Improving Industrial Quality Control with Computer Vision

Valeo Data Challenge, ENS 2025
MVA Master, ENS-Paris-Saclay

Emilio Picard, Rosalie Millner
emilio.picard@etu.u-paris.fr
rosalie.millner@ens-paris-saclay.fr

1 Introduction

Quality control in industrial settings plays a crucial role in ensuring product reliability while preventing defective components from propagating through production lines. Traditionally, this process relies on human inspection, where images of manufactured parts are analyzed to detect potential defects. However, manual inspections are prone to human error and can represent additional workload, making them inefficient for large-scale industrial applications. Deep learning modeling has recently made significant strides in the world of computer vision, greatly extending foundation models to industrial applications. This technologies promise good control over image classification tasks. Image classification systems rely on image inputs being classified as a class or a label.

In this project, as part of a challenge proposed by Valeo, we integrate deep learning-based computer vision models into the quality control workflow. The aim is to classify product images into known defect categories while also detecting anomalous images that may not fit predefined defect classes.

To achieve this, we explore advanced machine learning techniques, including self-supervised learning for anomaly detection (PaDiM with SimCLR), CNN classification (ResNet101), and confidence-based filtering to enhance decision-making reliability.

The report is organized as follow: after presenting the problem framework and describing the dataset, highlighting challenges such as class imbalance and unlabeled anomalies, we introduce the baseline model provided by the challenge organizers, and then expose our proposed improvements and final methodology. In the last section, we discuss our experimental results, benchmark different architectures and analyze their implications.

2 Problem Background

Deep learning techniques provide a strong solution for enhancing both the speed and accuracy of quality control. A current approach involves capturing images of products using a camera, and applying computer vision models to control their characteristics, and make sure they present no defect. In this section, we present the problem framework, which serves as the basis for our proposed improvements.

2.1 Provided Data

The dataset comprises 8,278 training images and 1,055 test images, covering five distinct defect classes along with a "good" class. Figure 1 presents one representative sample from each class in

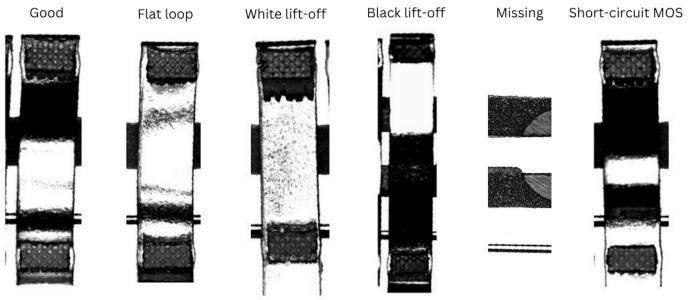


Figure 1: Example for each class from the training set, after rotation and cropping pre-processing

the training set. The images depict a component fixed within a product, with each class indicating whether the component is missing or specifying the type of defect present.
Additionally, the test set includes a drift class, representing anomalies that were not present in the training data. This introduces an additional challenge for deep learning-based approaches, as the data is unlabeled. Consequently, this task requires an unsupervised learning approach.

In addition, the training set is very imbalanced, as each class has a significantly different number of samples. Class distribution is shown in Table 1.

Good	Flat loop	White lift-off	Black lift-off	Missing	Short-circuit MOS
1,235	71	270	104	6,472	126

Table 1: Original distribution of image classes

This poses a challenge for classification models, as they must learn from an imbalanced dataset with varying amounts of data for different classes. In general, models risk becoming biased toward the majority class, leading to poor generalization for the rarer ones. This can result in lower recall for underrepresented classes, where the model may fail to detect certain defects.

2.2 Baseline model

In order to tackle the problem, we first followed the challenge's baseline. First, the industrial images are rotated and cropped to obtain an image concentrated around the region of interest, which aims to be more informative than the original raw image. The next step is to pass this image in both an **anomaly detector**, based on PaDiM Defard et al. [2020], and a **classifier** (here a Resnet50).
The role of PaDiM is to determine whether an image contains an anomaly. It functions as a binary classifier, distinguishing between normal and anomalous samples. If an anomaly is detected, the image is classified as "drift". Otherwise, we go through a classifier that classifies the corresponding class to the image (either "good" or a defect). The baseline architecture is shown in Figure 8.

