Severstal Steel Defect Detection

Tevfik Çağrı Dural

Springboard Data Science Career Track

Branko Kovac

May 2021

Abstract

Steel production is one of the world's leading industries. The product is used in many fields, from aviation to construction, from household items to cargo. Therefore, having a reliable product will support many industries and help the world's industrial development.

This work develops a model to classify the defects on a steel image. Later on to be used in production-line and quickly define the problem on the product

Keywords: steel, production, neural network, transfer learning, image processing, image classification

Problem Statement

"Severstal is leading the charge on efficient steelmining and production. They believe the future of metallurgy requires developmentacross the economic,ecological, and social aspects of the industry—andthey take corporate responsibilityseriously. The company recently created the country'slargest industrial data lake,with petabytes of data that were previously discarded.Severstal is now looking tomachine learning to improve automation, increase efficiency,and maintain highquality in their production."[1]

How to build a model that classifies the images with defects? Therefore Severstal can improve its product quality and efficiency.

[1] Severstal kaggle competition overview, Severstal:Steel Defect Detection, 04/04/2021

Dataset

Data is provided by Severstal on the Kaggle Competition page.[2] Data consists of train and test sets. Two CSV file for training the model and a sample submission file.

- Train images, with around ~12k images either with or without defects.
- train.csv file for training. Mapping the train images but only with the ones with defects.

  Consists of three columns
  - `ImageId` file name of the image
  - `ClassId` type of the defect
  - `EncodedPixels` representing the location of the defects on the image.
- Test images. Images for model testing
- Sample_submission.csv. Expected output type of the model

Exploratory Data Analysis

In this part provided data is explored to understand the computer vision behaviour on such images and preparing the preprocessing steps.

Accordingly, the below questions asked;

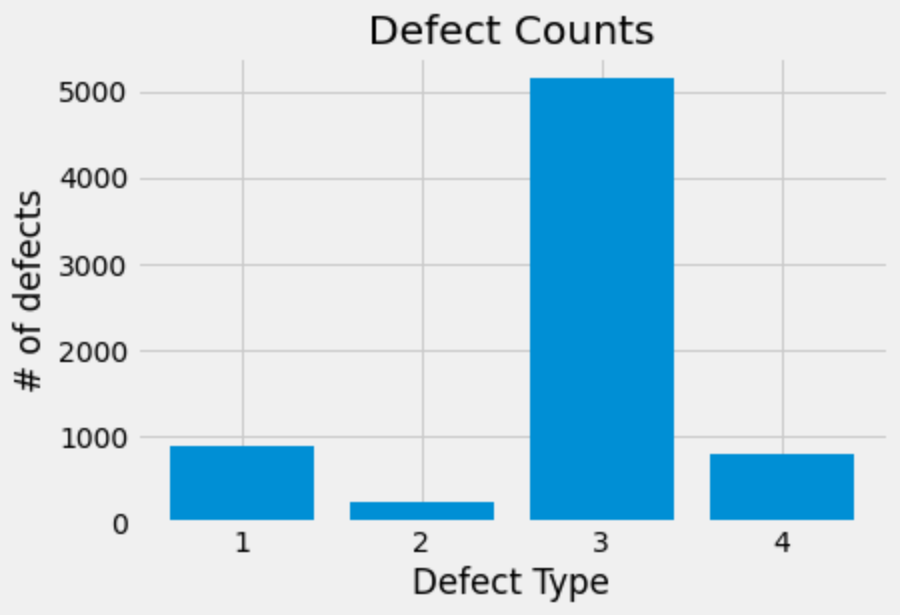
- Are all images of the same size?

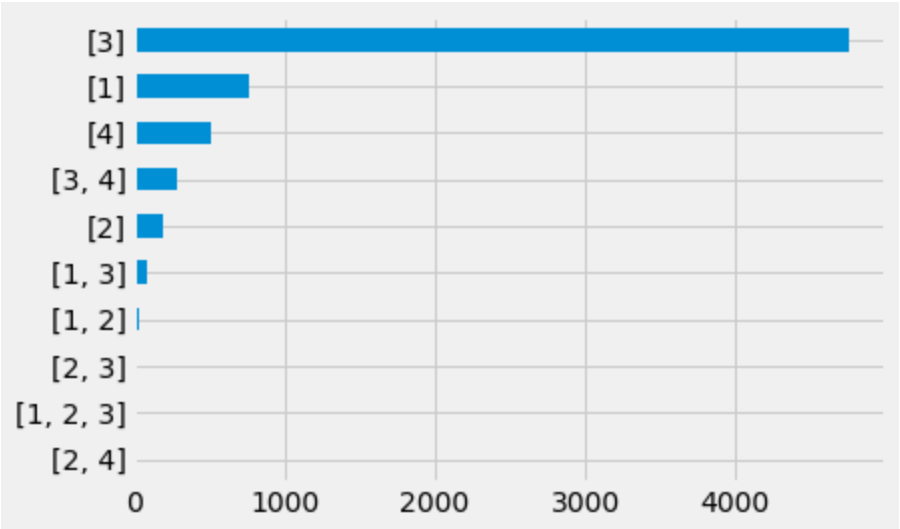
  All train and test images were in the same shape.

