INTP22-ML-4

STEEL DEFECT DETECTION USING COMPUTER VISION

PHASE-1 REPORT

Submitted by:

PAVAN KUMAR M

Under the mentorship of Devesh Tarasia

PHASE-1 OBJECTIVE

1. **Understanding the given Dataset**
2. **Performing Exploratory Data analysis**
3. **Drawing Conclusion from the performed EDA**
4. **Image Processing and preparing image for Training**
5. **Based on the conclusion deciding the model to be used in Phase-2**

INTRODUCTION

In the process of industrial production, the detection of steel surface defect possesses a vital effect in ensuring the industrial products quality in market. The major reason is that industrial parts are widely utilized in the sequence of industries for example machinery, aerospace, electronics and automobile and so on, and they are the basic parts of various industrial manufacturing products. In addition, in the fierce competition of modern industrial products, the industrial parts quality is directly associated with the products' ultimate quality. In the automatic and mechanical processing engineering, the features of material itself, improper polishing process, tool damage and vibration, together with the change of tool path may lead to scratch, dent and deformation on the surface of machined parts. These defects can result in problems, for example, poor reflection characteristics and damage. The surface defects for the industrial parts will make the workpiece appearance ugly, at the same time influence the property of the workpiece.

Traditionally, the control of surface quality is conducted manually, and workers are trained in order to identify the complicated surface defects. Nevertheless, this kind of control is inefficient and time-consuming, and its accuracy of detection is affected by the experience, energy and subjectivity of inspectors. With the aim of overcoming the shortcomings of manual inspection, the automatic detection of surface defects on the basis of machine vision came into being. In the last decade, many approaches [1, 2, 3, 4] have been utilized for automatic detection of surface defects on steel. The major principle is to utilize the shape or pixel values of steel surface to predict defects; nevertheless, it is time-consuming and complex to set threshold and obtain feature parameters.

Therefore, the aim of this project is to implement and deploying deep learning model which helps in identifying defects in steel.

UNDERSTANDING THE DATASET

The dataset consists of 4 folders, namely:

- Train_images – Images that are used to train the model
- Test_images – Images that are used to test the trained model before deploying
- Train.csv – comma Separated Values contains the images with defects and the encoded pixels with defect class type
- Sample_submission.csv – Contains the test images name which is used once the model is trained and ready for prediction. The images are scanned and corresponding pixels with defects and Class type is updated into this.

With this knowledge, We can proceed to performing Exploratory Data analysis and image Processing.

EXPLORATORY DATA ANALYSIS

The dataset used contains high-frequency images of steel plates, corresponding defect classes, and defect regions. The size of the image provided is 1600 × 256, and the total number of training images is 12568. Among all images, there are 5902 images which have defects and 6666 distinct images. There exist four classes of defects. Fig.1 demonstrates that there are a total of 7,095 labeled mask instances consisting 897 class 1 defects, 247 class 2 defects, 5150 class 3 defects, and 801 class 4 defects. Therefore, the dataset is very imbalanced. Data augmentation and resampling techniques will be required to perform the defect detection. Moreover, there are multi-class defects in the training set. From Fig. 1.1, it seems like the combinations of two labels in a single image are reasonably frequent and even a combination of three labels exists in the training set. In fact, classes 3 and 4 appear together more often than 2 does on its own. The sample images are shown in Fig. 1.2.

