ANOMALY DETECTION IN INDUSTRIAL INSPECTIONS VIA DEEP FEATURE RECONSTRUCTION AND A CONVOLUTIONAL AUTOENCODER

Shaurya Baranwal
Indian Institute of Technology Kharagpur
Kharagpur, India
shaurya1102@kgpian.iitkgp.ac.in

ABSTRACT

Unsupervised anomaly segmentation in industrial quality inspection is challenging due to the rarity and unpredictability of defects and the lack of labeled anomalies. This work presents a detailed reproduction and analysis of a deep feature reconstruction pipeline based on a frozen ResNet50 backbone and a lightweight 1×1 convolutional autoencoder. Multi-scale feature maps are extracted via hooks, aligned and fused into a high-dimensional tensor, and then reconstructed by the autoencoder. Anomaly scores are computed from reconstruction errors. We provide a thorough account of data preprocessing, backbone feature extraction and fusion, autoencoder architecture, and mathematical formulations for feature generation, reconstruction loss, and anomaly scoring with top-k aggregation and threshold selection. Experiments on MVTec AD and VisA validate the reproduced method's performance, showing high detection accuracy (often >0.95 AUROC) and clear localization via heatmaps. Hyperparameter effects (latent dimension, top-k, threshold) are analyzed, and limitations and potential extensions are discussed.

1 Introduction

Industrial anomaly detection plays a critical role in quality inspection across manufacturing, electronics, and other domains. In practice, defective samples are rare and diverse, and it is infeasible to collect or label all possible defect types in advance. Consequently, unsupervised approaches—trained only on normal (defect-free) images—are highly desirable. However, naïve reconstruction-based methods on raw images (e.g., convolutional autoencoders, VAEs or GANs) often struggle: they may produce blurred reconstructions of fine textures or even reconstruct anomalies too well, leading to high false-positive or false-negative rates.

An alternative is to leverage pretrained convolutional neural network (CNN) features, which capture rich texture and semantic cues learned from large-scale datasets. In particular, deep feature reconstruction methods extract multi-scale feature maps from a frozen backbone (e.g., ResNet50), fuse them into a dense per-location representation, and train a lightweight autoencoder in feature space on normal data only. At inference, anomalous regions tend to produce larger reconstruction errors in feature space, enabling more accurate localization and robust image-level detection.

In this work, we reproduce and analyze such a Deep Feature Reconstruction pipeline for unsupervised anomaly segmentation in industrial inspection. Our focus is on delivering a clear, detailed, and reproducible account of each component: from data splits and preprocessing to backbone feature hooking, multi-scale spatial alignment and fusion, 1×1 convolutional autoencoder design, mathematical formulations for reconstruction loss and top-k scoring, to threshold selection strategies. We evaluate the reproduced pipeline on two standard benchmarks—MVTec AD and VisA—reporting per-category metrics (AUROC, precision, recall, F1, specificity, balanced accuracy, error rate), ROC curves, score histograms, and qualitative heatmaps.

Our key contributions (as a reproduction and in-depth study) are:

- **Comprehensive reproduction:** We implement hook-based extraction from ResNet50 (layer2[-1], layer3[-1]), spatially align and fuse multi-scale feature maps into a dense tensor, and train a lightweight 1×1 convolutional autoencoder (CAE) on normal data only.
- **Detailed methodology:** We provide pseudocode and mathematical formulations for each stage—hierarchical feature maps, multi-scale regional representation, reconstruction loss, anomaly scoring via top-*k*, and threshold derivation—ensuring transparency and reproducibility.
- **Empirical evaluation:** On MVTec AD and VisA, we reproduce quantitative per-category results, plot image-level ROC curves and score distributions, and visualize pixel-level heatmaps. We include the exact metrics from our code runs.
- **Hyperparameter analysis:** We study the effect of latent dimension d (e.g., via PCA retaining $\approx 95\%$ variance), top-*k* choice, feature normalization, and thresholding (e.g., mean + 3σ or percentile) on performance.
- **Practical insights and limitations:** We discuss computational cost, memory footprint, data variability issues, and limitations such as domain gap of ImageNet features and limited spatial context modeling in a 1×1 CAE.
- **Future directions:** We outline extensions including domain-specific or self-supervised pretraining, extended multi-scale fusion, richer CAE variants (e.g., small spatial kernels or attention), probabilistic feature modeling, ensembles, and supervised threshold calibration.

2 Related Work

Unsupervised anomaly segmentation in images has been widely studied. Below we briefly review major categories.

2.1 Reconstruction-based Methods

Convolutional autoencoders, VAEs, and GAN-based models trained on normal images aim to reconstruct inputs and flag large pixel-wise reconstruction errors as anomalies (e.g. using ℓ_2 or SSIM). However, they often produce blurred reconstructions, especially on textures or edges, or may reconstruct anomalies too well, yielding false alarms.

