

Machine Learning Engineer Nanodegree

Capstone Proposal

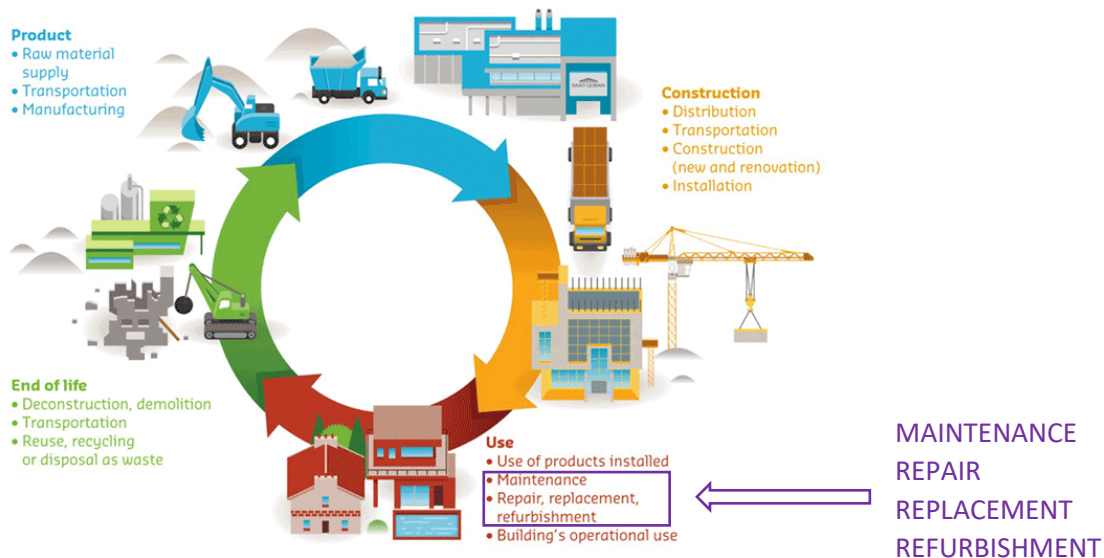
Gaylord Marville
December 5th, 2018

Proposal

Domain Background

The life cycle of a construction product is roughly made of four parts. After the construction and before the end of life of the building, there is the most common and most extended part of this cycle – “the used part.” During this part, the structure is held by a client who exploits it. Sometimes severe defects appear on this structure, and the client must repair it. For this purpose, the client engaged specialists to understand the cause and find a solution. Some clients are more cautious and inspect their property periodically before problems occur. Whatever the situation is, when it happens, the previously mentioned specialists are producing a written report stating the issues and how the client should handle them. Then the client engages a process of reparation, but that is something else.

THE LCA OF A CONSTRUCTION PRODUCT



Life cycle of building and major civil engineering works such as tunnels or bridges

Recently the Morandi bridge collapses in Genoa a city of Italy and did 41 casualties. To avoid those kinds of tragedy, French legislation fixed the terms of mandatory periodic diagnosis on public structures every 5 to 6 years. Those diagnosis consist of exhaustive inspections of the structural integrity, by detecting and locating and then rating all possible disorders a construction may be affected by. The written report I mentioned later is a 5 part one. One part of it prints out the disorder's pictures in A4 pages. Sometimes disorders like crack are barely visible due to the degradation of the printing process on the small sized images. When I was a student in training session, I had a mission among other which was to underline those disorders on the pictures and the only goal was to make them visible on the final paper report. It was a tedious task especially for cracks.



Concrete cracks



Underlined cracks



Spalling



Underlined spalling

The two cases above are easily readable in a report which is not necessarily the case for the one below, even after it has been underlined.



Concrete cracks

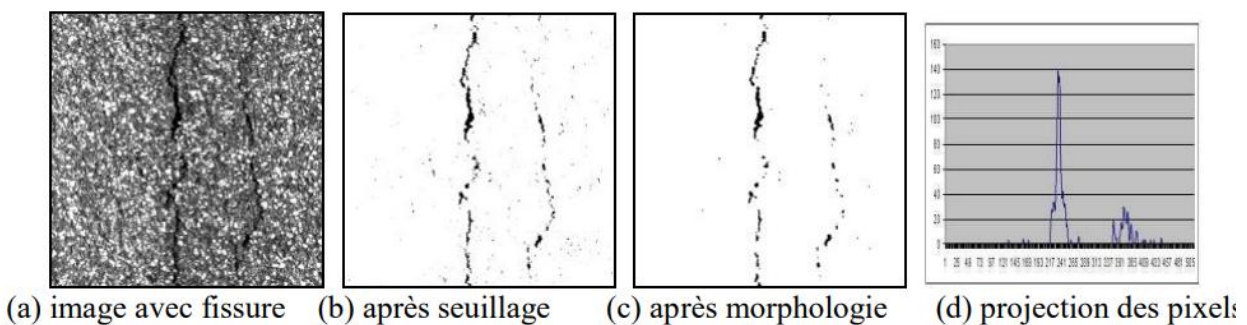


Underlined cracks

This goal may seem futile, but this is what motivated me to find a solution to accomplish this underlining by computer. The use case of such a technique coupled with accurate localization of those disorders could radically change the way we are collecting them.

