Model-guided contrastive fine-tuning for industrial anomaly detection

Aitor Artola\textsuperscript{1,2}, Yannis Kolodziej\textsuperscript{2}, Jean-Michel Morel\textsuperscript{3}, Thibaud Ehret\textsuperscript{1}
\textsuperscript{1}Universit\'e Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, France
\textsuperscript{2}Visionairy
\textsuperscript{3}University of Hong Kong, Department of Mathematics, Kowloon, Hong Kong
\{aitor.artola,thibaud.ehret\}@ens-paris-saclay.fr, yannis.kolodziej@visionairy.io, jeamorel@cityu.edu.hk

Abstract

State-of-the-art industrial visual anomaly detection now relies on modeling the distribution of pre-trained neural network features. To this goal, most of the work has focused on how to model features of normal data and the choice of the pre-trained network. The current trend is to use a network pre-trained using self-supervised contrastive learning so that the same network can be used for all possible downstream applications. However, this also means that the network is object and task agnostic, meaning that features are very generic and not optimized with the detection model. In this paper, we propose to look at how to specialize features for a given application so as to improve performance and propose a fine-tuning process taking advantage of the differentiability of some popular models. This fine-tuning is performed following a contrastive learning framework meaning that no real anomalies are necessary during the process. We demonstrate the improvement on both localization and quality of detection on the MVtec dataset.

1. Introduction

Defect and anomaly detection is a crucial step in production lines. Historically, this task has been performed by human operators, who are able to discriminate between expected samples and anomalous ones based on very few samples, or even on a single one. This task is however not as simple to automatize. To remain general, the detection of such anomalies cannot be considered as a classification task: anomalies are few and do not necessarily repeat, which prevents the construction of a supervised training dataset. Automatizing the process would accelerate, reduce the production cost, and smooth out the performance variations caused by the operators’ fatigue [27, 40].\footnote{Work supported by a CIFRE scholarship of the French Ministry for Higher Studies, Research and Innovation.}

Initially, automatic anomaly detection focused on narrow domain-specific solutions, for example for texture-like objects [1, 42] (texture, metallic surfaces, \ldots). However, the main problem is that these solutions make strong assumptions about normality that prevent them from being generalized. Over time, the community has proposed solutions that extract application biases from the model design. These models make general assumptions about normality and anomalies, and adapt to each application with a training phase on anomaly-free data that is assumed to be abundant. In the end, the problem of visual anomaly detection in industrial images has been redefined as an outlier detection problem. Recent developments in deep learning and the publication of benchmark datasets [5, 22, 50] have inspired the community and many new solutions have been proposed in the past few years, focusing on the robustness and generality of the method by taking advantage of the latest deep learning innovation such as using a pre-trained backbone for image representation.

In this work, we propose to overcome the difficulty of choosing the right backbone for each object representation by fine-tuning it. Recent development in self-supervised learning is an opportunity for anomaly detection, as it allows us to define a completely end-to-end self-supervised fine-tuning that does not require the knowledge of real anomalies, thus in line with recently proposed anomaly detection methods. By taking advantage of the differentiability of some detection models applied on the pre-trained features, we define a contrastive loss that compares normal samples with random images from ImageNet [35]. ImageNet images offer the diversity required to discriminate potential anomalies from normal features. Since ImageNet potentially also includes parts similar to normal data, we also show how to filter the corresponding features to prevent the training from degenerating.
2. Related work

There are multiple anomaly detection reviews in the literature [16, 26, 34] that try to summarize and organize the proposed solutions. The usual first distinction is between non-deep learning and deep learning based methods. A second distinction is between those that train a network, often using supervision or synthetically generated examples, and those that use generalist pre-trained networks.

Trained deep-learning methods. Adapting deep learning for anomaly detection is a challenging task because the training of such methods often relies on fully labeled datasets for supervised training. Such datasets are often difficult, or even impossible, to obtain in that case. Draem [48] addresses this problem by generating artificial anomalies and expecting the network to generalize to real anomalies. Most other methods involve training that is not directly related to detection. In encoder-decoder methods [6, 25, 39, 43], a network is trained to compress and decompress normal data. During inference, it should fail to reconstruct never-seen structures such as anomalies and highlight them in the residual between the input and output of the network. Student-Teacher [7, 33, 38, 44, 47] are two neural networks with different levels of knowledge. The teacher is a generalist image encoder that distills its knowledge to the student, but only for normal images. Since the teacher has not shared its entire representation, both networks are not expected to give the same result for anomalies. The normalizing flow [18, 31, 32] network maps the complex and unknown distribution of anomaly-free data to a much simpler known distribution, for example the normal distribution, thanks to learned invertible mappings. In that framework, anomalies are singled out as what deviates from the learned distribution.

