Industrial Anomaly Detection with Localization: A Comparative Study under Clean and Shifted Domains

Ivan Necerini
s345147
s345147@studenti.polito.it

Jacopo Rialti
s346357
s346357@studenti.polito.it

Fabio Veroli
s336301
s336301@studenti.polito.it

Abstract

[?]

1. Introduction

Visual anomaly detection in manufacturing aims to automatically identify defective products from images acquired under controlled conditions. This task is commonly framed as *one-class classification*, where a model learns only from normal samples and must identify deviations at test time [1]. Since anomalous examples are often scarce or unavailable during training, supervised approaches are impractical, motivating unsupervised methods that learn what constitutes normality from defect-free data.

A practical anomaly detection system must address two complementary goals: (i) *image-level detection*, determining whether an image is normal or anomalous, and (ii) *pixel-level localization*, identifying which regions contain defects. State-of-the-art methods such as **PatchCore** [7] and **PaDiM** [2] achieve strong results on standard benchmarks like MVTec AD [1] by modelling patch-level feature distributions from pre-trained networks. However, their robustness to distribution shift and sensitivity to threshold calibration remain less explored.

In this work, we present a comparative study of PatchCore and PaDiM on MVTec AD under both clean and synthetically shifted conditions. Our contributions are:

1. **Baseline comparison.** We evaluate PatchCore (custom implementation) and PaDiM (anomalib-based) on three MVTec AD categories (Hazelnut, Carpet, Zipper), reporting both image-level and pixel-level metrics.

2. **Synthetic domain shift.** We construct *MVTec-Shift* by applying photometric and geometric transformations inspired by MVTec AD 2 [4], and measure the performance drop when models trained on clean data are tested on shifted data. Measuring also the hyperparameter influence on the performance obtained.

3. **Threshold adaptation strategies.** We compare three regimes: (a) no adaptation, (b) threshold-only adaptation on shifted validation data, and (c) full adaptation with model re-training. This ablation isolates the benefit of threshold recalibration from feature-level adaptation.

4. **Model-unified setting.** Following recent literature [8, 3, 5], we train a single model on pooled normal data from all categories while using per-class thresholds at inference. This setting enables investigation of the *identical shortcut* problem [8], where normal samples from one class may be misclassified as anomalies when evaluated against another class's threshold due to overlapping feature distributions.

The paper is organized as follows: Section 2 reviews related work; Section 3 describes the methods; Section 4 details the experimental setup; Section 5 presents results and analysis; Section 6 concludes with limitations and future directions. Figure 1 provides a high-level overview of our experimental pipeline.

2. Related Work

Unsupervised anomaly detection methods for industrial images can be broadly categorized into four families: reconstruction-based, distribution-based, memory-based, and unified approaches. Reconstruction methods [1] learn to encode and decode normal images, flagging samples with high reconstruction error as anomalous. While intuitive, these approaches often struggle to localize subtle defects when the autoencoder generalizes too well. Distribution-based approaches model the statistical properties of normal features, whereas memory-based methods store and retrieve representative embeddings at test time. In this work we focus on two state-of-the-art methods from the latter categories, **PaDiM** and **PatchCore**, which both leverage features from pre-trained convolutional networks.

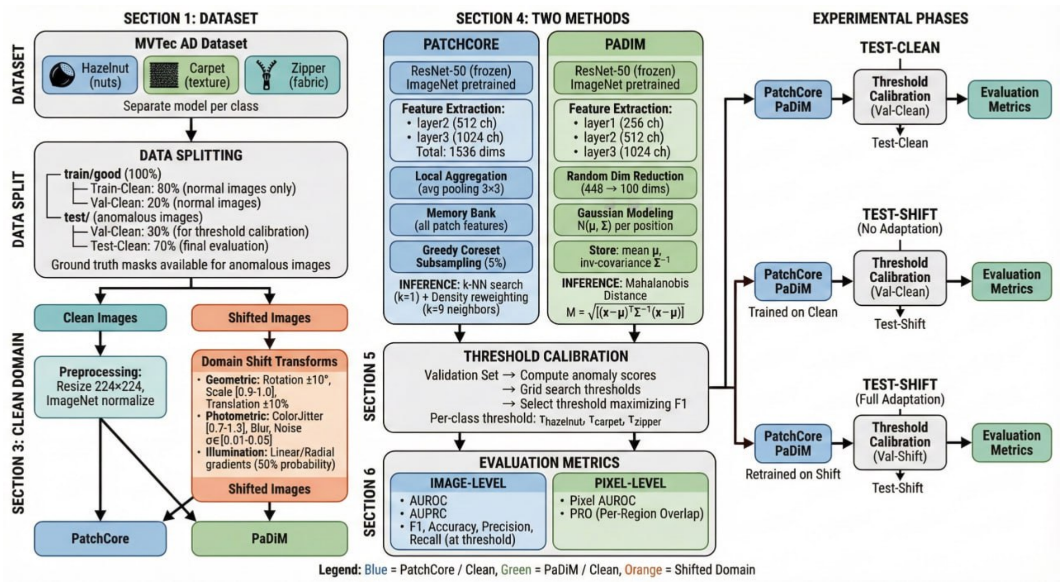
SECTION 1: DATASET
DATASET
MVTec AD Dataset
Hazelnut (nuts)
Carpet (texture)
Zipper (fabric)
Separate model per class
DATA SPLIT
DATA SPLITTING
train/good (100%)
Train-Clean: 80% (normal images only)
Val-Clean: 20% (normal images)
test/ (anomalous images)
Val-Clean: 30% (for threshold calibration)
Test-Clean: 70% (final evaluation)
Ground truth masks available for anomalous images
Clean Images
Shifted Images
SECTION 3: CLEAN DOMAIN
Preprocessing: Resize 224×224, ImageNet normalize
Domain Shift Transforms
Geometric: Rotation ±10°, Scale [0.9-1.0], Translation ±10%
Photometric: ColorJitter [0.7-1.3], Blur, Noise σ∈[0.01-0.05]
Illumination: Linear/Radial gradients (50% probability)
Shifted Images
PatchCore
PaDiM
SECTION 4: TWO METHODS
PATCHCORE
ResNet-50 (frozen) ImageNet pretrained
Feature Extraction: layer2 (512 ch), layer3 (1024 ch) Total: 1536 dims
Local Aggregation (avg pooling 3×3)
Memory Bank (all patch features)
Greedy Coreset Subsampling (5%)
INFERENCE: k-NN search (k=1) + Density reweighting (k=9 neighbors)
PADIM
ResNet-50 (frozen) ImageNet pretrained
Feature Extraction: layer1 (256 ch), layer2 (512 ch), layer3 (1024 ch)
Random Dim Reduction (448 → 100 dims)
Gaussian Modeling N(μ, Σ) per position
Store: mean μ, inv-covariance Σ⁻¹
INFERENCE: Mahalanobis Distance
M = √[(x−μ)ᵀΣ⁻¹(x−μ)]
SECTION 5
THRESHOLD CALIBRATION
Validation Set → Compute anomaly scores → Grid search thresholds → Select threshold maximizing F1
Per-class threshold: τhazelnut, τcarpet, τzipper
SECTION 6
EVALUATION METRICS
IMAGE-LEVEL
AUROC
AUPRC
F1, Accuracy, Precision, Recall (at threshold)
PIXEL-LEVEL
Pixel AUROC
PRO (Per-Region Overlap)
EXPERIMENTAL PHASES
TEST-CLEAN
PatchCore PaDiM
Threshold Calibration (Val-Clean)
Test-Clean
Evaluation Metrics
TEST-SHIFT (No Adaptation)
PatchCore PaDiM
Trained on Clean
Threshold Calibration (Val-Clean)
Test-Shift
Evaluation Metrics
TEST-SHIFT (Full Adaptation)
PatchCore PaDiM
Retrained on Shift
Threshold Calibration (Val-Shift)
Test-Shift
Evaluation Metrics
Legend: Blue = PatchCore / Clean, Green = PaDiM / Clean, Orange = Shifted Domain

