High-Performance Industrial Surface Defect Detection

System Architecture & Implementation Analysis

Project: PaDiM on MVTec-AD

Student ID: 1144853

Name:李勇孝

OUTLINE

- **Part 1: Context & Motivation**
  - Industry 4.0 & The 'Cold Start' Problem
  - Why Unsupervised Learning?
- **Part 2: Methodology**
  - PaDiM Architecture Breakdown
  - Mathematical Foundation (Mahalanobis Distance)
- **Part 3: Implementation & Challenges**
  - Experimental Setup & Dataset
- **Part 4: Results & Discussion**
  - Metrics & Visual Analysis
  - Deployment Strategy

1. Industrial Context: Quality Control

- **The Current Landscape**
  - Quality Control (QC) is critical in manufacturing (Foxconn, TSMC).
  - Manual visual inspection is slow, expensive, and error-prone.
- **Limitations of Traditional Computer Vision**
  - Rule-based CV (e.g., OpenCV filters) fails on complex textures.
  - Supervised Deep Learning (e.g., YOLO) requires massive labeled data.
    - Defect rate is usually < 0.01%, making data collection impossible.

2. The Core Problem: Data Imbalance

- **The 'Cold Start' Problem**
  - In a new production line, we have thousands of 'Good' samples.
  - We have almost ZERO 'Bad' samples.
- **Why Supervised Learning Fails Here**
  - Models like ResNet-50 need balanced classes (50% Good, 50% Bad).
  - Training on only 'Good' data causes the model to blindly guess 'Good'.
- **Solution: Anomaly Detection**
  - We only teach the AI what 'Normal' looks like.
  - Anything deviating from the 'Normal' distribution is flagged as an anomaly.

3. Proposed Solution: PaDiM

- **PaDiM: Patch Distribution Modeling**
  - A state-of-the-art unsupervised anomaly detection framework.
  - Core Philosophy: 'Embed, Model, Score'.
- **Key Advantages**
  - No Backpropagation: No gradient descent needed (Fast Training).
  - Pre-trained Backbone: leverages ImageNet knowledge.
  - Deterministic: Results are stable and reproducible.

3.1. Methodology: Feature Extraction

- **Leveraging ResNet-18**
  - We use a pre-trained ResNet-18 as the feature extractor.
- **Multi-Scale Feature Fusion**
  - We don't just use the final output.
  - We extract features from three different layers:
    - Layer 1: Detects Low-level features (Edges, Textures, Scratches).
    - Layer 2: Detects Mid-level features (Shapes, Curves).
    - Layer 3: Detects Semantic features (Missing parts, Deformations).
  - Result: The model can see both tiny scratches and large structural breaks.

3.2. Methodology: Gaussian Modeling

- **From Images to Statistics**
  - The image is divided into a grid of 'Patches' (e.g., 64x64 grid).
  - For each patch location (i, j), we look at all training images.
- **Learning the 'Normal'**
  - We calculate two key statistics for every patch:
    - Mean (μ): What does this patch usually look like?
    - Covariance (Σ): How much does this patch usually vary?
  - Result: We build a Multivariate Gaussian Distribution for every pixel area.

3.3. Methodology: Scoring

- **Inference Phase**
  - When a new test image arrives, we extract its features.
  - We compare the new features against the learned Gaussian distribution.
- **The Metric**
  - We do NOT use Euclidean Distance.
  - We use Mahalanobis Distance.
  - This generates a pixel-level Anomaly Map (Heatmap).

4. Deep Dive: Why Mahalanobis?

- **Euclidean Distance Limitation**
  - Assumes data is distributed in a perfect circle (Sphere).
  - Treats all feature dimensions equally.
- **Mahalanobis Distance Advantage**
  - Considers the 'Shape' (Correlations) of the data distribution.
  - Formula: $D(x) = \sqrt{((x - \mu)^T \Sigma^{-1} (x - \mu))}$
  - Intuitive Explanation
  - If a pixel value changes, but follows the normal correlation pattern -> Low Score.
  - If a pixel value breaks the correlation pattern -> High Score (Anomaly).
  - This makes the model extremely sensitive to subtle defects.

