Research Square

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

Visual Defect Detection for Manufacturing Quality Control Using Resnet-18

Omkar Addepalli

2300030012@kluniversity.in

Koneru Lakshmaiah Education Foundation

Research Article

Keywords: Anomaly detection, industrial inspection, machine vision, manufacturing quality control, ResNet-18, unsupervised learning, visual defect localization.

Posted Date: November 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11865546/v1

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

Additional Declarations: The authors declare no competing interests.

Visual Defect Detection for Manufacturing Quality Control Using ResNet-18 and Unsupervised Patch-Based Anomaly Analysis

Dr. Natha Priya, Assistant Professor, KL University
Addepalli Omkar, Nuvvula Chaturya, and Maddi Monisha Lakshmi

Abstract—Automated visual inspection is an important component of manufacturing quality assurance, yet supervised defect classifiers are difficult to deploy when defect examples are scarce, diverse, or expensive to annotate. This paper describes VisionGuard AI, a visual defect inspection system designed around unsupervised anomaly detection using normal-product imagery. The proposed workflow uses a ResNet-18 feature extractor and patch-level representations to characterize expected product appearance, then compares test-image features against a reference memory to estimate anomalous regions. The project is evaluated using the original MVTec Anomaly Detection (MVTec AD) benchmark, which contains object and texture categories and pixel-level anomaly annotations. The system combines image-level anomaly decisions with heatmap visualization, contour-derived localization, sensitivity adjustment, a conveyorline simulation, alerts, and inspection report generation. The team has reported preliminary aggregate values of 98.8% mean AUROC, 82.5% localization IoU, and latency below 70 ms; these values are not independently verified in this manuscript and require complete category-wise evaluation records, measurement definitions, and reproducible benchmarking before publication. The paper documents the design, methodology, implementation, evaluation protocol, limitations, and validation requirements so that the system can be assessed transparently and extended toward practical inspection scenarios.

Index Terms—Anomaly detection, industrial inspection, machine vision, manufacturing quality control, ResNet-18, unsupervised learning, visual defect localization.

I. Introduction

Manufacturing quality control is concerned with identifying products that do not meet visual, dimensional, material, or process specifications. In many production environments, inspection is performed at high throughput and requires consistent decisions across large numbers of items. Human inspection remains valuable because operators can adapt to unfamiliar situations, but repeated inspection tasks may be affected by fatigue, subjective judgment, lighting, viewing angle, and throughput constraints. Automated vision systems offer a way to assist inspectors by analyzing image evidence consistently, documenting decisions, and highlighting regions that warrant review.

Project Guide: Dr. Natha Priya, Assistant Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation (KLEF), Andhra Pradesh, India.

Student Authors: Addepalli Omkar, Nuvvula Chaturya, and Maddi Monisha Lakshmi, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation (KLEF), Vaddeswaram, Andhra Pradesh, India. Student author order and institutional details should be confirmed before submission.

Traditional supervised defect classification assumes that examples of the classes of interest can be collected and labelled. This assumption is often unrealistic for industrial defects. Defects may be rare, newly emerging, or caused by changing machinery, material batches, environmental conditions, and process settings. The cost of producing defective samples intentionally can be high, while annotating defect boundaries requires time and domain expertise. A classifier trained on a fixed list of known defects may therefore fail when an unanticipated defect appears. These constraints motivate learning the distribution of acceptable products and treating deviations from it as potential anomalies.

Unsupervised and one-class anomaly detection approaches address this setting by learning from normal observations. A model is exposed to defect-free examples during training and attempts to identify test inputs that differ from the learned normal representation. The objective is not to enumerate every defect class but to estimate whether a sample or region is unusual relative to the reference data. This paradigm is attractive for manufacturing because defect labels are not required during model fitting. However, anomaly detection remains challenging: normal products can exhibit legitimate variation, defects may be visually subtle, and the system must avoid false alarms while retaining sensitivity to small anomalies.

VisionGuard AI is a project implementation that combines patch-based feature analysis with an inspection-oriented application. The model workflow uses ResNet-18 as a feature backbone, extracts spatially organized representations, compares test features with a normal reference memory, and produces anomaly scores and heatmaps. The application layer exposes the results through a user interface intended to support review rather than replace human responsibility. The interface includes image inspection, overlays, adjustable sensitivity, a simulated conveyor workflow, warnings, and report generation.

This manuscript is written as an extended IEEE-style research draft. It documents the known project architecture and establishes an evaluation framework. It does not assume that every reported implementation detail has been independently reproduced. The exact feature layers, image resolution, memory-bank construction, threshold selection, and

benchmark results must be filled from the source code and experiment logs before submission.

The main aims are: (1) to describe a normal-only patchfeature inspection workflow; (2) to explain the role of pretrained residual features and nearest-neighbor anomaly scoring; (3) to document memory reduction and vectorized distance computation as implementation optimizations, subject to verification; (4) to connect pixel-level anomaly maps with contour-based localization and an inspection interface; and (5) to provide a transparent evaluation protocol for accuracy, localization, runtime, memory use, and system behavior.

II. Problem Definition and Motivation

Let an image x represent a product or surface captured by a camera. A production inspection system must assign an imagelevel decision and, when possible, identify the spatial locations that appear abnormal. A binary decision is convenient for downstream handling, but a single label does not explain the visual evidence. Localization is therefore important for operator trust, corrective action, and root-cause investigation.

The normal-only learning condition is formalized as follows. Let $\mathcal{D}_N = \{x_i\}_{i=1}^{N}$ denote training images known to be defect-free. No defect categories or anomalous masks are used to fit the anomaly detector. During evaluation, the system receives a test image x drawn from a mixture of normal and anomalous examples. It estimates an anomaly score $s(x)$ and a spatial anomaly map $A(x)$. A threshold τ converts the score into a binary decision: $\hat{y} = 1$ if $s(x) > \tau$, and $\hat{y} = 0$ otherwise, where 1 indicates suspected anomaly.

The central challenge is distinguishing abnormal appearance from normal variation. Surface texture, object pose, illumination, focus, manufacturing tolerances, and image acquisition noise can all influence the feature representation. An anomaly detector that is overly sensitive may flag harmless variations; one that is too conservative may miss real defects. A threshold is therefore not universally optimal across product categories, and deployment should include category-specific validation and operator review.

The practical motivation for VisionGuard AI is to make normal-only anomaly analysis accessible through a workflow that includes visual evidence and a simulated production response. The system is intended to be adaptable to product categories without collecting labeled defect examples for every possible defect. The benchmark evaluation on MVTec AD provides a controlled starting point, while industrial deployment would require additional images collected under the target camera, optics, lighting, line speed, and product variation.

III. Scope, Assumptions, and Research Questions

The study focuses on two-dimensional RGB or grayscale image-based inspection. It does not directly measure dimensional tolerances, subsurface defects, chemical composition, or mechanical properties. An anomaly detected in the image is a visual deviation and should not automatically be interpreted as a confirmed manufacturing fault. The application is framed as an inspection-support tool that can flag candidates for rejection or human review.

The initial assumptions are: (a) training images represent defect-free products; (b) the camera and imaging conditions are sufficiently stable for feature comparisons; (c) the test image contains a product or surface similar to the modeled category; (d) defect regions are spatially meaningful in the image; and (e) the ground-truth masks in the benchmark are used only for evaluation, not for fitting the normal reference memory. These assumptions should be explicitly revisited when transferring the method to a production line.

The manuscript addresses the following research questions. RQ1: Can pretrained patch features support anomaly detection without defect labels? RQ2: How effectively does patchlevel nearest-neighbor distance identify anomalous regions? RQ3: What trade-offs arise when the reference memory is compressed? RQ4: What inference latency and memory requirements result from the implementation on the specified hardware? RQ5: How can the anomaly map be integrated into an inspection workflow that communicates decisions and evidence to an operator? These questions guide the evaluation and help prevent claims that are not supported by measurements.

