Anomalib

Anomaly detection is crucial in quality control and automated inspection, especially in manufacturing, where identifying defects like scratches, dents, or missing parts is essential. The MVTec-AD dataset is a standard for benchmarking anomaly detection algorithms in industrial settings, **anomalib** is a library tailored for anomaly detection, providing various pretrained models and a streamlined interface for training, evaluation, and visualization.

Categories

The MVTec-AD dataset consists of various product categories, each containing images labeled as either normal or anomalous. For this project, we focus on three flat surface categories: tile, leather, and grid (Flat Surfaces).

Models

There are several pre-trained model available in anomalib, but in this assignment we will focus on PatchCore and Efficient AD

1. PatchCore

- PatchCore is a model designed specifically to address the challenges of anomaly detection in large datasets by reducing memory usage without compromising on accuracy.
- PatchCore processes images by dividing them into patches, allowing it to focus on local image features rather than the entire image at once. This is particularly useful for anomaly detection because it enables the model to identify small, localized anomalies
- After extracting features from these patches, PatchCore applies **coresets**—a technique for selecting a small subset of representative data points. Coresets help the model retain the essential structure and patterns of the original data while significantly reducing the number of data points it needs to store in memory.
- This subset is chosen such that it covers the diverse patterns in the data while excluding redundant information.
- PatchCore calculates anomaly scores by comparing new images against the features in the coreset. If a new patch is far from the closest patches in the coreset, it's more likely to be classified as anomalous.

2. Efficient AD

- EfficientAD is typically designed for speed and lower computational costs, often optimized for deployment in environments with limited computational resources.
- EfficientAD uses a lightweight convolutional neural network (CNN) architecture to extract features from input images. The architecture is designed to be compact, using fewer layers and parameters compared to heavier models
- EfficientAD focuses on extracting just enough features to identify anomalies without going into high-dimensional representations, which would increase computational load. It selectively learns essential patterns associated with normal images and filters out unnecessary details.
- EfficientAD is optimized for deployment in low-resource environments, making it ideal for real-time applications or on-device anomaly detection where speed and resource efficiency are critical.

AUROC Scores

Category	PatchCore	EfficientAd
tile	0.8585858345031738	0.9635642766952515
leather	1.0	0.879755437374115
grid	0.9766082167625427	0.9807853698730469

Workflow

1. Data Preparation

- **Dataset Selection**: Use only the specified categories (tile, leather, and grid) from the MVTec-AD dataset.
- **Data Loading**: Utilize the anomalib library to load and preprocess the dataset, ensuring all images are resized, normalized, or standardized as required by the models.

2. Model Setup and Training

- **PatchCore Model**:
    1. Initialize the PatchCore model.
    2. Train the model on the data.

- **EfficientAD Model**:
    1. Initialize the PatchCore model.
    2. Train the model on the data.

AUROC Scoring:

- Calculate AUROC scores for each model on each product category (tile, leather, and grid) to assess how well each model differentiates between normal and anomalous samples.
- Compute the average AUROC score across the three categories for a general performance measure.

Similarity Search

1. Select a category and a model.
2. Extract feature for the category using the model.
3. Store the feature in qdrant vector database.
4. Select a image as a query image.
5. Extract its feature and sent it as a query for similarity search.
6. Get the results from the similarity search.

Similarity search should fetch results similar to the query image that we passed, if it's a normal good image, it should get similar good images, or if it is an anomaly image, it should find similar anomaly images.