Figure 1. Overview of the experimental pipeline. The workflow encompasses data preparation (splitting and domain shift generation), model training (PatchCore memory bank construction and PaDiM Gaussian fitting), threshold calibration on validation sets, and evaluation under multiple scenarios (clean domain, shifted domain with/without adaptation, and model-unified setting).

2.1. Embedding-Based Methods

PaDiM. Defard *et al*. [2] propose *Patch Distribution Modeling* (PaDiM), which extracts multi-scale features from intermediate layers of a pre-trained ResNet and models the distribution of patch embeddings with a multivariate Gaussian at each spatial location. At inference, anomaly scores are computed via the Mahalanobis distance between a test patch and its corresponding Gaussian. Because it stores only statistical summaries (mean and covariance) rather than individual embeddings, PaDiM is highly memory-efficient and fast at inference. However, the Gaussian assumption may be violated when the underlying distribution is non-unimodal, limiting its capacity to capture complex normal variations.

PatchCore. Roth *et al*. [7] present **PatchCore**, which constructs a *memory bank* of locally-aware patch embeddings extracted from a pre-trained network. To reduce storage and retrieval costs, the memory bank is subsampled using greedy coreset selection. At test time, each image patch is compared to its nearest neighbor in the memory bank, and the maximum distance across patches determines the image-level score. PatchCore achieves near-perfect AUROC on MVTec AD, demonstrating excellent localization thanks to its explicit patch-level comparison.

Efficiency vs. Accuracy Trade-off. Both methods exploit transferable features from ImageNet-pretrained backbones, enabling training without gradient-based optimization. PaDiM's parametric model yields constant-time inference independent of dataset size, whereas PatchCore's non-parametric memory bank scales with the number of stored embeddings (mitigated by coreset selection). Empirically, PatchCore tends to achieve higher detection and localization accuracy at the cost of increased memory footprint and inference time. In our experiments we quantify this trade-off across multiple categories and domain conditions.

2.2. Unified Multi-Class Anomaly Detection

Conventional anomaly detection follows a *one-model-per-class* paradigm: a separate model is trained for each product category. This approach becomes impractical when the number of categories grows, motivating research into *unified* models that handle multiple classes simultaneously.

Identical Shortcut Problem. You *et al*. [8] introduce **UniAD**, a transformer-based architecture that reconstructs features of different object classes with a single model. A key contribution is the identification of the *identical shortcut* problem: when a model is trained on multiple classes, it may learn to reconstruct any input identically, thereby losing the ability to detect anomalies. UniAD addresses this by injecting neighbour-masked attention and layer-wise query decoders. This insight motivates careful evaluation of unified settings in our work.

Model-Unified vs. Absolute-Unified. Heo and Kang [5] further distinguish between *Model-Unified* and *Absolute-Unified* settings. In the Model-Unified setting, a single model is trained on pooled normal data from all categories, but separate, per-class thresholds are calibrated at inference. In the stricter Absolute-Unified setting, a single global threshold is used regardless of class. The latter is significantly harder because classes may have different "normal" score distributions. Our experiments adopt the Model-Unified paradigm to study how PatchCore and PaDiM behave when exposed to heterogeneous training data, and whether the identical shortcut manifests in embedding-based methods.

