Supplementary:
ReConPatch : Contrastive Patch Representation Learning for Industrial Anomaly Detection

A. Implementation details

ReConPatch was implemented with Python 3.7 [37] and PyTorch 1.9 [27]. Experiments are run on Nvidia GeForce GTX 3090 GPU. We used ImageNet-pretrained models from `torchvision` [23] and the PyTorch Image Models repository [38]. By default, following [7] and [9], ReConPatch uses WideResNet50-backbone and WideResNet101-backbone [42] for direct comparability. Patch-level features are taken from feature map aggregation of the final outputs in f layer of ReConPatch. We use `faiss` [16] to compute all nearest neighbor retrieval and distance computations same as PatchCore [29].

B. Additional experiments on MVTec AD

This section contains the details of setting up hyper-parameters and experiments with the projection layer of ReConPatch on MVTec AD [4] dataset. We experimented to find the optimal hyper-parameters with the following values: k value of k-nearest neighborhood for calculating contextual similarity, the repelling margin m in RC loss, and applying ratio α between pairwise and contextual similarity. Additionally, to verify the effectiveness of projection layer g, we tested our proposed method without projection layer g.

All experiments have the same conditions. Each images are resized to 256×256 and center-cropped to 224×224 and ImageNet-pretrained WideResNet50-backbone [42] from `torchvision` [23] was employed as the feature extractor. The f layer output size was set to 512, and the coreset subsampling percentage was set to 1%. The ReConPatch was trained for 120 epochs per each class on MVTec AD [4] with AdamP [14] optimizer with cosine annealing [22] scheduler.

B.1. The k value of k-nearest neighbor for contextual similarity

We tested several k - nearest neighborhood value k for finding the best performance of anomaly detection on MVTec AD [4] dataset. In Eq. 2, smaller k values will use features that are closer together to determine contextual similarity, while larger k values will use features that are further apart in the embedding space to determine contextual similarity. To find an optimal k value, we fixed other hyper-parameters $m = 1$, $\alpha = 0.5$. Table S1 shows the results of experiments.

ReConPatch (WRS-50, 224×224, $\alpha = 0.5$, $m = 1.0$)						
k value	3	5	7	10	15	20
Detection	99.5	**99.56**	99.55	99.54	99.5	99.49
Segmentation	**98.09**	98.07	98.07	98.07	98.04	98.07

Table S1. Results of anomaly detection(image-level AUROC) and segmentation(pixel-level AUROC) with various k-nearest neighborhood on MVTec AD dataset. ($\alpha = 0.5$, $m = 1.0$).

B.2. The repelling margin m of relaxed contrastive loss

We tested margin value m in Eq. 7 repelling term of relaxed contrastive loss with various m values to obtain the performance of anomaly detection on MVTec AD [4]. All other hyper-parameters are fixed($k = 5$, $\alpha = 0.5$). Table S2 is the results of experiments. Changing the margin m has no effect on performance of anomaly detection(image-level AUROC) but it does slightly affect on segmentation(pixel-level AUROC). So we use the repelling margin m to 1.0.

ReConPatch (WRS-50, 224×224, $k = 5$, $\alpha = 0.5$)				
m value	0.5	1	1.5	2
Detection	99.55	**99.56**	99.55	99.5
Segmentation	98.04	**98.07**	98.05	98.04

Table S2. Results of anomaly detection(image-level AUROC) and segmentation(pixel-level AUROC) with various repelling margin m values of RC loss. ($k = 5$, $\alpha = 0.5$).

B.3. The applying ratio α between pairwise and contextual similarity

We tested α in Eq. 6 which the ratio of linear combination between pairwise and contextual similarity within range $[0, 1]$. If α is 0 then only use contextual similarity, and if α set to 1.0 then use pairwise similarity only. Table S3 shows the best detection(image-level AUROC) performance when using pairwise and contextual similarity with same ratio.

ReConPatch (WRS-50, 224×224, $k = 5$, $m = 1.0$)					
α	0.0	0.25	0.5	0.75	1.0
Detection	99.51	99.51	**99.56**	99.52	99.5
Segmentation	98.02	98.07	98.07	98.1	**98.12**

Table S3. Results of anomaly detection(image-level AUROC) and segmentation(pixel-level AUROC) with various α values within range $[0, 1]$ of ReConPatch. ($k = 5$, $m = 1.0$).

