Robustness of Surface Anomaly Detection Methods to Domain Shift

Matej Dobrevski, Jakob Božič, Danijel Skočaj

Faculty of Computer Science, University of Ljubljana, Slovenia.
e-mail: matej.dobrevski@fri.uni-lj.si

Abstract

In many realistic visual inspection scenarios it is to be expected that the distribution of images may change through time, due to, e.g., gradual changes in illumination, background, or acquisition settings. In this work we investigate how such a domain shift influences the performance of visual surface anomaly detection methods. We analyse three recent state-of-the-art unsupervised-learning-based methods and evaluate their response to the domain shift, caused by applying five different transformations to the original dataset. We also investigate the number of training images needed to build a good model, as well as the amount of images from the shifted domain needed to be added to the original training set in order to robustly build a good model. The obtained results can be used for addressing different issues in the field and are useful for both researchers and practitioners working in the field of surface anomaly detection.

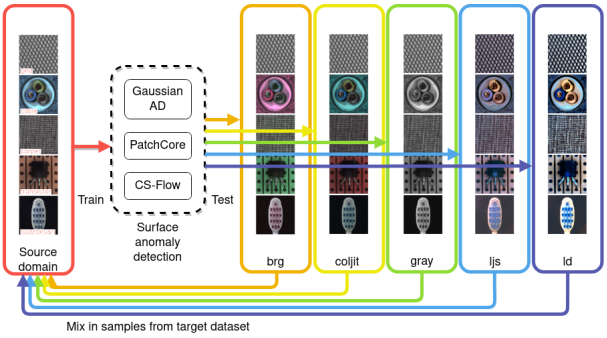

Figure 1: We investigate the robustness of anomaly detection methods to the domain-shift by training a model on a source domain and evaluating it on multiple target (shifted) domains. We also investigate the number of samples needed for building an accurate model, as well as the number of samples needed to adapt the model to the target domain.

1 Introduction

The field of surface anomaly detection is an enabling technology for various industrial and commercial applications. The problem is formulated as detection of parts of an image (anomalies, defects), depicting a surface of an object, that are inconsistent with a typical (normal) appearance of this surface. Keeping up with the times, the recent state-of-the-art surface anomaly detection methods are all based on CNNs, used in a variety of manners. In the standard deep learning paradigm, such a method would require labeled sets of ”normal” and ”anomalous” images. This supervised approach can quite often turn to be problematic — data labeling is costly, anomalies might be rare, or one might not know beforehand all the possible anomalies likely to appear in the future.

The field of unsupervised surface anomaly detection addresses these issues by requiring only ”normal” training images, that are used to build some type of a model that can discriminate between images that belong to the distribution observed during the training and images that are out of distribution or ”anomalous”. However, in many applications the distribution of ”normal” images is likely to change with time due to changes in the environment in which the images are taken, the objects of interest themselves might change, or the equipment used for taking the images might itself change.

In this work we evaluate the robustness of three of the most popular unsupervised surface anomaly detection methods, Gaussian AD [3]. PatchCore [4] and CS-Flow[5], to different domain shifts. For this purpose we use the popular MVTec [1] dataset and create five domain shifted datasets by changing the images in various ways. We evaluate the number of images needed for building a good model, the drop in performance when the domain shifts, as well as the number of images from the shifted domain that is needed to be included in the training process to get a performance comparable to the baseline.

The main contribution of the paper are therefore the results of the analysis and the insights derived from them, which we believe to be useful for researchers and practitioners working on the problem of visual surface anomaly detection.

2 Related work

In recent years the field of surface anomaly detection has been dominated by the methods based on unsupervised learning of specific CNN models. Various approaches to solving this problem have been proposed. A very popular approach used to be the analysis of the image reconstruction (e.g., like in RIAD [9]), also in combination with reconstruction approach (such as DREAM [8]]). Several

methods use pre-trained CNNs for feature extraction and then create models that take the extracted features [3, 4, 5] on the input and process them further to address the problem of consistency in the training images. In this paper we will analyse three different approaches to this paradigm, that turned to be among the most successful unsupervised methods for surface anomaly detection.