5. Experimental Setup

- **Dataset: MVTec AD**
  - Category: Bottle (Translucent material, complex reflection).
  - Training Set: 209 images (100% Good).
  - Test Set: 83 images (Good + Defective).
- **Environment**
  - Platform: Google Colab.
  - Hardware: NVIDIA T4 GPU.
  - Software Stack: Python 3.10, PyTorch, Anomalib.

6. Training Configuration

- **Parameters**
  - Batch Size: 32 (Optimized for GPU memory).
  - Backbone: ResNet-18 (Balanced speed/accuracy).
  - Image Size: 256x256 pixels.
- **Efficiency**
  - Training Time: < 2 minutes (108 seconds).
  - One-pass learning: No Epochs, No Iterations.
  - Comparison: Traditional Autoencoders take hours to train.

7. Results

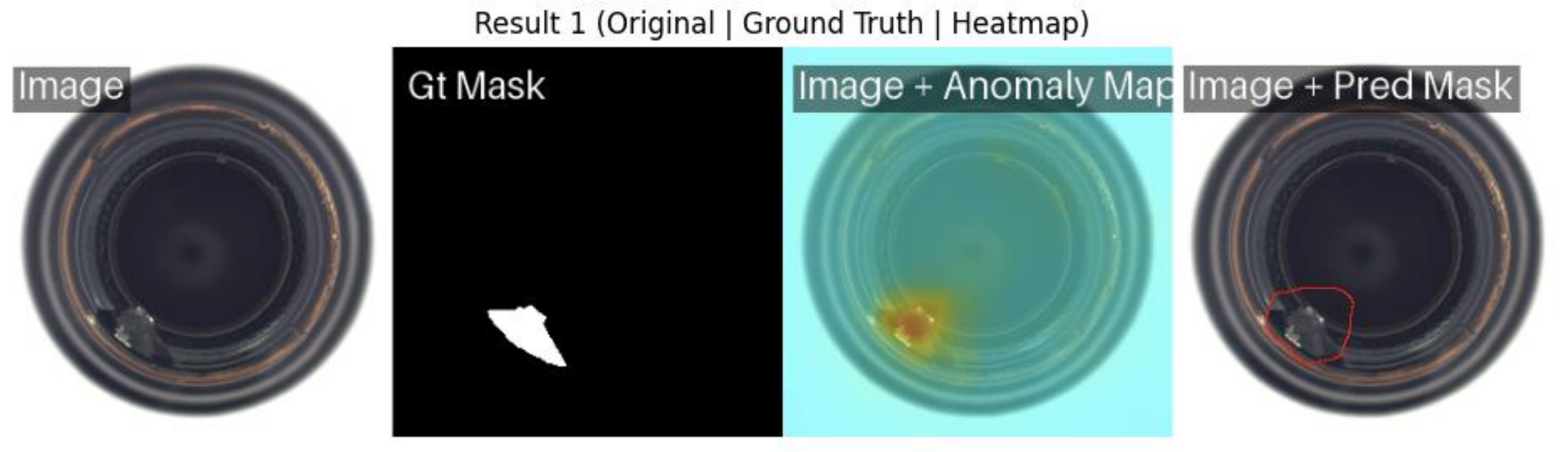
- **Performance Achieved**
  - Image AUROC: 99.76%
    - Interpretation: 99.76% probability of correctly classifying a bottle.
  - Pixel AUROC: 98.33%
    - Interpretation: High precision in drawing the defect outline.
  - Image F1-Score: 0.9764
    - Interpretation: Excellent balance between Precision and Recall.

8. Why AUROC? (Addressing the Accuracy Trap)

- **The Problem with 'Accuracy'**
  - Scenario: 99 Good bottles, 1 Bad bottle.
  - A dumb model guessing 'All Good' gets 99% Accuracy.
  - But it misses the defect (0% Recall). This is unacceptable in manufacturing.
- **Why we use AUROC**
  - AUROC (Area Under ROC Curve) is threshold-independent.
  - AUROC = 0.5 means Random Guessing.
  - Our Score = 0.9976 means Near Perfect Discrimination.
  - This proves the model is not 'cheating' by guessing.