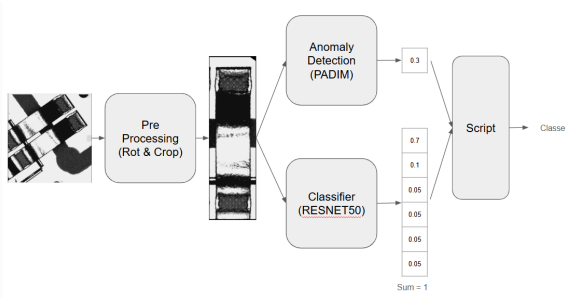


Figure 2: Baseline architecture

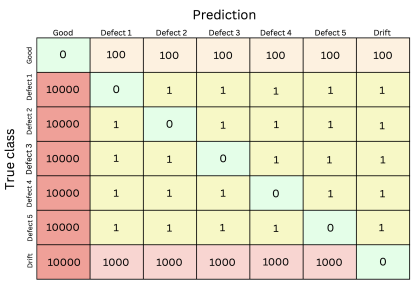


Figure 3: Penalty Matrix

2.3 Evaluation metric

The final process results in 7 possible label predictions, which are then compared to the ground truth.

The challenge evaluates model performance based on a weighted scoring system which prioritizes accurate defect/anomaly detection. It imposes higher penalties for misclassifying defective/anomalous products as good, as such errors can have worse consequences in quality control. On the other hand, the misclassification of a good product is penalized less severely. The penalty matrix used to compute the weighted accuracy is illustrated in Figure 9.

The baseline model obtains a penalty-weighted accuracy of: `0.9659`.

3 Methodology

With the aim of finding the best possible model for the task, we explored a variety of different approaches and architectures. The following section presents our final method selection.
We first explain how we handled class imbalance, followed by an overview of our anomaly detection pipeline which employs PaDiM trained through a self-supervised technique. Then, we show how we conducted our classification with a pre-trained ResNet101, using a two-step training strategy.

Data Pre-Processing & Handling Class Imbalance

For the pre-processing stage, we maintained the same approach as the baseline, by preserving the rotation and cropping strategy, using the same parameters. This pre-processing is well-justified, as it focuses on the region of interest, which is considered to contain the essential information for distinguishing between classes.

Regarding the issue of class imbalance, we first noticed that the "missing" class, despite being the most represented, exhibited minimal variation in the images. This class also stands out more easily from the others, which made it less challenging for the model to classify. As a result, we decided to reduce the number of "missing" images by 75%, decreasing the total from 6,472 to 1,294 images. We assessed that this reduction did not lead to an actual loss of information. Similarly, we reduced the number of "good" class images by 20%, lowering the count from 1,235 to 988 images. After these adjustments, the total number of images in the training set was 2,853. The class proportions in the dataset remain imbalanced, but way less than before.

The main strategy to address class imbalance was the introduction of weighted loss functions. Specifically, we employed a Class-Weighted Cross-Entropy Loss inside our classification models, which assigns higher penalties to misclassifications of minority class samples. This approach helps mitigate the bias toward the majority classes and encourages the model to pay more attention to the underrepresented classes. Further details on this can be found here.

Another approach to addressing class imbalance is applying data augmentation to minority classes (i.e., all except "good" and "missing"). We implemented contrast augmentation with a 0.4 probability of being applied. However, we chose not to use RandomCrop, RandomFlip, or RandomRotation augmentations, as the images were already preprocessed to focus on the region of interest. Applying such transformations could make the components harder to recognize, potentially hindering model performance.

For the subsequent training phase, we used the adjusted dataset of 2,853 images. For the classification model, we randomly split the dataset into training and validation sets with an 80%-20% ratio.

