Friedrich-Alexander-Universität Erlangen-Nürnberg

AnomalyAnything: Promptable Unseen Visual Anomaly Generation

Course: Advanced Machine Learning for Anomaly Detection

Submitted by:

Sajjadul Alam
Master's in Data Science
Sajjadul.alam@fau.de
Matriculation number: 23129194

Abstract

This project investigates prompt-driven synthetic anomaly generation for industrial defect detection and studies how artificially created anomalies influence feature learning and generalization capability. Building on the AnomalyAnything concept, I implement a full evaluation pipeline combining promptable anomaly creation, a denoising autoencoder (AE), and PatchCore-based anomaly detection. Experiments on MVTec AD and VisA compare synthetic-only training, real-only baselines, and contaminated training regimes with mixed real and synthetic anomalies. Results indicate that prompt-based synthetic defects enable viable anomaly detection even when real anomalies are limited, while mixed training yields the most robust performance.

I. Method and Code Extensions

Two main components are used: a denoising autoencoder (AE) that models the manifold of normal images, and PatchCore, a memory-based anomaly detection method.

A. Dataset Integration

I implemented unified loaders for MVTec AD and VisA with a normalized directory structure:

```
data/mvtec/<class>/train/good/
data/mvtec/<class>/test/good/
data/mvtec/<class>/test/<anomaly_type>/
data/visa/<class>/train/good/
data/visa/<class>/test/good/
data/visa/<class>/test/<anomaly_type>/
```

This removes the need for the full original AnomalyAnything code base and allows experiments to run with or without the original diffusion models.

B. Synthetic Anomaly Generator

A standalone synthetic anomaly module (`src/synthgen.py`) was implemented with three severity levels:

- **Mild**: subtle texture and color changes,
- **Moderate**: visible surface changes,
- **Severe**: strong structural damage or missing regions.

The module can either call a prompt-based generator (when available) or fall back to an internal mask- and noise-based corruption scheme. Generated images and their metadata are stored per severity and per seed in the `outputs/` directory.

C. Autoencoder + PatchCore Pipeline

A lightweight convolutional AE is trained per dataset and class on normal images. Its role is to denoise and regularize inputs; it is not used as the final detector. PatchCore then extracts features (e.g., from a ResNet-18 backbone), constructs a memory bank from normal embeddings, and computes nearest-neighbor distances as anomaly scores.

I extended the training and evaluation pipeline with:

- `train_ae.py`: AE training and checkpointing,
- `train_patchcore.py`: PatchCore training and scoring,
- `eval_utils.py`: ROC/PR computation and plotting,
- `experiments/run_exp1.py, run_exp2.py, run_exp3.py`: automated experimental scripts,
- `scripts/run_all.sh`: one-command reproducibility.

All experiments export tables (`results.csv`), curves (ROC/PR), sample visualizations, and model checkpoints.

II. Experimental Design

Three experiments were conducted on MVTec AD (bottle class) and optionally VisA:

A. Experiment 1: Severity-Controlled Synthetic Training

PatchCore is trained exclusively on synthetic anomalies, generated at mild, moderate, and severe levels, along with normal images. Evaluation is performed on real test anomalies. This tests whether synthetic data alone can support anomaly detection.

B. Experiment 2: Real vs. Synthetic vs. Mixed Training

Here, three regimes are compared:

1) Real-only: normal and real anomalies from MVTec/VisA,
2) Synthetic-only: normal plus synthetic anomalies,
3) Mixed: normal plus both real and synthetic anomalies.

This addresses whether synthetic anomalies can replace or should complement real data.

C. Experiment 3: Contaminated Training

To simulate noisy industrial datasets, real anomalies are injected into the "normal" training set at contamination ratios of 0%, 5%, 10%, and 20%. PatchCore is trained on these contaminated sets and evaluated on clean test splits. This reveals the robustness of the model under imperfect data assumptions.

[0]Project Repository: https://github.com/sajjadulalam/AnomalyAnything

III. Results and Discussion

Across experiments, synthetic-only training achieved reasonable ROC-AUC scores, confirming that promptable synthetic anomalies carry useful signal. However, real-only training generally remained slightly stronger when the evaluation defects matched the training distribution.

Mixed training, combining real and synthetic anomalies, produced the most robust performance, balancing coverage of diverse defect patterns with realistic distributions. In the contamination study, small amounts of contamination (up to about 5–10%) did not catastrophically harm performance and sometimes acted as a regularizer, while higher contamination ratios began to degrade generalization, as the model partially learned to treat some anomalies as normal.

The autoencoder improved stability and reduced artifacts in the features, but the final anomaly scores and metrics were always derived from PatchCore, which is the primary detector in this pipeline.

References

[1] T. Roth *et al.*, "Towards Total Recall in Industrial Anomaly Detection," in *Proc. CVPR*, 2021.

[2] Y. Mao *et al.*, "AnomalyAnything: Promptable Unseen Visual Anomaly Generation," 2024, arXiv:xxxx.xxxxx.

[3] P. Bergmann *et al.*, "The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection," *Int. J. Comput. Vis.*, 2019.

[4] Y. Zhang *et al.*, "VisA: A Versatile Benchmark for Visual Anomaly Detection," in *Proc. ECCV*, 2022.

[5] https://github.com/sajjadulalam/AnomalyAnything