Anomaly detection project

Teaching Assistant: Francesco Cappio Borlino

email: francesco.cappio@polito.it

reference paper: PatchCore

TASK OVERVIEW

The goal of this project is to delve into the anomaly detection task focusing on the industrial setting and in particular the “cold-start” problem. This task requires to build a model that is able to automatically detect anomalies in industrial products, despite having been trained on normal samples only.

The main difficulty of this task is the fact that anomaly types can range from subtle detail changes, such as small scratches, to large structural defects, such as missing components. This large variety of anomalies makes it extremely hard to define in advance and a priori exactly which kind of problems industrial products could show and thus it makes particularly difficult and costly to collect datasets of anomalous samples. As a result the standard approach (*cold-start* anomaly detection) consists in collecting datasets of nominal (*normal*) samples only, and cast the problem as an out-of-distribution detection task: a model is trained on normal data only and should be able to detect at test time all the samples that do not belong to the training data distribution.

Reference paper

This project focuses on the paper: “Towards Total Recall in Industrial Anomaly Detection” [1] by Roth et al. This paper proposes a new approach for the cold-start anomaly detection problem called “**PatchCore**”. This method belongs to a category of approaches which leverage the highly general representations that can be extracted by a standard network trained for classification on ImageNet in order to perform feature matching between nominal and test samples. The hierarchical features structure of deep networks is used to perform comparison at different levels in order to spot both small and large anomalies.

Contributions

PatchCore aims at overcoming some of the problems of the previous state of the art approaches in order to outperform them. Main contributions:

1. **Use of mid-level features**. While using features at various network layers is generally useful in order to detect both small and large anomalies, the use of representations extracted from a network trained for a different task (ImageNet classification) may cause harm if the layers' choice is not carefully considered. Indeed when going from the first layers to the deepest ones the extracted features become more and more biased towards the task on which the model has been trained. In our case this means that the deepest features (e.g. those of the last backbone layer) are certainly perfectly suited to represent objects belonging to ImageNet's classes, but much less suited to perform features matching for industrial anomaly detection. For this reason PatchCore discards high-level semantic features from the pretrained backbone and exploits only mid-level ones. The expressiveness of these features is improved by local context and multi-level aggregation. This strategy allows to extract for each image a set of features each one representing a specific image patch.
2. **Coreset-reduced patch-feature memory bank**. At inference time in order to define if a test sample is anomalous it is necessary to compare its patch-features with the patch features of nominal samples. Of course if the training dataset is particularly large, keeping in memory patch-features for all of the nominal images may be exceedingly costly, making also the comparison of test patch-features with nominal ones extremely computationally expensive. It is thus of paramount importance to reduce the size of the set of nominal patch-features which should be kept in memory. In order to obtain an effective reduction without hurting the performance, the nominal feature coverage encoded in the memory bank should be retained, thus a random subsampling is not possible. PatchCore uses therefore a *coreset* reduction strategy exploiting an iterative greedy algorithm to perform selection of patch-features maximizing the coverage of the original set.

Anomaly detection with PatchCore

Once a coreset memory bank of nominal patch-features has been defined it is possible to exploit it to perform anomaly detection and segmentation, by comparing the patch-features of each test sample with the nominal ones.

Anomaly detection

An image-level anomaly score should be defined for each test sample. This is done by finding the nearest nominal neighbor for each test sample patch-feature and then computing the maximum distance between a test patch-feature and the corresponding nominal nearest neighbor. This strategy allows one to easily find out which test patch-feature is responsible for guiding the decision about the normality of the test sample itself. Indeed for a test sample to be anomalous it is enough for it to have a single anomalous patch.

Anomaly segmentation

In many cases finding out if a test sample is anomalous or not is not enough as the anomaly should also be localized and segmented. This can be easily done with PatchCore, obtaining a segmentation map by simply computing the normality score for each test patch-feature and realigning them with their respective spatial location.

PROJECT GOALS

The initial goal of this project is to give you the opportunity to get familiar with the anomaly detection task, in particular considering the cold-start problem and industrial setting. To reach this result you should thoroughly study the reference paper [1] and any other paper that you may find among its references and that could be useful to get a better insight on the task.

First part: PatchCore reimplementation

You may have noticed that the PatchCore method by itself does not include a training phase. Indeed it is an approach that is designed to exploit a pretrained network in order to perform industrial anomaly detection by performing feature-matching between nominal and test data. As a result the first part of the project is pure code writing and does not involve performing trainings.

You should reimplement the PatchCore code from scratch and arrive at replicating the paper results. The minimum requirement for this part involves implementing the anomaly detection procedure without considering the anomaly segmentation one. Of course implementing also the anomaly segmentation part may lead to obtaining a better mark.