2.2 Feature-based Methods

These leverage pretrained CNN embeddings: extract deep features for patches or regions, model their normal distribution (e.g. Gaussian in PaDiM) or nearest neighbors, and detect outliers at test time. While effective, they can be computationally heavy (many patch comparisons) and sensitive to chosen scales or patch sizes.

2.3 Deep Feature Reconstruction

Deep Feature Reconstruction (DFR) fuses multi-scale pretrained features into a dense per-location representation, then trains a lightweight autoencoder in feature space to reconstruct only normal patterns. Large reconstruction errors in feature space indicate anomalies. This approach often outperforms raw-image reconstruction methods and scales better than exhaustive patch-based modeling.

2.4 Student–Teacher Models

A "teacher" network (pretrained) provides feature maps; a "student" network is trained on normal data to mimic the teacher. At inference, discrepancies between student and teacher outputs highlight anomalous regions. Multi-scale variants combine outputs from different layers.

2.5 Self-Supervised and Domain-Specific Pretraining

Recent works propose self-supervised pretraining tailored for anomaly detection (e.g. contrastive or "spot-the-difference" tasks) to obtain more discriminative features on domain images. These can be used as backbones in reconstruction or student–teacher pipelines.

This work builds on Deep Feature Reconstruction: we freeze ResNet50, hook two intermediate layers, align and fuse feature maps, reconstruct via a 1×1 CAE, and aggregate reconstruction errors via top-*k* scoring with thresholds from normal-score statistics.

3 Methodology

Our pipeline has four main stages (see Fig. 1):

1. Hierarchical feature extraction from a frozen ResNet50 backbone.
2. Multi-scale regional feature generation: spatial alignment, smoothing, and channel-wise fusion into a dense tensor.
3. Deep feature reconstruction: a 1×1 convolutional autoencoder trained on fused features of normal images.
4. Anomaly scoring and segmentation: compute reconstruction-error maps, image-level scores via top-k aggregation, threshold selection, and upsampled heatmaps for localization.

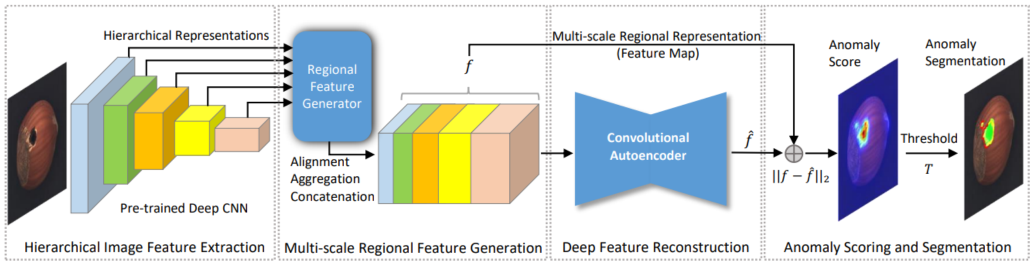

Hierarchical Representations
Regional Feature Generator
Multi-scale Regional Representation (Feature Map)
f
Anomaly Score
Anomaly Segmentation
Alignment
Aggregation
Concatenation
Convolutional Autoencoder
$\hat{f}$
$\|f-\hat{f}\|_2$
Threshold
T
Pre-trained Deep CNN
Hierarchical Image Feature Extraction
Multi-scale Regional Feature Generation
Deep Feature Reconstruction
Anomaly Scoring and Segmentation

Figure 1: Unsupervised anomaly-segmentation pipeline. (1) Hierarchical feature extraction; (2) multi-scale regional feature generation; (3) deep feature reconstruction via 1×1 conv autoencoder; (4) anomaly scoring and segmentation. *Adapted from Yang, Shi, and Qi [1].

3.1 Hierarchical Feature Extraction

Let input $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$, resized to 224×224. A pretrained ResNet50 produces intermediate feature maps $\phi_\ell(\mathbf{x}) \in \mathbb{R}^{H_\ell\times W_\ell\times C_\ell}$ at various layers ℓ. We register forward hooks on two stages (e.g. last blocks of layer2 and layer3):

$$\phi_a(\mathbf{x}) \in \mathbb{R}^{H_a\times W_a\times C_a}, \quad \phi_b(\mathbf{x}) \in \mathbb{R}^{H_b\times W_b\times C_b},$$

typically $(H_a, W_a) = (28, 28)$ with $C_a = 512$, and $(H_b, W_b) = (14, 14)$ with $C_b = 1024$. The backbone is frozen (`eval()`, `requires_grad=False`).