Problem Statement

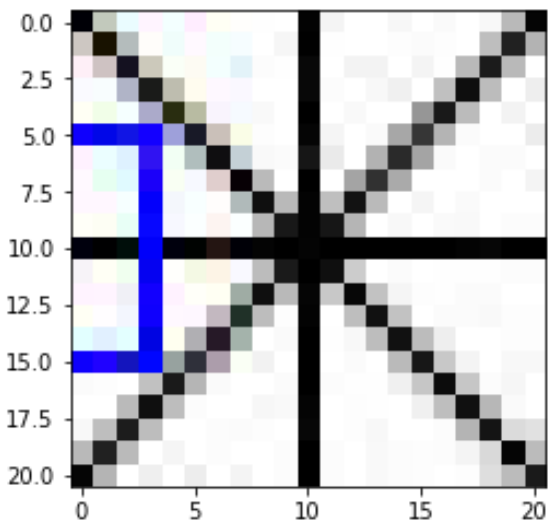
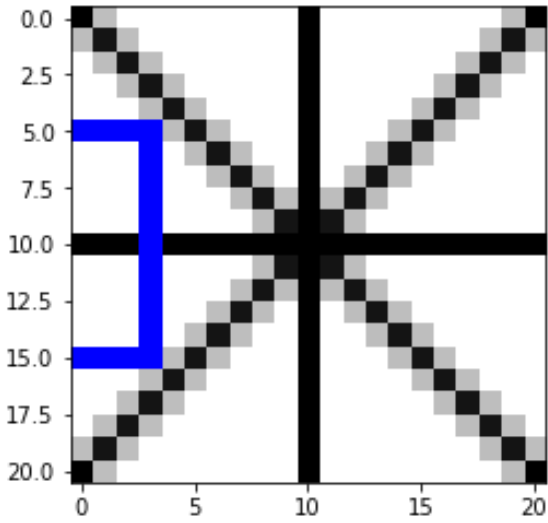
The problem to be solved here is a computer vision one. It has been subjected to many research papers these past few years before the emergence of deep neural network and technics mainly based on convolutional neural networks to work on images. Before that, the problem was addressed using traditional image processing technics such as Canny edge detection or Hough transforms coupled with a non-destructive rigid transformation like rotation and flipping without success. It was mainly resulting in non-consistent results. Indeed, it was difficult to predict those kinds of tasks may need a new type of computer science.



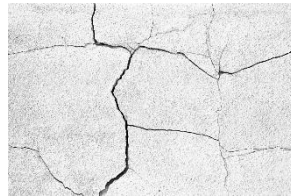
Traditional technics attempts

Datasets and Inputs

To work with images, it is more than wise to consider the PNG compression format instead of JPG since the last one is degrading images during compression. We want to delete noise due to compression during the process of creation of the labeled-masked images. Indeed neural network may be sensitive to that perturbation which could prevent it to converge to the solution or at best slow the progression.

	Image	First channel of the corresponding numpy array
JPG		<code>[[222 241 255]]</code>
		<code>[[224 250 255]]</code>
		<code>[[246 255 255]]</code>
		<code>[[232 255 246]]</code>
		<code>[[0 14 10]]</code>
		<code>[[240 255 239]]</code>
		<code>[[240 240 246]]</code>
		<code>[[243 246 254]]</code>
		<code>[[227 255 255]]]</code>
PNG		<code>[[255 255 255]]</code>
		<code>[[255 255 255]]</code>
		<code>[[255 255 255]]</code>
		<code>[[255 255 255]]</code>
		<code>[[0 0 0]]</code>
		<code>[[255 255 255]]</code>
		<code>[[255 255 255]]</code>
		<code>[[255 255 255]]</code>
		<code>[[255 255 255]]]</code>

There are dozens of different disorders in civil engineering, but here I chose to work with the most representative one, the concrete crack. I had to build a dataset quickly, so I scrapped Google image using googleimagesdownload -k "concrete crack" (<https://github.com/hardikvasa/google-images-download>), in all the language Google could translate and only looking for large pictures.