Pre-trained backbone based methods. Considering the difficulty to control the non-direct training methods, multiple approaches propose to bypass the training by using pre-trained neural networks. In these proposals, deep learning is used only for image representation and the background modeling is done with non deep learning solutions. Early features of neural networks trained on generalist datasets such as Imagenet [14] allow to improve the performance of classical models. SPADE [12] is a global homogeneous model that looks at the residual between a tested image and its nearest neighbor in an anomaly-free set in the feature space of an encoder. Patchcore [30] switches to a non-local model by constructing a cleverly subsampled normal feature set and searching for the farthest feature of a new image with the normal set. Instead of using sample-to-sample distance, others suggest estimating stochastic parametric models. The most common is the Gaussian distribution, which is used globally in MahalanobisAD [29] and locally in Padim[13]. GLAD[2] extends this to multi-modality by using Gaussian mixture models. More recently, Heckler et al. [21] studied the importance of the choice of the backbone.

Self-supervised network training. The pre-training step is crucial to make the previous backbones generic and discriminative enough for industrial anomaly detection. There are currently two popular options available for training such networks in an unsupervised manner. First, the masked auto-encoders [4, 15, 20, 41, 45, 46] (MAE), which mask parts of the input data and train the network to predict the missing pixels. The last layer, predicting pixel color from the learned features, is then discarded when applying this pre-training to other tasks such as classification. Another option for generic self-supervising training is contrastive learning [11, 17, 19, 49]. The idea is to encourage features from an image and its augmented version to be close to each other while features from two different images should be far from each other. More details about contrastive learning is provided in Section 3.1.

3. Model-guided contrastive fine-tuning

While the strength of contrastive learning comes from its generality, one can wonder whether this wouldn’t hinder performance for some applications. Indeed, the features learned this way are completely task and scene agnostic, meaning that the training is not influenced by the model that will be used for the anomaly detection in our case. This is why we propose, in this section, a model-guided fine-tuning to improve the quality of the features for a given object for the specific task of anomaly detection.

In particular, we show that the normality models used for anomaly detection by pre-trained backbone based methods, presented in Section 2, can be leveraged to define a pseudo-labeling procedure for feature space learning, leading to application-specific feature spaces. In this work, we consider Gaussian model based methods like MahalanobisAD [29], Padim [13] or GLAD [2]. These models have the advantage of being differentiable, an important requirement to make the process end-to-end. They are characterized by their weight $(\pi_k, \mu_k, \Sigma_k)_{k \in \{1,...,M\}}$. For Padim, $\pi_k = 1$ and each position is characterized by a single Gaussian $\mathcal{N}(\mu_k, \Sigma_k)$. For GLAD, $\pi_k$ depends on the position and each position is characterized by a Gaussian Mixture Model (GMM) $\sum_{k=1}^{M} \pi_k \mathcal{N}(\mu_k, \Sigma_k)$, with Gaussians shared between position. In both cases, the Gaussians are fitted on the normal data (either directly or with an EM algorithm). Note that using a GMM as a self-labeling process has already been used by self-supervised learning methods based on clustering, competitive with contrastive
methods [3, 9, 10]. The proposed method should nevertheless be compatible with other normality models as long as they are differentiable (or made differentiable, for example by replacing the k-NN search of PatchCore [30] by a differentiable approximation). Thanks to the differentiability of the normality models, the pseudo-labels allow us to use a contrastive learning framework to fine-tune the features for a specific application in an end-to-end process: the network and the clustering models are updated jointly.

The general concept of the model-guided contrastive fine-tuning is illustrated in Figure 1.