[2] https://www.kaggle.com/c/severstal-steel-defect-detection/data

➢ What kind of problem is this?

Plotting the number of defect types showed that it's highly imbalanced data. Also, in some cases, one image could have more than one defect. Having that information, this is a multi-label imbalanced classification problem





- Does any preprocess required?

  As the defect locations are given as decoded, they needed to be encoded. Detailed information about decoding can be found on the competitions' evaluation page.[3]

- Does all train images have defects?

  Having around ~12k images in the train images folder and ~6k rows in the train CSV file shows that almost half of the images does not have any defects. However, the following model will be prepared only with the ones with the defects to reduce the problem complexity.

- What is the computer vision algorithm's approach to the images?

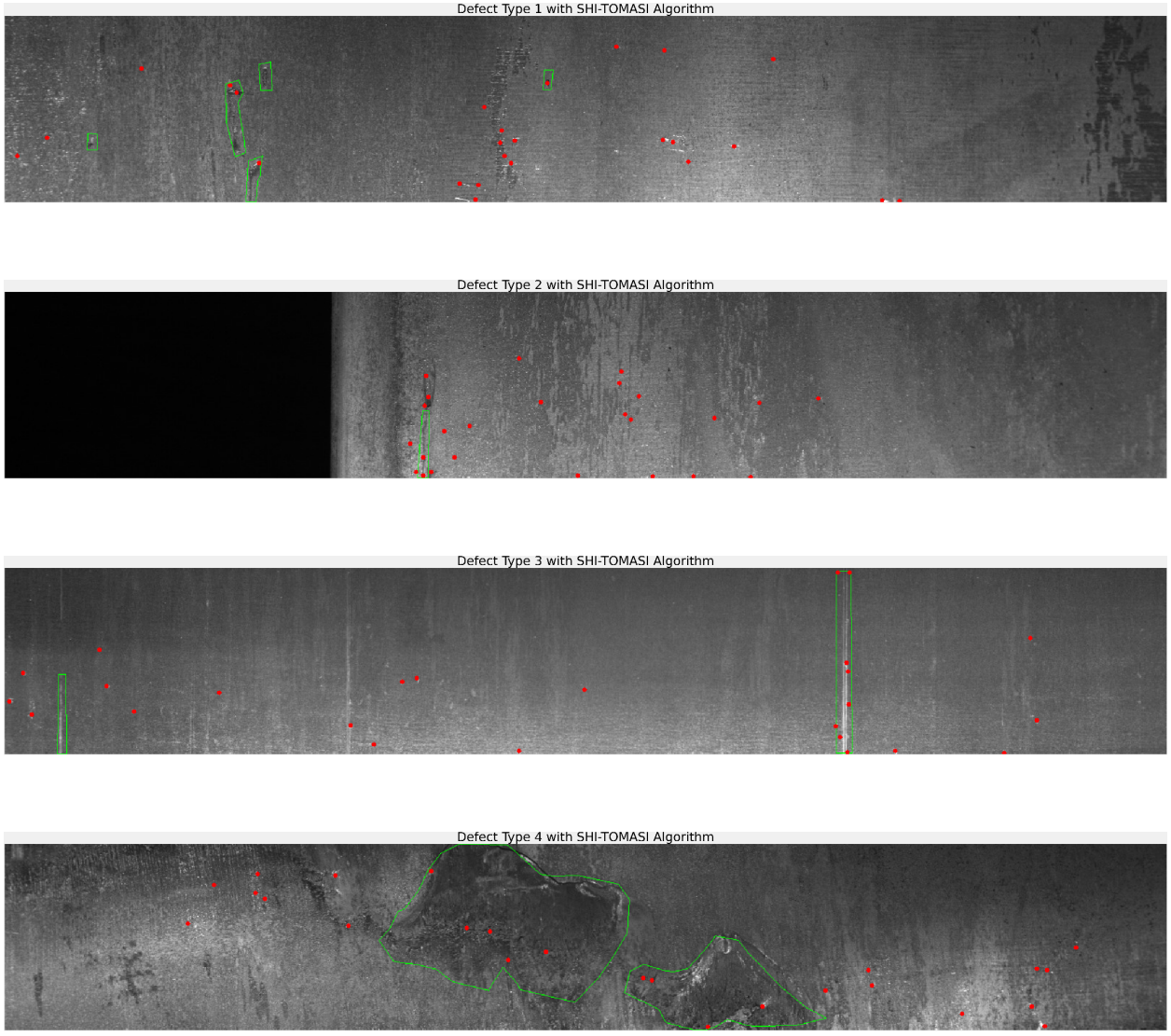
  For this question OpenCV's four different feature detection algorithms used. Feature detection algorithms are a very low-level approach to computer vision problems.

