# EDM Machined Surfaces: Visual Defect Detection on Whole Images Final Report

Fabrizio Patuzzo, IDSIA-SUPSI

September 2019

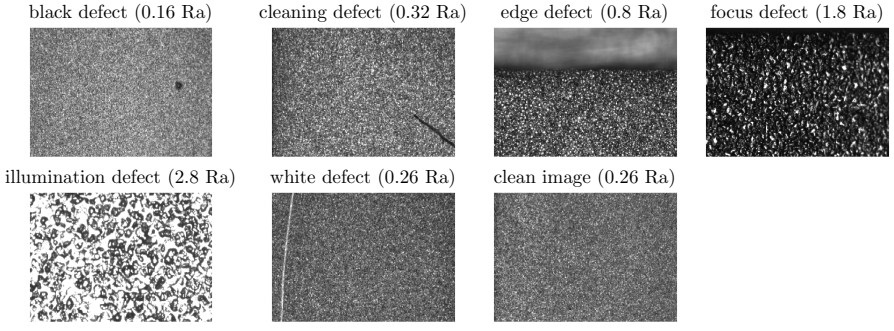
## Introduction

The aim of this project was to detect anomalies on EDM-Machined metallic surfaces, using machine learning techniques at the level of whole images: an entire image has to be classified as either good or defected. This approach is orthogonal to the main defect detection approach implemented by IDSIA-SUPSI, which aims to solve a segmentation problem, i.e. aims at detecting and outlining defects in images.

This project was developed at the Dalle Molle Institute of Artificial Intelligence (IDSIA) in Manno, in collaboration with MEMTI-SUPSI and Georg Fischer SA.

## 1 The dataset

Our dataset comprises 200 black and white images, with a size of 480x752 pixels whose values range from 0 to 255. They represent metallic surfaces with different roughnesses (from 0.12 to 5.6 Ra). Some of them have defects, others not. The defects are of seven possible types: black defects, white defects, cleaning problems, illumination problems, focus defects, edges and a roughness not uniform. Here are a few representative images from our dataset:



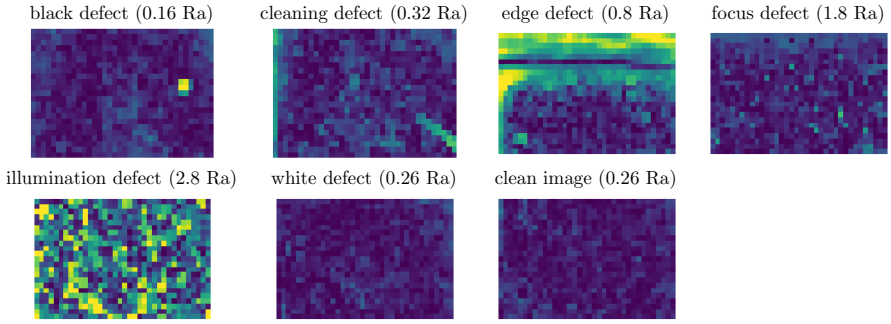
Of these 200 images, two thirds were randomly selected to build our training set, and the remainder to create a testing set. These images will be augmented using horizontal and vertical flips, resulting in a total of 800 images.

## 2 A simple classifier

To start with, we developed a simple classifier which works as follows. First, it performs a preprocessing phase in order to simplify these images as much as possible, while keeping the defects visible and making the images easy to handle by a computer:

1. It rescaled the pixel values to real numbers between 0 and 1. This step is useful to prevent the problem of vanishing gradients.
2. It applied a convolution with all identical, non-zero numbers in the kernel (a so-called 'box blur') to blur the image, using a kernel size of 20x20.
3. It applied a maxpool (with the same pool size as the convolution), to reduce the dimension of the input and prevent overfitting.
4. It subtracted the mean value of the resulting image from each pixel.
5. It computed the absolute value of the result, producing a 'heatmap' (these two steps ensure that black and white defects are highlighted in the image.

This preprocessing phase produced the following images (applied to the above inputs). They are rendered using the colormap 'viridis'.




Most of the defected images generated contain at least some high intensity pixels, while the clean ones are mainly dark.

To decide if an image is defected, the classifier then computes the maximum value of these pixels, and computes an ideal threshold to distinguish defected from non defected items. This baseline classifier yielded the following results:

| | training set | | testing set | |
|---|---|---|---|---|
| | Accuracy | AUC | Accuracy | AUC |
| black defects, Ra<0.4 | 0.95 | 0.95 | 0.97 | 0.94 |
| black defects, Ra<1.26 | 0.92 | 0.94 | 0.91 | 0.90 |
| all defects, any Ra | 0.77 | 0.80 | 0.72 | 0.74 |

With a Ra below 0.4, the classifier correctly labelled all non defected images, but made a few mistakes labelling defected images, like the one shown in Figure 1.

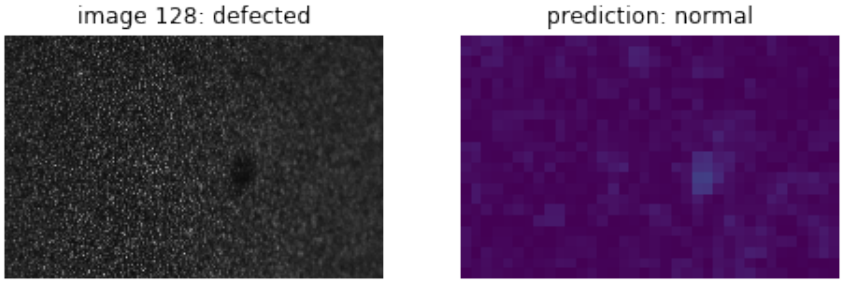



Figure 1: A classification error

Since the background was dark, the defect did not produce a high intensity pixel in the heatmap.

