Anomaly Detection Models: PatchCore and EfficientAD

In Assignment 2, we use two different visual anomaly detection models (or systems if one prefers to take that view given the each utilizes multiple subsystems like pre-trained neural networks). Both models are used in modern industrial applications and provide state-of-the-art performance. PatchCore's approach to anomaly detection leverages a memory bank of patch-level features, and PatchCore is widely used in industries like manufacturing where anomalous data is rare or difficult to obtain (since PatchCore only uses normal, non-anomalous images during training). Meanwhile, EfficientAd focuses on computational efficiency, using a student teach framework to detect anomalies with millisecond-level latency. EfficientAd is ideal for real-time applications where speed and efficiency are critical. Below we dive into the specifics of each anomaly detection model.

PatchCore

PatchCore is an advanced anomaly detection model that employs patch-level representation learning. It divides images into small patches and learns features from these patches to identify deviations from normal patterns. The model leverages a memory bank to store normal patch representations and compares incoming patches to this memory bank to detect anomalies. By focusing on local regions of images, PatchCore achieves high precision in identifying subtle anomalies that might be overlooked by global feature analysis.

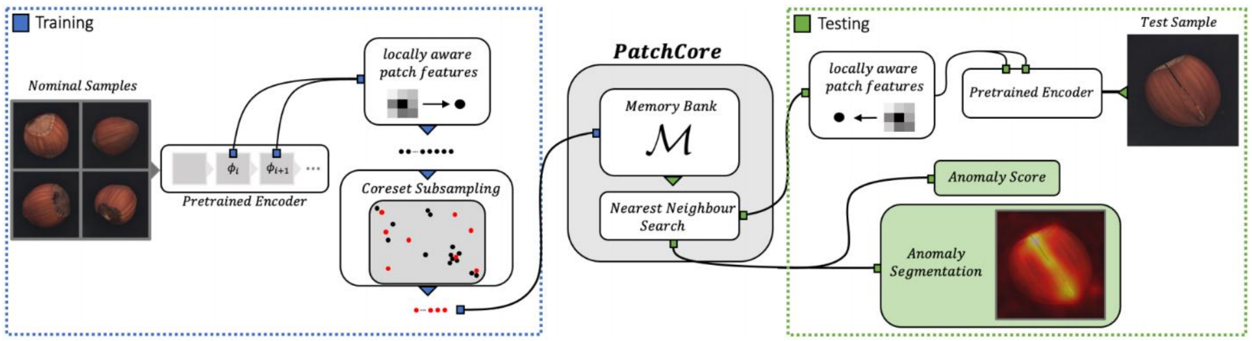

Figure 1. Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., & Gehler, P. (2022). *Towards total recall in industrial anomaly detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 14318-14328).*

During the training phase only normal (nominal) images are fed into PathCore's pipeline. First the training images pass through a pretrained encoder (or the "backbone" model like "wide_resnet50_2" as we specified in *anomalib*) which is typically pre-trained convolutional neural network (CNN) trained on popular image datasets like ImageNet. Instead of relying on global (entire) image classification, the pretrained encoder is said to "extract locally aware patch features" from each input image. What this technically means is that the output feature maps from a chosen hidden layer's output volume (H_{out} * W_{out} * D_{out} where D_{out} is the number of filters in that hidden layer) are sampled and each spatial location (mapped to a receptive field in the original input image) from the output feature map is treated as a patch or representation of a local region in the original image. What is extracted are various *feature vectors* that encompass the entire depth (D_{out}), one for each spatial location in the output feature map. Each of these feature vectors then represent a distinct local patch of the original image.

After feature extraction, the PatchCore pipeline employs *Coreset Subsampling* to choose a subset of the feature vectors in order to reduce computational and storage costs. Coreset Subsampling selects a diverse and representative subset of feature vectors using common methods like k-Center Greedy which ensures that the retrained subset covers the original feature space with minimal redundancy. After Coreset Subsampling, the remaining, most-informative feature vectors are then stored in the memory bank.

After training on the normal input image set (e.g. MVTec AD dataset's normal images like in our assignment), a new test image (which may be anomalous) is processed through the same backbone model and its patch-level feature vectors are extracted in the same way. Each patch-level feature vector from the training image is then compared against the feature vectors in the memory bank using a *Nearest Neighbor Search* in which the distance (Euclidean distance in k-Nearest Neighbors) between each test feature vector ("patch") is calculated and the closest stored feature in the memory bank is identified. This closest distance is said to be the *Anomaly Score* and if the score is too high (i.e. the distance of the test patch is too far from any training image patch/feature vector within the memory bank) then the test image is classified as anomalous; otherwise, the test image is classified as normal.

EfficientAD

EfficientAD is a highly efficient visual anomaly detection model designed for scenarios where computational resources are limited, and real-time performance is critical (e.g. when millisecond processing and detection latency is crucial). The EfficientAD model operates using a student-teacher framework that ensures quick and effective anomaly detection, especially relative to other anomaly detection models/systems.

During the training phase, EfficientAD utilizes a pre-trained teacher network (a CNN such as ‘WideResNet-50’) to extract high-level features from normal images. These features serve as the ground truth annotations for training a smaller, lightweight student network (with few parameters/filters) that mimics the teacher’s feature representation.

The training process involves passing normal images through the teacher network. For each training image, features vectors from multiple hidden layers are extracted and then aggregated together into a *multi-scale feature representations* that capture both low-level information (e.g. edges and object parts) and high-level information (semantic object-specific information). The teacher network's output (multi-scale feature representations) is then used to supervise the training of the student network by serving as the ground truth labels for the student network. During the supervised learning of the child network, the same images used to train the teacher network are passed through the student network, each producing their own output feature maps. A loss function (typically a MSE loss[1]) is computed between student network’s output feature map (from various selected hidden layers corresponding to the selected layers of the teacher) and the teacher’s feature representations for each image separately. Ultimately, the student network is trained to minimizing the loss between the multi-scale feature representations generated by the teacher and the student networks. This goal ensures that the student network can accurately replicate the teacher network's feature extraction process.

Once the student network is trained, it can be deployed for real-time anomaly detection. In the inference phase, test images are passed through the student network to obtain their feature representations. These representations are then compared to the features extracted from normal images during training. Anomalies are detected based on the reconstruction error or the deviation between the test image features and the expected normal features. If the discrepancy exceeds a predefined threshold, the test image is classified as anomalous; otherwise, it is considered normal.

EfficientAD’s primary advantage lies in its computational efficiency. The student network, being significantly smaller and faster than the teacher network, allows for low-latency anomaly detection, making it ideal for applications that require real-time processing. This model is particularly suited for environments where rapid detection of defects is essential, such as automated quality control in manufacturing or real-time monitoring in surveillance systems.

[1] MSE loss for single, selected layer *l*:

$$\mathcal{L}^l = \frac{1}{H^l W^l C^l} \sum_{i=1}^{H^l} \sum_{j=1}^{W^l} \sum_{k=1}^{C^l} (T^l_{i,j,k} - S^l_{i,j,k})^2$$