Improvement on Image Anomaly Detection Focusing on Background Anomaly

Mina Son
20208077
Department of Chem.

Daewon Kim
20180061
School of Computing

Jaeyoung Kang
20190011
Department of M. E.

Changmin Lee
20190519
School of Computing

Abstract

Anomaly detection is one of the classical problems, identifying unusual pattern in a given data. It is especially important in industry field, which is cost sensitive. Since anomalous case is too rare and diverse, unsupervised learning is a typical way to overcome this problem. MVTec AD dataset is a anomaly detection benchmark focusing on industrial inspection. All training images are defect-free and should not contain any anomaly while the test images have both good and anomalous samples. However, many images contain background anomaly. The background anomaly is random and easy to find, which has high possibility to drop an anomaly detection performance. There are many state-of-the-art approaches of the MVTec dataset, but they do not address background defects. Therefore, this paper focuses on how the background anomaly affects performance. The *PatchCore*(Roth et al., 2022) is the model on the top of an image classification, which is simple and intuitive. Based on the PatchCore model, our image preprocessing achieves AUROC score of 99.07% image classification on the MVTec AD capsule category. Moreover, we find how a background anomaly is related to the anomaly detection performance.

1 Introduction

The topic of anomaly detection refers to identification of unusual patterns in a given image. Examples include recognizing defects on products on an industrial scale as well as recognizing lesions for patients in a medical image. Doing this manually would be too costly, and there is a chance that human error can miss anomalies based on subjective judgements. Thus, computer vision in the field of artificial intelligence can help overcome these issues.

For most cases, the nominal situation is the absolute majority. So an unsupervised learning technique would be proper for handling these type of issues. Previous works focusing on this problem can be categorized into two methods. One is the use of normalizing flow, and the other is use of nearest neighbor methods. Details of the previous works are introduced in section **2**.
In this paper, we have utilized the nearest neighbor method based on PatchCore(Roth et al., 2022) and devise a scoring system for the implementation. The model is then evaluated with image-level and pixel-wise area area under the receiver operating characteristics(AUROC) curve. Furthermore, image preprocessing focusing on the background noise removal have been tested and the results are discussed.

2 Related Works

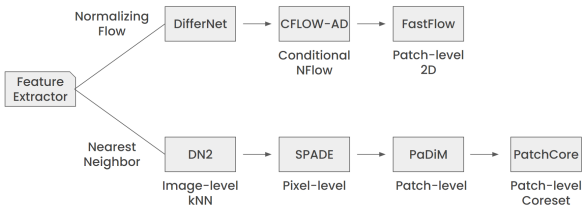


Figure 1: History of Models for Anomaly Detection

In this section, a brief summary of related works is given focusing on two main procedures after feature extraction: 1) Use of normalizing flow, and 2) use of the nearest neighbor method.

2.1 Works Based on Normalizing Flow

For the modeling of high-dimensional data, the method of *normalizing flows*(NFLOWs) (Dinh et al., 2016) to efficiently reduce and sample the data have been introduced. It is a real-valued non-volume preserving method which has the capability of efficiently and accurately reconstructing the input images.
DifferNet(Rudolph et al., 2021) has adapted the method of NFLOW for image-level anomaly detection. A feature extractor was used do reduce

the data dimension to make the images suitable for the use of NFLOW. With this, likelihoods are generated and a scoring function is devised to indicate defects. This method was applicable for small training sets.

CFLOW-AD(Gudovskiy et al., 2022) has expanded the DifferNet model to pixelwise anomaly detection and demonstrates that the use of an conditional NFLOW(CFLOW) framework is suitable for this task, producing similar results with less computational cost.

In the FastFlow(Yu et al., 2021) model, a 2D NFLOW with fully convolutional networks and a two-dimensional loss function is suggested. Attempts were made to scale down the model size by alternatively stacking large and small convolutional kernels. This can be used as a probability distribution estimator. The tractable distribution obtained in the training phase can then be used for calculating the likelihood for anomalies in inference phase.