  1. **Shi-Thomasi**

     This is a corner detection algorithm. Which can provide an understanding of how the very early levels of a NN model reacts to the image. Detailed information can be found on the Wikipedia page

[3] https://www.kaggle.com/c/severstal-steel-defect-detection/overview/evaluation

Some images with the points detected by the algorithm and the defect;

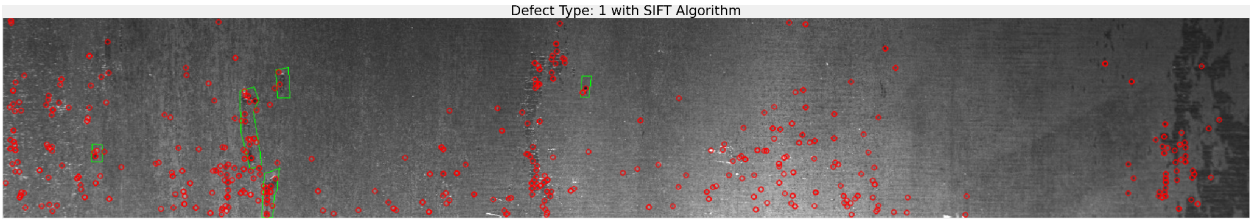


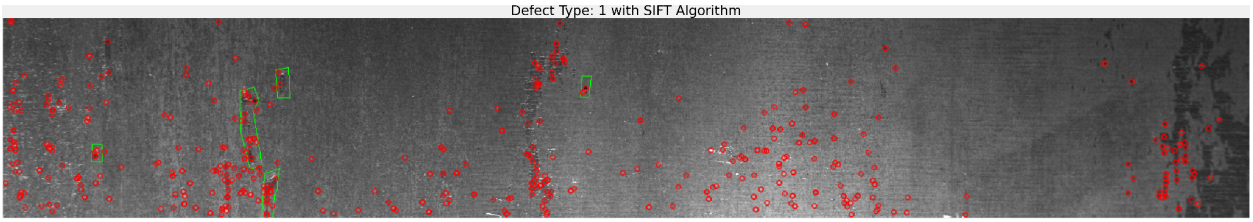
At this level, it doesn't seem like the algorithm catches anything specific.

