Project Proposal¶

Summary¶

Inside this document you will find a capstone proposal for the final project of Machine Learning Nanodegree. It is included below a brief introduction to the problem statement, objectives, potential methods for problem solving and expected results. The first part is dedicated to explain and explore the chosen dataset, which includes PreOps (cleaning and wrangling). The second part shows the machine learning classical steps: preparing data, splitting data in training, selecting a model and then feeding with data. The third and last part shows the results and final remarks.

Table of Contents¶

1. Domain Background¶

The industry of steel is one of the most important sectors around the world, accounting for massive jobs, high cash flow and delivering steel for many applications. Since industrial revolution in late 17th century, steel is a raw material in high demand for infrastructure, vehicles, ships, etc. Steel can be produced in many shapes, such as: plates, tubes, wires and beams so quality control is utmost important to assure its functionality and safety along its life-cycle.

2. Problem Statement¶

Steel surface analysis is one of the techniques developed to watch the material condition by revealing cracks, imperfections and patterns that can indicate the need for replacement. Surface analysis is generally performed by a professional inspector, trained and certified for this task. However, this activity can be overwhelming since the number of image taken of a building or and industry plant. Moreover, sometimes this activity can be dangerous, for example, in top of buildings (height) and inside equipment (limited space).

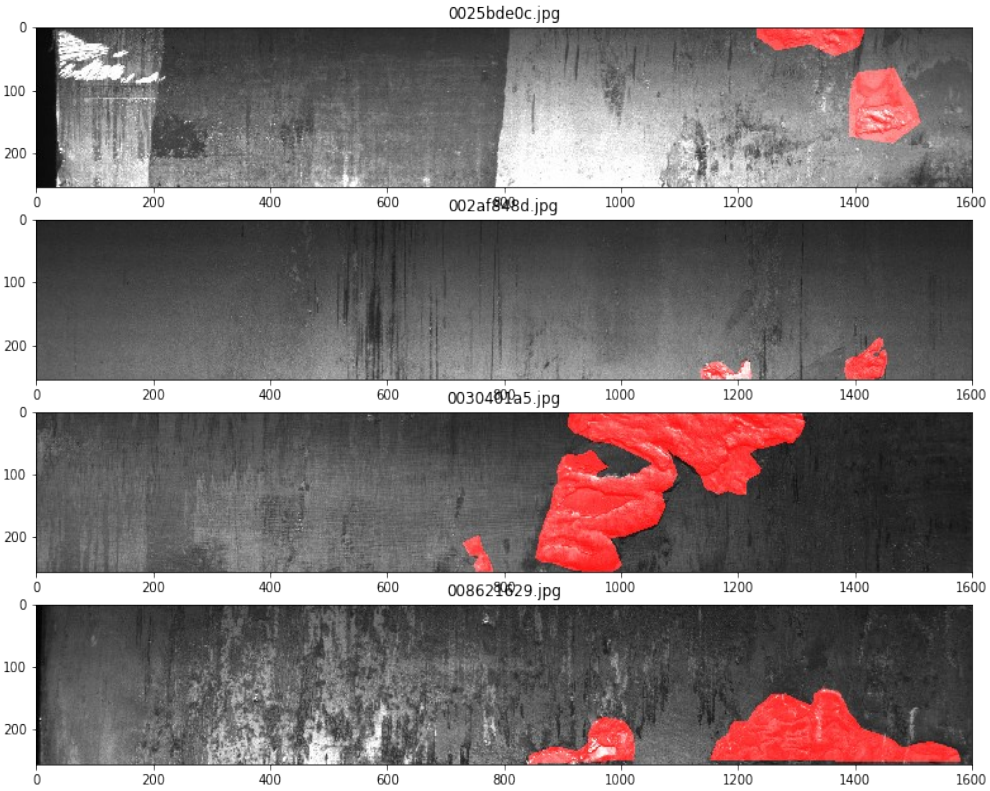
One alternative to improve surface analysis is to train a model to learn steel surface patterns in order to classify wheter a surface has a an imperfection or not, and if possible, what kind of imperfection it has, using machine learning methods. Important notice that the early detection of any issue in steel might prevent incidents and help the structures and equipment to perform as it is intended.

3. Dataset and Inputs¶

Although, Udacity has provided some data examples, I've opted to search for a more specific dataset towards my professional formation: chemical engineer. The dataset I've chosen can be find in Kaggle and is about a collection of steel plates images that can or cannot present an imperfection on its surface. Some images have a label with respect to the class of imperfection it has.

The dataset is originally divided in 2 folders: train and test. Train folder contains approximately 12k images of steel plates in shades of gray, most images shows parts of steel and some contains imperfections that are classified 1 up to 4. While test folder contains about 5,5k images. Further investigation in dataset will be done in exploratory data analysis step.

