# Capstone Proposal

Ahmed A. Ahmed

Machine Leaning Engineer Nanodegree

20 August 2019

# Severstal: Steel Defect Detection – Kaggle

This document is a proposal for the machine learning capstone project. It will define the proposed project problem, data, proposed solutions, and refences of external materials used. The project is based on a competition posted in Kaggle named “Severstal: Steel Defect Detection” [1].

## Domain Background

The problem falls in the domain of semantic segmentation of image data which is a subfield of computer vision. It requires classifying each pixel in the input image to a set of two or more classes. This filed of research has witnessed notable progress over the past four to five years with fully convolutional network architectures which employ an encoder-decoder structure such as U-Net [2] and LinkNet [3].

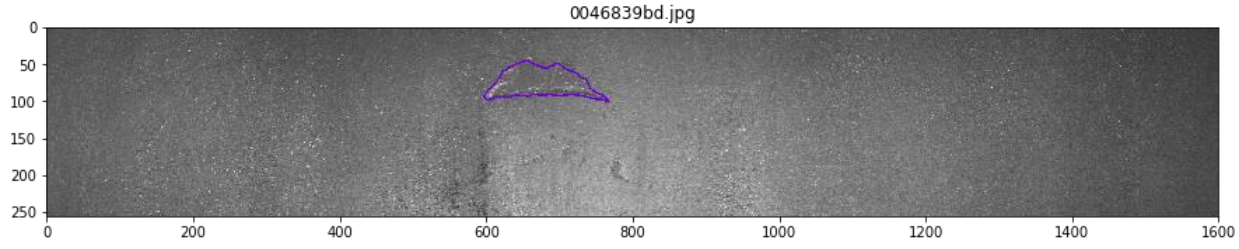
## Problem Statement

The goal is to detect surface defects in images of flat steel sheets on the production line by labeling the specific pixels representing defects classified into four types.

## Datasets And Inputs

The dataset is generated and provided by Severstal as part of the aforementioned Kaggle competition and consists of 12,568 labeled images. The data is labeled in a separate csv file using run-length encoding and referring to each image and class of defect by name.

About 53% of the training images have at least one defect and the vast majority of these only have one defect. Below is one example image from the dataset with the boundary of one defect type shown in purple.



## Solution Statement

Implement, tune, and test fully convolutional network architectures such as U-net and LinkNet in the following two configurations:

- Two steps: One standard convolutional network which performs binary classification (defect or no defect) followed by a fully convolutional network trained only on images with defects which performs segmentation. During inference, if the first model predicts no defects, the image is not passed to the second model
- A single step: one fully convolutional network which is trained on the entire dataset.

## Benchmark Model

U-net [2] is the most famous semantic segmentation model which can be used as a benchmark. It uses an encoder-decoder architecture where the encoder is a series of convolution blocks and the decoder uses up-convolution blocks to up-sample the image back to its original size and it also features cross connections between corresponding blocks in the encoder and the decoder where the encoder blocks are concatenated with the decoder blocks.

## Evaluation Metrics

The dice coefficient is the metric to be used to evaluate the performance of the model. It is the most widely used in semantic segmentation models and it takes into account false positives and false negatives. The loss-function of the model will incorporate the dice coefficient to improve performance.

## Project Design

The preliminary work-flow to achieve this project is as follows:

- Visualize and explore the dataset (will use a modified version of open-source code) [4]
- Preprocess the csv file to create a dataframe suitable for generating a training set
- Create functions which perform run-length encoding and decoding (will use slightly modified version of open source code) [5]
- Generate training and validation sets (will use modified version of open source code) [6]
- Create and train model architectures in Keras
- Perform inference and evaluate the results on the validation set
- Tune the model's architecture and hyperparameters and retrain it
- Perform evaluation on the test set and submit the results and the inference kernel to the Kaggle competition

## References

[1] Kaggle competition: https://www.kaggle.com/c/severstal-steel-defect-detection/overview

[2] U-Net: Convolutional Networks for Biomedical Image Segmentation: https://arxiv.org/abs/1505.04597

[3] LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation: https://arxiv.org/abs/1707.03718

[4] Clear mask visualization and simple: https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

[5] RLE functions – Run-Length Encode & Decode: https://www.kaggle.com/paulorzp/rle-functions-run-lenght-encode-decode

[6] Simple Keras U-Net Boilerplate: https://www.kaggle.com/xhlulu/severstal-simple-keras-u-net-boilerplate