3.1. Principles of contrastive learning

The idea of contrastive learning is to learn representations that are invariant to aspect variations to force the network to focus on the semantic content of the image. Sample-to-sample contrastive methods [11, 17, 19, 49] are composed of two twin networks: one \( f_0 \) that generates features and another \( f_{EMA} \), called the momentum encoder, that doesn’t backpropagate the gradient but instead generates pseudo-labels used for training. Consider two random transformations \( T_1(I) \) and \( T_2(I) \) of a given image \( I \). The idea is that the contrastive loss on one image batch of size \( M \) will encourage the feature \( f_0(T_1(I_j)) = q^j \) of image \( I_j, j \in [1, M] \) to move closer to its momentum counterpart \( f_{EMA}(T_2(I_j)) = k^j \) and farther away from the features of the other images of the batch, i.e.

\[
\text{loss}(I^1, \ldots, I^M) = \sum_{j=1}^{M} \log \frac{e^{q^j k^j / \tau}}{\sum_{i=1}^{M} e^{q^i k^j / \tau}} \tag{1}
\]

where the temperature \( \tau \) acts as a scaling parameter. Note the resemblance to a cross-entropy loss.

Despite not being backpropagated, the momentum encoder is not fixed during the training process and is updated by exponential moving average (EMA). This is done using a convex combination, of parameter \( \gamma \in [0, 1] \),

\[
\theta_{EMA}^{t} = (1 - \gamma)\theta_{EMA}^{t-1} + \gamma \theta^t. \tag{2}
\]

This slower update has proven to stabilize the training.

3.2. Pseudo-labeling of training examples

Instead of directly comparing samples like in Section 3.1, we propose to leverage the detection model to perform the contrastive learning. Indeed, a good contrastive loss in our case should push the features corresponding to the normal data toward the normality model while pushing other samples far away from it. For this reason, we consider two sets of training samples: on one side the normal samples (also referred to as inliners in the rest of the text) that are traditionally used to train normality models but also negative samples that we will detail in Section 3.5 (also referred to as outliers).

In order to take into account these negative samples, we adapt the normality model, characterized by Gaussians as mentioned in Section 3, and add an outlier component as suggested by Bonnaire et al. [8]. In practice, this outlier component is modeled by a uniform distribution \( U \) over the volume covered by the GMM and is combined with the GMM with an arbitrary mixture weight \( \alpha \in [0, 1] \). The uniform distribution is chosen because no prior on the negative samples distribution is available.

This leads to the following distribution of samples

\[
\begin{align*}
\mathbb{P}(u^+) &= GMM(u^+) = \sum_k \pi_k N(u^+ | \mu_k, \Sigma_k) \tag{3} \\
\mathbb{P}(u^-) &= U(u^-) = \frac{1}{L} \mathbbm{1}_{u^- \in U} \tag{4} \\
\mathbb{P}(u) &= (1 - \alpha)GMM(u) + \alpha U(u) \tag{5}
\end{align*}
\]

where \( u^+ \) is an inliner, \( u^- \) is an outlier and \( u \) a sample without a priori.

3.3. Network fine-tuning

The goal of the fine-tuning is to adjust the network so as to enhance as much as possible the contrast between \( u^+ \) and \( u^- \). The first intuition would be to define the loss as the lower bound of the likelihood, similarly to the quantity maximized by the parameters of the GMM, with an additional outlier component. While this loss will encourage \( u^+ \) to move toward the center of the mixture model in order to maximize it, it will have no effect on \( u^- \). In fact, since \( u^- \) is uniform and has a zero gradient, it has no potential to climb. Indeed, the likelihood has no particular effect on the distance between inliners and outliers.

A better choice is therefore to maximize the log-likelihood of the latent variable \( z, \Theta \in \{+, -\} \) for which we defined a pseudo target in Section 3.2. This is similar to training a binary classifier. In that case, we define a positive and a negative loss for both inliners and outliers:

\[
L_{pos}(u^+) = -\log \mathbb{P}(z = +| u^+, \alpha, \Theta) \tag{6} \]
\[
L_{neg}(u^-) = -\log \mathbb{P}(z = -| u^-, \alpha, \Theta) \tag{8}
\]

We can note that this is similar to minimizing the cross-entropy, presented in Section 3.1, except that the classifier is not a linear one but a mixture of densities instead. Furthermore, we can see in \( L_{neg} \) that the denominator will push the negative examples out of the clusters, thus increasing the contrast between positives and negatives while pulling the positives back into the clusters. It is important to remark
that optimizing the cross-entropy is not equivalent to maximizing the likelihood of the GMM and thus, the optimal fitting requires an additional GMM fitting step. Indeed, the cross-entropy loss will modify the features but its long-term effect is also to modify the mixture model, which has no reason to maximize the likelihood. We describe the update step in Section 3.4.