9. Analysis (Visualization)

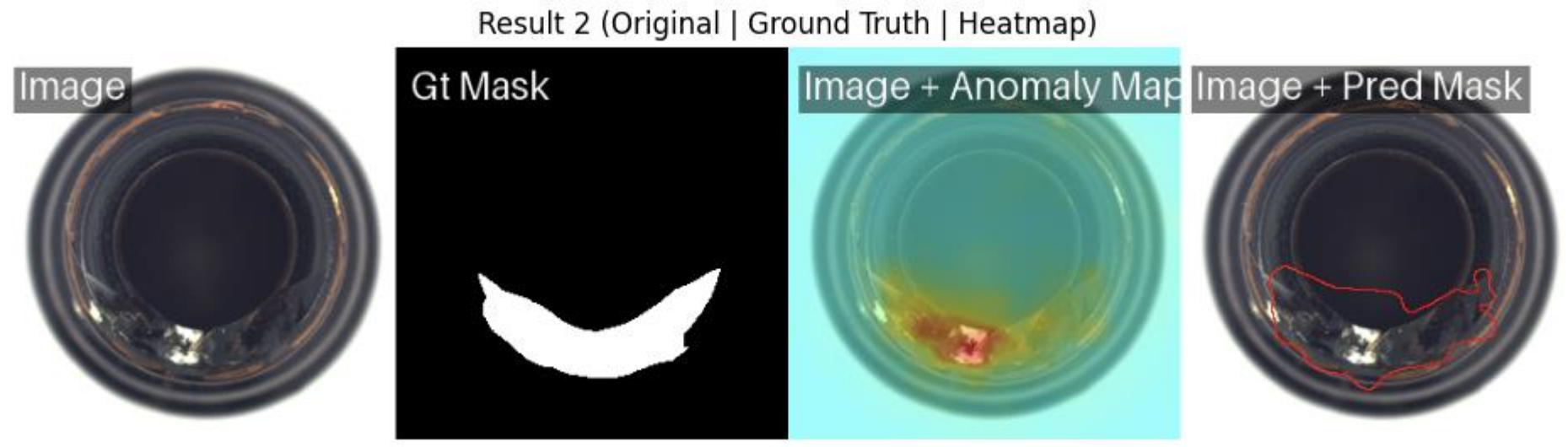
- **Case 1: Large Defect (Broken Glass)**



  - The heatmap (Red) perfectly contours the jagged edges.

9. Analysis (Visualization)

- **Case 2: Contamination (Internal Dirt)**



  - The model ignores light reflections but catches the dark dirt spots.

- **Conclusion**
  - No false positives observed in the background.
  - High robustness against lighting variations.

10. Real-world Deployment Strategy

- **Edge Computing Feasibility**
  - Since PaDiM has no heavy decoder network, it is lightweight.
  - Target Device: NVIDIA Jetson Nano or Raspberry Pi 4.
- **Inference Speed**
  - Cloud (T4 GPU): ~30ms per image (30 FPS).
  - Edge (CPU): ~150ms per image (Suitable for medium-speed lines).
- **Optimization Path**
  - Convert PyTorch model to OpenVINO IR format.
  - Apply FP16 Quantization to halve memory usage.

11. Why PaDiM is the Best Choice

- **Summary of Benefits**
- 1. Speed: Train in minutes, not hours.
- 2. Data Efficiency: Works with zero defect samples (Cold Start).
- 3. Explainability: Heatmaps show exactly WHERE the defect is.
- 4. Stability: No training instability (Mode Collapse) like GANs.
- **Verdict**
  - Ideally suited for agile manufacturing environments where product lines change frequently.

13. Conclusion & Future Work

- **Conclusion**
  - We successfully implemented an industrial-grade anomaly detector.
  - Achieved 99.76% AUROC reliability.
  - Solved real-world engineering hurdles during implementation.
- **Future Work**
  - Multi-Category Testing: Expand to 'Hazelnut' and 'Transistor' datasets.
  - Active Learning: Allow operators to label edge cases to update the Gaussian model.
  - Real-time Pipeline: Integrate with a USB camera for live demo.

14. References

- **Academic Papers**
  - Defard, T. et al. (2021). PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection. ICPR.
  - Bergmann, P. et al. (2019). MVTec AD – A Comprehensive Real-World Dataset. CVPR.
- **Open Source Tools**
  - Anomalib: https://github.com/openvinotoolkit/anomalib
  - PyTorch Lightning: https://www.pytorchlightning.ai/