2.2 Works Based on Nearest Neighbor

The SPADE(Cohen and Hoshen, 2020) model used k-Nearest neighbor(kNN) to enable anomaly segmentation based on alignment between an anomalous image and other nominal images. Using a pre-trained feature extractor, a multi-resolution feature pyramid is used. This uses a separate approach for image- and pixel-level anomaly detection.

Patch Distribution Modeling(PaDiM)(Defard et al., 2020) uses the concept of a patch, which is described by a multivariate Gaussian distribution. PaDiM then utilizes the correlation between different sematic levels of the pretrained feature extractor. The Mahalanobis distance measures are used to give an anomaly score for a given patch data.

The PatchCore(Roth et al., 2022) utilizes patch-scale coreset. The local patch features are aggregated into a memory bank. The memory bank is then subsampled, forming a coreset. Finally, using the features inside the coreset and a scoring function, an anomaly is detected and a pixelwise segmentation is made possible.

Our work focuses on the nearest neighbor method, and specifically, the PatchCore architecture. As of June 15th, 2022, the date this report is being written, PatchCore ranks 1st in anomaly detection and 3rd place in pixelwise anomaly segmentation. Also, the model is divided into clear modules, and is much affordable in the given computing resources compared to deep learning models. Also, the structure of NN is more interpretable as the model uses a well-defined scoring function, leaving more room for research.

3 Experiments

3.1 Dataset

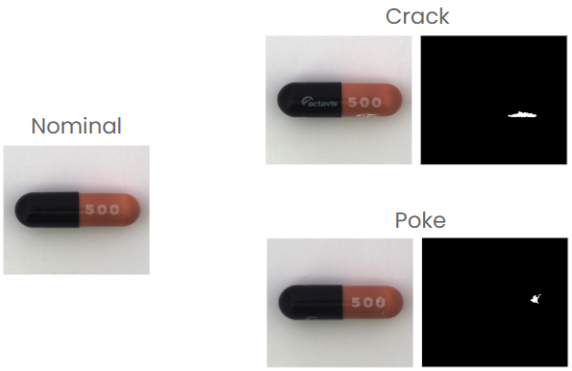


Figure 2: Examples of nominal and anominal capsule images in the MVTec-AD dataset

For the anomaly detection, the MVTec AD dataset (Bergmann et al., 2019, 2021), specifically, the capsule images, are used with no additional training images. The training dataset is composed of 219 1000x1000 RGB images of nominal capsules. The test dataset is composed of 23 good images and 132 anomalous images. The anomalities include cracks, faulty prints, pokes, scratches, and squeezed pills. The ground truth images are also provided in black and white. Some examples are given in Figure 2.

3.2 Baseline Model

For the baseline, the PatchCore architecture is used. A replication version of the code based on PyTorch is adapted.[1] Coreset ratio of 1% is used. All other options are kept same as in the original paper. Each experiment was carried out 7 times, and the average value is reported. This is to even out the deviation caused by the coreset subsampling.

3.3 Evaluation Metrics

Our model will be evaluated with the AUROC metrics. AUROC is an area under the receiver operating characteristics(ROC) curve. The ROC curve

[1] https://github.com/hcw-00/PatchCore_anomaly_detection
[2] https://commons.wikimedia.org/wiki/File:Roc_curve.svg

Capsule AUROC	Image detection	Pixel segmentation
Paper	0.980	0.988
Replication code	0.976	0.989

Table 1: Performance of the Patchcore replication code.