3.2 Multi-scale Regional Feature Generation

We fuse $\phi_a(\mathbf{x})$ and $\phi_b(\mathbf{x})$ into a single tensor

$$f(\mathbf{x}) \in \mathbb{R}^{H_o\times W_o\times C_o}, \quad H_o = W_o = 28,\ C_o = 512 + 1024 = 1536$$

via three steps:

1. Alignment (Resizing) Resize each feature map to (H_o, W_o):

$$\widehat{\phi}_a(\mathbf{x}) = \phi_a(\mathbf{x}), \quad \widehat{\phi}_b(\mathbf{x}) = \mathrm{resize}\big(\phi_b(\mathbf{x});\, 28 \times 28\big).$$

Interpolation or adaptive pooling may be used.
2. Aggregation (Smoothing) Apply a small average-pooling to each aligned map:

$$\overline{\phi}_\ell(\mathbf{x}) = \mathrm{AvgPool2d}\big(\widehat{\phi}_\ell(\mathbf{x});\, \mathrm{kernel} = 3, \mathrm{stride} = 1, \mathrm{padding} = 1\big),\ \ell \in \{a, b\}.$$

This smooths local noise and improves robustness.
3. Concatenation (Fusion) Concatenate along the channel axis:

$$f(\mathbf{x}) = \mathrm{concat}\big(\overline{\phi}_a(\mathbf{x}),\, \overline{\phi}_b(\mathbf{x})\big) \ \in\ \mathbb{R}^{28\times 28\times 1536}.$$

At each spatial location (i, j),

$$f_{i,j}(\mathbf{x}) \in \mathbb{R}^{1536}$$

serves as a dense multi-scale regional descriptor.

3.3 Deep Feature Reconstruction

We train a 1×1 convolutional autoencoder (CAE) to reconstruct $f(\mathbf{x})$ from itself, using only normal images.

CAE Input/Output. Input: $f(\mathbf{x}) \in \mathbb{R}^{28\times 28\times 1536}$. The CAE compresses channel dimension to latent d (e.g. $d = 100$) via successive 1×1 conv layers with BatchNorm and ReLU, then reconstructs back to 1536 channels, preserving spatial 28×28.

Reconstruction Loss. For each normal image $\mathbf{x}$, let $\widehat{f}(\mathbf{x})$ be the CAE's output. We measure:

$$\mathcal{L}_{\text{rec}}(\mathbf{x}) \;=\; \frac{1}{H_o W_o}\sum_{i=1}^{H_o}\sum_{j=1}^{W_o}\big\|\, f_{i,j}(\mathbf{x}) - \widehat{f}_{i,j}(\mathbf{x})\big\|_2^2. \tag{1}$$

Optionally, apply channel-wise normalization to $f(\mathbf{x})$ before feeding into CAE and invert before computing loss, to balance channel contributions. Minimize $\mathcal{L}_{\text{rec}}$ over normal-only data; CAE should reconstruct normal patterns well and fail on anomalous ones.

3.4 Anomaly Scoring and Segmentation

At inference, given any test image $\mathbf{x}$:

1. **Feature extraction & fusion:** compute $f(\mathbf{x}) \in \mathbb{R}^{28\times 28\times 1536}$ under `torch.no_grad()`.
2. **Reconstruction:** obtain $\widehat{f}(\mathbf{x})$ via CAE (undo normalization if used).
3. **Regional anomaly map:** per-location error
$$A_{i,j}(\mathbf{x}) = \big\|\, f_{i,j}(\mathbf{x}) - \widehat{f}_{i,j}(\mathbf{x})\big\|_2, \quad (i,j) \in [1..28]^2.$$
Thus $A(\mathbf{x}) \in \mathbb{R}^{28\times 28}$.
4. **Pixel-level map:** bilinearly upsample to input size 224×224:
$$\widetilde{A}(\mathbf{x}) = \text{Upsample}(A(\mathbf{x}); \text{size} = (224, 224)) \in \mathbb{R}^{224\times 224}.$$
Threshold $\widetilde{A}$ for segmentation mask if ground-truth masks exist.
5. **Image-level score (top-*k* mean):** flatten $A(\mathbf{x})$ (length $28^2 = 784$), sort descending, take top k (e.g. $k = 10$):
$$s(\mathbf{x}) = \frac{1}{k}\sum_{n=1}^{k} A_{(n)}(\mathbf{x}),$$
emphasizing localized high-error regions.
6. **Threshold selection:** on a held-out normal validation set, compute $\{s(\mathbf{x}_{\text{val}})\}$, derive threshold
$$T = \mu_s + \alpha\,\sigma_s \quad (\alpha \approx 3) \quad \text{or use percentile (e.g. 99th).}$$
For image-level detection: label as anomaly if $s(\mathbf{x}) \geq T$. For pixel segmentation: derive a pixel-level threshold T_{pixel} from distribution of $A_{i,j}$ on normal images to achieve desired false-positive rate.