Solution Statement

The idea will be to take a crack picture and use the Photoshop spot healing brush tool to remove the crack. We will then train the network in this task and later make the difference between the input image and the output one to obtain the crack. We could create a mask of cracks for each picture, but it is a much longer work, and here we can also take advantage of the uniformity of a concrete surface. Such a technique may not be reproduced for every situation like for example urban scene segmentation.

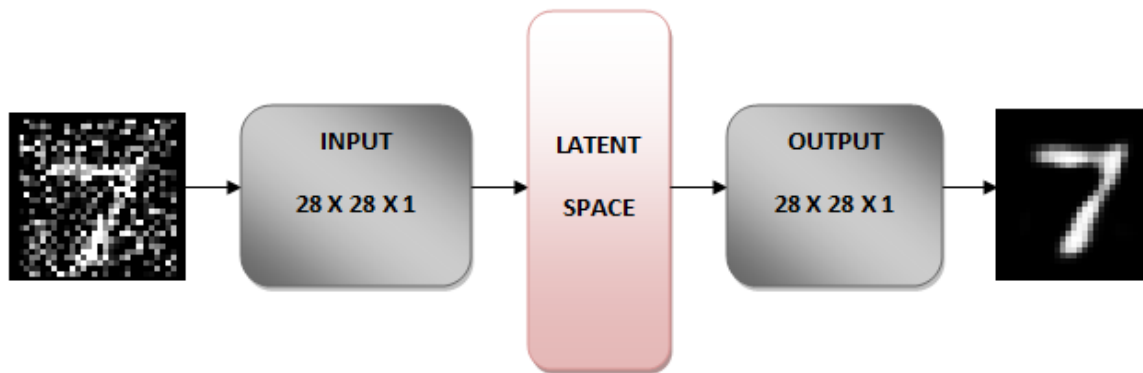


Original image

Same image after photoshop treatment

Benchmark Model

Denoising Autoencoder

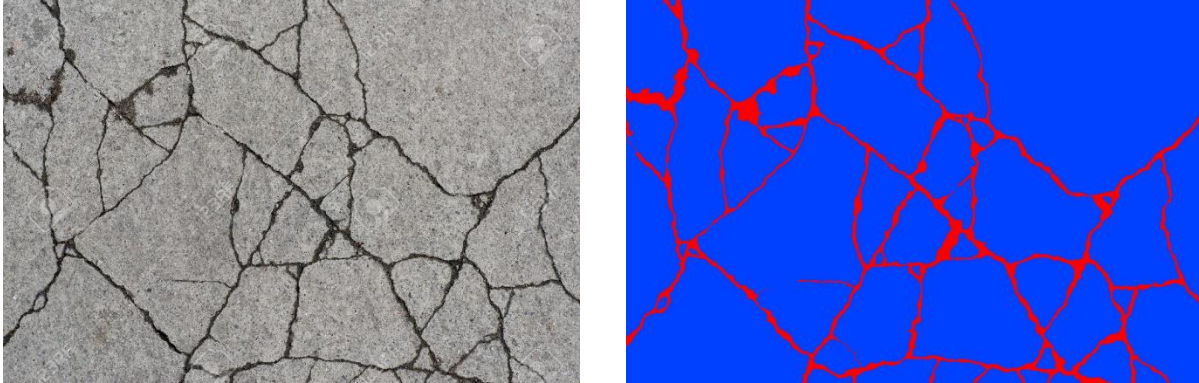


"The idea behind a denoising autoencoder is to learn a representation (latent space) that is robust to noise. We add noise to an image and then feed this noisy image as an input to our network. The encoder part of the autoencoder transforms the image into a different space that preserves the handwritten digits but removes the noise. As we will see later, the original image is 28 x 28 x 1 image, and the transformed image is 7 x 7 x 32. You can think of the 7 x 7 x 32 image as a 7 x 7 image with 32 color channels."

<https://www.learnopencv.com/understanding-autoencoders-using-tensorflow-python/>

Evaluation Metrics

To evaluate our model, the best is to manually mask a sample of image let say a hundred and, as a metric, calculate the dice or mean IOU between the model inference and the mask.