IV. Background and Related Work

A. Industrial Visual Inspection

Machine vision has long been used for measurement, surface inspection, assembly verification, and defect screening. Classical pipelines rely on carefully designed illumination, geometric rules, texture statistics, filters, or template matching. Such methods can be efficient and interpretable for stable processes, but may require extensive engineering for new products and may be sensitive to changes in appearance. Deep representations provide a flexible alternative by encoding image patterns at multiple levels, but supervised training remains limited when defective examples are not available or representative.

B. Normal-Only Anomaly Detection

Normal-only anomaly detection learns from observations of acceptable data and detects deviations at inference time. Methods can be broadly grouped into reconstruction-based methods, distribution modeling, embedding-based methods, memory-based methods, and teacher-student approaches. Reconstruction models learn to reproduce normal appearance and use reconstruction errors as signals; distribution-based approaches estimate feature statistics; embedding methods

use representations from pretrained networks; memory-based methods retain reference features and score distances; and teacher-student methods compare feature predictions. These approaches differ in the assumptions they make about normal variation and in their ability to localize small defects.

C. MVTec AD Benchmark

The MVTec AD dataset was introduced to support unsupervised anomaly detection research and includes 15 categories spanning object and texture inspection. Its training partitions contain normal images, while the test partitions contain normal and anomalous examples. Pixel-accurate masks are provided for anomalous regions, enabling both image-level detection and pixel-level localization evaluation. The benchmark contains 5,354 images, including 3,629 training images and 1,725 test images, as described in the dataset publications [1], [2]. The diversity of categories, defect types, and image appearance makes the benchmark useful for evaluating a detector beyond a single surface or object.

D. PatchCore

PatchCore is a memory-based anomaly detection method that uses pretrained feature representations and a representative memory of normal patch features. It extracts local feature embeddings, stores a subset of normal embeddings, and compares test features against that reference using nearest-neighbor distance. It is designed for the cold-start setting in which normal samples are available but anomalous examples are not [3]. PatchCore's design is directly relevant to VisionGuard AI because the project adopts a patch-feature reference approach. The use of a representative subset or coreset is part of the method family; therefore, the manuscript must avoid claiming coreset selection itself as a new invention.

E. Residual Networks

Residual networks introduce identity shortcuts that allow deeper convolutional networks to learn residual transformations and improve optimization of deep models [4]. ResNet18 is a comparatively compact backbone and can be used to extract intermediate features that retain spatial information. In anomaly detection, pretrained features may provide useful representations of textures, edges, shapes, and object parts even when the downstream task does not have defect labels. The suitability of the backbone depends on category, resolution, and the feature layers selected.

F. Related Methods and Positioning

Other widely studied approaches include PaDiM, which models patch-feature distributions, and FastFlow, which uses flow-based density estimation for anomaly detection. These methods provide useful comparison points because they represent different modeling choices. A valid comparison must use consistent dataset splits, image processing, metrics, hardware, and timing boundaries. Published numbers should be cited as literature results and not presented as direct measurements from VisionGuard AI unless the implementations are rerun under a controlled setup.

V. Dataset Description

A. MVTec AD Composition

The project uses the original MVTec AD dataset. The benchmark contains five texture categories and ten object categories. Texture categories include carpet, grid, leather, tile, and wood. Object categories include bottle, cable, capsule, hazelnut, metal nut, pill, screw, toothbrush, transistor, and zipper. The dataset contains high-resolution images and a variety of defect types such as scratches, contamination, dents, missing parts, and structural irregularities. Pixel-level ground truth masks are provided for anomalous test images [1], [2].

B. Official Splits and Data Integrity

For reproducible evaluation, the official split should be retained. Normal images in the training partition are used to construct the feature reference memory. The test partition contains normal and anomalous examples and is used to calculate metrics. No test anomaly masks, test images, or testderived thresholds should be used during fitting. The exact dataset version, download date, directory structure, and any preprocessing modifications must be recorded in the final paper. The current project description confirms use of the original dataset, but the exact split integrity and preprocessing configuration still require confirmation.

TABLE I
MVTec AD Benchmark Summary

Property	Value
Total images	5,354
Training images	3,629
Testing images	1,725
Categories	15
Texture categories	5
Object categories	10
Training anomaly labels	Not used
Test anomaly masks	Provided

C. Category Diversity

Texture categories present repeating patterns or stochastic surface appearance, while object categories have spatial structure and object-specific geometry. A single model design must account for both. Textures may contain local disruptions, missing repetitions, or irregularities. Objects may contain scratches, dents, cracks, stains, deformations, or missing components. Category diversity makes aggregate metrics useful but insufficient; per-category scores are needed to identify categories where the detector performs poorly.

D. Dataset Summary Table

The dataset totals in the table are sourced from the published benchmark description and are not generated by the project. The final experiment should verify that the local dataset files match these counts and that the implementation does not accidentally include test data in memory-bank construction.

VI. System Overview

VisionGuard AI is organized as a model pipeline and a presentation/inspection pipeline. The model pipeline accepts an image, applies preprocessing, extracts spatial features, aligns feature maps, constructs patch representations, compares each test representation with the normal memory, and converts distances into anomaly scores and a heatmap. The application pipeline receives the image-level score and anomaly map, applies a threshold, derives regions, renders visual overlays, and records inspection information.

The model layer and application layer should be kept separate. Model outputs include scores, feature distances, and maps. Interface outputs include labels such as pass/review/reject, overlays, controls, and report fields. Separating these layers makes it possible to test numerical behavior without relying on the browser, and to test interface behavior with deterministic mock predictions when model execution is unavailable.

The reported implementation uses a Python backend based on FastAPI and a React frontend styled with Tailwind CSS. FastAPI provides an interface for image submission and prediction responses, while the frontend is intended to provide interactive inspection. Exact endpoint paths, request formats, response schemas, deployment settings, and error handling are not yet documented and should be extracted from the source code.

A. Pipeline Stages

The stages are: input acquisition; image validation and preprocessing; backbone feature extraction; patch aggregation and spatial alignment; normal memory-bank construction; coreset selection; nearest-neighbor scoring; score-map interpolation and smoothing; thresholding; contour and boundingbox extraction; visualization; and reporting. A production implementation should log stage timings and define clear error behavior for corrupted images, unsupported dimensions, missing model weights, and unavailable compute devices.

B. Deployment Concept

The application is described as edge-ready because the inference workflow is designed to operate on a local machine rather than requiring a cloud service for every image. Edge readiness, however, is a deployment property that depends on measured throughput, model memory, hardware compatibility, thermal behavior, camera integration, and operational safety. The current project demonstrates a local software workflow; industrial readiness requires line integration and sustained testing.

VII. Methodology: Feature Extraction

A. Image Preprocessing

An input image is converted to the expected color format and normalized using the preprocessing convention associated with the pretrained backbone. If resizing or cropping is applied, it must be documented because spatial transformations can alter defect size and localization. Resizing can reduce compute but may suppress tiny defects. The input resolution used by the current implementation is not yet confirmed and is therefore a required experimental field.

Preprocessing should be applied consistently during reference construction and inference. Inconsistent normalization, image orientation, color ordering, or resizing can produce feature differences unrelated to defects. The preprocessing function should be version-controlled and tested on representative examples from each category. The final paper should specify interpolation, aspect-ratio handling, normalization means and standard deviations, and whether grayscale images are converted to three channels.