2.3. Robustness to Domain Shift

Despite strong benchmark performance, the robustness of anomaly detection models to distribution shift remains under-explored. Real-world deployment often involves variations in lighting, sensor noise, or imaging conditions that differ from the training environment. The MVTec AD 2 dataset [4] explicitly addresses this gap by introducing controlled illumination and pose variations, revealing substantial performance degradation for methods trained on standard MVTec AD. Motivated by this, we construct a synthetic *MVTec-Shift* domain using photometric and geometric augmentations and systematically study the drop in detection and localization accuracy. We further investigate whether simple threshold recalibration on shifted validation data can recover performance without retraining feature extractors, offering practical insights for deployment scenarios with limited access to shifted anomalous samples.

3. Methodology

This section presents the formal problem setup, the domain shift simulation strategy, and the technical details of the two anomaly detection methods employed in our study: PatchCore and PaDiM.

3.1. Problem Formulation

We consider the one-class classification paradigm for visual anomaly detection. Let $\mathcal{D}_{\text{train}} = \{x_i\}_{i=1}^{N}$ denote a training set consisting exclusively of nominal (defect-free) images, where $x_i \in \mathbb{R}^{H\times W\times 3}$. The objective is to learn a scoring function $s : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}$ that assigns low scores to normal samples and high scores to anomalous ones. For pixel-level localization, the goal extends to producing a spatial anomaly map $\mathcal{A} : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{H\times W}$, where $\mathcal{A}(x)_{i,j}$ indicates the anomaly intensity at pixel (i, j).

Preprocessing. All images undergo a deterministic preprocessing pipeline before being fed to the feature extractor. Images are resized to 224×224 pixels using bilinear interpolation, converted to tensors with values in $[0, 1]$, and normalized using ImageNet statistics:

$$\mu = (0.485, 0.456, 0.406), \quad \sigma = (0.229, 0.224, 0.225) \tag{1}$$

where normalization is applied channel-wise. Ground truth masks, when available, are resized using nearest-neighbor interpolation to preserve binary values and binarized at threshold 0.5.

3.2. Domain Shift Simulation (MVTec-Shift)

To evaluate robustness under distribution shift, we construct a synthetic shifted domain by applying controlled transformations to the clean MVTec AD images. This approach is inspired by the acquisition variations introduced in MVTec AD 2 [4], which include illumination changes and pose perturbations. Our transformation pipeline comprises three categories, applied with synchronized random seeds to ensure reproducibility:

Geometric Transforms. Applied to both images and ground truth masks to maintain spatial consistency:

- **Rotation:** uniformly sampled from $[-10°, +10°]$
- **Scale:** uniformly sampled from $[0.9, 1.0]$ (slight zoom-out)
- **Translation:** uniformly sampled within $\pm 10\%$ of image dimensions

Photometric Transforms. Applied only to images (not masks) to simulate sensor and illumination variations:

- **Color Jitter:** brightness, contrast, and saturation factors sampled uniformly from $[0.7, 1.3]$
- **Gaussian Blur:** kernel size $\in \{3, 5\}$, sigma $\in [0.1, 2.0]$, applied with probability 0.5
- **Gaussian Noise:** additive noise with $\sigma \in [0.01, 0.05]$, applied with probability 0.5

Illumination Gradients. To simulate non-uniform industrial lighting conditions (*e.g.*, spotlight effects from MVTec AD 2 scenarios such as Fabric and Wall Plugs), we apply smooth illumination gradients with probability 0.5:

- **Linear gradients:** simulate side lighting by darkening one edge (left, right, top, or bottom) with strength factor sampled from $[0.4, 0.7]$
- **Radial gradients:** simulate center/edge illumination variations
- Gradients are smoothed with a Gaussian filter ($\sigma = 80$) for natural transitions

These transformations collectively simulate realistic sensor degradation and environmental changes encountered in industrial deployment, providing a controlled way to evaluate model robustness.

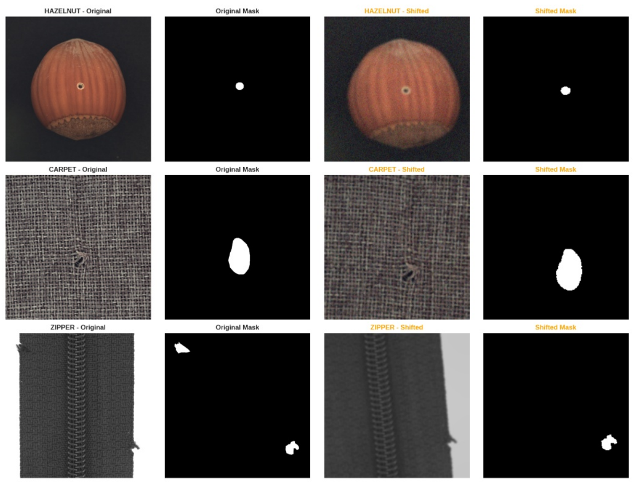

Figure 2. Comparison of clean (left) and shifted (right) samples from MVTec AD. The shifted domain includes photometric degradation, geometric perturbations, and non-uniform illumination to simulate realistic industrial variations.

3.3. PatchCore

PatchCore [7] is a memory-based anomaly detection method that achieves state-of-the-art results through efficient storage and retrieval of nominal patch embeddings. We implement PatchCore following the original paper with hyperparameters verified against our configuration.

Feature Extraction. We employ a ResNet-50 backbone pre-trained on ImageNet with frozen weights. Features are extracted from two intermediate layers, `layer2` (512 channels, 28×28 spatial resolution for 224×224 input) and `layer3` (1024 channels, 14×14 resolution), to capture both fine-grained and semantic information.