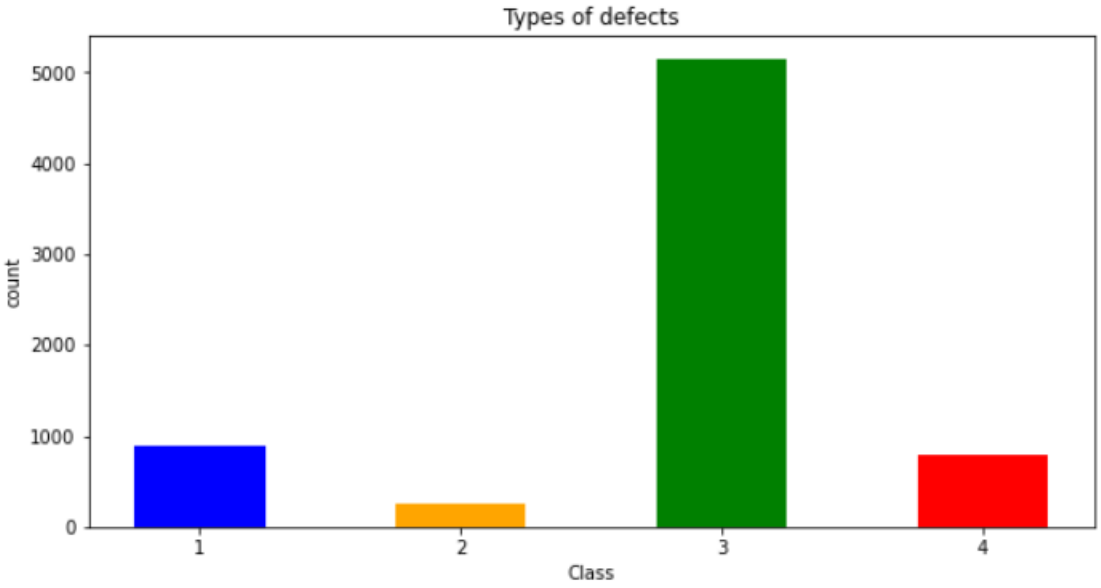

Fig 1: Number and frequency of defect classes.

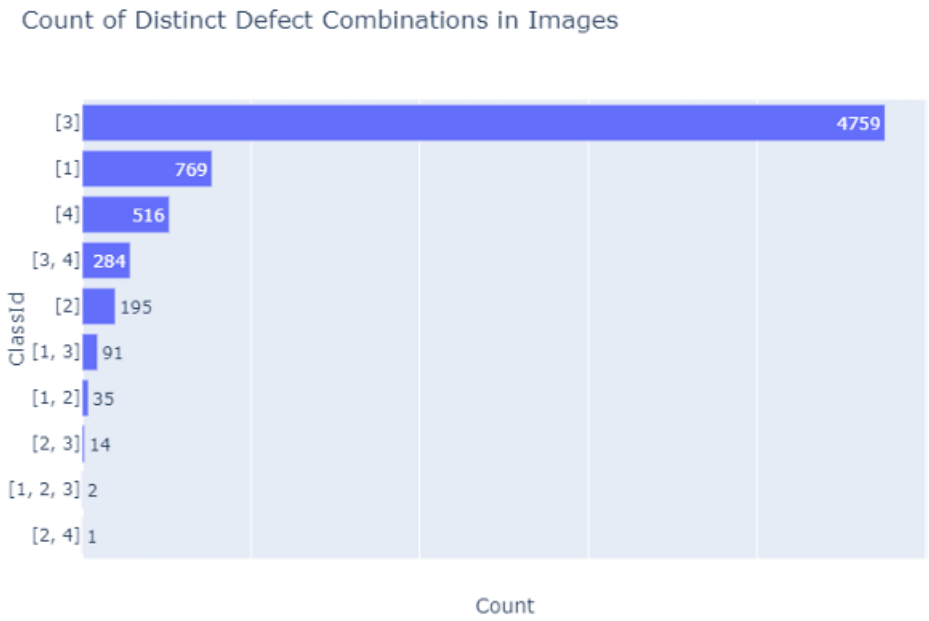

Fig 1.1: Count of Distinct Defect Combinations in Images.

