Improving Industrial Quality Control with Computer Vision

Génération de données en IA par transport et débruitage Challenge

Mouad ID SOUGOU
Master MVA
mouadidsougou@gmail.com

Abstract

This report aims to describe the overall pipeline used to classify images in a manufacturing quality control challenge. We begin by reviewing the dataset, then we detail the model architecture and the decisions made during the training process. Finally, we present the performance results of our model and discuss its strengths and limitations.

1. Overview & Dataset

1.1. Introduction

Valeo, a leading global automotive supplier, aims to enhance industrial performance through advanced production systems. In the context of production lines, traditional automatic inspection systems are used to assess whether a part is *good* or *defective*. However, while these systems can handle common defects effectively, there is a challenge when it comes to rare or previously unseen defects.

To address this, Valeo uses computer vision (CV) models to automatically sort parts rejected by Automated Optical Inspection (AOI) machines. Given that training cannot cover all cases, it is crucial to detect images that do not match any trained categories, especially when part appearance or process changes cause drift.

The implementation code is available here [1]

1.2. Problem Statement

In this context, the main objective of this challenge is to classify images of manufactured parts into predefined categories: However, the challenge becomes more complex due to the presence of rare defects that are not represented in the training data. The problem is twofold:

- **Classification:** The model needs to identify images as either belonging to the *good* class or one of the defect classes.
- **Anomaly Detection:** The model must also detect images that do not belong to any of the predefined classes, which could indicate defects that were not part of the training dataset.

[1] https://github.com/midsougou/Improving-Industrial-Quality-Control

1.3. Dataset Description

The training dataset contains a total of 8,278 images, divided into **six** classes. The distribution of images across the classes is as follows:

- **GOOD (Label 0)**: 1235 images
- **Defect1 (Label 1)**: 71 images
- **Defect2 (Label 2)**: 270 images
- **Defect3 (Label 3)**: 104 images
- **Defect4 (Label 4)**: 6472 images
- **Defect5 (Label 5)**: 126 images

Figure 1. Example of a train sample.

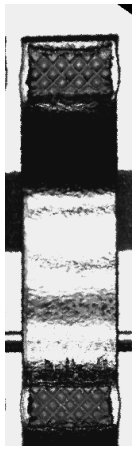

Figure 2. Example of a train sample.

The test dataset consists of 1,055 images, divided into seven classes:

- **GOOD (Label 0)**: Images of good parts.
- **Defects (Labels 1-5)**: The same defect classes as in the training set.
- **Drift (Label 6)**: A class containing images that do not belong to any of the known defect or good categories. This class includes anomalies in image capture, such as blurry

images, or rare defects that are grouped together to form this anomalous class.

Both the training and test datasets contain additional metadata in the form of two columns: `window` and `lib`:

- **window**: Represents one of two zones for inspecting the component in the image.
- **lib**: Indicates one of four different components present on the parts as shown in Figure 3, with variations in their orientation and size relative to the camera.

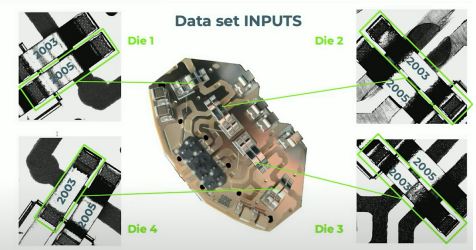

Figure 3. Dataset inputs

1.4. Evaluation Metric

The evaluation metric used for this challenge incorporates a weighted accuracy approach. A penalty is applied for misclassifying defective parts as *good*. Additionally, the metric strongly penalizes the failure to detect a drift, which refers to images that do not belong to any of the known classes. This ensures that the model not only classifies parts accurately but also detects anomalies effectively.

1.5. Data Augmentation

To enhance the model's generalization and prevent overfitting, we applied several data augmentation techniques using PyTorch's `transforms` module. Images were first resized to 256×256 pixels, ensuring uniform input dimensions. To introduce variability, we applied random horizontal flipping with a 50% probability, random rotation within ±15° to account for angular variations, and random affine transformations with slight translations to simulate distortions. Additionally, we applied color jittering to introduce variations in brightness, contrast, saturation, and hue, making the model more robust to different lighting conditions.