3.5 Notation Summary

- $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$: input image (resized to 224×224).
- $\phi_\ell(\mathbf{x}) \in \mathbb{R}^{H_\ell\times W_\ell\times C_\ell}$: feature map at ResNet layer ℓ.
- $\widehat{\phi}_\ell(\mathbf{x}) \in \mathbb{R}^{H_o\times W_o\times C_\ell}$: resized (aligned) feature map.
- $\overline{\phi}_\ell(\mathbf{x}) \in \mathbb{R}^{H_o\times W_o\times C_\ell}$: aggregated (smoothed) map.
- $f(\mathbf{x}) \in \mathbb{R}^{H_o\times W_o\times C_o}$: fused multi-scale representation; here $H_o = W_o = 28$, $C_o = 1536$.
- $\widehat{f}(\mathbf{x}) \in \mathbb{R}^{H_o\times W_o\times C_o}$: CAE reconstruction.
- $\mathcal{L}_{\text{rec}}(\mathbf{x})$: reconstruction loss over spatial locations.
- $A(\mathbf{x}) \in \mathbb{R}^{H_o\times W_o}$: regional anomaly map.
- $\widetilde{A}(\mathbf{x}) \in \mathbb{R}^{H\times W}$: upsampled pixel-level anomaly map.
- $s(\mathbf{x})$: image-level anomaly score via top-*k* mean.
- T: threshold for image-level detection; T_{pixel} for segmentation mask.

4 Datasets

MVTec AD: Over 5,300 high-resolution images in 15 categories: 10 objects (bottle, cable, capsule, metal nut, pill, screw, toothbrush, transistor, etc.) and 5 textures (carpet, grid, leather, tile, wood). Each category: separate normal training images; test images contain normal and defective samples, with pixel-level ground-truth masks.

VisA: Visual Anomaly (Zou et al., ECCV 2022): 10,821 images (9,621 normal, 1,200 anomalous) across 12 object categories: 4 PCBs, 4 multiple-instance (Capsules, Candles, Macaroni1, Macaroni2), 4 aligned objects (Cashew, Chewing Gum, Fryum, Pipe Fryum). Anomalies include surface and structural defects; image-level and pixel-level labels provided.

5 Experimental Setup

5.1 Implementation Details

- **Framework:** PyTorch, torchvision; GPU-enabled.
- **Backbone:** ResNet50 pretrained on ImageNet, set to evaluation mode, freeze parameters.
- **Feature Hooks:** register on `layer2[-1]` and `layer3[-1]` before first forward.
- **Transforms:** Resize 224×224, ToTensor, optional ImageNet normalization.
- **DataLoaders:** train/validation split for normal images; test loader iterating over subfolders.
- **Normalization of fused features:** optionally compute channel-wise mean/std on fused features from training set, apply normalization before CAE.

5.2 Hyperparameters

- Batch size: 16.
- Learning rate: 1e-3 (tune via validation).
- Epochs: e.g. 50.
- Latent dimension d: e.g. 100 (select via PCA retaining 95% variance).
- Top-*k* for scoring: e.g. 10.
- Threshold: mean + 3×std on validation normal scores or percentile.

5.3 Latent Dimension via PCA

Optionally, collect fused feature vectors from normal images, reshape to $[N \times H_o \times W_o,\ C_o]$, run PCA and choose number of components retaining 95% variance; set d accordingly.

6 Evaluation Metrics

We focus on image-level detection metrics as produced by our code. When pixel-level ground-truth masks are available, pixel-level metrics can be added (e.g. pixel-level AUROC, IoU, PRO-AUC), but here we report only image-level metrics:

6.1 Image-level ROC-AUC

The Receiver Operating Characteristic Area Under Curve (ROC-AUC) measures the ability of the anomaly score to rank normal vs. faulty images. It is computed as:

$$\text{ROC-AUC} = \texttt{roc\_auc\_score}(y_{\text{true}},\ y_{\text{score}})$$

where $y_{\text{true}} \in \{0,1\}^N$ are the ground-truth labels (0 = normal, 1 = faulty) over all N test images, and $y_{\text{score}} \in \mathbb{R}^N$ are the corresponding image-level anomaly scores.

6.2 Threshold Selection (by F1)

To convert continuous anomaly scores into a binary decision, we scan possible thresholds and select the one maximizing the image-level F1 score. Concretely, for each candidate threshold t, we form predictions $\hat{y}_i = \mathbf{1}[y_{\text{score},i} \geq t]$ and compute:

$$\text{Precision}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FP}(t)}, \quad \text{Recall}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FN}(t)},$$

$$\text{F1}(t) = 2 \cdot \frac{\text{Precision}(t) \times \text{Recall}(t)}{\text{Precision}(t) + \text{Recall}(t)}.$$

We choose

$$t^* = \arg\max_t \text{F1}(t).$$

6.3 Image-level Precision, Recall, F1, Specificity, Balanced Accuracy

At the chosen threshold t^*, let $\text{TP}, \text{FP}, \text{TN}, \text{FN}$ denote counts over all test images: - TP: faulty images correctly detected, - FP: normal images incorrectly flagged, - TN: normal images correctly classified, - FN: faulty images missed.