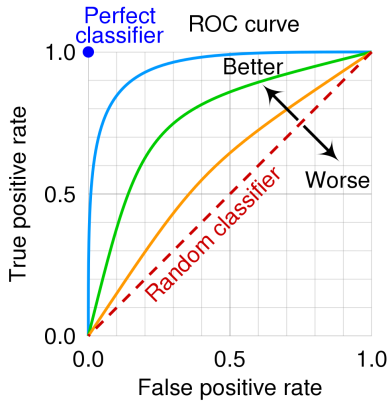


Figure 3: Area under the ROC Curve[2]

is a graphical plot to judge a binary classifier in varying the discrimination threshold. The x axis is the false positive rate, while the y axis shows the true positive rate. The curve moves to the upper-left corner as a classifier gets closer to being perfect. We will evaluate our model with both image classification and pixel-wise segmentation AUROC.

4 Score modification

4.1 Concept

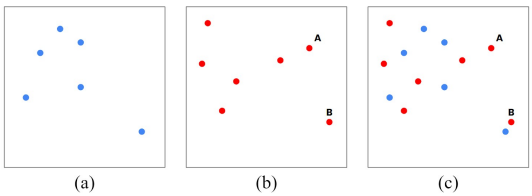


Figure 4: Example feature vectors of the patches in (a) the memory bank (blue dots), (b) the test image (red dots), and (c) both sets.

Our first approach for this problem was to edit the scoring function for the anomaly detection. In Figure 4, the feature vectors of the patches in the memory bank (blue dots), and the test image (red dots) are represented as 2D data. As mentioned earlier, the memory bank is a representative subset of the data of the training dataset consisting of only nominal data. Although B has a closer point in the memory bank than A, that point seems to be already anomalous.

$$m^{\text{test},*}, m^* = \underset{m^{\text{test}} \in \mathscr{P}(x^{\text{test}})}{\operatorname{argmax}} \underset{m \in \mathscr{M}}{\operatorname{argmin}} \left\| m^{\text{test}} - m \right\|_2 \tag{1}$$

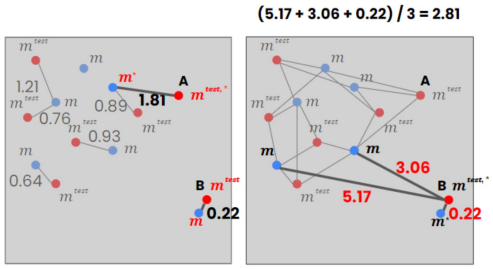
$$s^* = \left\| m^{\text{test},*} - m^* \right\|_2 \tag{2}$$

However, Patchcore finds the closest data in the memory bank for each patch in the test image. And that distance becomes the anomaly score of each patch. You can see the exact formula in the Equation 1 and 2. So PatchCore predicts that A is the most anomalous and B is the most nominal. It doesn't care about an already rare nominal occurrence of the data in the memory bank. We will call these rare nominals 'pre-anomaly' from now on. To overcome this issue, we have devised a new scoring function with kNN approach.

$$m^{\text{test},*}, m^* = \underset{m^{\text{test}} \in \mathscr{P}(x^{\text{test}})}{\operatorname{argmax}} \underset{m \in \mathscr{M}}{\operatorname{argmin}} \left(\frac{1}{b} \sum_{m \in \mathscr{N}_b(m)} \left\| m^{\text{test}} - m \right\|_2 \right) \tag{3}$$

$$s^* = \frac{1}{b} \sum_{m \in \mathscr{N}_b(m^*)} \left\| m^{\text{test},*} - m^* \right\|_2 \tag{4}$$

Actually, PatchCore uses a scaling on the score to increase the anomaly score, if m* is a "pre-anomaly". However, the scaled score is used only for image-level AUROC, so it is not considered when we get a pixel-wise AUROC score. So we suggest the new scoring method which measures the score with kNN algorithm in equation 3 and equation 4. Like PatchCore, new scoring method finds the closest data in the memory bank for each patch in the test image. But, it also finds the distances to several neighbors of the data in the memory bank and calculates the average of those distances. As a result, contrary to PatchCore, our scoring concludes that B is the most anomalous data. This is because the closest nominal data was considered as a "pre-anomaly". So our new scoring method seems to produce better results when "pre-anomaly" exists in the training dataset.