Gaussian AD uses an ImageNet pre-trained EfficientNet [6] to extract features from the training set of images, and build a Multivariate Gaussian distribution (MVG) model of the distribution of the features. During the test time the features of the input image are extracted and then the Mahalanobis distance to the MVG distribution of "normal" images is calculated. If the distance is above a set threshold the image is categorized as anomalous, without the possibility of localizing the anomaly.

PatchCore uses different types of pre-trained ResNet [2] architectures for feature extraction from all the images in the training set. Then it uses a *minimax facility location* method to extract a core-set of features that represent the distribution of "normal" images. At the test time, the features of the input image are compared to this core-set, and if the divergence is large enough the anomalies are detected and localized in the image.

CS-Flow uses an ImageNet pre-trained EfficientNet [6] architecture through which three scales of the input image are passed to get three feature representations of the image. Then, these features are used by a custom Cross-Scale Normalizing Flow [5] neural architecture to model the distribution (of the features) of "normal" images. During the test time we simply extract the features of the image and then pass them through the Cross-Scale Flow model and the outputs directly represent anomalous regions in the image.

3 Experiment design

Our analysis is centered around the MVTec dataset, containing images of 15 different objects. The dataset is the most common dataset in the field, used in virtually all published works on the topic of unsupervised surface anomaly detection. Five of the objects in this dataset can be seen in the leftmost column in Figure 2.

In order to measure the effects of the domain-shifts on the performance of the three anomaly detection algorithms we created five domain-shifted versions of the MVTec dataset. The *MVTec-brg* dataset was obtained by simply permuting the channels of the original images as RGB→BRG. The *MVTec-coljit* dataset was generated using the color jittering technique for image augmentation; in our case the same image transformation was applied to all images. All images were first transformed into HSV representation, all channels were multipled with a constant factor, and then the images were reverted to the RGB representation. *MVTec-gray* is simply a grayscale version of the original. *MVTec-ljs* and *MVTec-ld* were generated using the WCT2 [7] method for style-transfer, where two arbitrarily chosen images (images of Ljubljana sunrise and Luka Dončič) were used as a style image. A number of images from these datasets, showing the influence of the applied transformations, are shown in Figure 2.

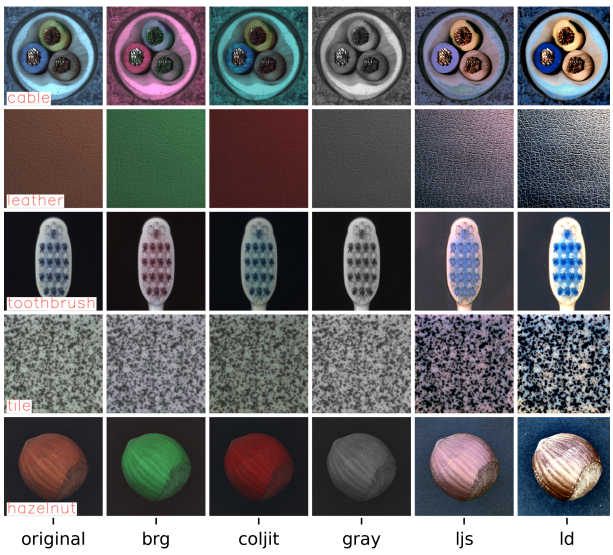

Figure 2: Examples from the domain-shifted dataset for five object categories. In the leftmost column we see the objects as they look in the original dataset. Other columns depict the domain shifted images.

In the *first experiment* we simply train all three methods on the original training set and evaluate the corresponding trained models on the original test set. This is a conventional setting, without a domain shift and the obtained results serve as a baseline for the other experiments. In the *second experiment* we examine the effect of reducing the size of the training set. We evaluate a model trained on 1, 5, 10, 50, 100, 200, and all available training images for each object from the original training set, and evaluate the performance on the complete original test set for each object. For the MVTec dataset the total number of training images for each object ranges between 219 and 280 with two exceptions; the "toothbrush" object contains only 60 training images and the "hazelnut" contains 391 training images.

In the *third experiment* all models that were trained on the source domain are evaluated on all shifted domains, and we record the difference in the performance in terms of the mean AUC metric. In the *fourth* and final *experiment*, we extend the training set consisted of all available original images (from the source domain) with 1, 5 or 10 images from the target domain in order to gauge the amount of images needed to adapt the trained model to the target domain by simply adding images from the target domain to the training set.