Anomaly Detection via PaDiM & Self-Supervised Learning

In this study, as evidenced by state-of-the-art, we use a CNN-based architecture for anomaly detection. Specifically, we employ a ResNet-50 model (He et al. [2015]), pretrained on the large-scale ImageNet dataset. To adapt the model to our task, we fine-tune it by unfreezing its

parameters and training it in a self-supervised manner using the SimCLR framework (Chen et al. [2020]). SimCLR leverages contrastive learning to learn meaningful feature representations, even in the absence of labels, by maximizing agreement between differently augmented views of the same image.

Several anomaly detection training tools are available for one-class training (Bergmann et al. [2020], Cohen and Hoshen [2021]). Recently, two main approaches have emerged: SimCLR and MoCo. In this work, we opted for SimCLR as our chosen method.
SimCLR achieves this by applying two different stochastic augmentation transformations (e.g. color jittering, Gaussian blurring, random flip, etc.) to an input sample and passing both augmented versions through a shared encoder network. We opted not to include cropping and rotation in the possible transformations, as the region of interest is already well-defined by our pre-processing. We believe introducing them in the SimCLR transformations could potentially degrade feature learning rather than enhance it. The resulting feature embeddings are then projected into a lower-dimensional space using an MLP projection head. The model is trained using the normalized temperature-scaled cross-entropy loss (NTXentLoss), defined in Chen et al. [2020] as:

$$\mathcal{L}_{\text{NTXent}} = -\sum_{i \in I} \log \frac{\exp(\text{sim}(z_i, z_i^+)/\tau)}{\sum_{k \neq i} \exp(\text{sim}(z_i, z_k)/\tau)} \tag{1}$$

where:
· z_i and z_i^+ are the normalized embeddings of two different augmentations of the same instance,
· $\text{sim}(z_i, z_i^+)$ represents the cosine similarity between these embeddings,
· τ is a temperature parameter that controls the sharpness of the similarity distribution,
· $k \neq i$ denotes all other samples in the batch, which act as negative examples.

The objective is to pull together embeddings of the same instance ("positive pairs") while pushing apart embeddings from different instances ("negative pairs"). By leveraging SimCLR's contrastive learning paradigm, our method enables the CNN to learn robust, transferable feature representations suitable for anomaly detection without requiring explicit supervision.

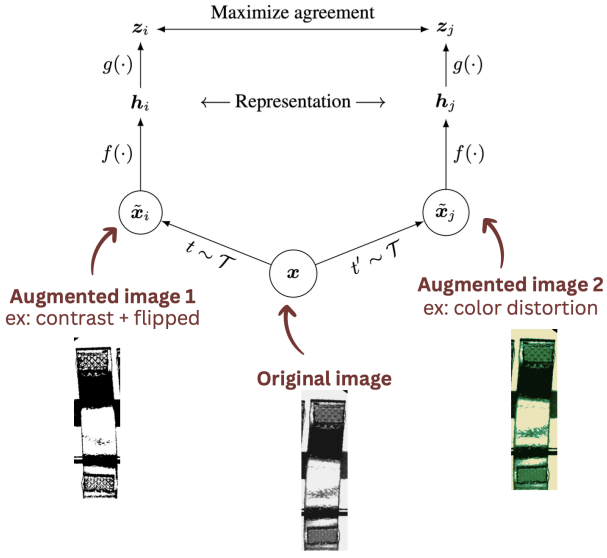


Figure 4: Contrastive Learning Framework

One could have also used MoCo (Momentum Contrast, He et al. [2020]), which also employs contrastive learning for self-supervised representation learning, and maintains a dynamic memory bank to store a large set of negative samples.