Local Neighborhood Aggregation. To incorporate spatial context into each patch embedding, we perform average pooling over a $p \times p$ local neighborhood:

$$f_{\text{agg}}^{(h,w)} = \frac{1}{p^2} \sum_{(i,j) \in \mathcal{N}_p(h,w)} f^{(i,j)} \tag{2}$$

where $\mathcal{N}_p(h,w)$ denotes the $p \times p$ patch centered at position (h,w) and $f^{(i,j)}$ is the feature vector at that location. We use $p = 3$ in our implementation. Features from `layer3` are bilinearly upsampled to match the spatial resolution of `layer2`, and the two feature maps are concatenated along the channel dimension, yielding a combined feature dimension of $512 + 1024 = 1536$ per patch.

Coreset Subsampling. Storing all patch embeddings from the training set would result in an impractically large memory bank. Following Roth *et al.* [7], we apply greedy coreset subsampling to select a representative subset. The algorithm iteratively selects patches that maximize the minimum distance to already selected patches (furthest-point sampling):

$$m^* = \arg \max_{m \in \mathcal{M} \setminus \mathcal{C}} \min_{c \in \mathcal{C}} \|m - c\|_2 \tag{3}$$

where $\mathcal{M}$ is the full memory bank and $\mathcal{C}$ is the current coreset. To accelerate distance computations, we apply Johnson-Lindenstrauss random projection to reduce dimensionality to 128 before sampling. We retain $\rho = 5\%$ of total patches (*i.e.*, `coreset_sampling_ratio = 0.05`), which provides an optimal balance between coverage and efficiency as validated on the clean domain. In following Section 5, we will show the effect of this hyperparameter on the performance of PatchCore.

Anomaly Scoring. At inference time, each test image x is decomposed into a collection of locally aware patch features $\mathcal{P}(x)$. Each patch embedding $q \in \mathcal{P}(x)$ is compared against the coreset memory bank $\mathcal{C}$ using a Nearest Neighbor (NN) search, implemented via the FAISS [6] library for high-performance indexing.

The raw anomaly score $s_{\text{raw}}(q)$ is defined as the L^2 distance to its nearest neighbor in the coreset:

$$s_{\text{raw}}(q) = \min_{m \in \mathcal{C}} \|q - m\|_2 \tag{4}$$

To enhance robustness against noise and outliers, we adopt a density-based reweighting scheme inspired by Roth *et al.* [7]. Specifically, we retrieve the k nearest neighbors of q from the coreset (where $k = 9$ in our implementation) and compute a weighting factor $w(q)$ that penalizes samples whose nearest neighbor is significantly more distant than the rest of the local neighborhood:

$$w(q) = 1 - \frac{\exp(-d_1)}{\sum_{i=1}^{k} \exp(-d_i)} \tag{5}$$

where d_i represents the L^2 distance to the i-th nearest neighbor. The final anomaly score for each patch is then $s(q) = s_{\text{raw}}(q) \cdot w(q)$.

The overall image-level anomaly score $S(x)$ is determined by the maximum patch-level score, reflecting the assumption that an image is anomalous if it contains at least one anomalous patch:

$$S(x) = \max_{q \in \mathcal{P}(x)} s(q) \tag{6}$$

Anomaly Localization. For defect localization, a spatial anomaly map is generated by reshaping the patch scores $s(q)$ to their respective grid positions. The map is then upsampled to the original input resolution via bilinear interpolation to produce the final pixel-level prediction.

3.4. PaDiM

PaDiM [2] (Patch Distribution Modeling) is a distribution-based anomaly detection method that models normal patch embeddings with multivariate Gaussian distributions. Our implementation uses the `anomalib` library's native `PadimModel`.

Feature Extraction. PaDiM extracts multi-scale features from three ResNet-50 layers: `layer1`, `layer2`, and `layer3`. Features are concatenated and projected to a common spatial resolution, yielding a high-dimensional embedding at each spatial position.

Dimensionality Reduction. To mitigate the curse of dimensionality and reduce computational cost, PaDiM employs random feature selection. From the concatenated feature vector, a random subset of $d = 100$ dimensions is selected (configured via `n_features = 100`). This selection is performed once at initialization and fixed for the lifetime of the model.

Gaussian Modeling. For each spatial position (h, w), PaDiM estimates a multivariate Gaussian distribution from the training embeddings:

$$\mathcal{N}_{h,w}(\mu_{h,w}, \Sigma_{h,w}) \tag{7}$$

where $\mu_{h,w} \in \mathbb{R}^d$ is the mean embedding and $\Sigma_{h,w} \in \mathbb{R}^{d\times d}$ is the covariance matrix. This position-specific modeling captures the expected appearance at each location, accounting for spatial structure in the images.

Anomaly Scoring via Mahalanobis Distance. At inference, the anomaly score for a test patch at position (h, w) is the Mahalanobis distance to the learned Gaussian:

$$\mathcal{M}(x_{h,w}) = \sqrt{(x_{h,w} - \mu_{h,w})^\top \Sigma_{h,w}^{-1} (x_{h,w} - \mu_{h,w})} \tag{8}$$

where $\Sigma_{h,w}^{-1}$ is the inverse covariance (precision) matrix. The image-level score is computed as the maximum Mahalanobis distance across all positions, and the spatial anomaly map is obtained by upsampling the distance values to the original image resolution.

Comparison with PatchCore. Unlike PatchCore's non-parametric memory bank, PaDiM stores only statistical summaries (mean and covariance per position), resulting in constant memory and inference time independent of training set size. However, the Gaussian assumption may not hold for all normal variations, potentially limiting its discriminative power compared to PatchCore's explicit nearest-neighbor matching.

4. Experimental Setup

This section describes the dataset, evaluation protocol, and experimental scenarios employed to validate the proposed anomaly detection methods. We detail the data splits, threshold calibration procedure, evaluation metrics, and the different domain settings under which models are tested.