B. ResNet-18 Feature Backbone

ResNet-18 is used as a pretrained feature extractor. The network comprises convolutional stages with residual connections and progressively lower spatial resolution. Intermediate feature maps can encode texture, edge, shape, and contextual information. The implementation description indicates that features are extracted from intermediate layers; the exact layers must be stated based on code inspection rather than assumed. The pretrained weight source, framework model version, and whether the network is frozen are also required for reproducibility.

C. Multi-Level Feature Representation

Multi-level feature extraction can combine features with different receptive fields. Shallow layers retain localized appearance cues, while deeper layers represent broader context and semantic structure. To combine layers, feature maps are typically resized to a common spatial grid and concatenated or otherwise aggregated. If the project combines two layers, the paper should specify the layer identifiers, output dimensions, interpolation method, normalization, and resulting feature dimension. These values are intentionally left as implementation placeholders.

Let feature maps from two selected layers be $F^{(a)} \in \mathrm{R}^{H_a \times W_a \times C_a}$ and $F_{(b)} \in \mathrm{R}^{H_b \times W_b \times C_b}$. A possible aligned

representation is

$$F = \mathrm{Concat}(F^{(a)}, \mathrm{Resize}(F^{(b)})), \tag{1}$$

where the spatial dimensions are aligned to a chosen grid. The exact aggregation used in the implementation must replace this generic formulation if it differs.

D. Patch Embedding

Each aligned feature map is interpreted as a set of spatial patch descriptors. If the feature grid contains $H \times W$ locations, it yields $N = HW$ descriptors, each with dimension D. The descriptor at location j is denoted $z_j \in \mathrm{R}^D$. This local representation is useful because defects may affect only a small portion of an image and may be diluted by global pooling. Patch-level descriptors preserve a correspondence between anomaly evidence and image location.

VIII. Methodology: Memory Bank Construction

A. Normal Reference Feature Collection

For each category or configured product type, the normal training images are passed through the feature extractor. Their patch descriptors are collected into a set $\mathrm{M}_{\mathrm{full}} = \{z_i\}_{i=1}^{M}$. The resulting memory represents the observed normal feature space. It is not a set of defect prototypes and does not require labels for defect types. The memory size depends on image resolution, number of training images, feature grid dimensions, and selected layers.

B. Coreset Subsampling

Large memories increase storage and nearest-neighbor computation. Coreset selection attempts to retain a subset that represents the full feature set. The project description reports kCenter Greedy selection and reduction from more than 160,000 patch vectors to 1,500 vectors. These figures are team-reported and require verification against logs and the actual featurebank dimensions. The selection procedure should document the initial seed, distance metric, number of retained points, and reproducibility seed.

C. k-Center Greedy Objective

Given the full set $\mathrm{M}_{\mathrm{full}}$, a selected subset C is intended to minimize the maximum distance between any full-memory feature and its nearest selected representative:

$$\min_{\mathcal{C} \subseteq \mathcal{M}_{\mathrm{full}},\ |\mathcal{C}|=k} \max_{z \in \mathcal{M}_{\mathrm{full}}} \min_{c \in \mathcal{C}} d(z, c). \qquad (2)$$

The greedy approximation iteratively selects a point with a large distance from the current selected set. The objective is representative coverage rather than classification accuracy directly. Reducing memory can lower runtime and storage but may remove rare normal variations, potentially increasing false positives. The effect must be measured by comparing performance and memory across coreset sizes.

D. Reported Compression

The project description reports a reduction to 1,500 reference vectors from a collection exceeding 160,000 vectors. If the original count is exactly 160,000, this would retain less than 0.94% of the original vectors; because the stated count is greater than 160,000, the exact retention percentage cannot be calculated without the actual initial count. The final paper should include exact counts and the memory size in bytes or megabytes, not only the number of vectors.

IX. Methodology: Anomaly Scoring and Localization

A. Nearest-Neighbor Distance

For a test patch feature q, the nearest-neighbor distance to the memory is

$$d(q) = \min_{m \in \mathrm{M}} \|q - m\|_2. \qquad (3)$$

A larger distance indicates that the test patch is less represented by normal reference features. The memory bank is searched for the nearest feature. Depending on the implementation, an image-level score may be based on the maximum patch distance, a selected neighborhood, or an adjusted score. The precise score function must be verified and reported because different formulations can yield different AUROC values.

B. Image-Level Decision

An image-level score aggregates patch distances. A simple maximum aggregation is $s(x) = \max_j d(z_j)$, but PatchCore variants can use additional neighborhood-based reweighting. VisionGuard's exact scoring rule is not confirmed in the information currently supplied. It must be described according to the code and not replaced by the simple equation if the implementation differs. A threshold τ is applied to the score to produce an anomaly decision. The threshold may be useradjustable in the interface, but benchmark AUROC should be calculated from continuous scores rather than thresholded labels.

C. Spatial Anomaly Map

Patch scores are arranged on the spatial feature grid to form an anomaly map. A map can be resized to the original image dimensions using interpolation. Optional smoothing can reduce discontinuities, but smoothing also changes localization sharpness and should be specified. The heatmap is intended to represent relative anomaly evidence, not a calibrated probability of defect unless calibration has been evaluated.

D. Thresholding and Sensitivity

The interface reportedly allows the inspector to change sensitivity using a slider and update the decision clientside. This is an interaction feature; its numerical behavior depends on how the threshold is mapped to score values. A higher sensitivity setting should generally lower the decision threshold and increase the number of flagged regions, while a

lower sensitivity should generally increase the threshold. The implementation should define the mapping, units, default setting, and whether pixel-level and image-level thresholds are independent. The phrase “0 ms recalculation” should be avoided in a publication unless measured; a safer description is “client-side recalculation with no additional model inference.”

E. Contour-Based Localization

The anomaly map can be thresholded into a binary mask. Connected components or contour analysis then provide candidate regions. For each component, a bounding rectangle can be calculated and drawn over the original image. The system may compute the percentage of image pixels flagged as anomalous, but this quantity is a mask-area fraction rather than a physical surface area unless camera calibration and surface geometry are available. Small components may be filtered by area; this filtering must be documented and tuned only on a validation protocol that does not use test masks for optimization.

X. Implementation Optimization

A. Vectorized Distance Computation

The project description states that native PyTorch torch.cdist is used for vectorized nearest-neighbor distance computation. Tensor operations can reduce Python-level loops and exploit optimized kernels, but speed depends on tensor sizes, device, memory pressure, precision, and library version. A correct speedup claim requires timing the same computation under controlled conditions and reporting the baseline implementation, measurement scope, warm-up policy, synchronization, and number of repetitions.

For query matrix $Q \in \mathrm{R}^{n \times D}$ and memory matrix $M \in \mathrm{R}^{k \times D}$, pairwise Euclidean distances can be represented as $D_{ij} = \|Q_i - M_j\|_2$. A vectorized routine computes a distance matrix or blocks of it, after which nearest-neighbor distances are obtained by reducing along the memory dimension. If the full matrix is too large, chunking query features avoids peak memory growth. Implementation choices such as batch size and chunk size must be recorded.

B. Reported Speedup

A 64-times speedup is reported by the project team. The baseline is not identified in the current information. It could refer to a previous Python loop, a non-vectorized implementation, a CPU computation, or another configuration. These baselines are not interchangeable. The paper should not publish the 64× figure as a validated result until a benchmark script and timing log establish a fair comparison. The latency below 70 ms also requires a clear boundary: feature extraction only, distance computation only, or end-to-end image upload through response.

C. Memory and Runtime Trade-Offs

Memory-bank reduction can decrease distance-search cost, but inference time also includes image decode, resize, backbone computation, feature alignment, heatmap interpolation, and response serialization. A complete runtime breakdown should separate these stages. A compact bank may also alter anomaly scores. Therefore, runtime optimization is evaluated jointly with detection and localization quality. The goal is not only speed but an acceptable accuracy-runtime-memory balance.