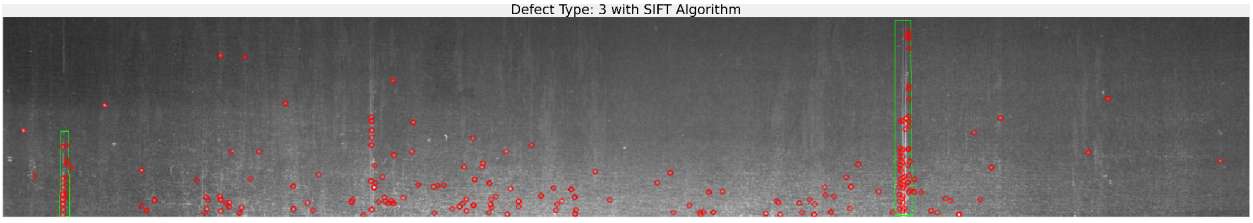
2. SIFT

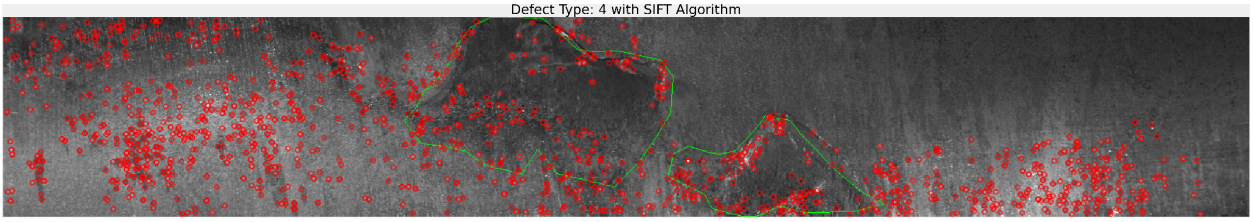
Scale-invariant feature transform (SIFT) is a feature detection algorithm. This algorithm is more complicated and extracted features can be used with scaled or noise added images too. Detailed information can be found on the Wikipedia page

Some images with the points detected by the algorithm and the defect;







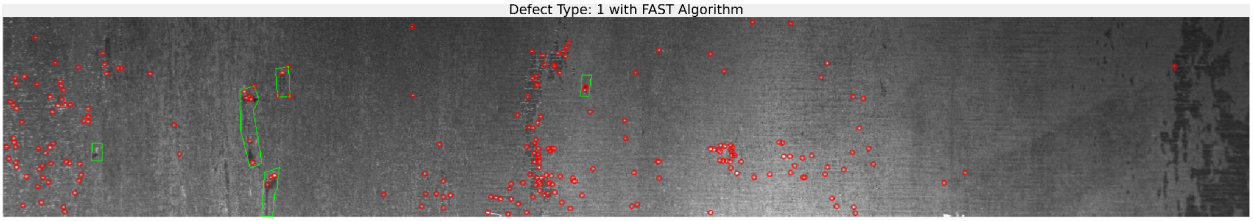


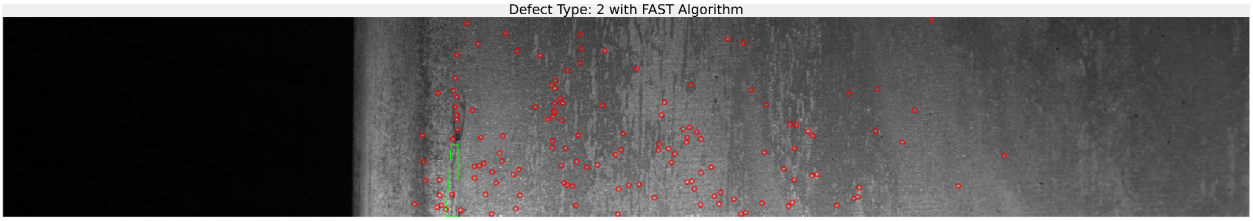
As being more complicated SIFT algorithm can detect more structural features in the images. Especially works well with defect type 3.

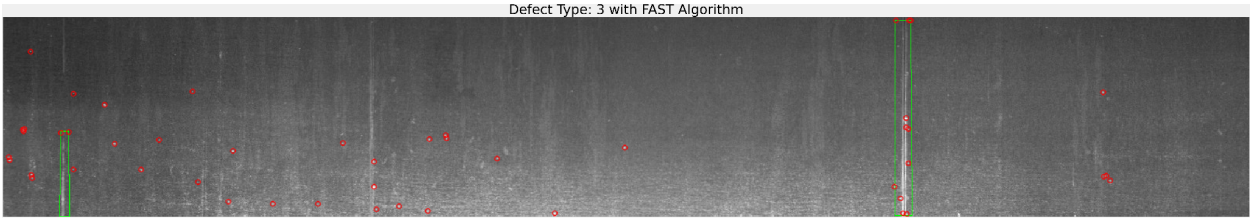
3. FAST

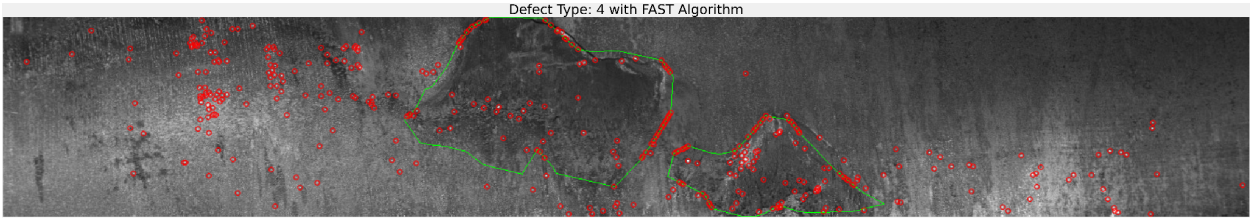
Features from Accelerated Segment Test (FAST) is another corner detection algorithm. Detailed information can be found on the Wikipedia page