(a) (b)

Figure 5: Anomaly score of patches in (a) the original score and (b) the new score.

4.2 Experiments

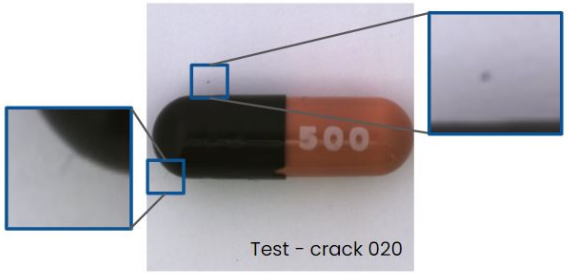


Figure 6: Background noise, the typical example of pre-anomaly.

We implemented new scoring function on PatchCore replication code. Then we checked the result whether the pre-anomaly was detected exactly. A typical example of pre-anomaly would be background noise as shown in Figure 6. Since our baseline model does not detect "pre-anomaly" well, we experimented with the hypothesis that the new scoring method would handle this problem better.

4.3 Result

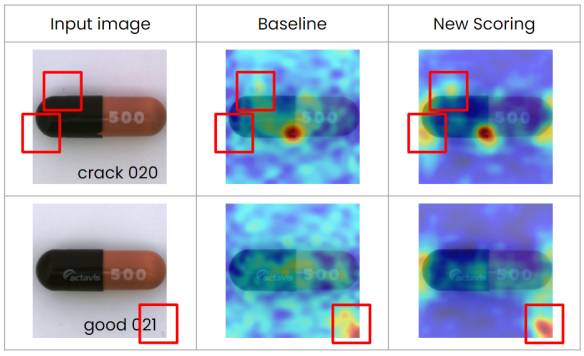


Figure 7: Examples of anomaly segmentation with the new scoring method

Figure 7 shows that our scoring function worked as intended. The crack 020 file had two notable background noises. and the good 21 file also has a noticeable background noise in the lower right corner. The baseline model did not detect this properly, but the new scoring methods are making more clear segmentation. And, even in the case of the good 21 file, the background anomaly appears much more clearly in the new scoring than in the baseline model, without confusion.

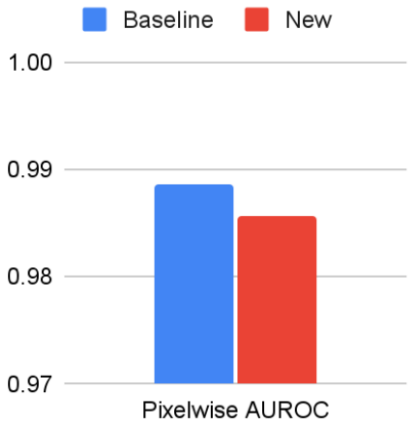


Figure 8: Results of the baseline moel and the new scoring

Figure 8 shows the final result of our new scoring. We didn't adjust the Image-Level score, so we only compared the pixelwise AUROC. We could conclude that new scoring detected pre-anomaly well as we intended, but since the ground truth of our test data did not include background noise as anomaly, so the AUROC of new scoring decreased.

5 Background Anomaly Removal

5.1 Concept

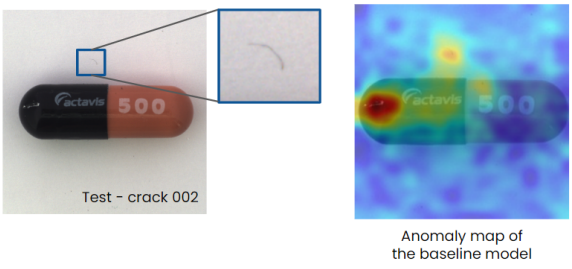


Figure 9: Example of background anomaly

Our second approach is to use preprocessing to remove the background anomaly. As can be

seen in Figure 9, many images contain dust or spots in the background section. The baseline model assigns high anomaly score to those dust. While this is in fact an anomaly in the data, this is not the anomaly that is of our interest. Thus, detection of these fake anomaly drops the performance both in image classification and pixel segmentation. To prevent background anomaly detection, we propose a solution of removing background anomaly.