4.1. Dataset and Splits

MVTec AD. We conduct experiments on the MVTec Anomaly Detection (MVTec AD) dataset [1], a comprehensive benchmark for unsupervised anomaly detection in industrial inspection. We select three representative categories: **Hazelnut** (object), **Carpet** (texture), and **Zipper** (mixed), covering the diversity of defect types and visual structures present in the dataset. Each category contains defect-free (normal) training images and a test set with both normal and anomalous samples, accompanied by pixel-level ground truth masks for localization evaluation.

Data Splitting Protocol. Following the one-class classification paradigm, we construct separate train/validation/test splits for each category. The original MVTec AD structure provides a `train/good` folder (normal images only) and `test/` folder containing both normal (`good`) and anomalous images organized by defect type. We partition these as follows:

- **Train-clean:** 80% of images from `train/good`, used exclusively to build the nominal representation (*i.e.*, memory bank for PatchCore, Gaussian parameters for PaDiM).
- **Val-clean:** Remaining 20% of `train/good` (normal) plus 30% of the anomalous images from `test/<defect>`, used for threshold calibration and hyperparameter tuning.
- **Test-clean:** All remaining normal images from `test/good` and all remaining anomalous images from `test/<defect>`, with ground truth masks. Reserved strictly for final evaluation.

The 80/20 train/validation ratio ensures sufficient training data for robust feature extraction while retaining a representative validation set. Crucially, **anomalous samples are included in the validation set** (at 30% of available anomalies) to enable threshold calibration that accounts for the score distribution of both classes. This design follows established practice in anomaly detection, where access to a small number of labeled validation anomalies is realistic and necessary for setting decision thresholds [7].

All splits are generated reproducibly using a fixed random seed (42), with split indices stored in JSON format to ensure consistency across experiments.

Table 1. Dataset split statistics for the three selected MVTec AD categories. Val-clean includes both normal (from `train/good`) and anomalous samples (30% from `test/`) for threshold calibration.

Category	**Train-clean**		**Val-clean**		**Test-clean**	
	Normal	Anom.	Normal	Anom.	Normal	Anom.
Hazelnut	312	0	79	21	40	49
Carpet	224	0	56	26	28	63
Zipper	192	0	48	36	32	84
Total	728	0	183	83	100	196

Split Statistics. Table 1 summarizes the dataset composition for each category and split.

4.2. Evaluation Protocol

Threshold Selection. For each class and method, we calibrate a decision threshold on the validation set using an F1-optimal grid search. Specifically, we:

1. Compute image-level anomaly scores for all validation samples (normal + anomalous).

2. Search over 1000 uniformly spaced thresholds between the minimum and maximum validation scores.

3. Select the threshold τ^* that maximizes the F1 score on the validation set:

$$\tau^* = \arg\max_{\tau} \mathrm{F1}(\tau) = \arg\max_{\tau} \frac{2 \cdot \mathrm{Prec}(\tau) \cdot \mathrm{Rec}(\tau)}{\mathrm{Prec}(\tau) + \mathrm{Rec}(\tau)} \quad (9)$$

Image-Level Metrics. We evaluate detection performance using both threshold-independent and threshold-dependent metrics:

- **AUROC:** Area Under the Receiver Operating Characteristic curve, measuring the model's ability to rank anomalous images higher than normal ones across all thresholds.

- **AUPRC:** Area Under the Precision-Recall Curve, particularly informative for imbalanced datasets where anomalies are the minority class.

- **F1 Score:** Harmonic mean of precision and recall at the calibrated threshold τ^*.

Pixel-Level Metrics. For localization evaluation, we compute:

- **Pixel AUROC:** ROC-AUC computed over all pixels from the entire test set, measuring discrimination between defective and non-defective pixels.

- **PRO (Per-Region Overlap):** Following the protocol of Bergmann *et al.* [1], we compute the area under the per-region overlap curve integrated up to a false positive rate of 0.3. PRO penalizes predictions that miss or only partially cover ground truth anomaly regions.

All metrics are reported per-class and as macro-averages across the three categories to provide both granular and aggregate performance views.

4.3. Evaluation Scenarios

We evaluate models under four distinct scenarios to assess both baseline performance and robustness to domain shift:

1. **Test-Clean (Baseline):** Models trained on Train-clean are evaluated on Test-clean using thresholds calibrated on Val-clean. This represents the idealized scenario where training and test conditions are matched.

2. **Test-Shift (No Adaptation):** Clean-trained models are evaluated on the shifted test set (Test-shift) using the *same thresholds calibrated on Val-clean*. This scenario measures pure performance degradation without any adaptation mechanism.

3. **Test-Shift (Threshold-Only Adaptation):** Models remain trained on Train-clean (no retraining), but thresholds are *re-calibrated on Val-shift*. This isolates the contribution of threshold adaptation.

4. **Test-Shift (Full Adaptation):** Models are *retrained from scratch on Train-shift* and thresholds are calibrated on Val-shift. This represents the upper bound of adaptation performance.

4.3.1 Global Model (Model-Unified Setting)

In addition to per-class models, we train a **single global model** on pooled normal data from all three categories (Figure 3). This setup follows the *Model-Unified* setting as defined by Heo and Kang [5]:

- **Training:** One shared model is fitted on the concatenated training data from Hazelnut, Carpet, and Zipper.

- **Inference:** Class identity is assumed *known* at test time.

- **Thresholds:** *Per-class thresholds* are calibrated on each class's validation set using the global model's predictions.