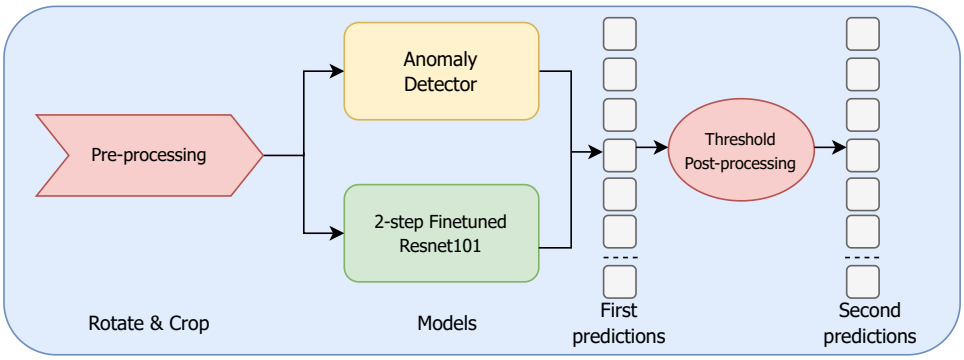


Figure 5: Overview of the proposed pipeline. Initially, the images are rotated and cropped for preprocessing. Next, they are passed through both detection and classification models, which generate logits. Finally, a post-processing step involving thresholding is applied to obtain the final predictions.

After training, the anomaly detection process is performed using the PaDiM technique (Patch Distribution Modeling, Defard et al. [2020]), a well-established method for anomaly detection. It is known to work well for industrial defect detection, as it detects local anomalies, rather than global ones. It is therefore useful for inspecting manufacturing images to detect defects or inconsistencies on products, such as in our case, which is why we decided to keep the PaDiM idea from the baseline model. PaDiM leverages a probabilistic modeling approach by extracting patch-level feature distributions from a pretrained CNN. Specifically, it uses Gaussian modeling to estimate the distribution of "normal" features within an image. During inference, test samples are compared against this learned distribution, and anomalies are identified based on deviations from expected statistical properties, based on the Mahalanobis distance:

$$D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

where μ and Σ represent the mean and covariance of the normal feature distribution. High Mahalanobis distances indicate potential anomalies.

To enhance localization accuracy, PaDiM applies Gaussian filtering to smooth the anomaly score map, reducing noise and improving robustness. The final anomaly map highlights regions with significant deviations from the learned normal feature space, allowing precise detection of defects.

Classification Model: Two-step training

In this work, we employ a CNN-based model to classify both defective and non-defective images across six distinct classes. Figure 5 provides an overview of the training setup.
The first training phase utilizes a ResNet-101 architecture pretrained on a large-scale dataset. To adapt the model to our problem, we modify the last fuly connected layer by another that outputs a tensor of size 6. We then freeze all network parameters except for *layer4* and the *final fully connected layer*, resulting in approximately 2 million trainable parameters. We thus used finetuning. To address class imbalance, we apply a weighted cross-entropy loss function.

$$\ell_{ce} = \sum_{i=1}^{n_{\text{classes}}} k_i \times y_i \log \hat{p}_i,$$

where $\hat{p}_i$ correspond to the predicted logit for class i, and k_i is a proportion value for each class of the dataset. In practice, we set the $(k_i)_i$ to [5, 35, 9, 23, 2, 19]. This is computed as follow: $\forall i \in \{0, \dots 5\},\ k_i = \dfrac{\#total}{\#class_i}$.
The model is trained for 15 epochs using an Adam optimizer, with an initial learning rate of 0.01, which is progressively reduced through a StepLR scheduler. All training hyperparameters can

be found Table 3. Training was conducted on a single RTX 4050 Laptop GPU and completed in approximately 20 minutes. The `best_model_weights` are saved based on the highest validation accuracy observed during training.

In the second phase, we reload the model with the fine-tuned weights from the first step and unfreeze all network parameters. The entire model is then further trained with a significantly lower learning rate (e.g., $10e^{-6}$) for an additional 5 epochs. This final refinement step ensures that all model weights are fine-tuned while preserving previously learned features. However, due to the need for full backpropagation across all layers, this step is computationally intensive and requires a longer training time. This second learning phase helped enhance our accuracy by around one/two percent.