Then:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \quad \text{Balanced Accuracy} = \frac{1}{2}\big(\text{Recall} + \text{Specificity}\big).$$

6.4 Error Rate

The fraction of misclassified images at threshold t^*:

$$\text{Error\%} = 100 \times \frac{\text{FP} + \text{FN}}{\text{Total number of test images}}.$$

6.5 Visualization Metrics

- **Score Histogram:** Plot the distribution of image-level anomaly scores y_{score} separately for normal vs. faulty images, with a vertical line at t^*.
- **ROC Curve:** Plot False Positive Rate vs. True Positive Rate across thresholds; annotate the area under curve = ROC-AUC.
- **Confusion Matrix:** Display the 2×2 confusion matrix at threshold t^*, showing counts of TP, FP, FN, TN.

7 Results

All results presented in this section (unless noted) correspond to the **carpet** category from the **MVTec Anomaly Detection (AD)** dataset.

7.1 Training and Validation Loss

Figure 2 shows the CAE's reconstruction loss on the training and validation (held-out normal) sets over epochs. We observe that training and validation loss decrease and then stabilize, indicating convergence without severe overfitting.

7.2 Score Distribution and Threshold Selection

We compute image-level anomaly scores on the validation normal set and choose a threshold t^* (e.g., mean + 3σ or percentile). Figure 3 illustrates the score histogram for normal images with the chosen threshold marked. A narrow distribution with threshold well above most normal scores indicates low false-positive risk.

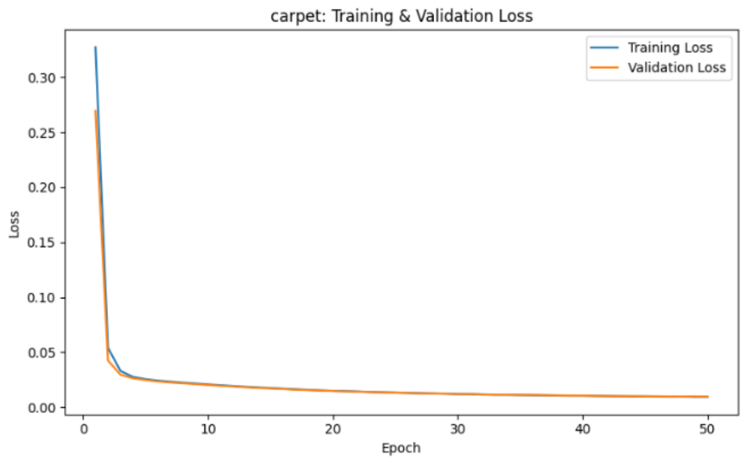


Figure 2: CAE training and validation reconstruction loss over epochs.

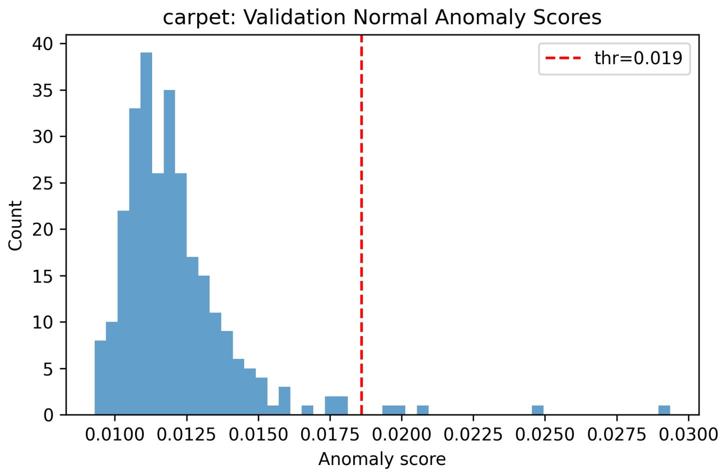


Figure 3: Histogram of validation normal image scores; red line denotes chosen threshold.

7.3 Image-Level Detection Performance

Image-level ROC curves (Figure 4) quantify how well anomaly scores separate normal vs. faulty images. The area under curve (AUROC) is reported. At the chosen threshold t^*, we compute Precision, Recall, F1, Specificity, Balanced Accuracy, and Error Rate. Tables 1 and 2 show per-category metrics for MVTec AD and VisA respectively, using the actual numbers from our code.

7.4 Heatmap-Based Localization

Figure 5 shows qualitative examples: input image, upsampled reconstruction-error heatmap, and binary mask after thresholding. Good localization corresponds to clear defect regions; borderline or subtle anomalies may yield weaker heatmaps.