5.2 Experiment

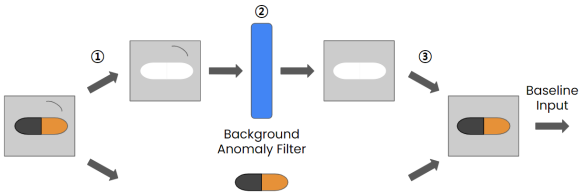


Figure 10: Image preprocessing for background removal

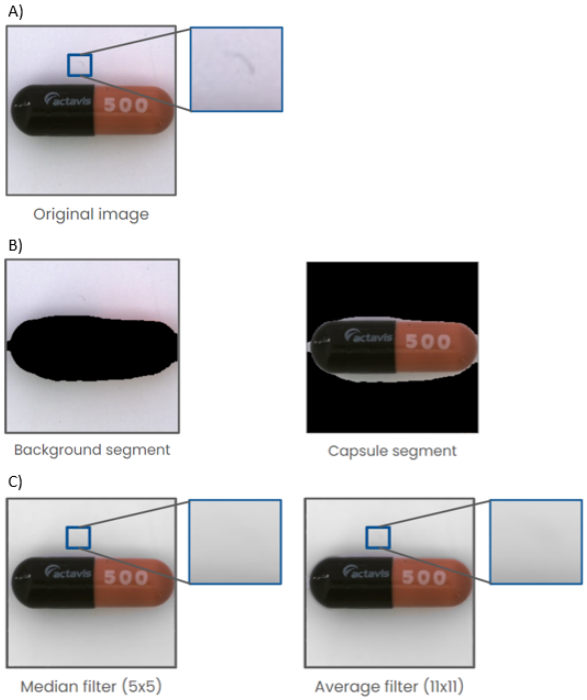


Figure 11: Background recognition and application of filters

As demonstrated in Figure 10, background anomaly removal can be achieved in 4 steps. First, the white background part is removed from the input image. Second, we apply filter on the background to remove anomaly. Third, replace filtered background instead of the original one. Finally, the anomaly-free image is fed into the model.

The actual implementation is described in Figure 11. The input image is divided into two segments; a background segment and a capsule segment. It is based on the assumption that background is light gray color. A pixel is considered as background if the RGB value is in certain range. In this experiment (150, 150, 150) and (255, 255, 255) are selected as the boundary values. Since applying only simple thresholding generates noisy and rough background, a morphological opening followed by a morphological closing is applied. From the original image (Figure 11A), the result of this procedure is shown in Figure 11B. The background segment is converted into grayscale, and a filter is applied to the background.
Two blurring filters provided by the OpenCV package was examined to remove background anomaly. First is an average filter. It reduces noise by taking the average of the values in a given kernel size. However, it cannot remove the noise perfectly. The second is a median filter. It can efficiently remove salt-and-pepper noise. The drawback of this method that the filter is sensitive to both kernel size and noise size. Since the performance highly depends on the filter size, we conducted the test with different sizes of 5, 7, 9, 11, 14 and 15. As can be seen in Figure 11C, the anomaly part in the original image is effectively removed with both filters.

5.3 Results

Figure 12 shows the AUROC scores for different filter types and sizes. Figure 12A) is a plot of image classification AUROC score. A median filter works better with small filter size, while an average filter performs better with large size. Median filter works best when the filter size is 5. The average filter shows the highest score, 0.991, when the size is 11x11. Background anomaly filtering improved the image classification capability. Figure 12B) shows the pixel-wise segmentation AUROC score. Unfortunately, this attempt could not improve the pixel segmentation.