Grad-CAM

Grad-CAM (Selvaraju et al. [2019]) was utilized to gain a deeper understanding of the dataset by visualizing the regions of interest that influence the model's decisions. By generating class activation maps, Grad-CAM highlights the areas within an image that contribute most to a given classification, allowing for an interpretability analysis of the model's focus. This technique helps assess whether the network is attending to the relevant features of defective and non-defective components, ensuring that the learning process is aligned with meaningful patterns in the data. Through this approach, we can refine our model and improve its reliability in identifying defects.
In addition, we initially applied it on one of our experimental models to gain a better understanding of the data, as we were not yet familiar with this type of industrial images.

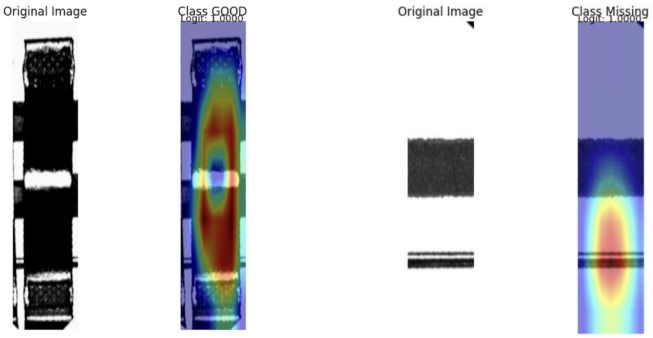


Figure 6: Two interpretation examples from Grad-CAM. The model appears to focus on relevant areas of the product. For the "good" class it assesses most of the piece, while for the "missing" class it highlights a background detail that effectively indicates the absence of the piece.

Additional insights from Grad-CAM is while we were visualizing our data, we noticed some very drift-looking images within the *training* set, potentially impacting classification performance. A few examples, along with their respective filenames, are given in Figure 10. If such samples are indeed anomalies that are misclassified as belonging to one of the classes from the training set, classification could overall be improved by removing them, especially regarding the anomaly detection model.

Post-Processing & Confidence-Based Filtering

To ensure robustness in classification, we apply confidence-based filtering, which leverages classification uncertainty in logits as an indicator of potential anomalies (Hendrycks and Gimpel [2018]). This method acts as a **secondary anomaly detection mechanism**.

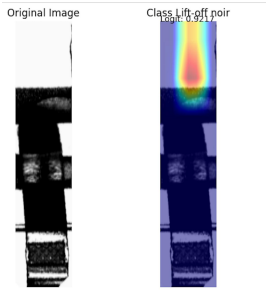


Figure 7: Grad-CAM applied to an anomaly-looking sample from the training set. The model incorrectly focuses on an irrelevant region (the background) rather than the actual piece.

We consider that our ResNet101 classification model has demonstrated strong performance on the training set. Therefore, if the maximum logit is not sufficiently high, we suspect that an anomaly is likely occurring. Since some misclassifications carry greater consequences—particularly for the "good" and "drift" classes (as defined by the penalty matrix in [9]), we introduce class-specific confidence thresholds. These thresholds are determined empirically to balance defect detection sensitivity and false positive rates.

The task at hand is to define a threshold τ_{drift} to classify samples as "drift" or not. If the logit value for p_{drift} exceeds this threshold, we classify the sample as a drift. On the other hand, if we determine that a sample is "normal," we apply a separate threshold, τ_{good}, to ensure we only confidently label it as "good" when the classification is "certain enough". This further ensures that predictions made on these critical classes are reliable, minimizing the risk of incorrect classification in potentially highly-penalized scenarios.

The process is summarized in Algorithm 1.