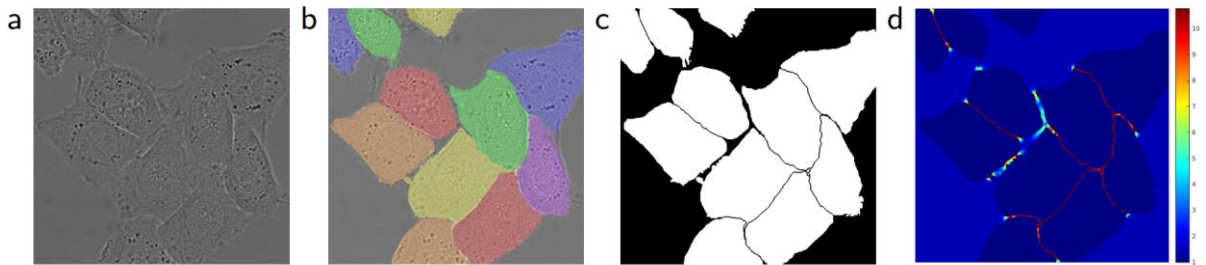


The IoU of a proposed set of object pixels and a set of true object pixels is calculated as:

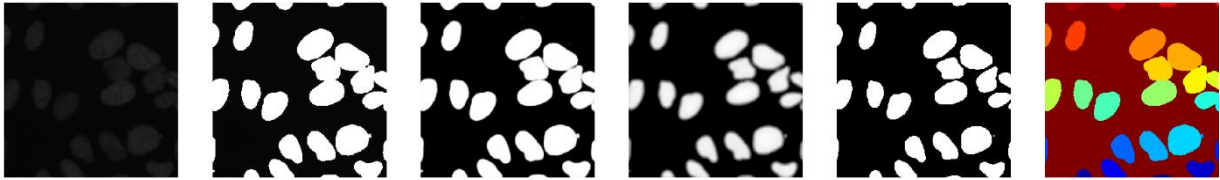
$$IoU(A, B) = \frac{A \cap B}{A \cup B}.$$

Project Design

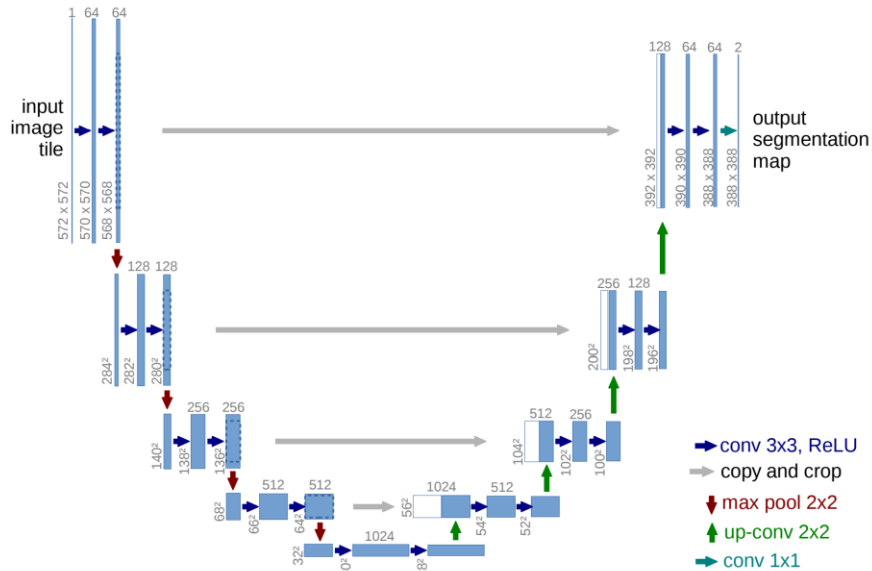
During the “2018 Data Science Bowl Kaggle competition”, U-Net an efficient architecture of image segmentation made the difference. We will reuse for cracks detection.



From the 2015 U-Net research paper <https://arxiv.org/pdf/1505.04597.pdf>

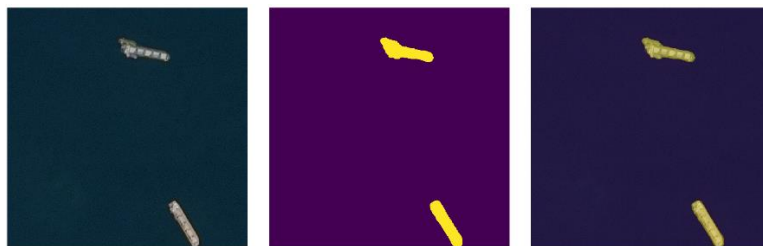


Kaggle data bowl challenge



U-Net architecture

U-Net is composed of an encoder and a decoder linked in the middle by concatenated connections which improve the model contouring sharpness. U-Net can be used on different CNN architecture and we I'm planning to use it on the ResNet34 which is a lightweight network. I will not use a pre-trained model of ResNet34 which is increasing the implementation difficulty and may not be pertinent as no dataset today is close enough from cracks.



U-Net recently used on the Airbus/Kaggle challenge