Figure 13 shows the anomaly map examples. The upper example, crack 2, has a background anomaly above the capsule. The baseline model assigns a high anomaly score to it while our two solutions effectively reduce faulty anomaly detec-

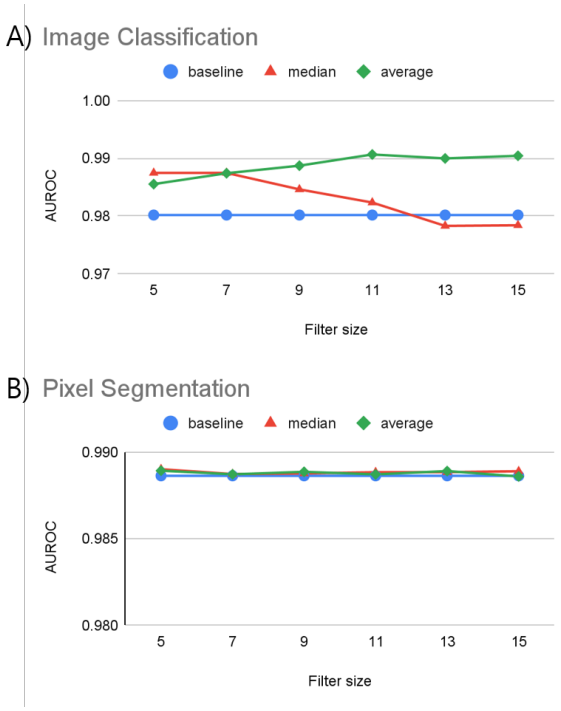


Figure 12: Results of the Background Preprocessing

tion. Below example is a case of a good capsule, which also has a large background anomaly in the right below corner. An anomaly map is gained by mapping the pixel anomaly scores into the certain range of color. It means that the color shows only relative pixel anomaly scores within a single image. Applying our filter to the good input reduces the range of the pixel anomaly score, which is getting closer to the anomaly map of anomaly-free input.

6 Result

The overall result of our experiment is displayed in Figure 14.

The graph on the left shows the image classification AUROC, and graph at the right shows the pixel-wise segmentation AUROC.

For the image classification, the average blur preprocessing showed the best performance. For pixel-wise segmentation, median blur showed the best performance, but the difference between baseline method and preprocessing methods were little.

7 Conclusion

7.1 Anomaly Score Function

The first solution we proposed, the new scoring function showed low performance in the pixel-