B.4. The projection layer g

We tested with linear projection layer g, without projection layer g. Fig. S1 describes the visualization of patch features obtained by ReConPatch which trained with projection layer g and trained without projection layer. For effective visualization of high dimensional feature vectors, we map the patch features into 2-dimensional space using UMAP [24]. With visualization result, we verify that the projection layer g of ReConPatch encourages the features with similar position together. We can also verify that using the projection layer has better anomaly detection performance than not using the projection layer by the results in Table S4.

ReConPatch (WRS-50, 224×224, $k = 5$, $m = 1.0$, $\alpha = 1.0$)		
Projection layer	w/o Proj. Layer	w/ Proj. Layer
Detection	99.42	99.56

Table S4. Results of Detection AUROC with and without projection layer g on the MVTec AD dataset [4] using our proposed ReConPatch model with a WideResNet-50 backbone [42], 224×224 input size, 512 dimensional f layer, $k = 5$, $m = 1.0$, $\alpha = 1.0$ and 1% coreset sampling.

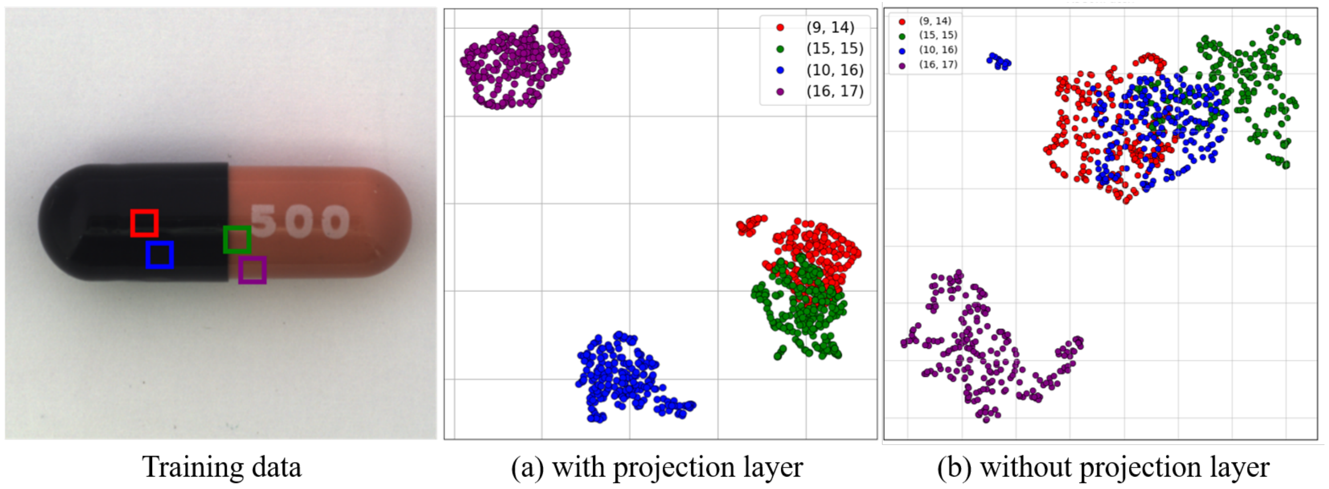

Figure S1. An illustrative comparison of features mapped by ReConPatch with projection layer (a) and without projection layer (b) using the MVTec AD dataset [4]. The scatter plot describes the feature space of each method, colored according to the pixel position.

Method	PatchCore			ReConPatch		
$\downarrow$ Class\Aug. Method $\rightarrow$	w/o Aug.	w/ Aug.	Diff	w/o Aug.	w/ Aug.	Diff
Bottle	100	99.68	0.32	100	100	0
Cable	99.5	96.59	2.91	99.83	99.01	0.82
Capsule	98.1	82.73	15.37	98.8	95.29	3.51
Hazelnut	100	100	0	100	100	0
Metal nut	100	98.92	1.08	100	99.9	0.1
Pill	96.6	92.09	4.51	97.49	94.73	2.76
Screw	98.1	89.75	8.35	98.52	89.96	8.56
Toothbrush	100	95	5	100	99.17	0.83
Transistor	100	98.37	1.63	100	99.33	0.67
Zipper	99.4	95.51	3.89	99.76	99.13	0.63
Object classes	99.17	94.86	4.31	99.44	97.65	1.79
Carpet	98.7	96.47	2.23	99.6	99.04	0.56
Grid	98.2	87.89	10.31	100	98.5	1.5
Leather	100	100	0	100	100	0
Tile	98.7	96.64	2.06	99.78	100	-0.22
Wood	99.2	99.47	-0.27	99.65	99.82	-0.17
Texture classes	98.96	96.09	2.87	99.81	99.47	0.34
Average	99.1	95.48	3.62	99.56	98.56	1.0

