CS230

SimCLR framework for defect detection on unknown context

Divyaj Shah
Stanford Center for Professional Development
Stanford University
divyaj28@stanford.edu

Po-Ting Lin
Department of Materials Science and Engineering
Stanford University
ptlin84@stanford.edu

Abstract

Defect detection is important in quality controls in manufacturing. However, the current technologies are limited in a way that they can only detect defects that are known and defined. This work aims to provide an innovative way of defect detection with SimCLR framework that can detect defects on unknown context, i.e., detect unseen types of defects in different manufacturing objects. The SimCLR model was first pretrained with different types of objects. After pretraining, two dense layers are appended to the pretrained encoder for downstream binary classification task. The results show good promise in replacing the VGG16 and ResNet50 models with SimCLR object detection task.

1. Introduction

Defect detection is an important process for quality controls in manufacturing. As a result, there exist different methods, such as contact measurement, computer vision, and 3D laser measurement, for defect detection in production lines. However, without the knowledge of specifications of a good object, it becomes extremely difficult for machine to define and detect defects from objects it has not seen before. The ability to detect defects without previous training on all possible defects can be very useful if we want to achieve defect detection in all kinds of circumstances outside of a production line.

To address this problem, we propose a contrastive learning approach for defect detection in manufacturing objects. The input to our algorithm is an image and a label indicating the object in the image is "good" or has "anomaly". The output is prediction on the image about whether it contains defects or not. The outcome will be a model which will be able to identify defects on backgrounds and context which are unseen, i.e., backgrounds and context that are different from what the model is trained on.

Most of the contrastive learning-based models do well on image classification where the object occupies more than 75% of the image. There is very limited research data available on detecting defects in objects which normally occupy < 5% of the image size. We aim on achieving this by making the model learn to differentiate breaks in pattern at a low-resolution level. Our motivation behind achieving this is to easily identify if a particular image has a defect or is defect free, which enables various quality-inspection applications outside of a factory.

2. Related Work

Given the importance of defect detection in manufacturing, a decent amount of previous work on improving defect detection with deep learning models can be found. Chen et al. reviewed different defect detection methods for industrial products [1]. They discussed different methods, including traditional machine vision and deep learning-based technologies. They also pointed out the unbalanced sample identification problem

typically in training deep learning models with manufacturing objects data, which causes the model to pay more attention to the majority of data that are defect-free.

Rudolph et al. proposed a novel fully convolutional cross-scale normalizing flow that jointly processes multiple feature maps of different scales [3]. They also presented DifferNet by utilizing a normalizing-flow-based density estimation of image features at multiple scales.

image features at multiple scales

3. Dataset and Features

The dataset we are using is the MVTEC AD data set consisting of multiple images of various object types (Paul Bergmann1, 2021). This set includes 5354 high resolution images which include 15 different types of industrial objects (such as bottle, capsules, grid, carpet and so on). Each object type has multiple and varying defect free images. In addition, there are sets of images for each object type, with various kinds of minute defect (70 different types in total). These defects include contaminations, scrapes, cavities, and other structural changes.

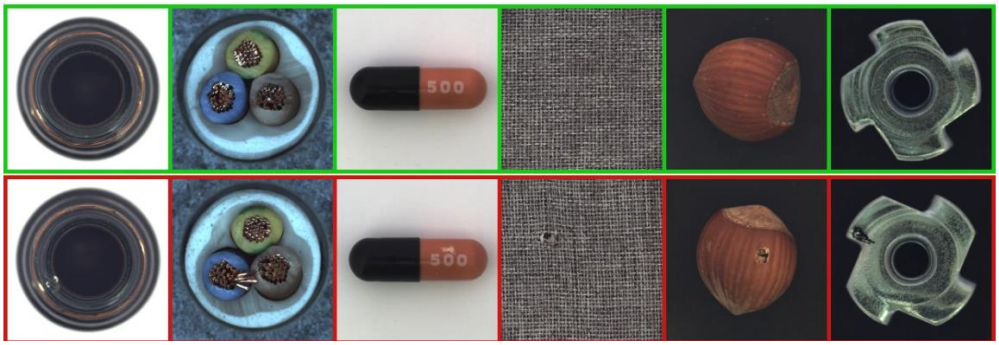

Figure 1 - Images from the MVTEC AD Dataset

Most images in the dataset are of size 1024×1024. However, to reduce the amount of computation and speed up iteration without sacrificing model accuracy, we resized the images to 224×224 as a data preprocessing step. There are in total 5274 labeled images from the dataset, with 74% of which being labeled “good” (without defects).

