

# Supplementary:

## Towards Total Recall in Industrial Anomaly Detection

### A. Implementation Details

We implemented our models in Python 3.7 [51] and PyTorch [37]. Experiments are run on Nvidia Tesla V4 GPUs. We used torchvision ImageNet-pretrained models from torchvision and the PyTorch Image Models repository [53]. By default, following [10] and [14], *PatchCore* uses a WideResNet50-backbone [57] for direct comparability. Patch-level features are taken from feature map aggregation of the final outputs in blocks 2 and 3. For all nearest neighbour retrieval and distance computations, we use `faiss` [27].

### B. Full MVTec AD comparison

This section contains a more detailed comparison on MVTec AD. We include more models and a more finegrained performance comparison on all MVTec AD sub-datasets where available. In the main part of the paper this has been referenced in §4.2. The corresponding result tables are S1, S2 and S3. We observe that *PatchCore*–25% solves six of the 15 MVTec datasets and achieves highest AUROC performance on most datasets and in average.

Figure S3 show Precision-Recall and ROC curves for *PatchCore* variants as well as reimplemented, comparable methods SPADE [10] and PaDiM [14] using a WideResNet50 backbone. We also plot classification error both at 100% recall and under a F1-optimal threshold to give a comparable working point. As can be seen, *PatchCore* achieves consistently low classification errors with defined working points as well, with near-optimal Precision-Recall and ROC curves across datasets, in contrast to SPADE and PaDiM.

Finally, Table S4 showcases the detailed performance on all MVTec AD subdatasets for larger imagesizes ( $280 \times 280$ ) and a WideResNet-101 backbone for further performance boosts using *PatchCore*–1%, which allows for efficient anomaly detection at inference time even with larger images.

### C. Additional Ablations & Details

#### C.1. Detailed Low-Shot experiments

This section offers detailed numerical values to the low-shot method study provided in the main part of this work (§4.5). The results are included in Table S5 and we find consistently higher numbers for detection and anomaly localization metrics.

#### C.2. Dependency on pretrained networks

We tested *PatchCore* with different backbones, the results are shown in S6. We find that results are mostly stable over the choice of different backbones. The choice of WideResNet50 was made to be comparable with SPADE and PaDiM.

#### C.3. Influence of image resolution

Next we study the influence of image size on performance. In the main paper we have used  $224 \times 224$  to be comparable with prior work. In Figure S4 we vary the image size from  $288 \times 288$ ,  $360 \times 360$  to  $448 \times 448$  and the neighborhood sizes (P) within 3, 5, 7, and 9. We observe slightly increased detection performance and the performance saturates for *PatchCore*. For anomaly segmentation we observe a consistent increase, so if good localization is of importance, this is an ingredient to validate over.

#### C.4. Remaining Misclassifications

The high image-level anomaly detection performance allows us to look into all remaining misclassifications in detail. We compute the working point (threshold above which scores are considered anomalous) using the F1-optimal point. With this threshold a total of 19 false-positive and 23 false-negative errors remain, all of which are visualized in Figures S1 and S2. Each segmentation map was normalized to the threshold value, so in some cases background scores are pronounced disproportionally.

Looking at Figure S1, we find that the majority of false-positive errors either stem from a) (in blue) ambiguity in labelling, i.e., image changes that could also be potentially labelled as anomalous, and b) (in orange) very high nominal variance,