2. General Observations

2.1. Addressing Imbalanced Datasets

When dealing with an imbalanced dataset, two primary solutions can be considered: generating similar samples or using a cost-sensitive loss function.

2.1.1 Generating Similar Samples

One approach to handle imbalanced datasets is to generate similar samples to augment the minority class. This can be achieved using techniques such as Generative Adversarial Networks (GANs) or other image generation methods. For instance, the SGBGAN model has been shown to effectively generate high-quality minority class images, overcoming the class imbalance restriction [7].

2.1.2 Cost-Sensitive Loss Function

Another approach is to use a cost-sensitive loss function, which assigns different misclassification costs to each class. This method modifies the learning algorithm's optimization function to minimize the overall cost of misclassification instead of the overall error rate. By strongly penalizing mistakes on the minority class, cost-sensitive learning improves their importance during the classifier training step [2].

For instance, it can be defined as :

$$L(\theta) = -\sum_{j=0}^{1} w_j \sum_{y_i=j} f(P(y_i|x_i,\theta))$$

where w_j is the weight for class j, y_i is the true label for the i-th sample, x_i is the feature vector for the i-th sample, and $f(P(y_i|x_i,\theta))$ is a function of the predicted probability for the true class y_i.

In scikit-learn, the weights w_j are calculated as follows:

$$w_j = \frac{N}{C \cdot |y_i = j|_{i=1\ldots N}}$$

where N is the total number of samples, C is the number of classes and $|y_i = j|_{i=1\ldots N}$ is the number of samples belonging to class j.

2.2. Model Selection for Image Datasets

When dealing with image datasets, several possibilities can be considered:

1. **Convolutional Neural Networks (CNNs)**: CNNs are particularly well-suited for image data due to their ability to capture local features through convolutional layers. They have been shown to perform well on small datasets [5].
2. **Vision Transformers (ViTs)**: ViTs have achieved state-of-the-art results in image classification tasks, particularly when trained on large datasets. They capture global relationships in images through self-attention mechanisms, making them a powerful alternative to CNNs [10].
3. **Hybrid Models**: Combining CNNs and ViTs can leverage the strengths of both architectures. Hybrid models can capture both local and global features, potentially improving performance on complex image classification tasks [3].
6. **Ensemble Methods**: Combining predictions from multiple models can improve the overall performance. Ensemble methods can include averaging the outputs of several CNNs or ViTs, or using stacking to combine different types of models [4].

2.3. Transfer Learning for Small Datasets

Given the limited amount of data, transfer learning is a valuable approach. Transfer learning involves using a pre-trained model on a large dataset and fine-tuning it on the smaller, target dataset. This method allows the model to **leverage the learned features** from the large dataset, improving performance on the smaller dataset [1].

2.3.1 Fine-Tuning Pre-trained Models

Fine-tuning a pre-trained model involves adjusting the weights of the model to better fit the target dataset. This process can significantly improve the model's performance, especially when the target dataset is small. By initializing the model with pre-trained weights and then fine-tuning it, the model can achieve high accuracy with less data [1].

2.4. Anomaly Detection for Unseen Classes

In some cases, the test set may contain instances of an unseen class (also known as a drift class) that was not present in the training set. Traditional classification models struggle in such scenarios, as they are trained only on known categories.

One effective approach is the use of **Isolation Forests**, an unsupervised anomaly detection method [8]. Isolation Forests operate by recursively partitioning the feature space, isolating observations based on how quickly they become outliers. Since drift classes are inherently different from known categories, their samples tend to be isolated in fewer steps, making them easier to detect.

However, applying Isolation Forests directly to raw image data can be challenging due to the high dimensionality and complex structure of images. To mitigate this, feature extraction techniques are commonly employed to transform images into a more manageable representation before applying Isolation Forests. For instance, utilizing pre-trained convolutional neural networks (CNNs) to extract feature embeddings has proven effective. These embeddings capture essential characteristics of images, reducing dimensionality while preserving relevant information for anomaly detection [9].