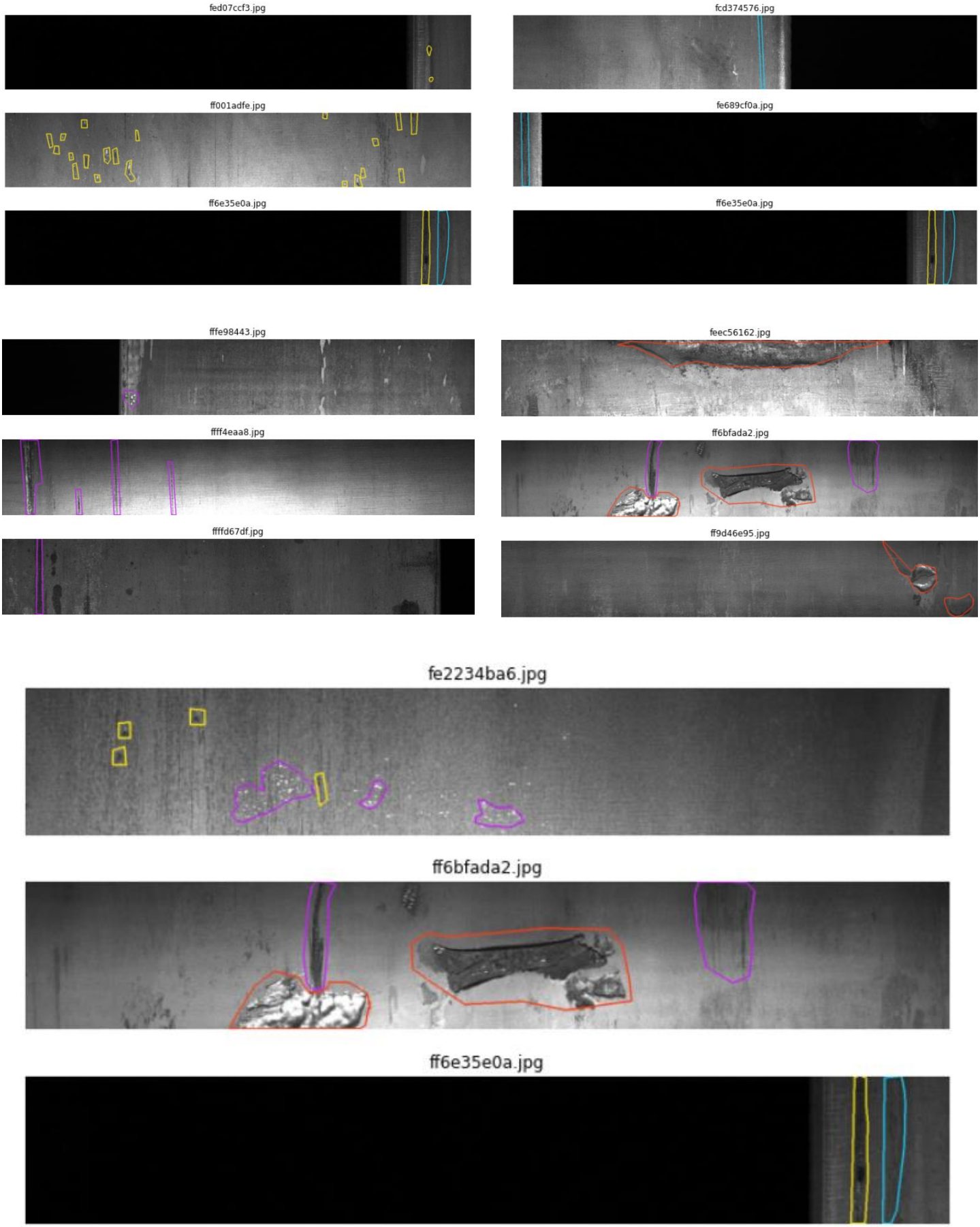

Fig 1.2: Sample image with instance-level defects.

IMPLEMENTATION:

Several software artifacts were developed using the Python programming language. Python was the chosen language for this project due to its simplicity, taking also into consideration the fact that it supports most of the recent deep learning frameworks. Keras and Pytorch are used for completing this project. The software modules that were mainly used in this project were responsible for data preprocessing, and training neural networks. The whole implementation process is demonstrated in the Fig.2.