D. Potential Optimization Controls

Relevant controls include feature-layer selection, input resolution, memory-bank size, distance computation precision, chunking, device placement, batch size, and map smoothing. Each control can change either compute cost or anomaly quality. An optimization should be introduced one at a time in an ablation study when possible. If multiple changes are combined, the paper should state that the final system is evaluated as a bundle and cannot attribute improvement to one component.

XI. Application Architecture

A. Backend Services

The project uses FastAPI as the backend framework. A backend service can expose a prediction endpoint that accepts an image and returns an image-level score, predicted status, anomaly map or encoded visualization, bounding boxes, and metadata. In a robust implementation, the service should validate image type and dimensions, enforce upload limits, handle model initialization, return structured errors, and avoid blocking the event loop with long CPU-bound work. Exact endpoints and schemas are pending source-code review.

B. Frontend

The user interface is built with React and Tailwind CSS and is described as a dark-mode inspection interface. It presents image upload or selection, results, anomaly overlays, and controls. The frontend should be treated as a client for model results rather than a second source of anomaly scoring. Client-side changes to the threshold should use the returned anomaly score or map and should not silently alter benchmark predictions.

C. Triple-View Inspector

The inspector reportedly presents three synchronized views: the original image with bounding boxes, an anomaly heatmap or overlay with an opacity control, and a ground-truth comparison when labels are available. The original image is needed to preserve visual context. Heatmap transparency enables inspectors to see whether the anomaly response corresponds to a visible defect. Ground-truth comparison is appropriate for evaluation and demonstration, but should be

unavailable as an inference input in the normal deployment workflow.

D. Inspection Certificate

The application can generate a PDF report containing timestamps, defect coordinates, and sign-off fields. A report should include an image identifier, model version, category, decision threshold, anomaly score, predicted result, localized regions, time of inspection, and reviewer identification. It should avoid claiming a certified quality status unless the inspection process has been validated and formally authorized. Report templates and the PDF-generation library should be recorded for reproducibility.

XII. Conveyor-Line Simulation and Operator Alerts

A. Simulation Purpose

The conveyor simulator is a software demonstration of inspection-triggered workflow behavior. It can continuously introduce items, classify each result, update pass and reject counts, and stop the simulated line when an anomaly is detected. A simulator helps demonstrate how model output might interact with a production process. It is not equivalent to a physical conveyor, PLC, industrial camera, or safety-rated stop controller.

B. Control Logic

A simulated item can be assigned a prediction result, which triggers a state transition. Normal items increment the pass count and continue; flagged items increment the reject or review count and may pause the conveyor. The implementation should document whether simulator defects are generated by real image predictions, selected test images, or scripted events. If scripted, simulator outcomes must be labelled as demonstration events and not mixed with benchmark performance.

C. Audio and Visual Alerts

The frontend reportedly uses the Web Audio API for audio warning chimes and flashing banners. These alerts can make the simulated decision visible and audible. Alert behavior should include controls for volume, mute, acknowledgment, and repeated-event handling. A browser-based audio warning is not a substitute for industrial alarm hardware and must not be described as a safety system without formal engineering validation.

D. Human-in-the-Loop Workflow

A useful inspection workflow is: capture image; obtain model score and map; apply threshold; display suspected regions; allow the operator to inspect evidence; accept, reject, or request reinspection; and store a report. The interface should communicate uncertainty and avoid implying that the anomaly score is a probability. Human review is particularly important for unfamiliar products, lighting changes, or low-confidence decisions.

XIII. Evaluation Protocol

A. Evaluation Principles

A reproducible benchmark requires fixed dataset splits, fixed preprocessing, a documented model configuration, and a defined metric implementation. All model-fitting operations should use the normal training partition. Test masks are reserved for metric computation and visual analysis. Threshold selection should not be performed on the test set if the resulting binary metrics are intended to estimate deployment performance. AUROC is threshold-independent and should be computed from continuous scores.

B. Image-Level AUROC

Image-level AUROC measures how well anomaly scores rank defective images above normal images. It can be interpreted as the probability that a randomly selected anomalous image receives a higher score than a randomly selected normal image, subject to tie handling. It is not the same as accuracy at a fixed threshold. Scores should be computed category-wise and then aggregated with the aggregation convention stated, such as unweighted mean over 15 categories. A reported mean of 98.8% must be accompanied by the per-category values and a precise definition of the average.

C. Pixel-Level AUROC

Pixel-level AUROC evaluates whether anomaly-map values rank anomalous pixels above normal pixels. It is useful for localization but can be affected by the large number of normal pixels and by image resizing. The exact resizing and mask alignment procedures must match. The paper should report pixel AUROC separately from image AUROC and should not use the term AUROC without specifying which level is intended.

D. Intersection over Union

IoU measures overlap between predicted and ground-truth masks after thresholding. For predicted mask P and groundtruth mask G,

$$\mathrm{IoU} = \frac{|P \cap G|}{|P \cup G|}. \tag{4}$$

IoU depends on the threshold, the binary mask generation process, and treatment of normal images. The reported 82.5% localization IoU requires the exact averaging procedure, mask threshold, category coverage, and whether only anomalous test images are included. These details are currently unknown.

TABLE II
Known Experimental Hardware

Component	Reported configuration

CPU	AMD Ryzen 7 5600H
GPU	NVIDIA GeForce RTX 3050
RAM	16 GB
Storage	1 TB (type not specified)
Operating system	*To be confirmed*
Python	*To be confirmed*
PyTorch	*To be confirmed*
CUDA	*To be confirmed*

E. Additional Metrics

Precision, recall, F1-score, false-positive rate, false-negative rate, average precision, and PRO may provide complementary views. Precision and recall depend on a threshold; AUROC does not. PRO evaluates region overlap across thresholds and is commonly used in anomaly localization. For operational deployment, false alarms per inspected item and missed-defect rates at a chosen operating point are often more interpretable than a single aggregate AUROC. The selected metrics should be justified by the inspection use case.

F. Runtime Measurement

Latency measurements should specify the image resolution, device, batch size, warm-up iterations, number of timed repetitions, synchronization method, and whether preprocessing and postprocessing are included. Median and percentile latency are useful because timing distributions can be skewed. Throughput should be reported separately from single-image latency. A single latency value without a measurement boundary is not sufficient to support a deployment claim.

XIV. Experimental Environment

The currently supplied hardware information is summarized in Table II. The CPU model, GPU model, and RAM capacity are provided by the project team. The storage capacity is listed as 1 TB, but the storage type (SSD/HDD) is not confirmed. Software versions and device configuration are still required. The exact RTX 3050 variant and available VRAM must be confirmed from system diagnostics. Hardware labels can refer to different memory configurations, and the measured inference device should be recorded. The CPUonly fallback, if supported, should be benchmarked separately. The environment should also include the Python dependency versions, pretrained-weight identifier, and configuration file or commit hash.

XV. Results: Known Project Values and Pending Measurements

The team has provided preliminary aggregate values: mean AUROC of 98.8%, localization IoU of 82.5%, and inference latency below 70 ms. These are treated as reported project values rather than independently verified experimental results. No random measurements are inserted into this paper. The following table deliberately separates supplied values from missing details. These values should not be interpreted as final benchmark results until the experiments are reproduced. The paper must state whether the 98.8% value represents image AUROC or pixel AUROC, and whether it is the arithmetic mean across all 15 categories. The IoU value requires a binary threshold and a mask evaluation convention. The latency figure requires timing logs. The speedup requires a reproducible baseline and an identical workload.