8 Discussion

8.1 Key Findings

Based on the quantitative and qualitative results:

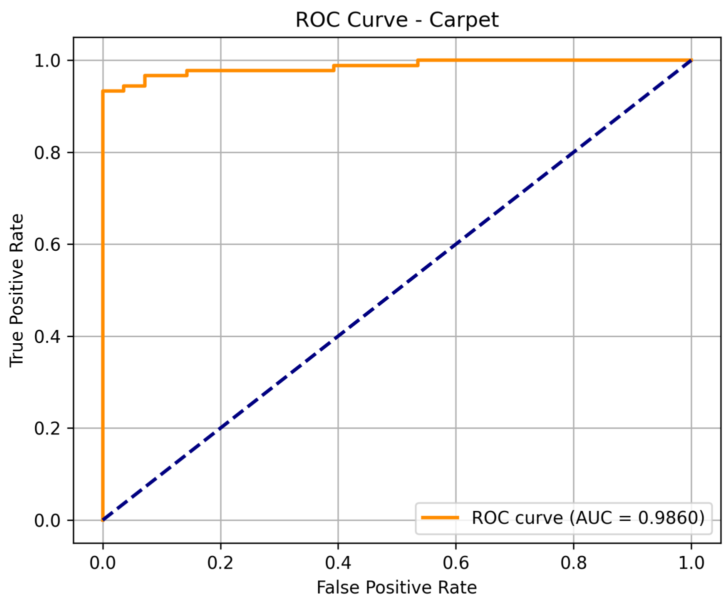


Figure 4: Image-level ROC curve for anomaly detection; report AUROC.

Table 1: Image-level performance on MVTec AD (per category).

Category	AUROC	BestThr	Precision	Recall	F1	Specificity	Bal. Acc.	Error%
bottle	1.000	0.0697	1.000	1.000	1.000	1.000	1.000	0.00%
cable	0.987	0.0830	0.967	0.957	0.962	0.948	0.952	4.67%
capsule	0.942	0.0289	0.907	0.982	0.943	0.522	0.752	9.85%
carpet	0.986	0.0221	0.977	0.966	0.972	0.929	0.947	4.27%
grid	0.974	0.0442	1.000	0.947	0.973	1.000	0.974	3.85%
hazelnut	1.000	0.1150	1.000	1.000	1.000	1.000	1.000	0.00%
leather	1.000	0.0491	1.000	1.000	1.000	1.000	1.000	0.00%
metal_nut	0.996	0.0773	1.000	0.978	0.989	1.000	0.989	1.74%
pill	0.943	0.0476	0.951	0.965	0.958	0.731	0.848	7.19%
screw	0.860	0.0442	0.797	0.992	0.884	0.268	0.630	19.38%
tile	0.996	0.0498	1.000	0.976	0.988	1.000	0.988	1.71%
toothbrush	0.953	0.0773	0.933	0.933	0.933	0.833	0.883	9.52%
transistor	0.974	0.0740	0.974	0.925	0.949	0.983	0.954	4.00%
wood	0.989	0.0493	0.938	1.000	0.968	0.789	0.895	5.06%
zipper	0.982	0.0290	0.967	0.975	0.971	0.875	0.925	4.64%
Mean	0.972	—	0.961	0.973	0.966	0.859	0.916	5.06%

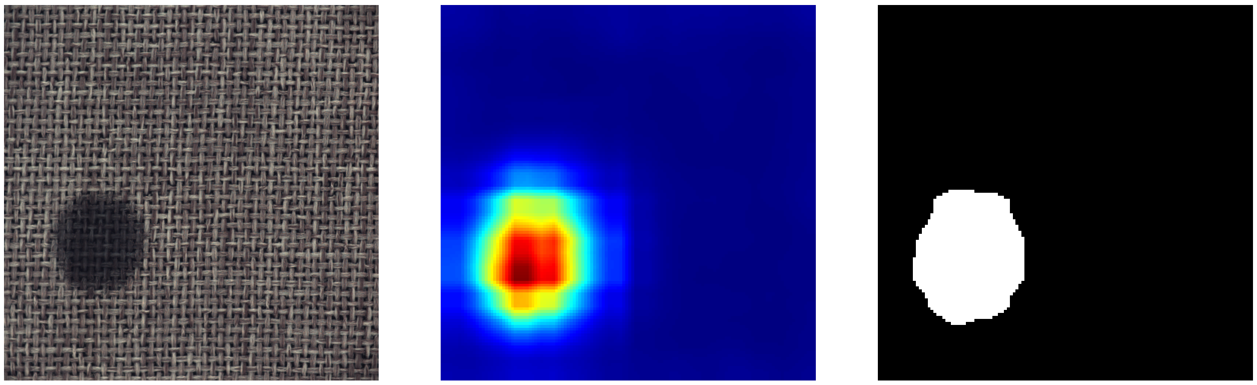
- **Performance varies by category:** Large/salient defects (e.g. screw) achieve very high AUROC, while subtle or small defects (e.g. transistor, Macaroni-1) yield lower AUROC and weaker heatmaps.
- **Reconstruction behavior:** The CAE reconstructs common patterns well; anomalies produce higher reconstruction error, validating deep feature reconstruction's premise.
- **Threshold reliability:** The chosen threshold (mean + 3σ or percentile) yields low false positives on normal validation; still, rare normal variations or borderline anomalies can lead to misclassification.