Table S5. Anomaly detection performance with data augmentation(random rotate, translate, brightness, contrast and gaussian blur) on the MVTec AD dataset [4]

C. Performance of ReConPatch

C.1. Applying data augmenatation

In the industrial domain, product images are collected in a well-controlled environment, but uncontrollable change in the environmental conditions at the time of acquisition can cause variation in the image data. This can lead to variation between normal image data, which can adversely affect the performance of industrial anomaly detection. To verify that ReConPatch is adaptable to environmental changes, we applied various data augmentation method to evaluate its anomaly detection performance. We randomly applied rotation($\pm 5 degree$), translation($\pm 1\%$), color jitter(brightness and contrast $\pm 30\%$), and gaussian blur($\sigma = [0.1, 1.0]$) which could occur in the real world. Table S5 shows the detail performance comparison of PatchCore [29] and ReConPatch with data augmentation on MVTec AD dataset [4].

C.2. Inference time

We measured inference time and memory usage of ReConPatch and PatchCore [29] with Intel Xeon Gold 6240 CPU and Nvidia GeForce GTX 3090 GPU. Test images are resized to 256×256 and center cropped to 224×224 on MVTec AD dataset [4]. The target embed dimension was set to 512 in both PatchCore [29] and ReConPatch. We experimented for 30 iterations with 10 warm-up times to measure the inference time of PatchCore [29] and our proposed ReConPatch. In Table S6, the average inference time of PatchCore [29] and ReConPatch are not significantly different.

Method	Avg. inference time(msec)
PatchCore	37.89
ReConPatch	38.09

Table S6. Inference time comparison between PatchCore [29] and ReConPatch on MVTec AD dataset.

C.3. Memory efficiency

Our proposed ReConPatch algorithm uses a single linear layer which trained with local patch features of normal samples. In Eq. S1, memory usage of memory bank $M_{coreset}$ is affected by dimension of f layer f_{dim} and percentage of coreset subsampling $c_{percentage}$. $H_{feature}$ and $W_{features}$ are size of outputs in blocks 2 and 3. The results of anomaly detection and segmentation with various f layer dimension shows in Table 2 and Fig. S2. In Fig. S2, the performance of ReConPatch for anomaly detection (image-level AUROC) and segmentation (pixel-level AUROC) is better than PatchCore [29] with same memory bank size.

$$M_{coreset} = f_{dim} * H_{features} * W_{features} * N_{train} * c_{percentage} \tag{S1}$$