As recommended by contrastive learning methods, we also keep the principle of two encoders because it allows for a more stable convergence of the model. The first encoder is a live encoder \( f_\theta \), which gets the gradient corresponding to the losses (7) and (9). The second encoder is a momentum encoder called \( f_{\theta_{EMA}} \), which blocks the gradient. Its purpose is to create features \( u_{EMA}^+ \), which are then used to update the parameters of the GMM. The momentum encoder is updated with the live encoder parameters using an exponential moving average (EMA) of the parameter \( \gamma_{EMA} \) as described in Eq. (2).

Because of the negative examples, some GMM components are encouraged to be pushed toward infinity. This is because the network is allowed to put as much distance as possible between the positives and negatives, what the training loss encourages as the goal is to separate as best as possible inliers and outliers, since the space is unconstrained. This can lead to an expansion of the feature space and to a reduction of the uniform probability at each point, thus also causing numerical instability. Our proposed solution is to constrain the features in a closed space by projecting the feature norm from \([0, \infty]\) to \([0, 1]\) by adding a sigmoid layer \( \text{sig}(x) \) with

\[
\text{sig}(x) = \frac{x}{1 + \|x\|}
\]

at the end of the network. This change makes it more and more difficult to spread the component and therefore the network is constrained to reorganize the feature space to minimize the loss. Another advantage of this constraint is that the uniform area is defined on the unit sphere and therefore is fixed during the entire fine-tuning process. Indeed, the volume \( L \) is fixed to the volume of the unit hyper-sphere of dimension \( d \):

\[
L = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2} + 1)}
\]

where \( \Gamma \) is the Euler’s gamma function.

### 3.4. Online update of the normality model

The fine-tuning process of the neural network features forces us to regularly update the normality model so that
it continues to fit the features after each network update. Instead of learning a new model from scratch every time, we propose to take advantage of online GMM learning methods. Traditional EM algorithms for GMM learning require that all samples are seen to compute a single step, meaning that they are costly. In online alternatives [36, 37], the update is done on a batch of data, similar to neural network training, by adding an intermediate step: computing and updating sufficient statistics. These sufficient statistics act as a global memory that aggregates information from all seen image batches.

Updating the GMM at iteration $t$ with the batch samples $u^i_{EMA}$, with $i \in [1, M]$, follows four steps:

1. **E-step**, assign each feature $u^i_{EMA}$ to a component by sampling its latent variable $z_i$,

   \[
   P(z_i = k \mid x, \Theta^t) = \frac{\pi_k^t \mathcal{N}(\mu_k^t, \Sigma_k^t)}{\sum_{k=1}^{K^t} \pi_k^t \mathcal{N}(\mu_k^t, \Sigma_k^t)}
   \]

   2. **Batch sufficient statistics** computation,

   \[
   s_{0,k}^t = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}_{z_i = k}
   \]

   \[
   s_{1,k}^t = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}_{z_i = k} u_i^t_{EMA}
   \]

   \[
   s_{2,k}^t = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}_{z_i = k} u_i^t_{EMA} u_i^t_{EMA}^T
   \]

3. **Global sufficient statistics** update by convex combination of parameter $\gamma_{GMM} \in [0, 1]$,

   \[
   S^t = (1 - \gamma_{GMM}) S^{t-1} + \gamma_{GMM} S^t.
   \]

4. **M-step**, compute the updated parameters from the sufficient statistics,

   \[
   \pi_k^t = \frac{S_{k,0}^t}{\sum_{k=0}^{K^t} S_{k,0}^t}
   \]

   \[
   \mu_k^t = \frac{S_{k,1}^t}{S_{k,0}^t}
   \]

   \[
   \Sigma_k^t = \frac{S_{k,2}^t - \mu_k^T \mu_k^t}{\sum_{k=0}^{K^t} S_{k,0}^t - \mu_k^T \mu_k^t}
   \]

3.5. **The choice of negative examples**

The choice of negative examples is crucial because they are the element that will define the contrast of the feature space and thus influence the detection model. Since using real anomalies is not an option in practice, an alternative is required.