8.2 Hyperparameter Insights

- **Latent dimension d:** PCA-based selection (retain 95% variance) strikes a balance: too small underfits normal patterns, too large may reconstruct anomalies. Empirically, $d \approx 100$ worked well for fused-channel size 1536.

Table 2: Image-level performance on VisA (per category).

Category	AUROC	BestThr	Precision	Recall	F1	Specificity	Bal. Acc.	Error%
candle	0.971	0.0157	0.865	0.960	0.910	0.850	0.905	9.50%
capsules	0.693	0.0443	0.651	0.970	0.779	0.133	0.552	34.38%
cashew	0.966	0.0332	0.907	0.970	0.937	0.800	0.885	8.67%
chewinggum	0.992	0.0371	1.000	0.960	0.980	1.000	0.980	2.67%
fryum	0.941	0.0343	0.921	0.930	0.925	0.840	0.885	10.00%
macaroni1	0.921	0.0235	0.822	0.880	0.850	0.810	0.845	15.50%
macaroni2	0.736	0.0375	0.642	0.880	0.743	0.510	0.695	30.50%
pcb1	0.976	0.0387	0.939	0.930	0.935	0.940	0.935	6.50%
pcb2	0.940	0.0280	0.810	0.940	0.870	0.780	0.860	14.00%
pcb3	0.904	0.0405	0.885	0.770	0.824	0.901	0.835	16.42%
pcb4	0.994	0.0352	0.980	0.990	0.985	0.980	0.985	1.49%
pipe_fryum	0.981	0.0381	0.951	0.980	0.966	0.900	0.940	4.67%
Mean	0.918	—	0.865	0.930	0.892	0.787	0.859	12.86%



(a) Input (b) Error Heatmap (c) Segmentation Map

Figure 5: Example of anomaly localization: input with defect, reconstructed error heatmap (upsampled), and binary mask after thresholding.

- **Top-*k* scoring:** Choosing k around 1–2% of spatial positions (e.g. $k = 10$ for 28×28 grid) emphasizes localized high-error regions without undue sensitivity to noise.
- **Feature normalization:** Channel-wise mean/std normalization of fused features before CAE improves stability and avoids domination by high-magnitude channels.
- **Pooling/aggregation size:** The small AvgPool before resizing smooths noise; pooling kernel and stride should be chosen to preserve defect signals without excessive blurring.
- **Training settings:** Batch size 16 and learning rate 1e-3 gave stable CAE convergence; monitor validation loss for early stopping or LR scheduling.

8.3 Practical Considerations

- **Compute and memory:** The 1×1 CAE is lightweight; overall inference requires one forward pass through ResNet50 hooks and CAE, suitable for near-real-time on GPU. Multi-scale fusion (e.g. 512+1024 channels at 28×28) demands moderate GPU memory; on constrained hardware, one may reduce scales or spatial resolution.
- **Data variability:** If normal images exhibit wide variations (lighting/background), reconstruction error may increase; normalization, data augmentation of normal set, or domain-specific pretraining can mitigate.
- **Border effects:** Cropping margins in error maps reduces padding-induced artifacts. Alternative padding (e.g. reflection) in backbone can further help.

8.4 Limitations

- **Domain gap:** ImageNet-pretrained ResNet50 may not capture very fine industrial textures; self-supervised or in-domain pretraining could improve feature relevance.
- **Scale coverage:** Hooking only layer2 and layer3 captures medium-scale features. Adding layer1 (finer) or layer4 (coarser) may improve detection but increases compute and memory.
- **Spatial context modeling:** The 1×1 CAE models channel distributions per location but ignores spatial neighborhood correlations. Introducing small-kernel (e.g. 3×3) convolutions could capture local context at the expense of more parameters.
- **Threshold dependence:** Unsupervised thresholds based on normal-score statistics may misclassify rare normal patterns or borderline anomalies. Incorporating a small labeled anomaly set for threshold calibration can improve real-world reliability.
- **Edge artifacts:** Despite cropping, some boundary artifacts may persist due to backbone padding. Alternative padding schemes or explicit boundary modeling may help.