4. Methods

Baseline Model

The baseline model is built on the work of Olga Chernytska and Olha Chernytska and their GitHub repository. For this baseline model, we used a modified VGG-16 model, in which the first four sets of “convolutional + pooling” layers are pre-trained on ImageNet. After the frozen layers, there are three convolutional layers, one pooling layer, one global average pooling layer, and one fully connected layes. Only the last 3 convolutional layers and a dense layer are finetuned with the MVTEC dataset. The loss function is Cross-Entropy; optimizer is Adam with a learning rate of 0.0001.

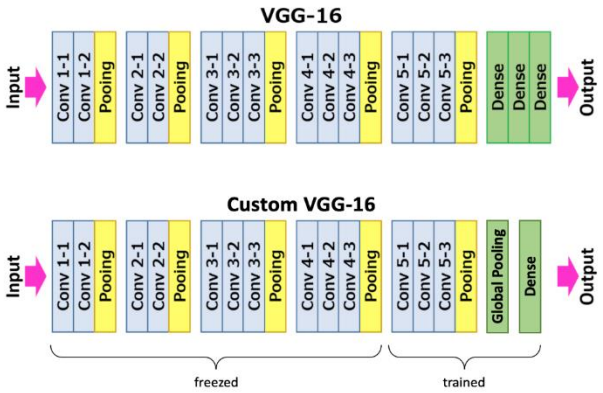

Figure 2 - Modified VGG-16 model architecture (image by Olga Chernytska and Olha Chernytska)

SimCLR Model

The framework of the SimCLR model is shown in Fig. 3,4. The method we used contains two parts, the pretraining with contrastive learning and the binary classification for the downstream defect-detection task.

The pretraining takes an image as input, transforms it into two different images, passes it through a CNN encoder and a projection head with two dense layers, and minimize the contrastive loss $l_{i,j}$:

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(z_i, z_k)/\tau)}$$

The SimCLR pretraining trains the parameters in the encoder and the projection head by maximizing similarity between two transformed images from one same original image and minimizing similarity between two transformed images from two different original images. After this pretraining of the SimCLR model, the downstream task of defect detection in images is simply binary prediction to predict that either the input image contains defect or not. The encoder along with the pretrained weights are connected to three dense layers with a binary output at the end.