**ImageNet as an outlier dataset.** Self-supervised backbones are often trained with a generic dataset like ImageNet [35]. This then suggests that such a dataset has sufficient diversity to both normal data as well as potential anomalies, thus preventing a potential overfitting. Following this analysis, we propose to use images from ImageNet to represent negative examples. Nevertheless, the contrastive task directly performed with ImageNet is not discriminative enough because of the difference in color distribution between ImageNet and the set of normal images, thus resulting in an imprecise feature space. Therefore, in order to avoid this problem, we process the negative data so as to align the two color spaces. For that, the color histogram of each negative sample is transported to the color histogram of the object using a linear color transport [28] $T$. Using a convex combination of the parameter $\rho \in [0, 1]$, it is possible to simulate an interpolation between the original image $I^{-}$ and the transported image $T(I^{-})$. Thus, each image used during the fine-tuning is randomly augmented by uniformly sampling $\rho$.

\[
I^-_\rho = \rho I^- + (1 - \rho) T(I^-)
\]

**Normal features in ImageNet.** Since we can’t exclude the possibility that some ImageNet images and normal images could be locally identical, it would be contradictory to force their representation to be able to distinguish them all. In our experiments, this casual similarity occurs especially in flatter regions of images, often corresponding to their background. Some examples are shown in Figure 4. If no additional restrictions are imposed, the network will try to differentiate a uniform background supporting an object with any uniform flat area of the same color in ImageNet. Therefore, this forces the network to focus on the fine noise/texture of the normal data background to learn its fingerprint. Thus, in general, at the end of the training process the network manages to perfectly classify positive and negative examples, but with unstable inference and hallucinated anomalies in the background.

This intuition is supported by studying the gradient of the loss, see the supplementary material. Such contradictory negative examples can generate gradient spikes and be overrepresented in the aggregated batch gradient. For example, we show in Figure 2 the detection probability map estimated using a network fine-tuned with Imagenet as negative examples and without any other restriction. This saliency map shows strong false positives in the upper right white background. To make sure that this phenomenon was not due
to a defective training, we performed several training with multiple ranges of hyperparameters to no avail. In particular, this specific image always yielded a false alarm. Therefore, this highly suggests that the network indeed detects an anomaly in the fine grain of the background texture.

**Normal features filter.** To overcome this problem we block the back-propagation for negative examples that are well within the normality model, i.e. too much inside the GMM in our case. We use the probability $P(z = -u^{-}, \Theta)$ (i.e. the negative loss) as reference. If it is above a given threshold, then the sample is used for back-propagation. However, since the target model, the GMM, is on the stable feature space of the momentum encoder $f_{\Theta_{EMA}}$, the back-propagation decision on a negative feature $u^{-}$ is made on the momentum feature space ($u^{-}_{EMA}$). In other words, we back-propagate the gradient on $u^{-}$ if $P(z = -u^{-}_{EMA}, \Theta)$ is higher than the threshold. Nonetheless, this solution requires us to compute a new probability since the probability corresponding to $u^{-}_{EMA}$ was not yet estimated, only the one corresponding to $u^{-}$ is computed for the loss during the process described in Section 3.3. This stabilizes the training since the network is no longer asked to extract all negative examples from the normality model, at the cost of only one extra probability computation.

Based on the definition of the loss, we expect that the backpropagated negative examples will pull the other examples that are close to them out of the GMM. This should lead to an extraction of most of the negative examples, leaving the ones that can’t be distinguished in. We show the evolution of the amount of filtered samples during training in Figure 3. We force the network to backpropagate on the 10% of the negative examples that are most outside the GMM, i.e. the 10% $u^{-}_{EMA}$ of higher $P(z = -u^{-}_{EMA}, \alpha, \Theta)$. This is similar to assuming that at least 10% of the negative examples in a batch are distinguishable from the object. We observe that, as expected, the network manages to extract the negative examples from the GMM, since the percentage of filtered examples converges to a value close to zero. We performed this experiment with and without color augmentation of the negative examples. With the augmentation, there are more samples that are filtered at the end of the training, which seems logical because the augmentation makes the Imagenet images look more similar to the normal data. Samples that remain filtered, for example those shown in Figure 4, are mostly logical because they have a uniform background of the same color as the background of the object.