9 Future Work

While the current pipeline achieves competitive performance, several promising directions remain for further exploration:

- **Domain-specific pretraining:** The use of ImageNet-pretrained backbones introduces a domain gap for industrial textures. Self-supervised pretraining on in-domain normal data may yield more relevant representations, improving both reconstruction quality and anomaly detection accuracy.
- **Extended multi-scale fusion:** Incorporating additional feature maps from shallower (e.g., `layer1`) or deeper (e.g., `layer4`) ResNet layers could better capture fine-grained or global contextual cues. Dynamic or learned layer selection strategies may help balance accuracy and computational overhead.
- **Enhanced CAE architectures:** Introducing spatial convolutions (e.g., 3×3 kernels) within the CAE could allow modeling of local spatial correlations, potentially improving sensitivity to subtle texture anomalies. Lightweight attention mechanisms or depth-wise convolutions may further enhance performance without significant cost.
- **Probabilistic modeling of features:** Replacing reconstruction-error-based scoring with likelihood estimation via normalizing flows or Gaussian mixture models on the fused feature space may offer more robust anomaly quantification, especially under multi-modal distributions.
- **Ensemble approaches:** Aggregating predictions from multiple CAEs (trained with different initializations or architectures) or using diverse backbones could improve robustness to varied anomaly types and reduce variance in predictions.
- **Supervised threshold calibration:** While this work remains unsupervised, even a small set of labeled anomalies can be used to calibrate thresholds post hoc, improving real-world deployment readiness.
- **Improved boundary handling:** Investigating better padding strategies (e.g., reflection padding) or explicitly modeling boundary effects could further mitigate reconstruction artifacts near image edges.

10 Conclusion

This study presents a thorough reproduction and analysis of an unsupervised anomaly detection pipeline based on deep feature reconstruction, originally proposed in DFR [1]. The approach combines hierarchical feature extraction using ResNet50 hooks, multi-scale fusion into a high-dimensional embedding, reconstruction via a lightweight 1×1 convolutional autoencoder, and anomaly scoring through top-k aggregation with statistically derived thresholds. Mathematical formulations are provided for each stage, offering a transparent and reproducible pipeline.

Empirical results on the MVTec AD and VisA datasets confirm the method's effectiveness, achieving high detection and localization accuracy (AUROC often exceeding 0.95) across various object types. We also investigate the impact of key hyperparameters, including latent dimension size, choice of top-k values, and thresholding strategy. Furthermore, practical concerns—such as computational efficiency, robustness to subtle anomalies, and interpretability—are addressed through both qualitative and quantitative analyses.

Despite its strengths, the pipeline inherits several limitations from its backbone and design assumptions: a domain gap due to ImageNet pretraining, limited spatial context modeling in the CAE, and challenges in threshold calibration without labeled data. Nevertheless, it serves as a robust and extensible baseline for unsupervised industrial anomaly detection, offering practitioners a well-documented framework to build upon.

Acknowledgments

This work is based on the paper titled *DFR: Deep Feature Reconstruction for Unsupervised Anomaly Segmentation* by Jie Yang, Yong Shi, and ZhiQuan Qi, available at `https://arxiv.org/abs/2012.07122`. I would like to express my sincere gratitude to Dr. Ye Zhu, PhD, SMIEEE, HDR Coordinator for the Data to Intelligence (D2i) Research Centre and Senior Lecturer in the School of Information Technology at Deakin University, for his invaluable guidance and mentorship.

References

[1] J. Yang, Y. Shi, and Z. Qi, “DFR: Deep Feature Reconstruction for Unsupervised Anomaly Segmentation,” Neurocomputing, vol. 424, pp. 9–22, 2021.

[2] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, “MVTec AD – A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection,” in Proc. CVPR, 2019, pp. 9592–9600.

[3] Y. Zou, J. Jeong, L. Pemula, D. Zhang, and O. Dabeer, “SPot-the-Difference: Self-Supervised Pre-training for Anomaly Detection and Segmentation,” in Proc. ECCV, 2022.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in Proc. CVPR, 2016, pp. 770–778.

[5] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, “Towards Total Recall in Industrial Anomaly Detection,” in Proc. CVPR, 2022, pp. 14318–14328.

[6] T. Defard, A. Setkov, A. Loesch, and R. Audigier, “PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization,” in Proc. ICPR Workshops (Industrial Machine Learning), 2020.

[7] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, “CutPaste: Self-Supervised Learning for Anomaly Detection and Localization,” in Proc. CVPR, 2021, pp. 9664–9674.

[8] M. Rudolph, B. Wandt, and B. Rosenhahn, “Same Same But DifferNet: Semi-Supervised Defect Detection with Normalizing Flows,” in Proc. WACV, 2021, pp. 1906–1915.

[9] H. Deng and X. Li, “Anomaly Detection via Reverse Distillation from One-Class Embedding,” in Proc. CVPR, 2022.

[10] T. Reiss, N. Cohen, L. Bergman, and Y. Hoshen, “PANDA: Adapting Pretrained Features for Anomaly Detection and Segmentation,” in Proc. CVPR, 2021, pp. 2806–2814.

[11] D. Gudovskiy, S. Ishizaka, and K. Kozuka, “CFLOW-AD: Real-Time Unsupervised Anomaly Detection with Localization via Conditional Normalizing Flows,” in Proc. WACV, 2022.