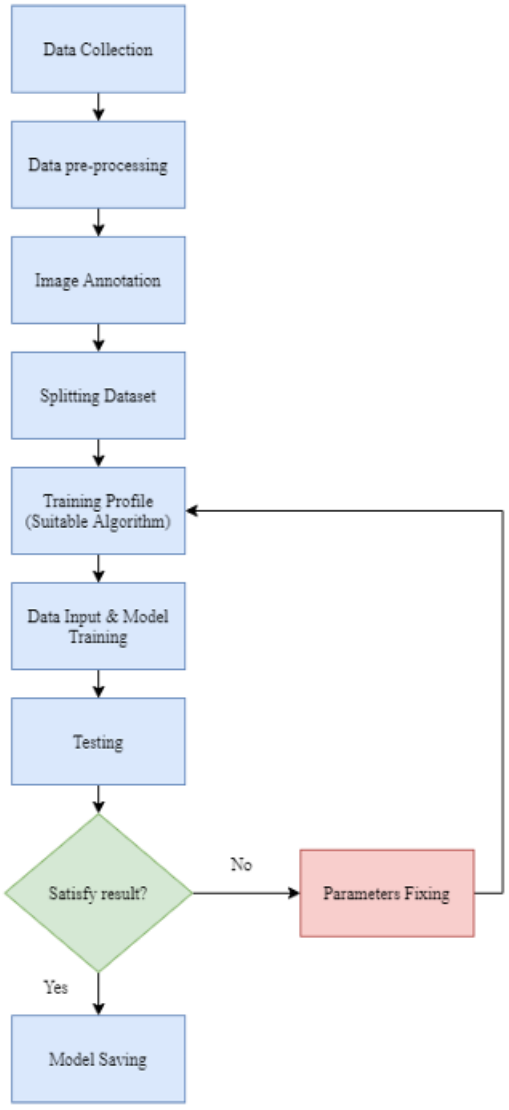

Fig 2: Implementation Process.

DATA PREPROCESSING

During the data preprocessing, the data is resized to the 128 × 800 × 3 for ResUnet due to the decoder channel. Also, the data is flipped horizontally and vertically. In the case of ResUnet, when trained on each of them separately, the augmented images were transformed for the assessment of the properties of the model.

SPLITTING DATASET

Based on generally accepted practice 20% of the original dataset was held out as a testing set, the remaining 80% was further divided by the ratio 80:20 into a training, respectively, validation set, resulting in three separated subsets of each original dataset.

SEGMENTATION MASK

A mask branch is a convolutional network which employs the positive regions chose via the RoI classifier and created masks for them. The acquiring mask resolution is very low, i.e. 28x28 pixels. Nevertheless, they are soft masks that are expressed as floating-point numbers, thus in contrast to binary masks, they have more detail. Smaller sizes of mask help maintain mask branches lighter. In the process of training, the ground-truth masks are reduced to 28x28 for the calculation of the loss. In the process of inference, the predicted masks are amplified to the RoI bounding box size, thereby offering us with the ultimate mask, one for each object.

MAP VISUALIZATION

By visualizing the map, we can more intuitively realize what the neural network has learned. We can able to visualize feature map from different layers of ResNet according to the same input image. Regarding RGB images, the darker of color regions are in the image, the more the neural network will pay attention to these regions (Darker from blue to red). In other word, when neural network is learning, it extracts more features from red regions.

Algorithm to convert the Annotations:

- Color palette is used to distinguish between the different class types.
- In the dataset EncodedPixel is provided which is nothing but the RLE (run length Encoding).
- As a first step is to convert RLE to mask and can be done using utility function.
- In order to convert annotations into contours or masks there are two approaches for generating these data:
  - **i)** Generate contours or an object segmentation mask image from a region defined by RLE.
  - **ii)** Generate Contours or object segmentation mask images from annotations contained within region-of-interest (ROI) annotations.
- Hence in next step mask is converted into contour.
- Third stage is to enlarge a mask
- The second and third together put blank space between defect and mask contour for better visualization.