This is distinct from the more challenging *Absolute-Unified* setting [3], where class identity is unknown and a single global threshold must be used. The Model-Unified

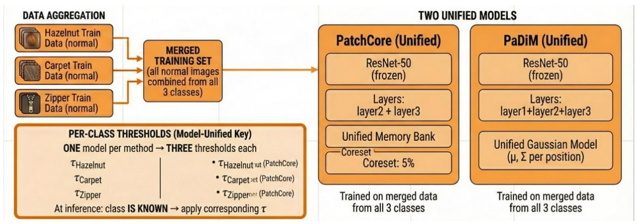
DATA AGGREGATION
Hazelnut Train Data (normal)
Carpet Train Data (normal)
Zipper Train Data (normal)
MERGED TRAINING SET (all normal images combined from all 3 classes)
TWO UNIFIED MODELS
PatchCore (Unified)
ResNet-50 (frozen)
Layers: layer2 + layer3
Unified Memory Bank
Coreset
Coreset: 5%
Trained on merged data from all 3 classes
PaDiM (Unified)
ResNet-50 (frozen)
Layers: layer1+layer2+layer3
Unified Gaussian Model (μ, Σ per position)
Trained on merged data from all 3 classes
PER-CLASS THRESHOLDS (Model-Unified Key)
ONE model per method → THREE thresholds each
τHazelnut
τCarpet
τZipper
• τHazelnut (PatchCore)
• τCarpet (PatchCore)
• τZipper (PatchCore)
At inference: class IS KNOWN → apply corresponding τ

Figure 3. Overview of the Model-Unified setting. Normal training data from all three categories (Hazelnut, Carpet, Zipper) is merged into a single training set. One unified PatchCore model and one unified PaDiM model are trained on this pooled data. At inference time, class identity is known and per-class thresholds are applied accordingly.

setting is specifically designed to investigate the **"identical shortcut" problem** [8]: when a single model is trained on multiple classes, it may learn to distinguish between classes rather than between normal and anomalous samples within each class. We analyze this phenomenon by examining cross-class confusion patterns and feature space separation (see Section 5).

4.4. Methods Under Evaluation

All evaluation scenarios described above are conducted on both PatchCore and PaDiM to enable direct comparison. PaDiM serves as a baseline method, providing a reference point for assessing PatchCore's performance across all experimental conditions. The two methods share the same preprocessing pipeline, data splits, and evaluation metrics, differing only in their core anomaly scoring mechanisms.

Coreset Ablation (PatchCore Only). The coreset ratio ablation study (Section 4.5) is conducted exclusively on PatchCore, as coreset subsampling is not applicable to PaDiM's parametric Gaussian modeling.

4.5. Ablation Study: Coreset Ratio

The coreset sampling ratio ρ is a critical hyperparameter for PatchCore, controlling the trade-off between memory bank coverage and computational efficiency. Following the range suggested by Roth *et al.* [7] (1–10%), we conduct an ablation study to evaluate the impact of this parameter.

Experimental Design. We train three PatchCore variants with coreset ratios $\rho \in \{1\%, 5\%, 10\%\}$ on the shifted domain (Train-shift), calibrate thresholds on Val-shift, and evaluate on Test-shift. This configuration isolates the effect of coreset size by keeping the domain fixed (shifted), avoiding confounding factors from domain mismatch.

For each configuration, we record:

- **Detection metrics:** AUROC, AUPRC, F1, and Accuracy on Test-shift.
- **Efficiency metrics:** Memory bank size (number of retained patches) and training time.

Smaller coreset ratios reduce memory usage and accelerate nearest neighbor search during inference, but may underfit by discarding representative patches. Larger ratios improve coverage but increase computational cost quadratically during coreset selection. The ablation quantifies this trade-off empirically, justifying the selection of $\rho = 0.05$ as the default configuration for all other experiments. Results are reported in Section 5.

4.6. Implementation Details

All experiments are executed on a single NVIDIA T4 GPU provided by Google Colab. The complete codebase, including data preprocessing, model training, and evaluation pipelines, is implemented as a collection of Jupyter notebooks designed for reproducibility. All random operations use a fixed seed of 42.

For implementation specifics of PatchCore (backbone, feature layers, coreset subsampling, score reweighting) and PaDiM (layers, dimensionality reduction, Gaussian modeling), we refer to Section 3. The `anomalib` library is used for PaDiM, while PatchCore is implemented from scratch following Roth *et al.* [7].

5. Results and Discussion

This section presents the experimental results across the evaluation scenarios described in Section 4.3.

5.1. Clean Domain Performance

Table 2 reports the detection and localization performance of PatchCore and PaDiM on Test-clean, using thresholds calibrated on Val-clean. Both methods achieve strong baseline performance, validating the experimental setup and implementations.

Table 2. Clean domain performance (Test-clean). Best per metric in **bold**.

Method	Class	AUC	PRC	F1	pAUC	PRO
PatchCore	Hazel.	**1.00**	**1.00**	**1.00**	**.988**	**.855**
	Carpet	**.973**	**.991**	**.951**	**.988**	**.836**
	Zipper	**.982**	**.994**	**.949**	**.986**	**.833**
	Avg	***.985***	***.995***	***.966***	***.987***	***.841***
PaDiM	Hazel.	.971	.975	.867	.963	.797
	Carpet	.959	.985	.923	.984	.534
	Zipper	.862	.942	.895	.972	.797
	Avg	*.930*	*.967*	*.895*	*.973*	*.709*

Detection Performance. PatchCore demonstrates superior image-level detection, achieving near-perfect performance on Hazelnut (AUROC = 1.000, F1 = 1.000) and strong results across all categories (macro-average AUROC = 0.985, F1 = 0.966). PaDiM, while competitive, shows a consistent performance gap, particularly on Zipper (AUROC = 0.862 vs. 0.982 for PatchCore). This difference in macro-average AUROC (0.985 vs. 0.930) confirms the advantage of PatchCore's non-parametric approach.