### 4. Experiments

#### 4.1. Implementation details

We fine-tune a truncated ConvNext-V1 Pico [23] using the contrastive learning described above. The backbone is truncated after the second block and outputs a feature map downsampled by 8 with a depth of 128. A sigmoid layer, Eq (10), is added to project the features in the unit circle. This step is necessary as otherwise the problem is ill-posed: Gaussian sizes can be indefinitely large since their distance can also grow at the same speed. The backbone is initialized with weights trained for Imagenet classification, which already has a good representation space. We train a specific backbone for each object during 1000 epochs. For that, we use the AdamW [24] optimizer with a learning rate of $10^{-4}$ and a cosine decay to $10^{-6}$. The first 100 epochs are used as warm-up with a linearly increasing learning rate. We use a weight decay of 0.05 as recommended for small backbones. A batch comprises 32 positive and 32 negative examples. Momentum weights are updated with $\gamma = 10^{-3}$, which follows the same decay as the learning rate (end at $\gamma = 10^{-5}$).

For the GMM setting, we assume a balanced distribution since we chose balanced batches. Therefore, we initialize $\alpha$ to $\alpha = 0.5$. We selected $K = 256$ is a tractable number of components, as shown by Artola et al. [2] there is only less than 0.5% AUROC difference with 1024 components. The update convex combination ratio for the sufficient statistics of the GMM is set to $10^{-2}$. The component suppression threshold is set to $10^{-5}$, as this is approximately the inverse of the total number of positive features. We try to restart a component at each iteration if the actual number of components is below 256.

#### 4.2. Experimental results and ablation studies

In this section, we evaluate the detection performance of the backbone, combined with the GMM-based detection model presented in Section 3.4, before and after fine-tuning. The proposed method is evaluated on the MVtec dataset [5]. The pseudo-labeling GMM used during the fine-tuning is ini-
Figure 4. Examples of difficult negative examples. For each object, we show an augmented Imagenet example that has some parts (in black in the threshold map) that are inside the GMM. The filter map shows the parts of the Imagenet sample that are outside the GMM in white, and the parts that are inside the GMM in black. Since this is like binary classification, the filter is set so that the gradient is applied to samples that have more than a 50% chance of being outside the GMM.

Table 1. Evaluation of contrastive fine-tuning in AUROC for image level detection. We use a single-layer GLAD model, initialized with $K = 256$ Gaussians, to evaluate the quality of the pretrained Convnext-V1 feature space and our fine-tuned version. In the second case, GLAD uses the GMM estimated at the end of the fine-tuning process.

<table>
<thead>
<tr>
<th>MVtec</th>
<th>Image-level AUROC</th>
<th>Pixel-level AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pretrained fine-</td>
<td>pretrained fine-</td>
</tr>
<tr>
<td></td>
<td>tuned</td>
<td>tuned</td>
</tr>
<tr>
<td>bottle</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>cable</td>
<td>82.5 95.6</td>
<td>91.6 95.6</td>
</tr>
<tr>
<td>capsule</td>
<td>95.4 95.3</td>
<td>98.7 98.5</td>
</tr>
<tr>
<td>hazelnut</td>
<td>99.9 99.4</td>
<td>97.4 98.9</td>
</tr>
<tr>
<td>metal nut</td>
<td>97.0 99.2</td>
<td>92.2 97.8</td>
</tr>
<tr>
<td>pill</td>
<td>92.0 95.6</td>
<td>96.4 98.6</td>
</tr>
<tr>
<td>screw</td>
<td>97.8 98.3</td>
<td>99.8 99.8</td>
</tr>
<tr>
<td>toothbrush</td>
<td>95.6 98.1</td>
<td>96.6 98.2</td>
</tr>
<tr>
<td>transistor</td>
<td>88.3 95.2</td>
<td>94.1 97.1</td>
</tr>
<tr>
<td>zipper</td>
<td>98.9 99.7</td>
<td>99.4 99.5</td>
</tr>
<tr>
<td>object</td>
<td>94.7 97.6</td>
<td>96.4 98.2</td>
</tr>
<tr>
<td>carpet</td>
<td>98.6 99.0</td>
<td>99.4 99.2</td>
</tr>
<tr>
<td>grid</td>
<td>99.2 98.9</td>
<td>99.7 99.6</td>
</tr>
<tr>
<td>leather</td>
<td>100 99.9</td>
<td>99.8 99.8</td>
</tr>
<tr>
<td>tile</td>
<td>98.1 99.7</td>
<td>94.1 98.4</td>
</tr>
<tr>
<td>wood</td>
<td>98.7 99.2</td>
<td>95.4 96.6</td>
</tr>
<tr>
<td>texture</td>
<td>98.9 99.3</td>
<td>97.6 98.8</td>
</tr>
<tr>
<td>All</td>
<td>96.1 98.2</td>
<td>96.8 98.4</td>
</tr>
</tbody>
</table>