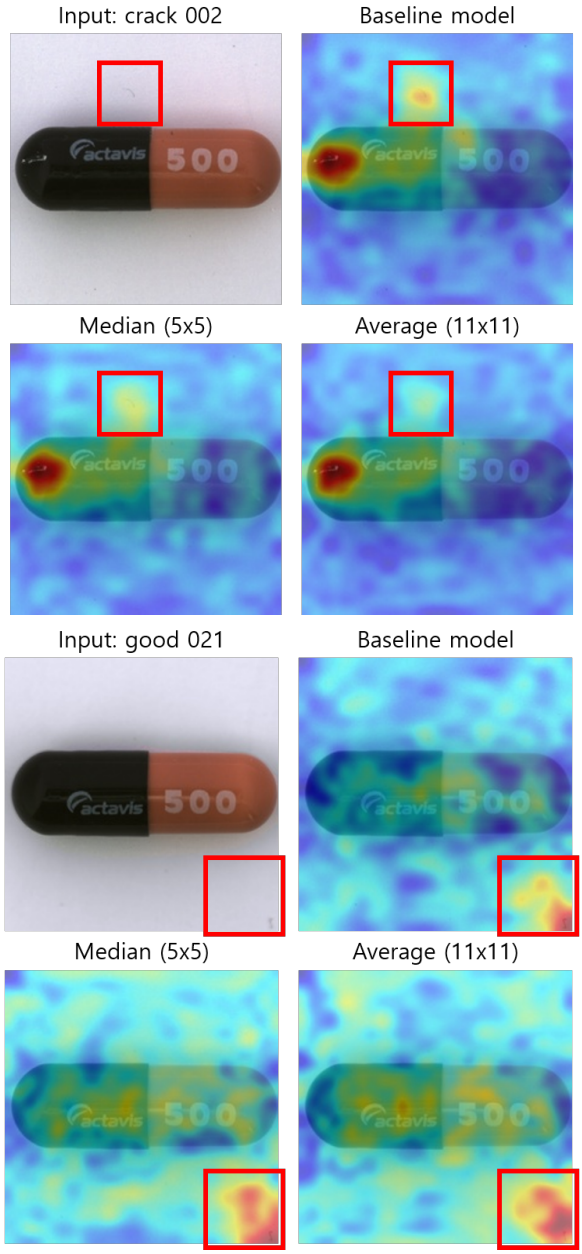


Figure 13: Examples of anomaly segmentation with filters

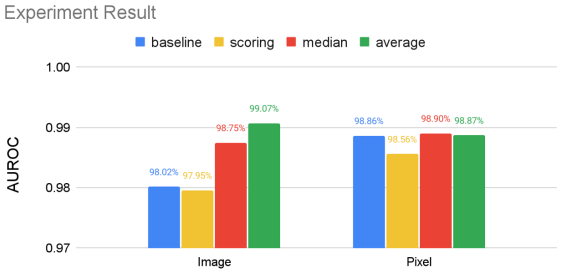


Figure 14: Image classification and pixel segmentation AUROC result

wise segmentation AUROC.

We found out that the new scoring function detected background anomaly accurately. The problem was that the ground truth of test data doesn't include the background anomaly, which led to the decrease of the performance of new scoring function.

7.2 Background Anomaly Removal in General

Inspired from the results of the first solution, we devised a second solution, a preprocessing to remove background anomalies.

This solution showed high performance in both Image-Classification and pixel-wise segmentation AUROC. We could conclude that the preprocessing effectively ruled out most of the background anomalies.

Specifically, in image classification, the best performance could be seen from the average blur with the largest kernel size. We assume that this is because the average blur with larger kernel makes the intensity of background anomaly to decrease in a large amount.

7.3 Weakness of Background Anomaly Removal

While the overall performance of background anomaly removal was satisfying, there were not much improvement on pixel-wise segmentation AUROC. The results between baseline and our solution differed less than 0.04 percent. We assume that this is because background preprocessing had small influence on actual anomaly site.

Median blur showed less performance of 0.32 percent at image classification. Our hypothesis is that this is because the median blur makes the border of blurred anomalies in rough manner, which could have made the hitmap more discretized.

Lastly, median blur showed lower performance in image classification with larger kernel. The median blur with large kernel can cause the wavy distortion at the blur sites. We assume that the low performance is caused by the wavy distortion of the capsule shadow.

References

Paul Bergmann, Kilian Natzner, Michael Fauser, David Sattlegger, and Carsten Steger. 2019. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9584–9592.

Paul Bergmann, Kilian Natzner, Michael Fauser, David Sattlegger, and Carsten Steger. 2021. The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. In *International Journal of Computer Vision*, pages 1038–1059.

Niv Cohen and Yedid Hoshen. 2020. Sub-image anomaly detection with deep pyramid correspondences. *CoRR*, abs/2005.02357.

Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. 2020. Padim: a patch distribution modeling framework for anomaly detection and localization. *CoRR*, abs/2011.08785.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real NVP. *CoRR*, abs/1605.08803.

Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. 2022. Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 98–107.

Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. 2022. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14318–14328.

Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. 2021. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1907–1916.

Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. 2021. Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows. *CoRR*, abs/2111.07677.