All experiments were performed with 5 fold cross-validation and we are reporting the mean values, as well as the standard deviations.

4 Results

All three methods perform relatively well on the original dataset. When trained on the complete training set of the original MVTec dataset, and tested on the complete test set of the same dataset, Gaussian AD achieves mean

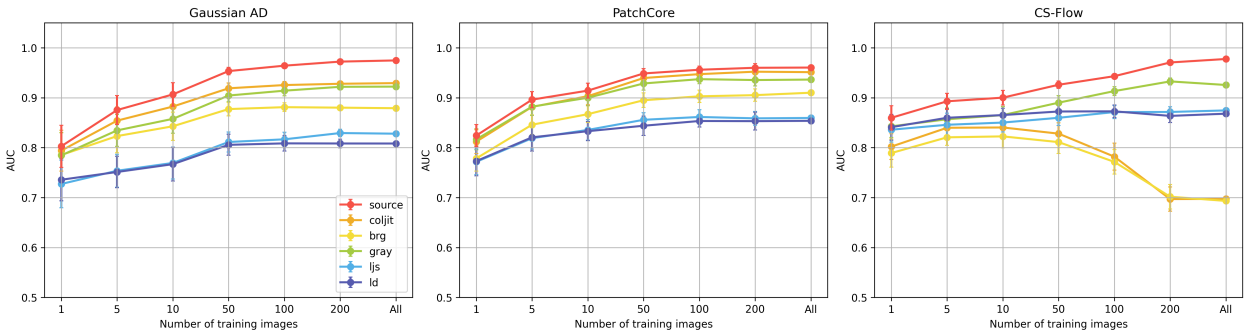

Figure 3: Performance of Gaussian AD, PatchCore and CS-Flow on the MVTec and domain shifted MVTec datasets. The horizontal axis shows how many images from the MVTec domain were used training learning the model. The vertical axis shows the mean AUC metric when evaluating on the complete test sets of each domain. Each color represents a different test domain, the red being the MVTec dataset.

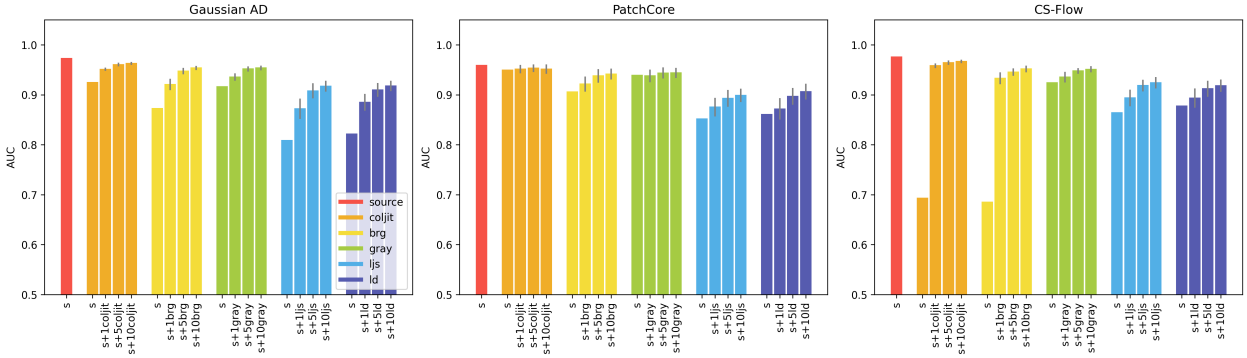

Figure 4: This image shows the results of adding images from the target domain to the training set of images from the source domain for each of the three examined methods. Results are shown for adding 1, 5 or 10 images to the complete training set of the original domain.

AUC of 0.97, PatchCore achieves mean AUC of 0.96, and CS-Flow has a mean AUC of 0.98. Certain object categories seem to be consistently easier for all three methods. For example, all three methods have a perfect score of 1 mAUC for the *leather* and *bottle* categories. The most difficult objects for all three methods are the *capsule* and *screw* objects. In general, the success of the *leather* category is probably due to its textural appearance, as all textural categories seem to be relatively easy to model. The good results on the *bottle* category as well as the bad results on the *capsule* and *screw* categories are probably due to the alignment between different samples – *bottle* is one of the best aligned categories, and *screw* and *capsule* are one of the mostly non-aligned categories. The results for these objects for the CS-Flow method are shown in Figure 5.