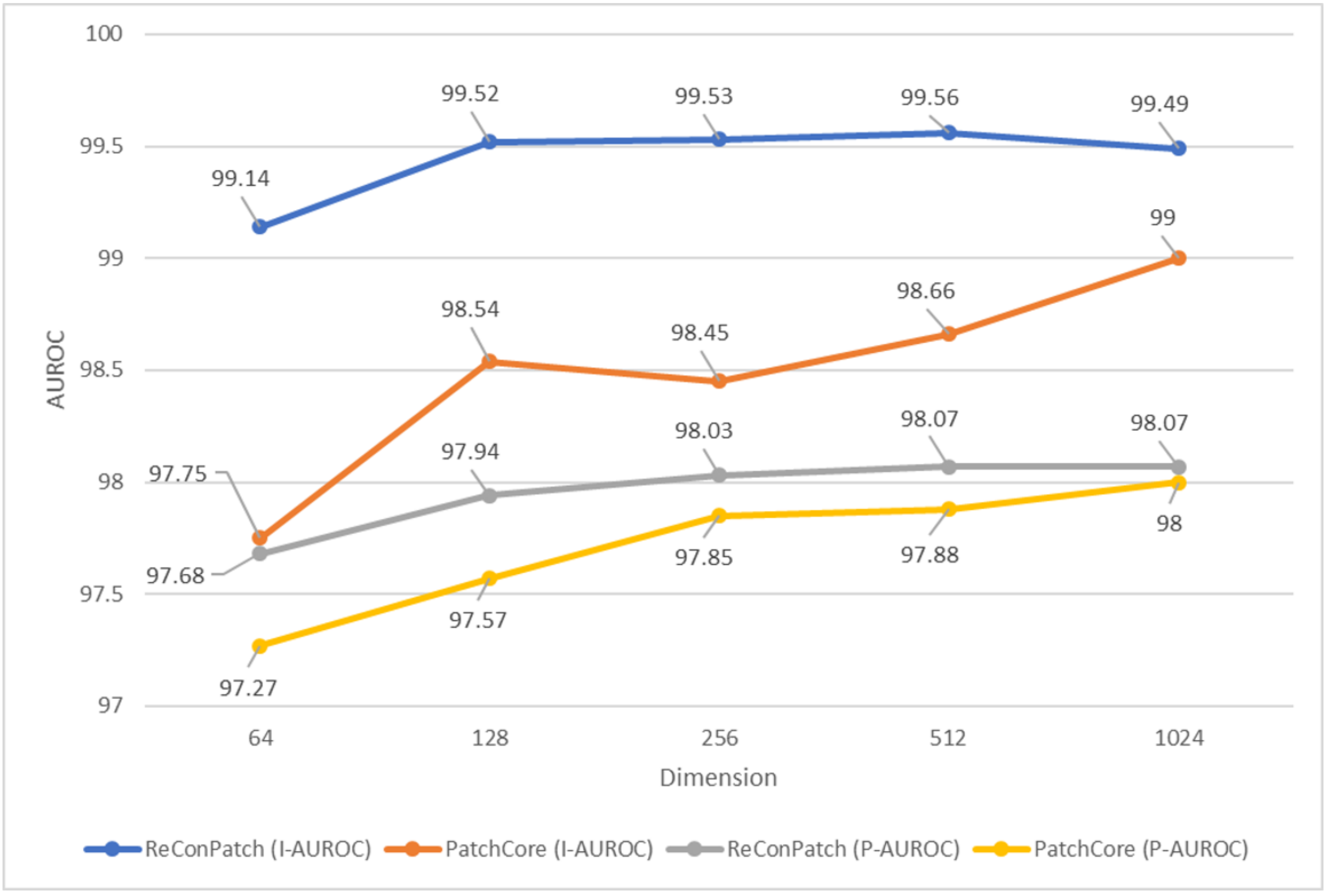

Figure S2. The results of Detection AUROC and Segmentation AUROC for the f layer dimension on the MVTec AD dataset [4]. Image size is 224×224, $k = 5$, $m = 1.0$, $\alpha = 1.0$ and coreset subsampling rate is 1%

C.4. Anomaly score maps

We provide the examples of the anomaly score map of MVTec AD dataset [4] and BTAD [25] dataset along with ground truth (orange line) overlaid input images in Fig.S3, S4. The anomaly map indicates the regions of the input image where ReConPatch has detected anomalies, with higher values indicating a higher likelihood of an anomaly being present. The green line indicates the threshold which is optimized by the F1 scores of anomaly segmentation, and the orange line indicates the ground truth of the anomalies.