TABLE III Preliminary Results Supplied by the Project Team

Metric	Reported value	Verification status
Mean AUROC	98.8%	Definition/splits pending
Localization IoU	82.5%	Threshold/averaging pending
Inference latency	<70 ms	Timing boundary/device pending
Speedup	64×	Baseline unspecified
Memory bank	1,500 vectors	Verify configuration

A. Category-Wise Result Reporting

A complete benchmark table should contain all 15 categories. Results should include image AUROC, pixel AUROC, and a localization metric where applicable. If a metric is not calculated, the cell should be marked N/A rather than estimated. Aggregate means should be recalculated from the category values using a stated method. Category-level reporting is necessary because high scores on easy categories can conceal poor performance on challenging categories.

B. Qualitative Results

Qualitative evaluation should show normal examples, anomalous examples, heatmaps, predicted masks, and groundtruth masks. Examples should include both successful detections and failure cases. The figure caption should identify the category and defect type when known, and should state whether the heatmap is normalized per image or globally. Selecting only visually strong examples can bias readers; a systematic selection procedure should be used, such as representative examples from each category and a fixed number of errors.

C. Failure Analysis

Likely failure modes include false alarms from benign texture variations, missed low-contrast defects, localization drift after resizing, edge artifacts, differences in pose or illumination, and feature mismatch caused by category-specific normal variation. Actual failure examples must be collected from the project's test run. Each failure should be classified by visual cause and by whether it affects image-level decision, localization, or both. This section remains a protocol until the error cases are inspected.

XVI. Ablation Study Plan

Ablation experiments are necessary to identify the effect of individual design choices. The project description emphasizes

memory reduction and vectorized distance computation. A controlled study should compare a full memory bank with progressively smaller coresets while holding the backbone, image resolution, and scoring method constant. The number of retained vectors could be varied across several values, for example 500, 1,000, 1,500, and 3,000, but these are proposed experiment settings—not results.

TABLE IV
REQUIRED CATEGORY-WISE EVALUATION TABLE (POPULATE FROM EXPERIMENT LOGS)

Category	Image AUROC (%)	Pixel AUROC (%)	IoU (%)
Bottle	TBD	TBD	TBD
Cable	TBD	TBD	TBD
Capsule	TBD	TBD	TBD
Carpet	TBD	TBD	TBD
Grid	TBD	TBD	TBD
Hazelnut	TBD	TBD	TBD
Leather	TBD	TBD	TBD
Metal nut	TBD	TBD	TBD
Pill	TBD	TBD	TBD
Screw	TBD	TBD	TBD
Tile	TBD	TBD	TBD
Toothbrush	TBD	TBD	TBD
Transistor	TBD	TBD	TBD
Wood	TBD	TBD	TBD
Zipper	TBD	TBD	TBD
Mean	TBD	TBD	TBD

A second ablation should compare the original distancesearch implementation with the vectorized implementation. The same queries, reference memory, device, precision, and output should be used. If the optimization changes only the computation implementation, output distances should agree within a documented numerical tolerance. The speedup should be calculated as baseline runtime divided by optimized runtime using the same timing boundary.

A third ablation can examine feature-layer selection. Shallow features may be better for local texture defects while deeper features can represent object structure. The study should compare single-layer and multi-layer feature configurations. A fourth ablation can examine image resolution. Increasing resolution may improve detection of small defects but increases feature extraction and distance computation cost. The trade-off should be summarized using accuracy and runtime curves rather than choosing the best accuracy alone.

Threshold and postprocessing ablations should evaluate the influence of heatmap smoothing, minimum component area, contour merging, and threshold selection. These parameters can improve visualization but may also alter IoU and falsepositive behavior. A threshold tuned on the test masks would create leakage; threshold selection must be separated from final evaluation or reported as exploratory analysis.

XVII. COMPARISON WITH PUBLISHED METHODS

The literature includes memory-based, statistical, flowbased, and reconstruction-based anomaly detection methods. PatchCore is the closest methodological reference because VisionGuard AI is described as PatchCore-inspired. PaDiM and FastFlow are reasonable comparison methods due to their use of pretrained features and different anomaly modeling strategies. The comparison table below is intentionally blank: published results are not directly comparable unless evaluation conditions are harmonized, and project results must be generated by actual runs. A literature comparison should include citations and clearly label reported results. A controlled comparison should use the same split, preprocessing, hardware, and metric implementation. If the team cannot rerun baseline methods, the paper may discuss published values in related work but should not claim a direct performance advantage over them.

XVIII. DISCUSSION

A. Interpretation of the Normal-Only Paradigm

The normal-only approach addresses a practical data constraint: the system can form a reference representation from acceptable products without requiring labeled defects. This does not eliminate data requirements. It still needs a representative set of normal samples and careful acquisition conditions. If normal training data omit valid variations, those variations may be treated as anomalous. Conversely, if training data contain unnoticed defects, the reference memory may learn them as normal.

TABLE V
COMPARISON FRAMEWORK FOR FUTURE CONTROLLED EXPERIMENTS

Method	Image AUROC	Pixel AUROC	Runtime
PaDiM	Literature / rerun	Literature / rerun	Measure
FastFlow	Literature / rerun	Literature / rerun	Measure
PatchCore reference	Literature / rerun	Literature / rerun	Measure
VisionGuard AI	TBD	TBD	TBD

B. Role of Patch-Level Features

Patch representations preserve local evidence and allow an anomaly map to identify candidate regions. This spatial property is useful for operator review and for calculating localization metrics. However, patch-level scoring can be sensitive to feature-grid resolution and interpolation. Very small defects may not occupy enough pixels or may be lost when input images are resized. The selected backbone and layers determine the scale of visual information retained.

C. Memory Compression

Memory compression is a practical consideration for edge deployment. A compact reference bank reduces storage and the number of nearest-neighbor comparisons. Yet aggressive compression can reduce coverage of normal feature variation. The reported 1,500-vector memory size should therefore be viewed as a design setting whose value must be supported by accuracy, runtime, and memory experiments. The best memory size may vary by category rather than being universal.

D. Interactive Thresholding

Interactive threshold adjustment can help an inspector understand the trade-off between sensitivity and false alarms. A threshold slider should be transparent: users need to know whether it affects image-level scoring, pixel masks, or both.
It can also create inconsistent inspection decisions if different operators choose different settings. A production system may therefore require controlled threshold profiles, access permissions, and saved configuration metadata for each inspection.

E. End-to-End System Value

A visual inspector that combines image score, heatmap, bounding box, report, and simulation may improve the usability of anomaly detection. The application features are engineering contributions but are not automatically evidence of better detection accuracy. Their evaluation should include response time, interface correctness, report consistency, accessibility, operator task completion, and error handling. A user study or factory pilot would be needed to substantiate claims about productivity or decision quality.

XIX. Threats to Validity

A. Internal Validity

Internal validity concerns whether reported performance is caused by the method being tested. Potential confounders include data leakage, inconsistent preprocessing, threshold tuning on test data, incomplete benchmark categories, nondeterministic coreset selection, and timing differences. Repeated runs with fixed seeds and a documented configuration can reduce these issues. The code and logs should be retained for audit.

B. Construct Validity

AUROC and IoU measure specific aspects of anomaly ranking and spatial overlap; they do not directly measure manufacturing safety, cost reduction, operator trust, or production yield. A bounding box around an anomaly is not necessarily a calibrated physical measurement. A simulated conveyor stop is not a safety-certified response. Claims should match the metric and evaluation setup.

C. External Validity

MVTec AD is a benchmark with controlled acquisition conditions and limited category coverage. Industrial products, cameras, materials, illumination, motion blur, reflections, and defect distributions may differ. A model that performs well on the benchmark may require retraining or memory-bank rebuilding for a new production line. Generalization to unseen product types should not be assumed.