Next, we evaluated the effect of the size of the training set. As can be seen in Figure 3 (red line), all three methods benefit from larger training sets, however, on average, Gaussian AD and PatchCore seem to reach a plateau at about 100 samples in the training set, while for CS-Flow it seems that a plateau has not been reached yet. It is also surprising to note there are some objects where a single image is enough for building a good model, for all three methods. On average, with a single image Gaussian AD reaches a mAUC of 0.80, for PatchCore the score is 0.82 and for CS-Flow the score is 0.86. It is also notable, that as the number of training samples grows, the score is less dependent on the samples that we have chosen, as the standard deviation is reduced. If we examine the individual categories, we see that the best performing categories across all methods are *leather*, *tile*, *carpet* and *bottle*. The first three are textural categories and the last is highly aligned across samples. Some of the worst performing categories are *screw*, *hazelnut*, *capsule*, *transistor* and *toothbrush* as can be seen in Figure 5. For all these categories we can say that they are non-textural and relatively unaligned categories.

In the next experiment we evaluated the performance of the methods, when they are trained on the original MVTec and tested on MVTec-brg, MVTec-coljit, MVTec-gray, MVTec-ljs and MVTec-ld. For Gaussian AD we can see that the smallest drop in performance happens on the MVTec-coljit and MVTec-gray domains, the score drops further for MVTec-brg, and performs the worst on the MVTec-ljs and MVTec-ld domains. For PatchCore, we can see that the method is most robust to changes in the MVTec-coljit and MVTec-gray domains. Then the performance drops on the MVTec-brg domain, and it performs the worst on the MVTec-ljs and MVTec-ld domains. For CS-Flow we can see that the smallest drop in performance is observed on the MVTec-gray domain, then additional drop for the MVTec-ljs and MVTec-ld domains, while the largest drop in performance happens on the MVTec-coljit

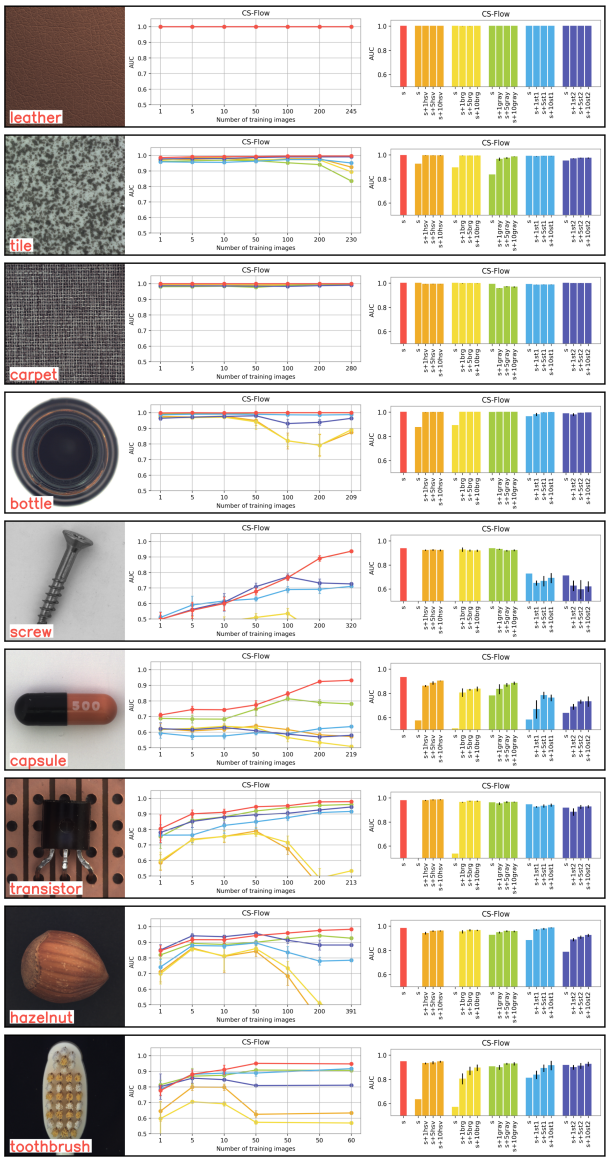

Figure 5: Results for specific objects for the CS-Flow method.

and MVTec-brg domains. Without any adaptation to the target domains, it seems that overall PatchCore experiences the smallest drop in performance. The reasons for this require additional investigation, however there are two obvious suspects, the feature extractor (ResNet vs. EfficientNet used by Gaussian AD and CS-Flow) and the core-set selection strategy.