Table 1. Evaluation of contrastive fine-tuning in AUROC for image level detection. We use a single-layer GLAD model, initialized with $K = 256$ Gaussians, to evaluate the quality of the pretrained Convnext-V1 feature space and our fine-tuned version. In the second case, GLAD uses the GMM estimated at the end of the fine-tuning process.

Figure 5. Evolution of the training loss and the log likelihood of the GMM during fine-tuning.

Losses are more adequate for the problem of anomaly detection. In particular, most of the improvement is on the objects, textures only improve by 0.4%. The most notable improvements are on cable and transistor, two difficult objects with high normal variability.

We show in Figure 6 the difference of pseudo-labeling (i.e. the probability of being in the uniform region) before and after fine-tuning for a few MVtec examples. Note again that no real anomalies have been seen during training. Before training, the uniform probability is lower than the GMM probability. The fine-tuning allows us to pull the anomalous samples out of the Gaussians representing the normal model, thus making it easier to identify them in the probability maps. This also has the added benefit of improving the localization of the detections. Some false alarms, such as printed characters on the capsule, are also better handled after training.
Figure 6. Probability of being an outlier $P(z = -|u, \alpha, \Theta)$ of MVTec anomalies, before and after training. No anomalies have been seen during the fine-tuning process. The first columns of each object show the object with its anomalies. The second columns are the probability of being a negative example before fine-tuning the network and the thirds are after fine-tuning the network. Yellow shows a probability of 1 and dark purple represents probability lower than $10^{-100}$.

We show the evolution of the training loss and the likelihood during fine-tuning in Figure 5. The loss improves most at the beginning of the training: negative examples are extracted from the GMM at this time, as we can see in Figure 3. However, this effect leads to a decrease in the likelihood. During this time, the Gaussian volume (eigenvalues) increases, probably because the positive examples are pulled out by the movement of the negative examples. This seems logical since we have seen that at the beginning of the training, negative examples are predominant in the aggregated gradient. Then, after about 100 epochs, the likelihood starts to improve again. This is likely because negative examples are far enough from the GMM so that both positive and negative examples have the same impact on the gradient and Gaussians can start to shrink again. Remark that around epoch 400 the loss rises and the likelihood goes down. This kind of event is due to the encounter of challenging negative examples.

Additional experimental results showing the distribution of wrong classification probability of positive and negative samples is shown in the supplementary material.

5. Conclusion and future work

Following the concepts of contrastive learning, we propose in this paper a method to fine-tune a pre-trained neural network to improve its performance on visual anomaly for a specific given object. Indeed, while generic plug-and-play backbones, pre-trained using regular contrastive learning, already offer good performance, they are not optimal when trying to represent a unique object for a specific task. The proposed fine-tuning allows us to improve the contrast of the neural network feature space between anomalies and normality and thus improve detection and localization of anomalies.

The proposed method is generic and should work with all detection frameworks based on feature model of pre-trained backbones, such as Padim [13] or GLAD [2], as long as the model is differentiable. Nevertheless, we plan on investigating more architectures, and especially transformers. Thanks to their positional encoding layer, their features may provide localization information that can be exploited for anomaly detection.

Another important research direction is the choice of negative examples. In this work, we simply chose natural images coming from ImageNet as negative examples, but another interesting option would be to generate artificial anomalies. For example, DRAEM [48] proposed to use synthetic generated anomalies to train their encoder-decoder segmentation network. These anomalies are generated using a convex combination of a normal image (from the set of anomaly free images) and an augmented texture image. The idea in that case would be to apply the contrastive fine-tuning between these anomalies and normal images. This might improve even more the specialization of the backbone.
References


[23] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the