Localization Performance. Both methods achieve excellent pixel-level AUROC (PatchCore: 0.987, PaDiM: 0.973), indicating strong discrimination between defective and normal pixels. However, the PRO metric reveals a more significant gap: PatchCore achieves 0.841 compared to PaDiM's 0.709. The PRO metric, which penalizes predictions that only partially cover defect regions, highlights PatchCore's more precise spatial localization. Notably, PaDiM's PRO on Carpet (0.534) is substantially lower than other categories, suggesting difficulty with fine-grained texture defects.

Method Comparison. The performance gap between PatchCore and PaDiM can be attributed to their fundamentally different modeling approaches:

- **Non-parametric vs. Parametric:** PatchCore's memory bank approach (Section 3.3) enables exact matching, while PaDiM's Gaussian assumption (Section 3.4) may not capture complex feature distributions.
- **Density Reweighting:** PatchCore applies density-based reweighting (see Section 3.3) to anomaly scores, reducing the influence of outliers in the memory bank. PaDiM lacks this mechanism, making it more sensitive to covariance estimation errors.
- **Trade-offs:** PaDiM offers lower memory footprint and faster inference (no nearest-neighbor search required), making it suitable for resource-constrained deployments. However, for maximum detection accuracy, PatchCore's memory bank approach proves more effective.

Figure 4 shows the ROC curves for both methods across all categories. PatchCore consistently achieves higher true positive rates at low false positive rates, reflecting its superior ranking ability. The confusion matrices (Figure 5) reveal that PatchCore's errors are primarily false positives (8 total across all classes), while PaDiM exhibits a higher false negative rate (9 missed anomalies on Carpet alone).

5.2. Impact of Domain Shift

We now evaluate the robustness of clean-trained models when tested on the shifted domain (Test-shift) *without any adaptation*. This scenario uses the same models trained on Train-clean and the same thresholds calibrated on Val-clean, isolating the pure impact of distribution shift.

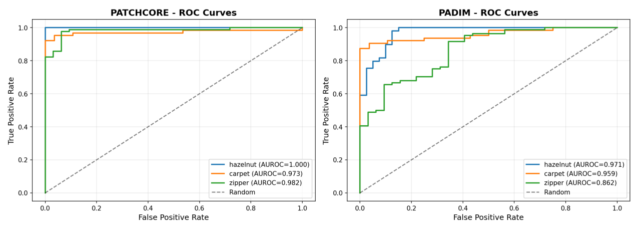

Figure 4. ROC curves on Test-clean for PatchCore and PaDiM across the three categories. PatchCore achieves near-perfect AUROC on Hazelnut and consistently outperforms PaDiM.

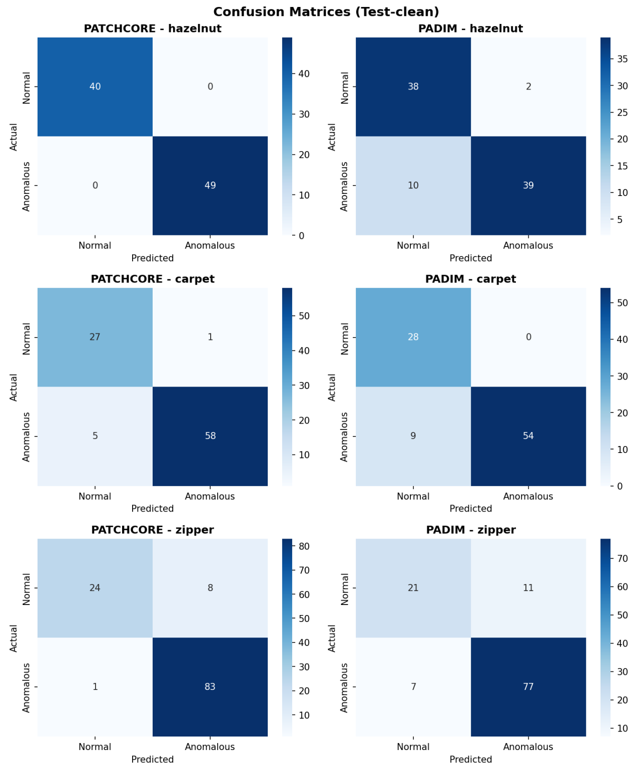

Figure 5. Confusion matrices on Test-clean at calibrated thresholds. PatchCore shows fewer false negatives, while PaDiM exhibits higher miss rates, particularly on Carpet and Hazelnut.

Experimental Setup. The domain shift transformations applied to Test-shift include geometric perturbations (rotation ± 10, scale variations, translations) and photometric changes (color jitter, Gaussian blur, noise, non-uniform illumination). See Section 3.2 for details.

Table 3 presents the performance degradation compared to the clean domain baseline.

Overall Degradation. Both methods suffer substantial performance drops when evaluated on shifted data with clean-domain thresholds. PatchCore's macro-average AUROC decreases from 0.985 to 0.892 ($\Delta = -0.093$), while PaDiM experiences a more severe drop from 0.930 to 0.684

Table 3. Domain shift degradation (No Adaptation). Macro-average metrics comparing Clean vs. Shift. Δ = Shift − Clean.

	AUROC			Specificity		
Method	Clean	Shift	Δ	Clean	Shift	Δ
PatchCore	.985	.892	-.093	.905	.223	-.682
PaDiM	.930	.684	-.246	.869	.183	-.686

$(\Delta = -0.246)$. This larger degradation for PaDiM indicates that Gaussian parametric models are more sensitive to distribution shift than memory-bank approaches.

Threshold Invalidity and Specificity Collapse. The most striking observation is the **catastrophic collapse of specificity**, dropping from ∼0.90 to ∼0.20 for both methods. For Zipper, *both methods achieve exactly 0% specificity*, meaning every normal image is misclassified as anomalous. This occurs because the domain shift causes anomaly scores to increase systematically across all samples, pushing normal images above the threshold calibrated on clean data.