There's also another folder from a similar source containing mask for the imperfections, we can use these files to improve our model, but this will be discussed later in model selection. The Figure below shows an example of steel plate marked with its respective mask.

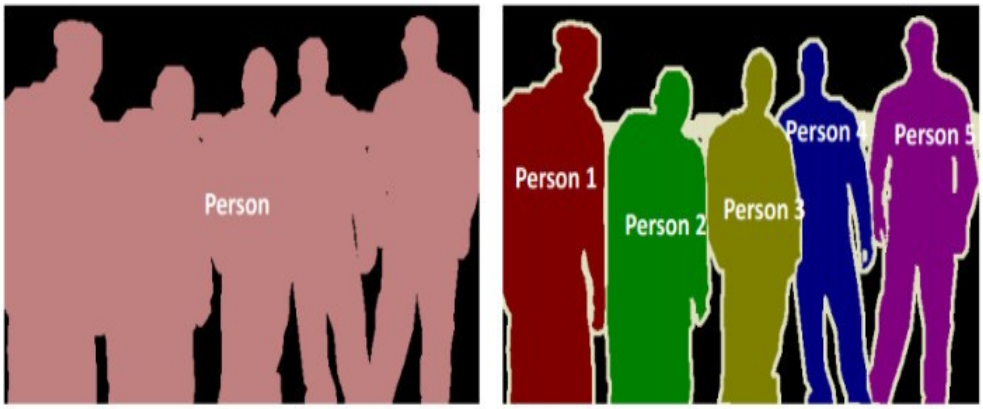


Link: https://www.kaggle.com/c/severstal-steel-defect-detection

4. Solution Statement¶

Since we have a dataset of images, it's a good approach to start working on a convolutional neural network (CNN) in order to get the images features that are more related to the imperfections. This stated, we can work on a classification network to categorize the results to show the imperfection over an image test.

There are many classification models available in literature, they basically do semantic segmentation, which is pixel classification towards a pre-trained feature or category. Some models can do better by doing instance segmentation, which is an object or subset classification, e.g., detect boundaries between a car, the road and the sky.



Nowadays, the most prominent image classification models are based on convolutional neural network (CNN) with some grade of modification in either input vector or hidden layer. For example, we can cite: FCN (fully connected), ParseNet (a modification in FCN), ResNet (microsoft net) and UNet (enconding-decoding). More recently, Faster R-CNN and Mask R-CNN were developed to account pooling schemes to speed up the computing process by selecting zones of interest instead of processing whole image.

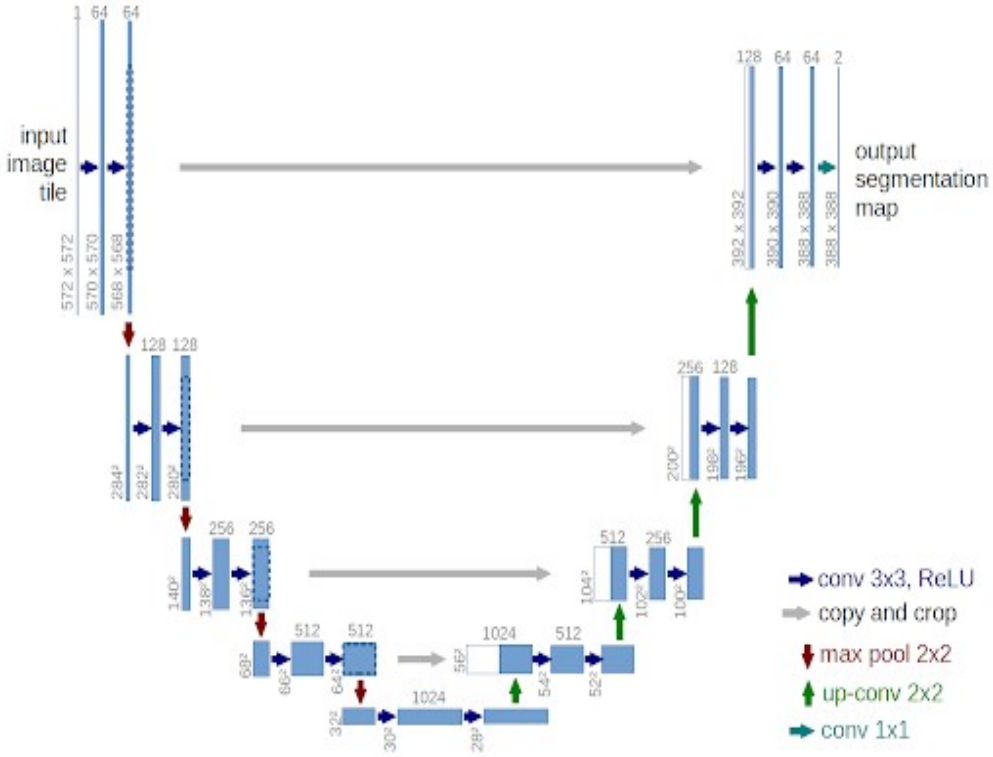
The abovementioned strategy can be achieved by the following steps: (i) PreOps (data cleaning and wrangling); (ii) train and test split; (iii) convert images to matrices (NxM; where N is height and M width); (iv) training a pre-selected model; (v) evaluate model hyperparameters; (vi) re-train model and/or tuning hyperparameters if applicable; (vi) analyze results and make predictions to validate model.