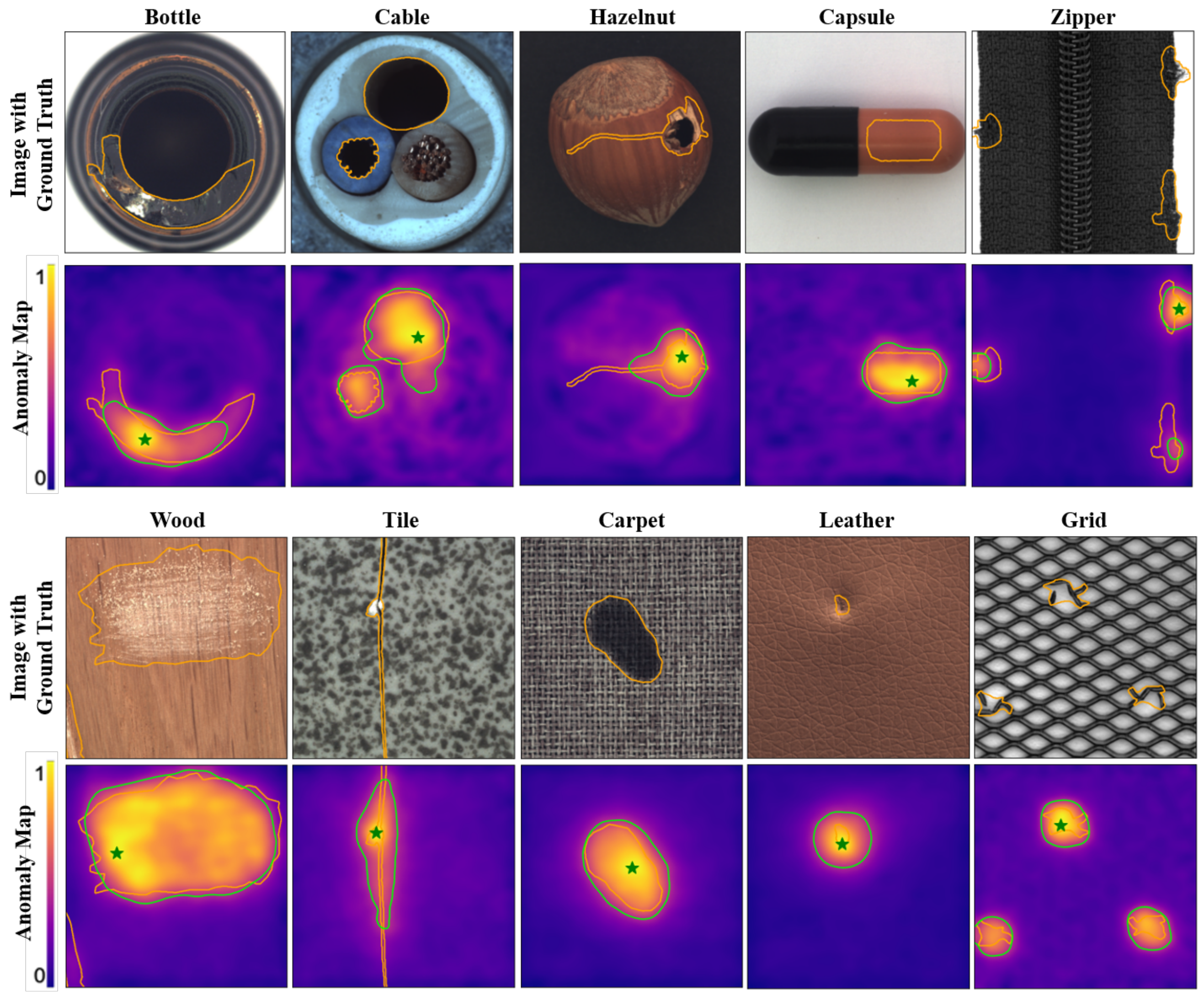

Figure S3. Examples of images with anomalies (top) and measured anomaly score maps (bottom) on MVTec AD dataset [4]. The orange line depicts the ground truth of the anomalies and the green line depicts thresholds optimizing F1 scores of anomaly segmentation. The green star indicates the maximal location of the anomaly score in the heatmap.

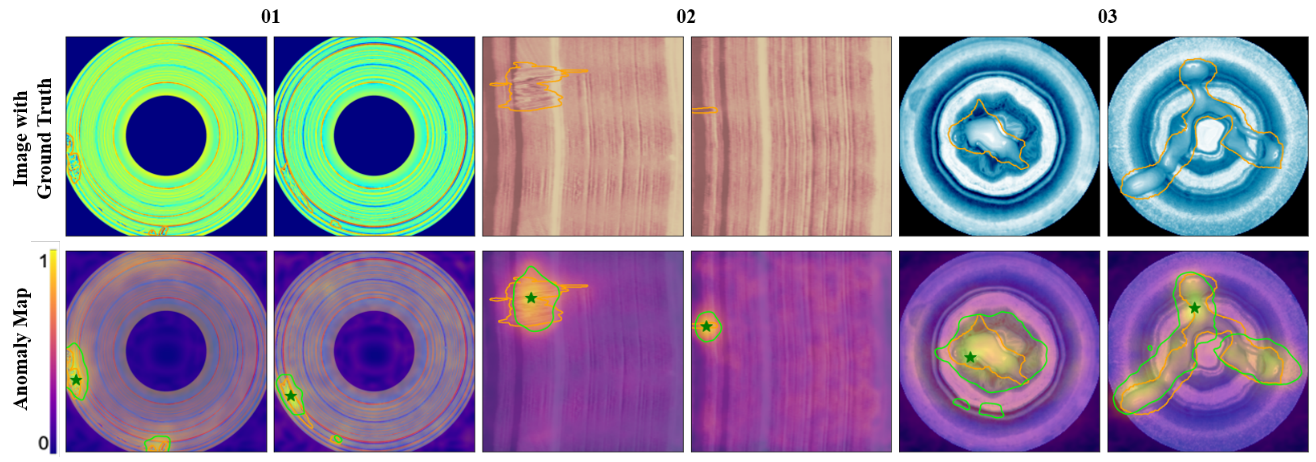

Figure S4. Examples of images with anomalies (top) and measured anomaly score maps (bottom) on BTAD dataset [25]. The orange line depicts the ground truth of the anomalies and the green line depicts thresholds optimizing F1 scores of anomaly segmentation. The green star indicates the maximal location of the anomaly score in the heatmap.