Another approach involves using a **threshold-based rule** on the model's confidence scores. If the highest softmax probability of a prediction falls below a predefined threshold, the sample can be classified as an anomaly. This method assumes that the model will assign lower confidence scores to instances from unseen classes compared to known categories.

3. Model Architecture

We used a pretrained ResNet-18 model [11] for this project because it is well-suited to our dataset, which shares similarities with ImageNet, and it helps save time and computational resources. To adapt the model, we replaced the final fully connected layer with a new one that matches the number of classes in our dataset. Specifically, we froze all layers of the pretrained model except for the final classification head and the last residual block (`layer4`), allowing them to be fine-tuned to better capture dataset-specific features. This approach leverages the general features learned by earlier layers while refining high-level representations to improve classification performance.

We used the Adam optimizer [6] with an initial learning rate of 0.001 and applied weight decay of 1.10^{-4} to prevent overfitting. To handle the class imbalance, we used a weighted cross-entropy loss function, where class-specific weights were computed and assigned to penalize misclassification of underrepresented classes.

During training, only the last residual block (`layer4`) and the classification head were updated, while the rest of the model remained frozen. The model was trained for 10 epochs, with early stopping implemented to halt training if validation loss did not improve for 5 consecutive epochs. At each epoch, we tracked both training and validation losses, along with validation accuracy. The best-performing model, determined by the lowest validation loss, was saved to prevent overfitting and ensure optimal generalization.

4. Results and conclusion

After training for 10 epochs, our model reached a validation accuracy of 98% (The labeled set was split into a separate training and validation set) . This outcome shows that using a pretrained ResNet-18 model, along with data augmentation was effective for our task.

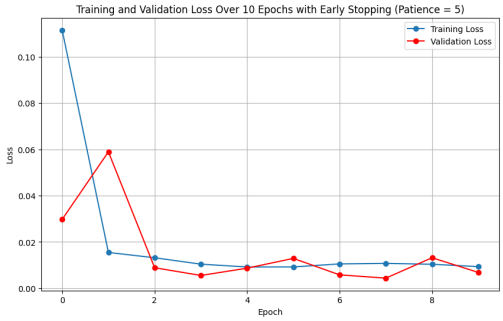

Figure 4. Training and validation loss over epochs.

As of the test score, we used a **thresholding rule of 0.5** on the output probabilities. We were ranked **3th** on the academic public leaderboard.

While we're pleased with these results, we believe there's potential for even better performance. Try modeling maybe by hand other parts of the neural network could enhance the model's accuracy further.

References

[1] Transfer learning for small and different datasets: Fine-tuning a pre-trained model affects performance. *Emerging Investigators*, 2020. 3

[2] Cost-sensitive learning for imbalanced datasets. *Reproducible Machine Learning for Credit Card Fraud detection - Practical handbook*, 2023. 2

[3] Cnn or vit? revisiting vision transformers through the lens of convolution. *ResearchGate*, 2023. 2

[4] What is the difference between ensemble methods and hybrid methods, or is there none? *Data Science Stack Exchange*, 2024. 2

[5] Hassaan Idrees. Vision transformer vs. cnn: A comparison of two image processing giants. *Medium*, 2024. 2

[6] P. Kingma and J. Ba. Adam. A method for stochastic optimization, 2014. 3

[7] Z. Li, Y. Jin, Y. Li, Z. Lin, and S. Wang. Sgbgan: minority class image generation for class-imbalanced datasets. *Machine Vision and Applications*, 2023. 2

[8] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. *Proceedings of the Eighth IEEE International Conference on Data Mining*, 2008. 3

[9] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep learning for anomaly detection: A review. *arXiv preprint arXiv:2007.02500*, 2019. 3

[10] Fahim Rustamy. Vision transformers vs. convolutional neural networks. *Medium*, 2024. 2

[11] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. Deep residual learning for image recognition, 2015. 3