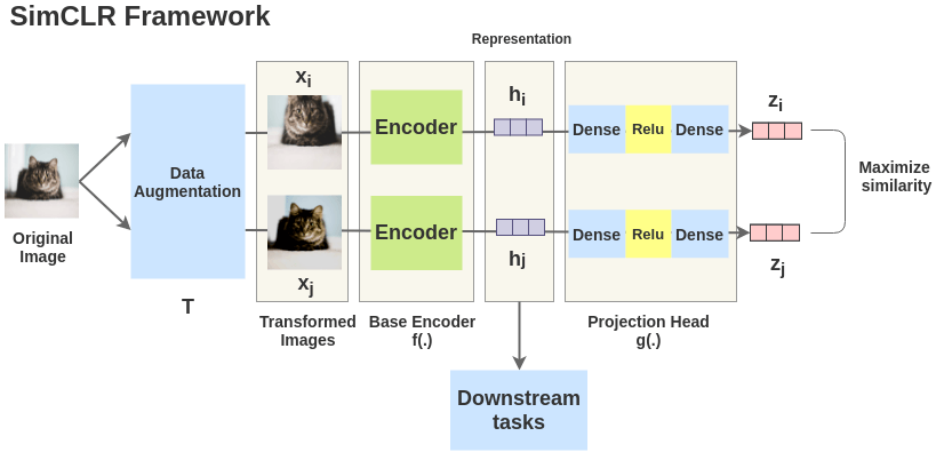

Figure 3 – Framework of SimCLR model training

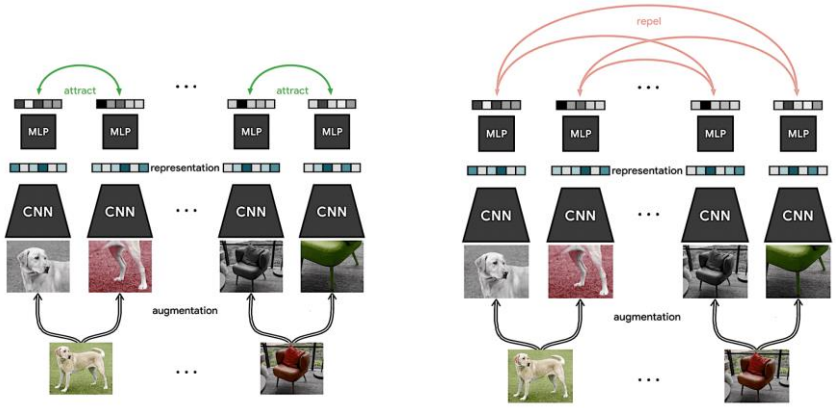

Figure 4 – Illustration of SimCLR framework

5. Experiments/Results/Discussion

Baseline Model

Ultimately, we want our model with SimCLR to be able to detect defects on unknown context. As a result, we trained our custom VGG-16 on 3 types of objects, which are Capsule (N=351 images), Hazelnut (N=501 images), and Leather (N=369 images). But for testing, the datasets of another 3 types of objects were used, which were Bottle (N=262 images), Pill (N=390 images), and Wood (N=293 images). The results show precision of 89.3%, recall of 63.4%, and **F1 score of 0.74**. **The test set accuracy is 66.6%**, and the **test set balanced accuracy is 70.2%.** This result suggests that the model is doing ok and a little better than guessing. Based on our findings for the baseline model, the accuracy is significantly affected by text engravings, unbalanced data distribution and insufficient context during training

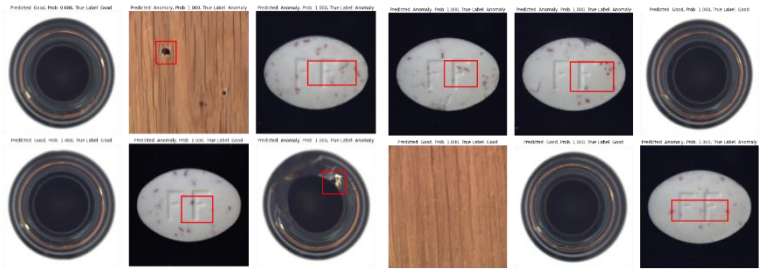

Figure 4 - Bottle, Pill, and Wood: Prediction on Test Set

SimCLR Model

As a part of the object detection problem, we have used the SimCLR model to obtain the encodings and subsequently learn the image classification and object detection task. The initial idea was to modify the SimCLR logic by adding a cut-paste augmentation and increasing the dissimilarity between the original image and augmented image but the idea was discarded due to potential wrong learning. Next, we decided to train the SimCLR on the MVTec-AD dataset for image classification problem and use the encoder for downstream task such as object detection. The image classification was trained on multiple number of objects. **For 3 objects the validation accuracy was 86.5% and for all 15 classes it was 89.6%. We even tested the model for different batch sizes. For batch size 512, the accuracy went as high as upto 98.6%**

6. Challenges Faced

In this section, we'd like to list down the challenges we faced during the duration of our project.

1. The original SimCLR paper by the authors was created in Tensorflow v1 and many functions were deprecated in Tensorflow v2. With limited understanding and the high complexity of Tensorflow, it was difficult to get the code running
2. To load the custom dataset into Tensorflow according to our own rules of labeling is not very straight forward as compared to PyTorch. Significant amount of time was spent to figure this out. In the end, we could successfully do it by creating our own class of custom dataset from scratch.
3. To compare our dataset with the baseline (which was developed in PyTorch), we needed to convert Tensorflow model to PyTorch. To do this we first converted Tensorflow to Onnx and from Onnx to PyTorch, with some existing issues still remaining to be debugged.

7. Conclusion/Future Work

The SimCLR model for image classification is a very promising method to train and get accurate encoding of your data, especially with small dataset like ours (~5K images). Once fully trained, this pre-trained model can be very useful in many downstream tasks such as object detection (which was the attempt of this paper). **SimCLR can achieve >85% accuracy** easily with a smaller number of classes to train from and by using large batch sizes, as compared to the traditional VGG which can only provide accuracy upto 70%. Although the final comparison could not be obtained with baseline due to many complexities in the code, an attempt to get good results will continue beyond the course requirements.