Link: https://medium.com/@arthur_ouaknine/review-of-deep-learning-algorithms-for-image-semantic-segmentation-509a600f7b57

5. Benchmark Model¶

As discussed above there are many models in literature that can do segmentation in images, however there are some particularities one might keep in mind before selecting a model. For instance, we can highlight 2 models that excell in this task: UNet and Mark R-CNN, so they are considered benchmark in computer vision. Note that both of these models are based on deep learning kernels so they are also considered heavy and demand high computational power. Furthermore, they do well in academic research, however for some real-time segmentation or some other edge device, one must rely on less sophisticated models.

In order to solve the classification problem we might choose a model that perform well in image segmentation as well as have a moderate processing time due to our dataset. Preliminary tests will be done first using a UNet model written in PyTorch, this model has good approach in segmentation and allow fully customization of its hidden layers. UNet is composed by a series on conlutional layers and pooling tasks, forming a U shape net (that's why the name U Net), the first part is responsible for encoding the image data, that is extract its features. The second part is responsible for decoding the image and re-localize its feature and achieve a 'meaning'. UNet has presented good results in many research papers and moderate computational effort since its network well known and pre-programmed. Figure below show UNet architeture, each step is either convolutional net or maxpooling sequence, originating a CNN with many hidden layers able to detect patterns in images.



Mask R-CNN is the state-of-art in image segmentation, some researchers have been using Mask R-CNN, which is an update from Faster R-CNN, to achieve very good results in industry. This method works in a similar fashion of most classification schemes, so, for a given image, Mask R-CNN, will return either a bounding box, class tagging and mask. The advantage of Mask R-CNN is that the extracted features are passed through a region proposal network (RPN) which return its bounding box and class. Combined bounding boxes are passed to a fully connected network for object identification and positioning in a 'meaning'.

In terms of baseline, we can compare our output to some Kaggle competitions that made use of either UNet or Mask R-CNN, remarkably, they got well evaluated in similar datasets with an average to good leaderboard position (LB).

Link: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/?utm_source=blog&utm_medium=computer-vision-implementing-mask-r-cnn-image-segmentation

6. Evaluation Metrics¶

The model will be evaluated using a couple metrics that make sense for image classification, namely, accuracy, precision, recall, dice and IoU. All these metrics are explained below:

(i) Accuracy, after model is computed and testing dataset is performed we can evaluate how many images were classified correctly over all images.

(ii) Precision, after model is computed and testing dataset is performed we can evaluate how many images were classified with a certain feature (positive) over all tagged images (either correct or incorrectly tagged), focus on detect overzealous models.

(iii) Recall, after model is computed and testing dataset is performed we can evaluate how many images were classified with a certain feature (positive) over all correct tagged images plus incorrect ones, focus on detect negligent models.

(iv) Dice, is a parameter that can serve as a loss function, geometrically, it represents the 2 times intersection area between [classification] vs [mask] over the sum of [classification] plus [mask] area.

For instance if the areas are completely overlaped, Dice is equals to 1, otherwise, if area don't match, Dice is null.

(v) IoU, intersection over union, very similar to dice but it is used a parameter for evaluate model after training step.

Below, it's possible to see math formulation where: tp - true positive, tn - true negative, fp - false positive, fn - false negative, I - intersection and [ ] area.

Accuracy = (tp + tn) / (tp + fp + tn + fn)

Recall = tp / (tp + fn)

Precision = tp / (tp + fp)

Dice = 2*I / ([class] + [mask])

7. Project Design¶

The aim of this project is to build a machine learning application from end-to-end, starting from dataset and pre-operations, training a model and then deploying it for validation. Some def functions will be developed to preprocess the dataframe, train the model and predict the results based on classification. Finally, the code needs to be posted in a github repository and all the progress written down in a tech blog. The project will follow classic ML project schematic:

(i) Data Cleaning and Wrangling: PreOps, data selection, reforming and replacement.

(ii) Exploratory Data Analysis: graph, histograms, general info.

(iii) Data Preparation: vectorizing, def function development, train / test split.

(iv) Model: Input preparation, model selection, hyperparameters, loss function, training.

(v) Results: endpoint creation, model evaluation and testing.

(vi) Prediction step: model validation and endpoint call.

(vii) Wrap up progress, write report and upload code to github.

Note that some steps can be iterative, since we have to return to previous step in order to change parameters values or resample data, for example.