Some images with the points detected by the algorithm and the defect;







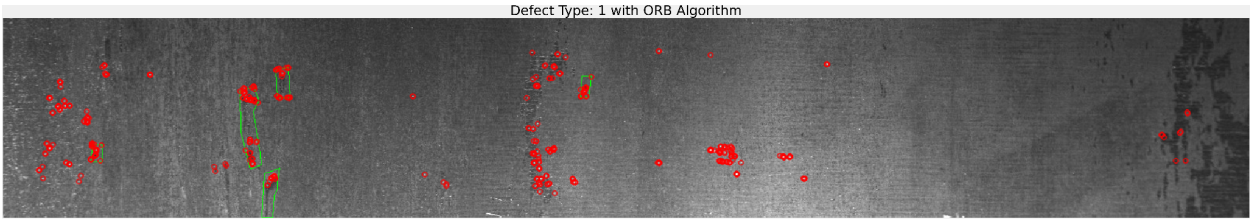


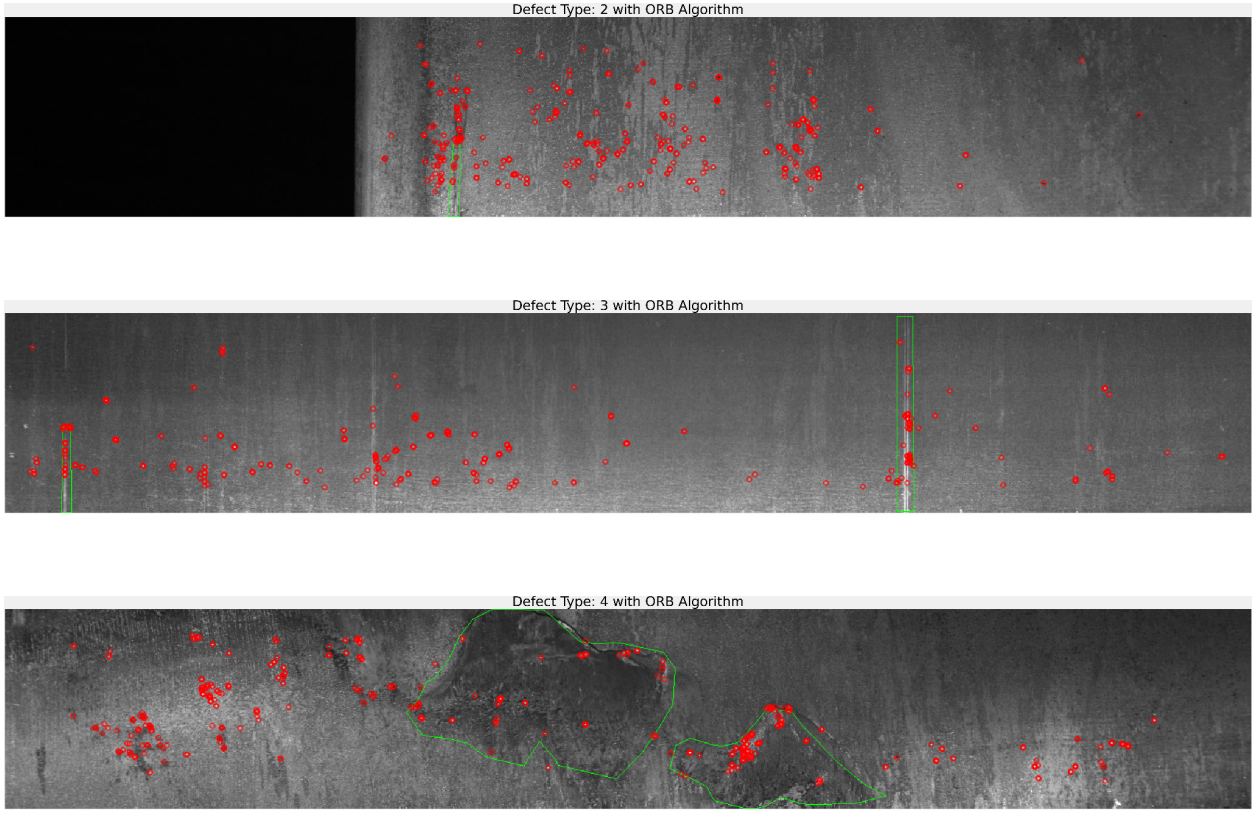
FAST is also better at detecting good features, especially for defect types 1 and 4. However, It doesn't seem to be good with defect type 3 and this is the biggest set of labels.