The results with a wider range of Ra values were still fine, while it achieved inferior results trying to detect any kind of defect on images with all sorts of roughness.

# 3 A convolutional extension

To improve on these results, we added a convolutional layer after the preprocessing phase, and before computing the maximum of the heatmap. Its purpose is to detect patterns in the heatmap that characterize defected items.

To be able to train it, we included the operations of the second phase in a convolutional network with the following architecture:

| LAYER | type | filters | neurons | kernel | activation |
|---|---|---|---|---|---|
| 0 | input | 1 | 23x36 | 5x5 | |
| 1 | convolution | 15 | 19x31 | 5x5 | |
| 2 | global maxpooling | 1 | 1 | 1x1 | |
| 3 | dense | 1 | 1 | 1x1 | sigmoid |

There was no need to add a relu activation function after the convolution, since it is followed by a global maxpool. The best results were obtained using 15 filters.

# 4 Results

The additional convolution significantly improved the results, in particular in the third task: detecting all types of defect on images with any roughness:

| | training set | | testing set | |
|---|---|---|---|---|
| | Accuracy | AUC | Accuracy | AUC |
| black defects, Ra<0.4 | 1.0 | 1.0 | 1.0 | 1.0 |
| black defects, Ra<1.26 | 1.0 | 1.0 | 0.98 | 0.99 |
| all defects, any Ra | 0.97 | 0.98 | 0.92 | 0.93 |

The classifier achieved very good results in the first two tasks. In particular, it correctly classified the defected image that had been previously wrongly labelled. One can generate a new heatmap by modifying the old heatmap, increasing the intensity of the pixels that exceeded some threshold in at least one of the maps of the convolution (see Figure 2).

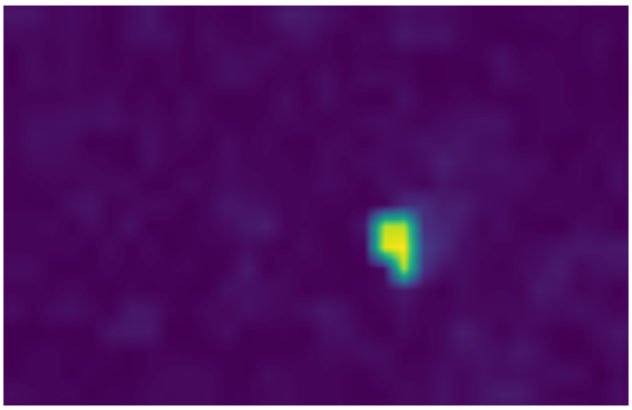


Figure 2: Heatmap generated by the convolution

In the third task, the classifier achieved an accuracy of 0.97 in training and 0.92 in testing. In testing, it misclassified 8 normal images out of 151 and 14 defected ones out of 119. These scores are obtained using a threshold of 0.5.

One can see these results also on the ROC curve (Figure 3). The $x$ coordinate of the red point indicates the percentage of clean images classified as defected (8/151), while its $y$ coordinate shows the percentage of defected items correctly

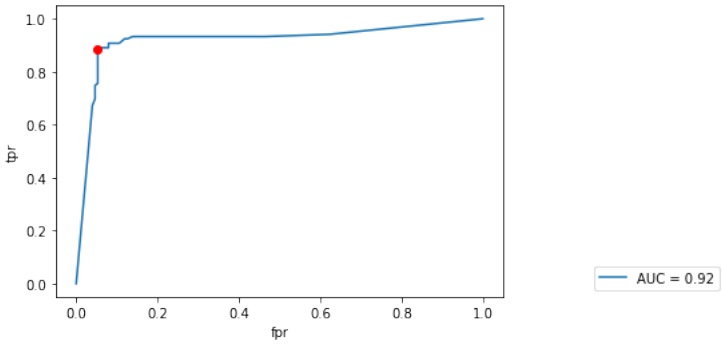



Figure 3: ROC curve

classified (105/119). If one changes the threshold, he obtains all the other points on the curve.

*Note*: During our experiments we noticed that, occasionally, our network did not start learning at all. This was due to the fact that when the input to the sigmoid exceeded a specific value (namely, 36), the gradients were all rounded to zero. To solve this problem, it was important to rescale the input to values between 0 and 1 and to initialize the kernels with small values. Another solution would be to use softmax instead of a sigmoid.

Even if it achieved better results, the new classifier also made some mistakes. Figure 4 shows one of these mistakes.

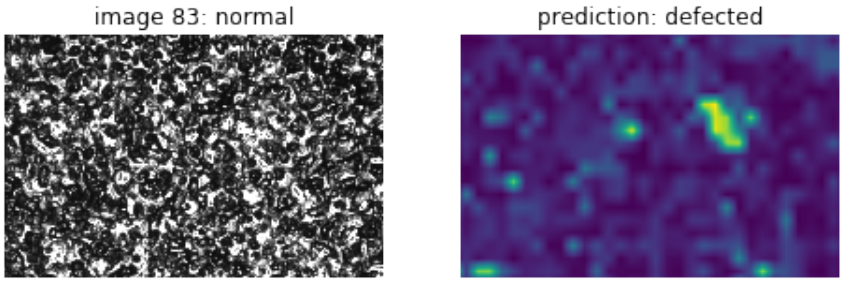



Figure 4: A mistake made by the enhanced classifier

# 5 Conclusion, Limitations and Further Work

The simple classifier obtained quite satisfactory results in the first two tasks, and mediocre ones in the third one. Its convolutional extension improved the results in all three tasks. However, the dataset used was limited in size. It should be increased to reach more conclusive results.