Algorithm 1 Logits Post-Processing for Each Sample

Input: Logits $p = (p_0, p_1, p_2, p_3, p_4, p_5)$ from classification and logit p_{drift} from PaDiM
Output: Final label prediction

if $p_{\text{drift}} > \tau_{\text{drift}}$ **then**
    Label $\leftarrow 6$ ▷ Anomaly detected
else if $\max(p) = p_0$ **then** ▷ Most likely class is "good"
    if $p_0 > \tau_{\text{good}}$ **then**
        Label $\leftarrow 0$ ▷ Confidently "good"
    else
        Label $\leftarrow 6$ ▷ Uncertain, reclassified as "drift"
    end if
else
    Label $\leftarrow \arg\max(p)$ ▷ Standard classification
end if
return Label

With the proposed methodology established, we now present the experimental results and evaluate the performance of our approach.

4 Results

The code of the project is available at https://github.com/emilio-pcrd/MVA_Valeo. The proposed method achieved a weighted test accuracy of 0.9877 on the public dataset, leading to the 4th position in the challenge competition.

As previously mentioned, all the following results were obtained using a ResNet101 architecture. Before selecting this model, we evaluated several alternatives, including ResNet34 and

ResNet50, as well as VGG19 (Simonyan and Zisserman [2015]) and DenseNet161 (Huang et al. [2018]), both recognized for their strong classification performance (Liu et al. [2023]). However, the validation set results for these models were lower compared to the three ResNet architectures, leading us to adopt ResNet-based models for our approach.
We thus provide here a benchmark of our different pos-processing approaches, using a ResNet architecture for all experiments. Table 2 is a benchmark of our experiments compared to the baseline. The benchmark compares different parameters settings, showing their associated test weighted accuracy (after submission online).

In table 3 (Appendix A), one can find all of our hyper-parameters that has been used for the experiments, from the data split to all parameters training.

Thresholds	Test Weighted Accuracy
Baseline	0.9659
Without thresholding	0.9702
$\tau_{\text{good}} = 0.8$, $\tau_{\text{drift}} = 0.5$	0.982
$\tau_{\mathbf{good}} \mathbf{= 0.95}$, $\tau_{\mathbf{drift}} \mathbf{= 0.5}$	**0.9877**
$\tau_{\text{good}} = 0.85$, $\tau_{\text{drift}} = 0.45$	0.9683
$\tau_{\text{good}} = 0.85$, $\tau_{\text{drift}} = 0.55$	0.9794

Table 2: The performance of models in classifying industrial images is evaluated, with all models built upon a ResNet101 architecture. They are distinguished by two thresholds, τ_{good} and τ_{drift}, corresponding to confidence values to ensure the reliability of predictions.

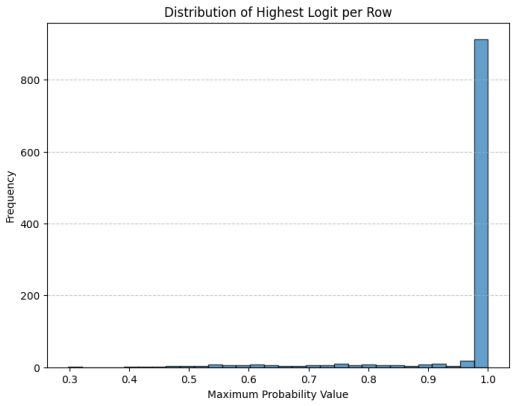


Figure 8: Distribution of highest logits from classification

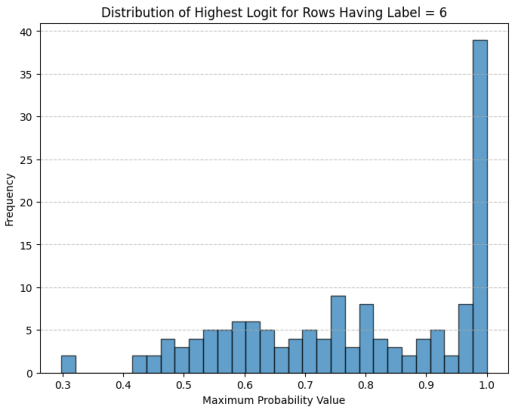


Figure 9: Distribution of highest logits from classification for the samples having $p_{\text{drift}} > 0.5$

5 Conclusion

We introduced an approach to classify industrial images from the Valeo Data Challenge. Our study demonstrates that integrating contrastive learning with PaDiM and fine-tuning ResNet101 with a dual step approach significantly improves quality control in Valeo's industrial setting. This approach improves the reliability of deep learning for automated defect detection compared to human inspection. To further enhance accuracy, future work could focus on improving anomaly detection by filtering potentially mislabeled training samples. Additionally, exploring alternative self-supervised techniques such as PatchCore (Roth et al. [2022]) or SPADE (Cohen and Hoshen [2021]) could provide valuable improvements in anomaly detection performance.