For Gaussian AD and PatchCore, somewhere about 50 and 100 samples a plateau is reached, and using more samples does not significantly improve the performance on the domain-shifted datasets. For CS-Flow we can see that the performance on the MVTec-gray domain only starts to drop after using 200+ samples. For the MVTec-ljs and MVTec-ld domains the performance does not change significantly with the number of training samples, nor does the score start to drop. For the MVTec-coljit and MVTec-brg datasets we can see that already after 10 training samples of the original domain the score starts dropping, which suggests overfitting on the source domain. CS-Flow is expected to be more prone to overfitting because the model that it builds is more complex than the models of the other two methods

In the final experiment, we evaluate the effect of adding 1, 5 or 10 images from the domain-shifted dataset to the complete training set of the source domain. The mean results over all categories are depicted in Figure 4. For Gaussian AD we see that adding images from the training set brings the performance close to baseline, for the MVTec-coljit, MVTec-brg and MVTec-gray domains, while the improvements on the MVTec-ljs and MVTec-ld domains are smaller. PatchCore performs almost equally well on the domains MVTec-coljit and MVTec-gray without adding any training images. The performance on the MVTec-brg domain can be close to baseline if we add anough images to the training set, while the performance on the MVTec-ljs and MVTec-ld domains does not come close to the baseline. For CS-Flow we can see that the sharp drop in performance on the MVTec-coljit and MVTec-brg domains can be alleviated with a single image in the training set. The performance on the MVTec-gray domain is relatively well without any adaptation, and further improved by adding images. Again, the worst performance happens on the MVTec-ljs and MVTec-ld domains. As expected, adding images from the target domain helps the performance for all three methods.

5 Conclusion

In this work we created domain-shifted variants of the popular MVTec dataset for surface anomaly detection, and evaluated the influence of the size of the training set and the domain-shift on the performance of three popular surface anomaly detection algorithms.

From the experiments presented in this work we can draw a few conclusions. Firstly, the MVTec dataset has been ”solved” to a great extent, so much so that its usage might bring about more methods that are fitted to the dataset, instead of bringing about fundamentally novel approaches to this problem.

Secondly, we can see that the tested domain-shifts do bring about a drop in performance, but this drop is largely non-catastrophic. Since this holds true for all three methods, the reason for this is probably in the power of the used feature extractors to ”absorb” such changes in the image space.

Lastly, we can see that certain domain shifts, like the tested BRG, COLJIT and GRAY domain shifts are easy to adapt to, needing only a few images from the target domain in order to get a performance comparable to the baseline.

In order to correctly interpret all the presented results, a further investigation is needed into the way the feature representation of an image changes when the certain visual properties of the image are changed.

References

[1] Bergmann, P., Fauser, M., Sattlegger, D., Steger, C.: Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[2] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[3] Rippel, O., Mertens, P., König, E., Merhof, D.: Gaussian anomaly detection by modeling the distribution of normal data in pretrained deep features. IEEE Transactions on Instrumentation and Measurement (2021)

[4] Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., Gehler, P.V.: Towards total recall in industrial anomaly detection. arXiv (2021), https://arxiv.org/abs/2106.08265

[5] Rudolph, M., Wehrbein, T., Rosenhahn, B., Wandt, B.: Fully convolutional cross-scale-flows for image-based defect detection. In: Winter Conference on Applications of Computer Vision (WACV) (2022)

[6] Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Proceedings of the 36th International Conference on Machine Learning (2019)

[7] Yoo, J., Uh, Y., Chun, S., Kang, B., Ha, J.W.: Photorealistic style transfer via wavelet transforms. In: International Conference on Computer Vision (ICCV) (2019)

[8] Zavrtanik, V., Kristan, M., Skočaj, D.: Draem - a discriminatively trained reconstruction embedding for surface anomaly detection. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2021)

[9] Zavrtanik, V., Kristan, M., Skočaj, D.: Reconstruction by inpainting for visual anomaly detection. Pattern Recognition (2021)