D. Reproducibility

The manuscript currently lacks software versions, exact preprocessing, model layers, threshold mapping, complete results, and benchmark scripts. These omissions limit reproducibility. A release should include configuration files, environment specification, code version, seeds, category-wise metrics, and timing procedures. Dataset licensing and redistribution rules should be respected; benchmark images should not be bundled unless permitted.

XX. Ethical, Safety, and Operational Considerations

The system is intended to assist quality inspection. A flagged item should be interpreted as a candidate anomaly, and the decision to reject or rework should follow the organization's established quality procedure. Automated decisions can have economic consequences, including unnecessary scrap or missed defects. False-positive and false-negative behavior should be measured at the operating threshold and reviewed by domain specialists.

The application should preserve traceability while minimizing unnecessary personal or sensitive data. Product images, inspection reports, and operator identifiers should be stored only for defined purposes and retained according to organizational policy. If the system is deployed on a network, access control, secure transport, upload validation, and audit logging should be implemented. Images may reveal proprietary product designs; therefore, dataset and report handling should follow confidentiality requirements.

Safety-related stop mechanisms require hardware and process engineering. Browser alerts and a simulated conveyor are demonstrative. A real conveyor stop must be implemented through appropriate industrial controllers and validated against machine safety requirements. The anomaly detector should not be the sole mechanism for safety-critical control without formal risk assessment and certification.

XXI. Limitations

The present work is limited by the use of a single benchmark dataset, a normal-only training paradigm, and an image-based inspection model. The exact implementation details are not fully recorded in the information provided for this manuscript. Category-wise metrics, uncertainty, threshold calibration, and controlled comparisons are not available. The reported aggregate values require verification. The system does not currently establish robustness under camera shifts, production speed, changing illumination, unseen materials, or real factory integration.

Patch-feature anomaly detection may be less effective when defects are extremely small, when normal appearance varies substantially, or when anomalous patterns are similar to normal features. Coreset reduction can omit rare but valid normal patterns. The detector may also respond to background, alignment, or acquisition artifacts. Heatmap smoothing and contour thresholds may improve display but can distort boundaries. The frontend features have not been evaluated through a formal usability study. These limitations should be retained in the final submission rather than hidden.

XXII. Future Work

Future work includes verifying all current metrics, collecting category-wise scores, and implementing controlled baselines. Additional experiments should evaluate coreset size, feature-layer choices, image resolution, threshold strategies, and inference precision. Robustness tests should vary lighting, pose, blur, scale, and background while preserving clear data separation. The model could be extended with categoryspecific calibration, adaptive normal-memory updates under controlled review, or complementary feature representations.

For practical deployment, the next stage should connect a camera feed and a real inspection station in a controlled pilot. The pilot should measure false alarms per unit, missed defects, line throughput, operator review time, system availability, and report accuracy. Edge deployment should include packaging, resource monitoring, recovery from service failures, and model version control. A formal human-factors study could assess whether heatmaps and threshold controls improve inspection consistency. Finally, the project should provide a reproducible benchmark script and a documented release with environment details.

XXIII. Conclusion

This paper presents VisionGuard AI, a visual defect inspection application based on normal-only patch-feature anomaly analysis using a ResNet-18 backbone and the MVTec AD benchmark. The project integrates anomaly scoring and localization with a frontend inspection workflow, heatmap visualization, contour-derived bounding boxes, sensitivity adjustment, simulated conveyor behavior, alerting, and PDF reporting. The approach is motivated by industrial settings in which defect samples are scarce and unknown defect types make supervised classification difficult.

The architecture provides a practical framework for learning a reference from normal images and identifying featurespace deviations at inference time. The application layer aims to make these deviations reviewable and traceable. The project team has supplied preliminary values for AUROC, IoU, latency, and speedup; these remain provisional until their definitions and underlying measurements are verified. The paper therefore emphasizes a reproducible evaluation plan, transparent reporting, and careful separation of benchmark evidence from software demonstration. With category-wise experiments, controlled optimization studies, and a factoryoriented validation phase, VisionGuard AI can be assessed more rigorously for its suitability as an inspection-support system.

Acknowledgment

The authors acknowledge the guidance of Dr. Natha Priya, Assistant Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation. The acknowledgment wording and author contribution statements should be confirmed by the listed authors before submission.

Author Contributions

Draft placeholder: Conceptualization, methodology, implementation, dataset preparation, experiments, visualization, and manuscript preparation were performed by the student team. The specific contributions of Addepalli Omkar, Nuvvula Chaturya, Maddi Monisha Lakshmi, and Dr. Natha Priya must be reviewed and assigned accurately. Authorship should reflect substantial contributions and approval of the final manuscript.

Data and Code Availability

The MVTec AD dataset is publicly released by its authors for research use subject to its applicable terms. The VisionGuard AI implementation, configuration, evaluation scripts, and logs are not currently linked in this draft. A repository URL, release tag, and instructions should be added if the code is made publicly available. If code cannot be released, the authors should describe what can be shared for reproducibility.

Appendix A Implementation Details to Be Confirmed

This appendix lists parameters that should be extracted from the code before finalization: input dimensions; color conversion; normalization; feature layer names; feature-map alignment; descriptor dimension; patch grid; memory-bank initial count; coreset implementation; seed; distance metric; scoring equation; map interpolation; smoothing kernel; binary threshold; component-area filter; inference batch size; device selection; and backend API. Recording these details in a single

configuration table is recommended. Unknown values should remain explicitly marked rather than inferred from common PatchCore settings.

Appendix B Reproducibility Checklist

The following checklist is recommended for the final research artifact: (1) verify the original dataset version and split; (2) confirm that only normal training images are used for fitting; (3) record the feature backbone and pretrained weight identifier; (4) specify all preprocessing and scoring operations; (5) set and record random seeds; (6) store category-wise continuous anomaly scores; (7) compute metrics using a versioned script; (8) benchmark with warm-up and synchronized timing; (9) report hardware and software versions; (10) archive logs, configuration, and model memory-bank files where licensing permits; and (11) disclose all known limitations and failures.

Appendix C Recommended Benchmark Tables

The final paper should include per-category image AUROC, pixel AUROC, and localization IoU; per-category falsepositive and false-negative counts at the operating threshold; mean and standard deviation across categories; runtime components; memory-bank size; peak RAM and VRAM; and an ablation table. Each number should be traceable to a run ID. The table should include units, whether larger or smaller is better, and the number of samples used. Do not fill missing cells with invented measurements.

Appendix D Deployment Validation Checklist

Before factory deployment, verify camera calibration and focus, illumination stability, product positioning, supported category selection, minimum defect size, operating threshold, false alarm rate, missed-defect rate, throughput, model load and recovery, network behavior, report integrity, audit logging, operator training, and safe integration with line controls. Run a shadow-mode pilot in which the detector records predictions without controlling production, then compare outputs with qualified human inspection. Only after process owners approve the validation should the system be considered for operational decision support.

Appendix E Appendix A: Image Acquisition and Preprocessing Protocol

A reproducible visual inspection experiment begins with a controlled acquisition procedure. For the benchmark, image acquisition is already determined by the dataset; the model must nevertheless document how files are loaded, how channel ordering is interpreted, and how the original spatial dimensions are preserved for localization. For a production camera, the procedure should define working distance, focal length, aperture, exposure, shutter, illumination arrangement, camera orientation, background, and item positioning. The camera should be mounted so that vibration and movement do not create inconsistent blur or framing. Illumination should be selected to reveal the relevant surface defects without introducing excessive glare or shadow. Where a product is reflective, diffuse lighting or multiple illumination angles may be needed, but the best setup depends on the product and defect types.

