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.

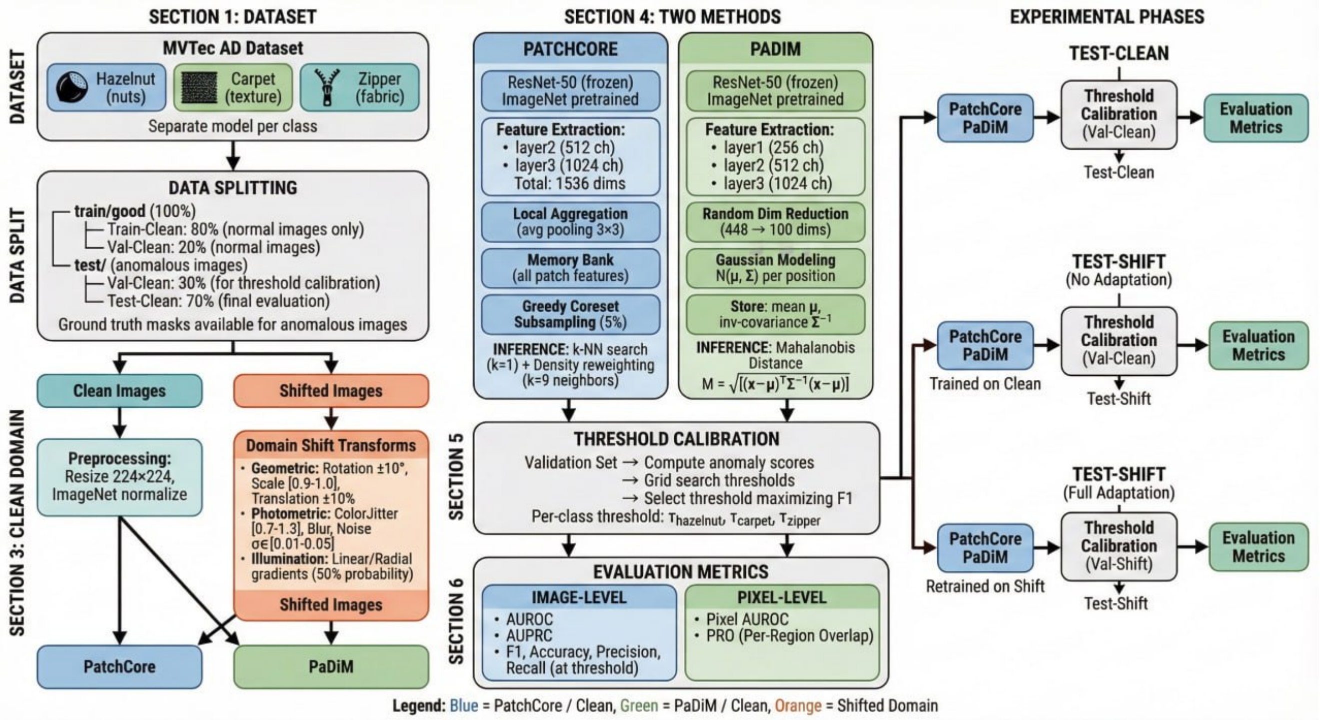
SECTION 1: DATASET
DATASET
MVTec AD Dataset
Hazelnut (nuts)
Carpet (texture)
Zipper (fabric)
Separate model per class
DATA SPLIT
DATA SPLITTING
train/good (100%)
Train-Clean: 80% (normal images only)
Val-Clean: 20% (normal images)
test/ (anomalous images)
Val-Clean: 30% (for threshold calibration)
Test-Clean: 70% (final evaluation)
Ground truth masks available for anomalous images
SECTION 3: CLEAN DOMAIN
Clean Images
Shifted Images
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)
Evaluation Metrics
Test-Clean
TEST-SHIFT (No Adaptation)
PatchCore PaDiM
Trained on Clean
Threshold Calibration (Val-Clean)
Evaluation Metrics
Test-Shift
TEST-SHIFT (Full Adaptation)
PatchCore PaDiM
Retrained on Shift
Threshold Calibration (Val-Shift)
Evaluation Metrics
Test-Shift
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 $[0.4, 0.7]$

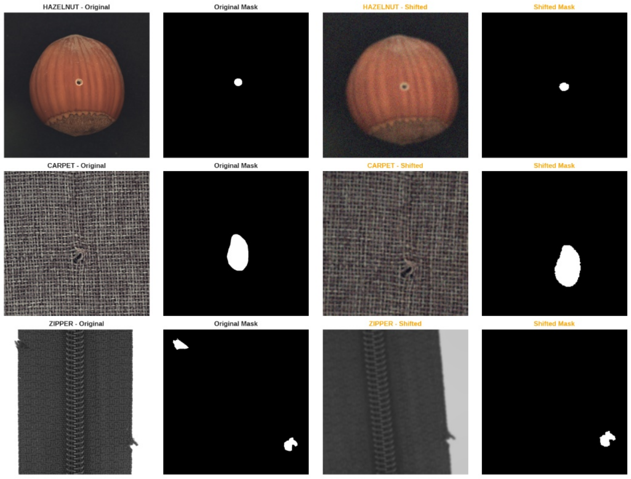

Figure 2. Comparison of clean (left) and shifted (right) samples from MVTec AD. The shifted domain includes photometric degradation, geometric perturbations, and non-uniform illumination to simulate realistic industrial variations.

- **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.

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

5. Results and Discussion

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.