4. ORB

The final algorithm is Oriented FAST and Rotated BRIEF (ORB). FAST was patented until the beginning of 2021 and the BRIEF algorithm was built on FAST. To provide users an alternate ORB algorithm was developed by OpenCV labs. Detailed information can be found in OpenCV documentary

Some images with the points detected by the algorithm and the defect;





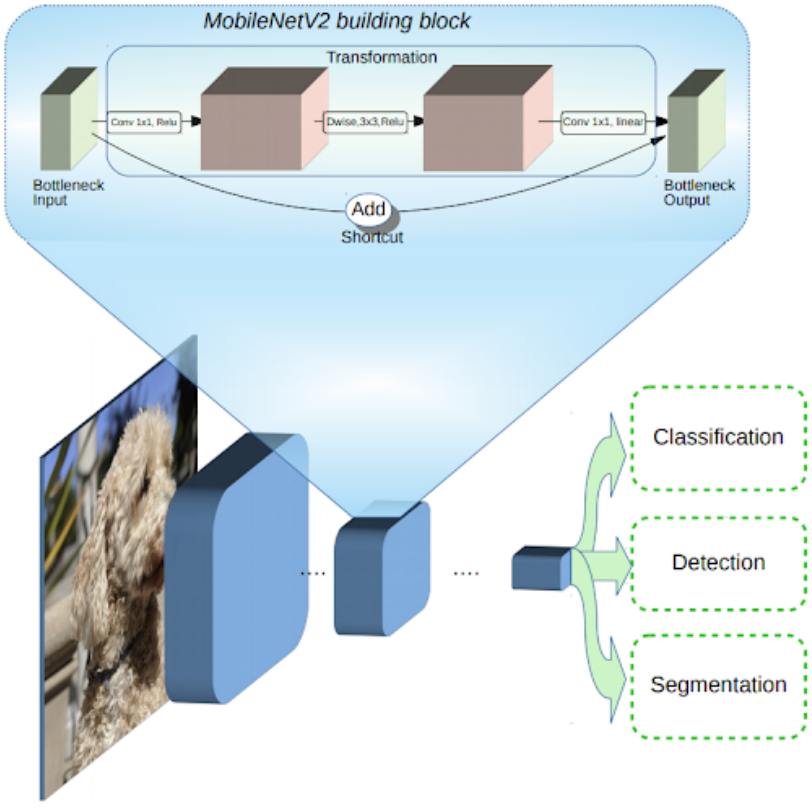
ORB is one of the best so far between the algorithms.

The final understanding of all algorithms provides that images carry some good features which can be detected by algorithms and no specific preprocessing is required. With this understanding, a neural network will be build to classify images.

Modelling

There are many state-of-art models developed and trained with millions of images. Instead of building a neural network, this work uses an already trained MobileNet V2[4] developed by Google. Then this neural network will be set for the problem type, predict with the base model. Finally fine-tune the model and get the predictions. This process is called transfer learning.

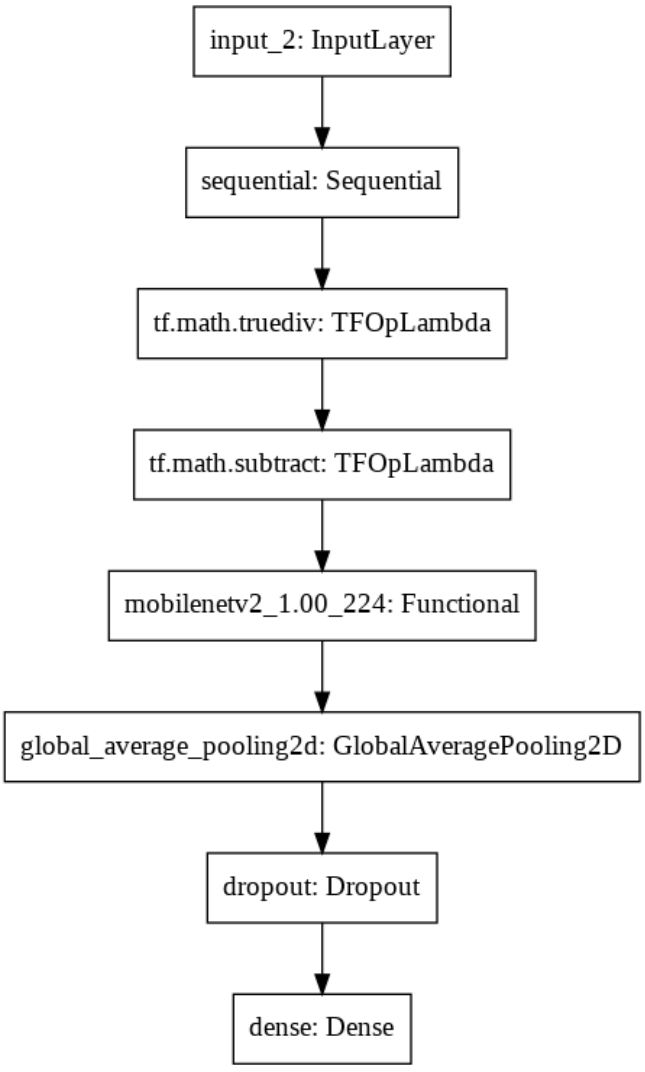
TensorFlow library is used to develop the model and the already trained model was imported from the TensorFlow hub.