The preprocessing pipeline should be represented as a deterministic function. Its operations should be applied in the same order during reference memory construction and inference. A preprocessing record should identify the image decoder, color conversion, resizing, crop policy, normalization, and tensor conversion. If the image is resized to a fixed square, the manuscript should explain whether aspect ratio is preserved or distorted. Aspect-ratio distortion may change local geometry; center cropping can remove border defects; padding can introduce artificial edges. The pipeline should therefore be tested with border anomalies and small defects.

The original benchmark resolution should be retained in metadata even if the model operates on a resized image. Localization masks must be mapped back to the original coordinate system using the same spatial transform. This mapping should be unit-tested by applying synthetic masks at known coordinates. For grayscale categories, channel conversion should be explicitly documented. The final paper should provide one example of the original image, the preprocessed image, and the corresponding coordinate transform. This appendix is a recommended protocol, not a claim that all these acquisition controls were implemented in the current prototype.

Appendix F Appendix B: Feature Tensor Shapes and Data Flow

The feature extraction process can be understood as a sequence of tensor transformations. An image tensor is represented as $X \in \mathrm{R}^{B\times 3\times H\times W}$, where B is batch size, H and W are spatial dimensions, and three channels are used for RGB input. The backbone produces feature tensors at multiple stages. These tensors generally differ in spatial resolution and channel count. The feature selection stage retrieves the layers chosen by the configuration. The selected layers are aligned spatially and combined to form patch descriptors.

If the aligned grid is $h \times w$, then the patch count is $N = hw$. A flattened patch feature matrix may be represented as $Z \in \mathrm{R}^{B\times N\times D}$. For memory-bank creation, the batch and patch dimensions are combined to produce a matrix of normal descriptors. For inference, the descriptors are retained with their image and spatial indices so that nearest-neighbor distances can be reshaped into an anomaly map. A common source of bugs is losing the spatial index during flattening or reshaping with an incorrect order; tests should verify that a known grid returns to the expected shape.

The exact tensor shapes depend on the input resolution and selected backbone layers. Those dimensions must be extracted from a model run and included in a final implementation table.

A recommended logging record is: input tensor shape; layer name; feature shape; alignment shape; concatenated descriptor dimension; patch count per image; total memory descriptors; selected memory descriptors; and anomaly map shape. Recording these values prevents a paper from describing an architecture that differs from the executed code.

Appendix G Appendix C: Nearest-Neighbor Search Pseudocode

The following conceptual algorithm describes the normalreference distance process. It does not assert the exact scoring implementation currently used by the project; the code should be reviewed and the pseudocode adapted to match it before submission.

The implementation should avoid materializing a pairwise distance matrix larger than memory permits. Query features can be processed in chunks. The memory dimension may also be chunked, while retaining the current minimum distance for each query. A chunked implementation should be tested against a small reference calculation and checked for numerical agreement. Device synchronization is necessary Algorithm 1 Patch Feature Anomaly Scoring

Require: Test image x, feature extractor f, normal memory M

Preprocess x using the configured transform
Extract and align feature maps $F = f(x)$
Convert F into patch descriptors $Q = \{q_i\}_{i=1}^{N}$
for each patch descriptor q_i do
    Compute distances from q_i to memory vectors
    Store the minimum distance a_i
end for
Reshape $\{a_i\}$ into a spatial score grid
Resize the score grid to image resolution
Apply configured smoothing, if enabled
Aggregate patch scores into image score $s(x)$
return image score and anomaly map

when measuring GPU timing; otherwise, the measurement may capture only the time needed to enqueue work. This pseudocode should be replaced by a code-accurate algorithm once the repository is available.

Appendix H Appendix D: Metric Definitions and Interpretation

Image-level AUROC evaluates ranking of anomalous images and normal images. For scores s_i and binary labels y_i, a receiver operating characteristic curve is constructed from the true-positive and false-positive rates across thresholds. The area under this curve summarizes ranking quality. It does not identify a threshold that is appropriate for a particular manufacturing line. A threshold is selected according to acceptable false alarms and missed defects, preferably using a validation set that represents expected deployment conditions.

Pixel-level AUROC evaluates the ranking of pixels by anomaly-map value against the pixel-level ground truth. The number of normal pixels can greatly exceed the number of anomalous pixels. This class imbalance is one reason that pixel AUROC should be supplemented by region-oriented metrics and qualitative inspection. IoU measures overlap at a single binary threshold and therefore is sensitive to threshold selection. Precision and recall describe the positive-mask quality at the selected operating point. If the benchmark includes images without anomalies, the evaluation should state how empty ground-truth masks are handled.

For localization, the threshold and postprocessing pipeline should be fixed before test evaluation. The binary predicted mask may be obtained by thresholding the anomaly map, and optionally filtering connected components. The IoU formula should be applied consistently. The paper must report whether IoU is averaged over images, categories, or all pixels, because these aggregation choices yield different results. A metric implementation should be version-controlled and validated on hand-constructed masks with known overlap.

Appendix I Appendix E: Proposed Runtime Benchmark Procedure

A runtime benchmark should be conducted on the stated Ryzen 7 5600H and GeForce RTX 3050 system after recording exact operating-system, driver, CUDA, PyTorch, and GPU memory details. The model should be loaded before timed iterations. A fixed representative image set should be used, with consistent resolution and batch size. Warm-up iterations should be performed to allow kernels, memory allocation, and caching to stabilize. The timing script should synchronize the GPU before and after the timed operation if GPU execution is used.

Report a distribution of runtimes rather than a single observation. At minimum, record median, mean, standard deviation, 90th percentile, and 95th percentile over repeated trials. Report separate timings for decode/preprocess, backbone, nearestneighbor search, map interpolation and smoothing, postprocessing, and end-to-end request. Throughput in images per second is a separate quantity from single-image latency. A batch throughput benchmark should not be used to claim single-image latency.

For a speedup claim, define the baseline and optimized versions. Both should use identical input data, feature memory, device, precision, batch size, and timing boundary. If the baseline uses a Python loop and the optimized version uses vectorized tensor operations, the benchmark should isolate that change. The speedup ratio is $T_{baseline}/T_{optimized}$. Store raw timing samples and benchmark code alongside the

manuscript. A claim of 64-times speedup should be retained only if these records support it.

Appendix J Appendix F: Proposed Coreset Ablation Protocol

The memory-bank ablation should evaluate several retained sizes while keeping the feature extractor and test set unchanged. Candidate sizes such as 500, 1,000, 1,500, and 3,000 vectors can be used as experiment settings; these are proposed values and are not experimental outcomes. The full bank should serve as a reference. For each size, report vector count, storage size, peak memory, runtime, image AUROC, pixel AUROC, and localization IoU. If the selection procedure is stochastic, repeat it using multiple seeds or report the seed and selection stability.

The experiment should investigate whether a reduced bank retains the diversity of normal features. A lower bank size may decrease nearest-neighbor search time but can create coverage gaps. Coverage can be summarized by the mean and maximum distance from full-bank descriptors to selected representatives, although such feature-space measures do not replace test performance. A plot of runtime versus AUROC or IoU can make the trade-off visible. The smallest bank should not automatically be considered best; it should be selected based on a predeclared quality requirement and resource constraints.

Appendix K Appendix G: API and Error-Handling Requirements

The backend interface should define a clear request schema and response schema. A request may contain an image file and optional category or inspection metadata. A response should include a request identifier, model version, category, image-level score, binary decision, threshold, anomaly-map representation, bounding boxes, and timing information. If the heatmap is returned as an image, its dimensions and encoding should be documented. If scores are returned as arrays, the schema should define shape, normalization, and coordinate convention.