Future work

1. Integrating the Tensorflow model into PyTorch baseline model with added linear heads to check the outcome of the classification as compared to VGG16.
2. Improve the projection head structure to be easily able to identify between good and anomaly using the generated heat map.
3. There is also good scope to modify the loss function for SimCLR to suit individual application needs but more research and discussion needs to be done in this regard.

8. Contributions

1. Po-Ting has worked on implementing the baseline model using CNN on a pre-trained VGG16 model. He has collected baseline data on the actual representation we expect to run our final model on. He also worked on creating the binary classifier for the downstream task for the final results and debugging Tesnorflow issues.
2. Divyaj has worked on implementing the basic SimCLR model that can run on our custom dataset. He implemented a custom Tensorflow dataset class from scratch. He also implemented a Cut-Paste image augmentation technique (although it wasn't used). He worked on integrating the Tensorflow model to PyTorch model and debugging many Tensorflow issues. So far, the model works for image classification and does an ok job in classifying images as good or anomaly.
3. **Git-hub for Baseline CNN model**: CNN Baseline for Defect Detection
4. **Final Github reop for project:** SimCLR Defect Detection

References

[1] Chen, Y., Ding, Y., Zhao, F., Zhang, E., Wu, Z., & Shao, L. (2021). **Surface defect detection methods for industrial products: A review**. Applied Sciences, 11(16), 7657.

[2] Lin, D., Li, Y., Prasad, S., Nwe, T. L., Dong, S., & Oo, Z. M. (2021). **CAM-guided Multi-Path Decoding U-Net with Triplet Feature Regularization for defect detection and segmentation**. Knowledge-Based Systems, 228, 107272.

[3] Schlegl, T., Seeböck, P., Waldstein, S. M., Langs, G., & Schmidt-Erfurth, U. (2019). **f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks**. Medical image analysis, 54, 30-44.

[4] Andrews, J., Tanay, T., Morton, E. J., & Griffin, L. D. (2016). **Transfer representation-learning for anomaly detection**. JMLR.

[5] Rudolph, M., Wehrbein, T., Rosenhahn, B., & Wandt, B. (2022). **Fully convolutional cross-scale-flows for image-based defect detection**. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 1088-1097).

[6] Rudolph, M., Wandt, B., & Rosenhahn, B. (2021). **Same same but differnet: Semi-supervised defect detection with normalizing flows.** In Proceedings of the IEEE/CVF winter conference on applications of computer vision (pp. 1907-1916).

[7] **A Simple Framework for Contrastive Learning of Visual Representations** [Conference] / auth. Ting Chen Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton // International Conference on Machine Learning. - Vienna : PMLR, 2020.

[8] **A Unifying Review of Deep and Shallow Anomaly Detection** [Journal] / auth. Lukas Ruff Jacob R. Kauffmann, Robert A. Vandermeulen, Gregoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, Klaus-Robert Muller. - [s.l.] : IEEE, 2021. - 11732 : Vol. 3.

[9] **CutPaste: Self-Supervised Learning for Anomaly Detection and Localization** [Journal] / auth. Chun-Liang Li Kihyuk Sohn, Jinsung Yoon, Tomas Pfister // Google Cloud AI Research. - 2021. - pp. 1-28.

[10] **MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection** [Journal] / auth. Paul Bergmann Michael Fauser, David Sattlegger, Carsten Steger // MVTec Software GmbH. - 2021. - pp. 1-9.

[11] **Review — SimCLR: A Simple Framework for Contrastive Learning of Visual Representations** [Online] / auth. Tsang Sik-Ho. - March 1, 2022. - https://sh-tsang.medium.com/review-simclr-a-simple-framework-for-contrastive-learning-of-visual-representations-5de42ba0bc66.

[12] **Self-supervised learning tutorial: Implementing SimCLR with pytorch lightning** [Online] / auth. Adaloglou Nikolas. - March 31, 2022. - https://theaisummer.com/simclr/.

[13] **The Illustrated SimCLR Framework** [Online] / auth. Chaudhary Amit. - March 4, 2020. - https://amitness.com/2020/03/illustrated-simclr/.

[14] **The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection** [Journal] / auth. Paul Bergmann1 2 · Kilian Batzner1 · Michael Fauser1 · David Sattlegger1 · Carsten Steger1 // International Journal of Computer Vision . - 2021. - pp. 129; 1038-1059.