Figure 6 visualizes this effect: the bottom-left quadrant (True Negatives) is nearly empty, while false positives dominate. Despite maintaining high recall (PatchCore: 1.000, PaDiM: 0.972), the models become useless for practical deployment due to excessive false alarms.

The Rotation Sensitivity. A good finding is PaDiM's disproportionate sensitivity to domain shift, particularly on Hazelnut (AUROC: 0.971 → 0.605, $\Delta = -0.366$) and Zipper (AUROC: 0.862 → 0.555, $\Delta = -0.307$). We hypothesize this is due to the ±10 rotations included in our shift transformations:

- **PaDiM's Position-Specific Modeling:** PaDiM estimates a separate Gaussian distribution for each spatial position (i, j) in the feature map [2]. When an image is rotated, features from position (i, j) in the test image correspond to a different physical location than in training. This spatial misalignment causes features to be compared against incorrect reference distributions, artificially inflating Mahalanobis distances.
- **PatchCore's Global Memory Bank:** In contrast, PatchCore's memory bank is position-agnostic: it stores patch embeddings without spatial indexing. During inference, each test patch is matched to its globally nearest neighbor regardless of position. Small rotations merely shuffle which memory bank patches are retrieved, without fundamentally breaking the matching process.

This *rotation sensitivity* explains why PaDiM, despite being competitive on clean data, degrades more severely

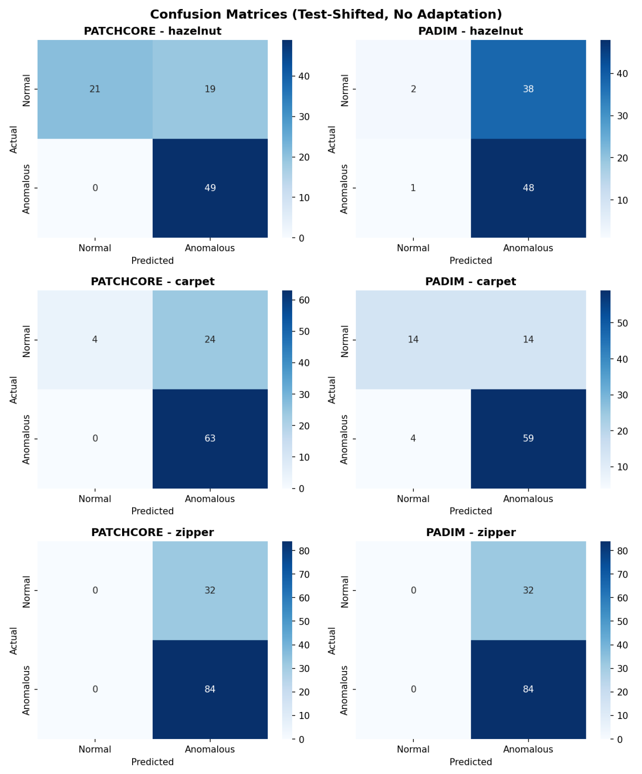

Figure 6. Confusion matrices on Test-shift (No Adaptation). Both methods exhibit near-zero specificity, with Zipper showing complete failure to identify normal samples. The threshold calibrated on Val-clean is no longer valid under domain shift.

under geometric transformations. The effect is less pronounced on Carpet, a texture category with inherent translation invariance, where PaDiM's degradation is more modest $(\Delta = -0.065)$.

Pixel-Level Degradation. Pixel-level metrics also degrade under shift, though less dramatically than image-level specificity. PatchCore's Pixel AUROC drops from 0.987 to 0.922 $(\Delta = -0.065)$, while PaDiM drops from 0.973 to 0.827 $(\Delta = -0.146)$. The PRO metric shows larger degradation (PatchCore: 0.841 → 0.645; PaDiM: 0.709 → 0.527), indicating that localization precision suffers more than pixel-level discrimination under distribution shift.

Implications. These results underscore the critical importance of threshold recalibration and/or model adaptation when deploying anomaly detection systems in environments with even modest acquisition variations. The next sections will examine whether threshold-only adaptation (Section **??**) or full model retraining (Section **??**) can recover the lost performance.

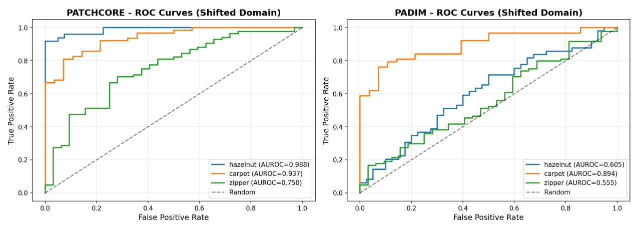

Figure 7. ROC curves on Test-shift (No Adaptation). PatchCore maintains reasonable ranking ability (macro AUROC = 0.892), while PaDiM's curves approach the diagonal (random classifier) for Hazelnut and Zipper.

6. Conclusion

References

[1] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9584–9592, 2019.

[2] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization, 2020.

[3] Jia Guo, Haonan Han, Shuai Lu, Weihang Zhang, and Huiqi Li. Absolute-unified multi-class anomaly detection via class-agnostic distribution alignment, 2024.

[4] Lars Heckler-Kram, Jan-Hendrik Neudeck, Ulla Scheler, Rebecca König, and Carsten Steger. The mvtec ad 2 dataset: Advanced scenarios for unsupervised anomaly detection, 2025.

[5] Jaehyuk Heo and Pilsung Kang. Multi-class image anomaly detection for practical applications: Requirements and robust solutions, 2025.

[6] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus, 2017.

[7] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection, 2022.

[8] Zhiyuan You, Lei Cui, Yujun Shen, Kai Yang, Xin Lu, Yu Zheng, and Xinyi Le. A unified model for multi-class anomaly detection, 2022.