Errors should be explicit and testable. Unsupported file formats, empty uploads, oversized images, invalid dimensions, corrupt files, unavailable model weights, inference failures, and invalid thresholds should return a structured response and should not crash the service. The API should avoid exposing stack traces or internal paths to clients. Upload size limits, request timeouts, and rate limits may be appropriate depending on deployment. The frontend should distinguish a failed inference from an anomaly result; a service failure must never be displayed as a normal product.

The exact endpoint names, HTTP methods, request payloads, response keys, and status codes are not known from the supplied description. They should be documented from the implementation. API documentation generated by FastAPI may be used as a reference, but the paper should include a concise architecture description rather than reproducing every endpoint.

Appendix L Appendix H: Functional Test Plan

Functional tests should cover image upload, valid inference, normal prediction display, anomalous prediction display, heatmap rendering, bounding-box rendering, threshold adjustment, ground-truth comparison, conveyor state changes, pass/reject counters, alert playback, report generation, and download behavior. Each test should have an identifier, initial state, input, expected behavior, observed result, and pass/fail status. The tests should be executed on the final build rather than described as successful before evidence exists.

Threshold-control tests should verify that changing the slider updates the visual decision consistently with the configured score threshold and does not unexpectedly call the model again if client-side recalculation is intended. Overlay-opacity tests should confirm that the original image remains visible and that changing opacity does not alter the stored anomaly map. Report tests should verify timestamps, coordinates, category, decision, model version, and sign-off fields. Conveyor tests should verify that scripted simulation events are clearly distinguished from actual predictions and that stop/resume state transitions are deterministic.

Backend tests should include invalid image files, large images, missing fields, model unavailable state, and timeout behavior. Frontend tests should confirm accessible labels, keyboard operation for sliders and buttons, and readable alerts.

A final table should summarize the number of test cases, passed tests, failed tests, and unresolved issues. No pass rate is included in this draft because test execution results were not provided.

Appendix M Appendix I: Example Inspection Record Schema

A consistent inspection record is useful for auditing and reproducing a decision. A suggested record contains the inspection identifier, timestamp, image filename or secure object reference, product category, model version, preprocessing version, score, threshold, decision, anomaly regions, operator action, and report identifier. Each region can include boundingbox coordinates in a clearly defined format, such as topleft x,y and width,height in original-image pixels. Coordinate conversion must be checked whenever resizing or cropping is performed.

The record should distinguish machine prediction from operator disposition. A machine prediction may be “anomaly suspected,” while the operator disposition may be “confirmed defect,” “false alarm,” or “reinspection requested.” This distinction supports later analysis of false positives and false negatives. If a report is generated, its hash or unique identifier can be stored to detect accidental replacement. The project

should not retain sensitive images longer than necessary and should restrict access to inspection records according to organizational policy.

A sample schema should be presented as a data-model illustration only. Example values should not be interpreted as actual inspection events. The final system can serialize this structure as JSON or store it in a database, but the storage implementation is not confirmed in the current project details.

Appendix N Appendix J: Glossary of Terms

Anomaly: A sample or region that deviates from the learned reference of normal appearance. An anomaly is not necessarily a confirmed manufacturing defect.

Anomaly map: A spatial map assigning anomaly scores to image locations.

AUROC: Area under the receiver operating characteristic curve; a ranking metric for binary labels and scores. Backbone: A feature-extraction network used to produce representations for a downstream task.

Coreset: A smaller subset of samples selected to approximate the coverage or structure of a larger set. Feature embedding: A numeric representation produced by a neural network from an image or local image region. Ground truth: Reference annotations used to evaluate predictions; in MVTec AD, anomalous regions have pixel-level masks.

IoU: Intersection over Union, a measure of overlap between predicted and reference masks.

Memory bank: A collection of reference feature vectors used during nearest-neighbor anomaly scoring. Normal-only training: Model fitting using normal examples without anomaly examples.

Patch feature: A descriptor associated with a local spatial region or feature-map location.

Threshold: A score boundary used to convert continuous anomaly evidence into a binary decision. Inference latency: Time required to process an input under a specified timing boundary.

Edge inference: Computation performed near the data source, such as on a local workstation or embedded device, rather than requiring a remote cloud service.

Appendix O Appendix K: Paper Completion and Submission Checklist

Before submission, the authors should confirm the title, author order, affiliations, institutional spelling, guide's role, corresponding author, email addresses, funding statement, acknowledgments, and authorship contributions. The manuscript should be checked for accurate technical descriptions, citations, and terminology. The results section should be updated only from verified experiments. All reported metrics must include definitions and aggregation rules. Comparisons should distinguish literature values from rerun baselines. Figures must be original project outputs or properly licensed materials with accurate captions.

The benchmark protocol should confirm all 15 categories, official split use, no test leakage, feature configuration, memory-bank size, coreset seed, image resolution, threshold policy, and metric implementation. Runtime reporting should confirm device, software versions, warm-up, timing boundaries, repetitions, and baseline definition. The application section should be checked against the actual repository, including API endpoints, report-generation library, simulator logic, and alert mechanism. The conclusion should be revised to reflect the final measured findings and should not repeat unverified claims.

IEEE formatting should be checked against the specific conference or journal template. Conference and journal requirements can differ in page limits, abstract length, title page details, and submission rules. This extended draft uses an IEEE-style two-column layout as a starting point, but it is not automatically compliant with any particular venue. Authors should download the venue's official template and verify the final submission requirements. The bibliography should be checked for correct volume, pages, DOI, venue, and author spellings.

Appendix P Appendix L: Suggested Figure Captions and Data Collection Plan

Fig. 1. Overview of the VisionGuard AI inspection workflow from image capture through feature extraction, memorybank comparison, anomaly map generation, region extraction, and operator review. This figure should be drawn from the actual implementation.

Fig. 2. ResNet-18 feature extraction and patch descriptor construction. Layer names, tensor shapes, and feature fusion should be annotated after confirming the implementation.

Fig. 3. Memory-bank construction and k-Center Greedy coreset selection. Include the full-bank count and retained count from experiment logs.

Fig. 4. Qualitative results showing a normal image, anomalous image, heatmap, predicted mask, and ground-truth mask. Examples should include both successful detections and failure cases.

Fig. 5. Frontend triple-view inspection interface. Use an actual screenshot and remove personal or sensitive data from the displayed record.

Fig. 6. Runtime breakdown and memory-bank size ablation. Plot measured data only and include units, error bars where appropriate, and the benchmark device.

The data collection plan is to save all per-category metrics, inference-time samples, bank-size configuration, memory usage, and qualitative test examples to a structured results directory. Each experiment should have a unique identifier and a configuration file. A result should be regenerable from the identifier, code version, configuration, and dataset split. The paper figures should be generated from those structured files

instead of manually entered values to reduce transcription errors.

References

[1] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, “MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 9592–9600.

[2] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, “The MVTec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection,” *Int. J. Comput. Vis.*, vol. 129, pp. 1038–1059, 2021, doi: 10.1007/s11263-020-01400-4.

[3] K. Roth, L. Pemula, J. Zepeda, B. Scholkopf, T. Brox, and P.¨ Gehler, “Towards total recall in industrial anomaly detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 14318–14328, doi: 10.1109/CVPR52688.2022.01392.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[5] T. Defard, A. Setkov, A. Loesch, and R. Audigier, “PaDiM: A patch distribution modeling framework for anomaly detection and localization,” in *Proc. Int. Conf. Pattern Recognit. (ICPR)*, 2021.

[6] D. Yu, X. Xie, Y. Wang, and others, “FastFlow: Unsupervised anomaly detection and localization via 2D normalizing flows,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021. [Bibliographic details to verify before submission.]