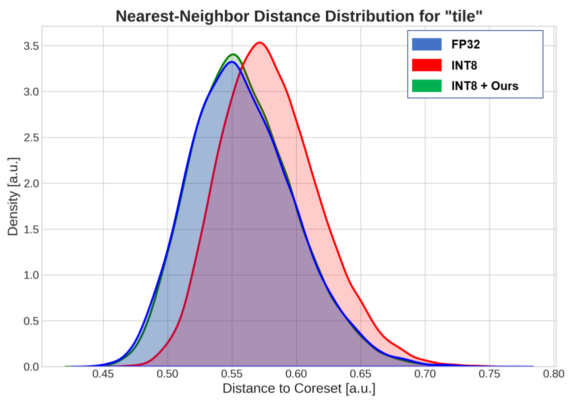

Fig. 3: Distribution of nearest-neighbor distances between normal training samples and the closest coreset member for the MVTec AD `tile` category. The x-axis denotes cosine distance to the nearest coreset member, and the y-axis is the probability density.

2) Distance Distribution Analysis: To complement the geometric view in Fig. 2, Fig. 3 shows the nearest-neighbor distance distribution between normal training samples and their closest coreset member. An ideal PatchCore model yields distances that are small (shifted to the left), indicating that normal samples lie tightly around the memory manifold.

- **INT8:** The INT8 model produces larger and more dispersed distances than FP32; the distribution shifts to the right and becomes wider. Many normal samples are pushed away from the memory manifold, leading to unstable and less reliable anomaly scores.
- **INT8 + Ours:** After applying our compensation, the distance distribution moves back toward the FP32 reference. Normal samples are again concentrated near the coreset, confirming that our method recovers the core distance statistics that drive PatchCore's kNN-based scoring.

V. Conclusion

In this work, we proposed PatchCore-Q, a quantized feature compensation framework for on-device industrial anomaly detection. We show that naive quantization induces structured distortions in intermediate features, breaking the local Euclidean geometry that PatchCore relies on for kNN-based scoring. Based on this, we argue that preserving FP32 feature geometry—rather than only minimizing layer-wise reconstruction error—is crucial for retaining anomaly detection accuracy under low-bit constraints.

Experiments support this claim both quantitatively and qualitatively. PatchCore-QA restores image-level AUROC from the mid-70% range (pure PTQ) to around 95%, while substantially reducing model size, latency, and runtime memory. Kernel PCA visualizations and nearest-neighbor distance distributions further indicate that our compensation recovers the separation between normal/abnormal patches and re-concentrates normal samples around the memory manifold. These results collectively suggest that local geometry preservation is the key mechanism enabling FP32-level accuracy in a compact form.

Finally, PatchCore-Q is platform-agnostic: it operates directly on quantized tensors and requires no device-specific kernels. While absolute latency/energy depends on hardware, the consistent reductions in storage and runtime memory—together with standard INT8/4 execution—make the method inherently on-device friendly. We will report additional on-device latency and energy results on ARM/Jetson platforms in future work.

References

[1] L. Roth, et al., "Towards Total Recall in Industrial Anomaly Detection," in *CVPR*, 2022.
[2] J. Wu, et al., "Quantization without Tears," *CVPR*, 2025.
[3] P. Bergmann, et al., "MVTec AD—A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection," in *CVPR*, 2019.
[4] S. Jacob, et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *CVPR*, 2018.
[5] S. Nagel, et al., "Up or Out: Quick Post-Training Quantization of Convolutional Networks," in *ECCV Workshops*, 2020.
[6] S. Esser, et al., "Learned Step Size Quantization," in *NeurIPS*, 2019.
[7] J. Choi, et al., "PACT: Parameterized Clipping Activation for Quantized Neural Networks," *arXiv preprint* arXiv:1805.06085, 2018.
[8] T. Defard, et al., "PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization," in *ICPR*, 2021.
[9] G. Hinton, et al., "Distilling the Knowledge in a Neural Network," in *NIPS*, 2015.
[10] A. Romero, et al., "FitNets: Hints for Thin Deep Nets," in *ICLR*, 2015.