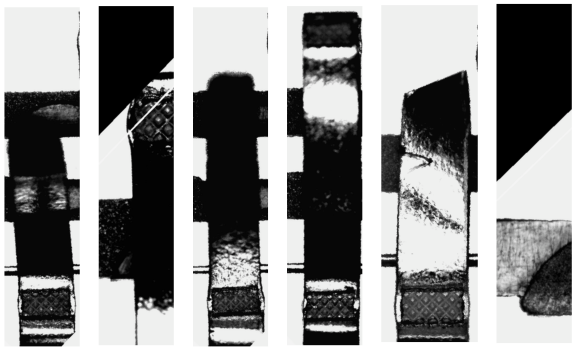
Appendix

A Hyper-parameter Table

data	2,853 for train and val (0.8/0.2). 1,055 for test
num epochs 1	15
num epochs 2	5
optimizer	Adam
optimizer momentum	$\beta_1, \beta_2 = (0.9, 0.999)$
learning rate 1	$1e^{-2}$
learning rate 2	$1e^{-5}$
scheduler	Linear, timesteps=5, $\alpha = 0.5$
batch size	64
loss	ℓ_{ce} proposed above

Table 3: Hyper-parameters tuning for training.

B Anomaly looking samples from the training set



filenames, from left to right:
f0fa1eef66a4dba5b9f5c060af0fc835fe77488c8aff498570326298c4848831.png
8b3a8f3c68228b3d427c9267b71ad4df9c18bc9d163ca1f748bea4e349630327.png
c53e06c710518bb02caabc4e1c5187501f85cd614ae9d685224ab5a2d477500c.png
ed72fee802ac1d3a1301cd6a719353c92ef9ede612335e72a728438a5a33af20.png
274c4cd2b4d0fa081c26045dd8c618e6c991abba3899c8945c255a0dcc51ef18.png
1a96f9c232636e5c82913efef78c82cbc8b23d35d3293a296f8078f7107805cd.png

Figure 10: Example of samples from the train set that seem anomalous. Some of them seem to have had wrong pre-processing settings (for rotation+crop), while others simply appear as anomalies, from a human perspective.

References

P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. doi: 10.1109/cvpr42600.2020.00424. URL http://dx.doi.org/10.1109/CVPR42600.2020.00424.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations, 2020. URL https://arxiv.org/abs/2002.05709.

N. Cohen and Y. Hoshen. Sub-image anomaly detection with deep pyramid correspondences, 2021. URL https://arxiv.org/abs/2005.02357.

T. Defard, A. Setkov, A. Loesch, and R. Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization, 2020. URL https://arxiv.org/abs/2011.08785.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning, 2020. URL https://arxiv.org/abs/1911.05722.

D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2018. URL https://arxiv.org/abs/1610.02136.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2018. URL https://arxiv.org/abs/1608.06993.

C. Liu, Y. Dong, W. Xiang, X. Yang, H. Su, J. Zhu, Y. Chen, Y. He, H. Xue, and S. Zheng. A comprehensive study on robustness of image classification models: Benchmarking and rethinking, 2023. URL https://arxiv.org/abs/2302.14301.

K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler. Towards total recall in industrial anomaly detection, 2022. URL https://arxiv.org/abs/2106.08265.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct. 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL http://dx.doi.org/10.1007/s11263-019-01228-7.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. URL https://arxiv.org/abs/1409.1556.