MobileNet V2 Architecture

[4] https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet

This model imported without its top layer and three layers added on top. First, a global average layer to have a bridge on reducing the number of parameters. Then a dropout layer to randomize the inputs for the final layer which helps the overfitting of the model. Then a final predictor layer with four outputs corresponding to the defect types.

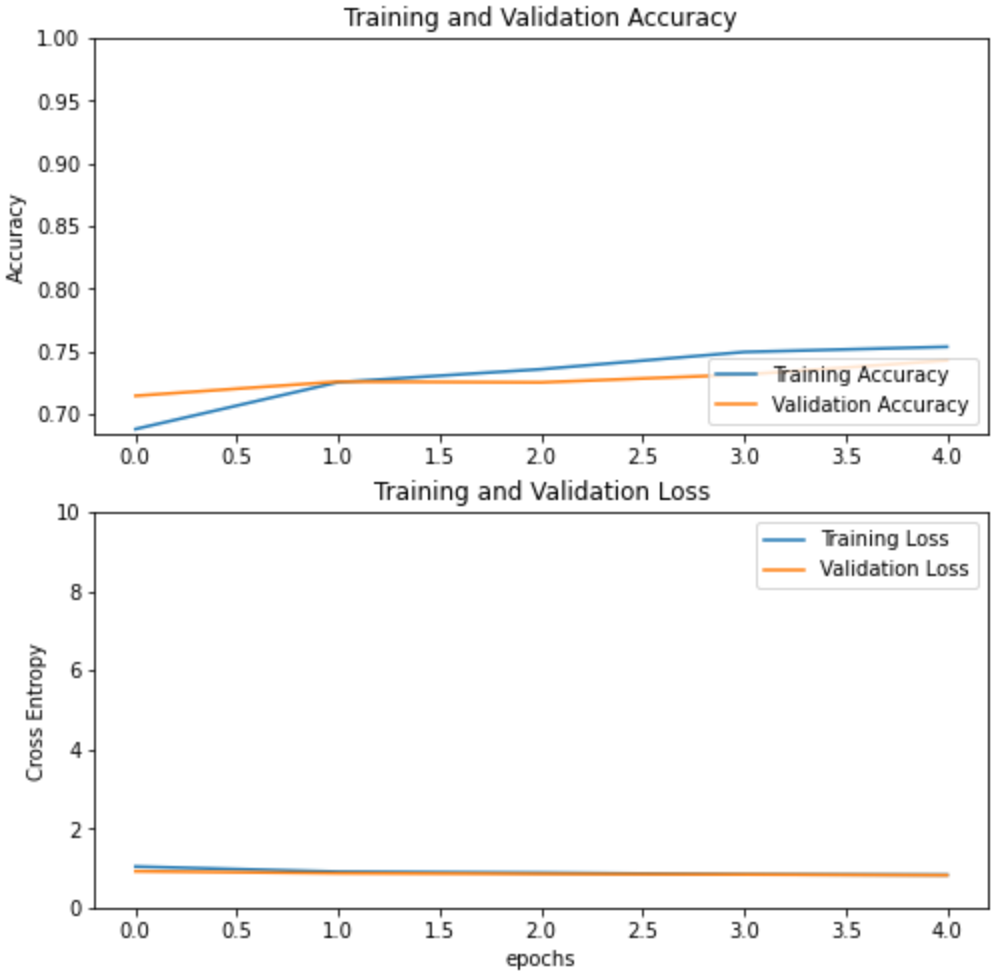


Final model architecture

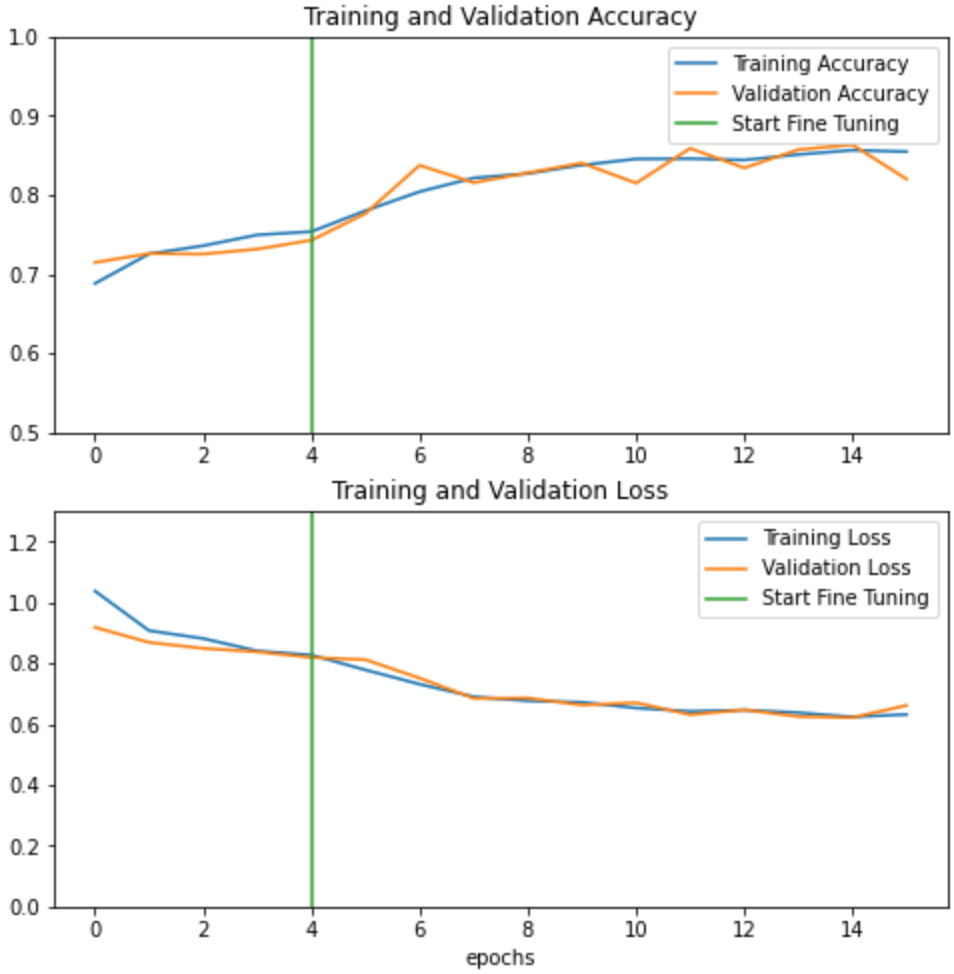
A prediction run on the validation set to have an intuition of how the general approach of model - data relationship

loss	1.40
accuracy	0.26

Then only the added layers trained with the train set for 5 epochs. And a significant can be observed.



For fine-tuning, the final 14 layers of the base model trained with the train data for another 10 epochs and results can be seen below.



The final accuracy of the model is 85% on the train set and 82% on the validation set.

Conclusion

This work is just a basic approach to the computer vision field and solves only the classification problem. The reason for selecting a pre-trained model is they are state-of-art well structured and already trained on millions of images from different type of problems. So if our problem was one of these, the out-of-box model would already score well at the beginning. But having a very low accuracy of ~26% shows that it was not.

Before fine-tuning one pooling and one dropout layer added to the model and only new layers trained for five epochs. This resulted in having ~75% accuracy. Then the top 14 layers opened for training and all these layers are trained again for another 10 epochs resulted in having the final accuracy of ~86%. This is a significant increase and can easily reduce the business workload.

Yet there are more works that need to be done. First, increasing the accuracy can be done with another model and/or changing the model's fine-tuning layers. Especially an experimental approach working with layers on different depths can even increase the accuracy. Having a different type of problem than the model pre-trained might need some fundamental changes and re-training low-level layers can help the model adapt to this kind of problem. Having this is an experimental process needs more time for competence. Additionally, the competition was expecting detected defects' location on the images and this is another problem named image segmentation.