It should be noticed that the use of a GPU may still be necessary even in this phase that does not involve training a deep model: indeed also the inference phase benefits from performing computations on a GPU.

Second part: PatchCore extension

An extension of the PatchCore method could lead to even better anomaly detection performance. In particular it would be particularly interesting to check if different or more tailored feature representations could lead to results improvements.

Thus there are mainly two possible extension approaches, you should choose one, implement it and compare its results with the ones you obtained in the first part of the project.

Extension alternative 1: a different pretraining

ImageNet classification pretraining has been a standard solution to obtain generalizable features to leverage for downstream tasks (directly or via finetuning) for a long time. Still, in the last few years some alternatives have been proposed to obtain even stronger and more generalizable representations. One of these alternative solutions is represented by CLIP [2]. The project's extension in this case consists in exploiting the pretrained Image Encoder of CLIP instead of the ImageNet pretrained one to see if it enables obtaining better AD performance with PatchCore.

Extension alternative 2: network finetuning

One of the main problems of using a network pretrained on ImageNet classification to perform industrial anomaly detection is the fact that when going towards deepest network layers the features become more and more object-centrics and biased towards ImageNet classes. Of course PatchCore excellent results show that a careful selection of network layers still allows to obtain representations which are good enough to perform effective feature-matching for anomaly detection.

Still, a finetuning of the network on the images of the specific task at hand could help in reducing the bias towards ImageNet objects features and push the approach to obtain better results. At the same time the finetuning strategy should be carefully designed in order to:

- keep the strong expressivity of the features;
- avoid pushing the model towards a different kind of bias which could hurt the performance.

Your task is therefore to design and implement a finetuning strategy to improve the network features. Your finetuning should use the nominal samples of the anomaly detection task at hand.

A possible solution involves finetuning the pretrained ImageNet network using a self-supervised task on the nominal data. Examples:

- (easy) **rotation recognition of image samples**. The rotation recognition task [3] is one of the easiest to implement self-supervised tasks and has been successfully used in a number of research works. A naive finetuning of the network via rotation recognition could make it better suited to extract good representations for the task at hand.
- (hard) **contrastive learning with synthetic anomalies obtained via CutMix**. Contrastive learning [4] has recently emerged as one of the most effective strategies for self-supervised learning. Like in the case above a simple contrastive learning approach could be used to finetune the network on the nominal samples of the task at hand. Contrastive learning however could also be used to design a specific finetuning strategy designed for the anomaly detection task. In particular synthetic anomalous samples could be easily obtained by applying CutMix [5] on the nominal samples and then the resulting

anomalous samples could be used as additional negative samples in the contrastive learning procedure.

SOME DETAILS

- Experiments will focus on the MVTec Anomaly Detection dataset [6]. This is a dataset designed specifically to benchmark industrial anomaly detection approaches and it can be freely downloaded from the project website. It is not necessary to perform experiments on other datasets for the project. Moreover, for simplicity in this work you can exploit the test set as validation set in order to find out the best values for hyperparameters and to compare methods.
- an official implementation of the reference paper is available online. Of course copying the code directly is discouraged, but you can look at it to search for inspiration on how to implement certain parts. Please be aware that you will have to provide the code of all your experiments and that during the oral exam I will expect you to know exactly how you implemented the chosen strategies, so I will easily notice if you have copied the code without even trying to understand it.

PROJECT REQUIREMENTS

Main requirements:

- you will need to **provide the code for every experiment you perform**. The code should be able to **replicate your results**. Keep your code clean and as simple as possible. Use launch params or configuration files in order to provide the values for the hyperparameters;
- when you implement a method that needs hyperparameters you should validate them by trying different values;
- besides the standard PatchCore, you should implement one of the extension alternatives and compare its performance with the original PatchCore;
- write a complete pdf report. The report must contain: a brief introduction, a related works section, a methodological section for describing the algorithms you are going to use, an experimental section with all the results and discussion. End the report with a brief conclusion.

References

[1] Roth et al., “Towards Total Recall in Industrial Anomaly Detection”, in CVPR 2022

[2] Radford et al., “Learning Transferable Visual Models From Natural Language Supervision”. In Arxiv, 2021.

[3] Gidaris et al., “Unsupervised Representation Learning by Predicting Image Rotations”, In ICLR 2018

[4] Chen et al., “A Simple Framework for Contrastive Learning of Visual Representations”, In ICML 2020

[5] Sangdoo et al., “CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features”, In ICCV 2019

[6] Bergman et al., “MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection”, in CVPR 2019