FSM Online Internship Completion Report
on

STEEL DEFECT DETECTION USING COMPUTER VISION

In

Machine Learning

Submitted by

PAVAN KUMAR M
BMS College of Engineering

Under Mentorship of
Devesh Tarasia

IITD-AIA Foundation for Smart Manufacturing

[1-June-2021 to 31-July-2021]

Steel Defect Detection using Computer Vision

ABSTRACT

Automatic detection of steel surface defects is very important for product quality control in the steel industry. However, the traditional method cannot be well applied in the production line, because of its low accuracy and slow running speed. The current, popular algorithm (based on deep learning) also has the problem of low accuracy, and there is still a lot of room for improvement. This paper proposes a method combining improved ResNet50 and enhanced faster region convolutional neural networks (faster R-CNN) to reduce the average running time and improve the accuracy. Firstly, the image input into the improved ResNet50 model, which add the deformable revolution network (DCN) and improved cutout to classify the sample with defectsand without defects. If the probability of having a defect is less than 0.3, the algorithm directly outputs the sample without defects. Otherwise, the samples are further input into the improved faster R-CNN, which adds spatial pyramid pooling (SPP), enhanced feature pyramid networks (FPN), and matrix NMS. The final output is the location and classification of the defect in the sample or without defect in the sample. By analyzing the data set obtained in the real factory environment, the accuracy of this method can reach 98.2%. At the same time, the average running time is faster than other models.

Keywords: **Steel defect, Neural Networks, Model, ResNet50, images.**

CONTENTS

CHAPTER 1: 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.

CHAPTER 2: PROBLEM STATEMENT

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.

The suitability of deep learning models will be investigated for steel defect detection. The main challenge of this task is to establish a robust system with a limited number of samples (12568 images). In this project, we will analyze the performance of different neural network architectures and data augmentation strategies in order to classify and compare their performance when solving the main challenges previously exposed. To address the identified problem, the following research questions have to be answered:

RQ1: What deep learning models are currently prevailing in the image processing especially in segmentation part and how to implement them?

RQ2: Are there significant differences in the detection over different kinds of deep learning models?

RQ3: What are the limitations of UNET

RQ4: How to improve the model of UNET when it is not able to perform well and how does it perform?

CHAPTER 3: EXPLORATORY DATA ANALYSIS

3.1 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.

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.

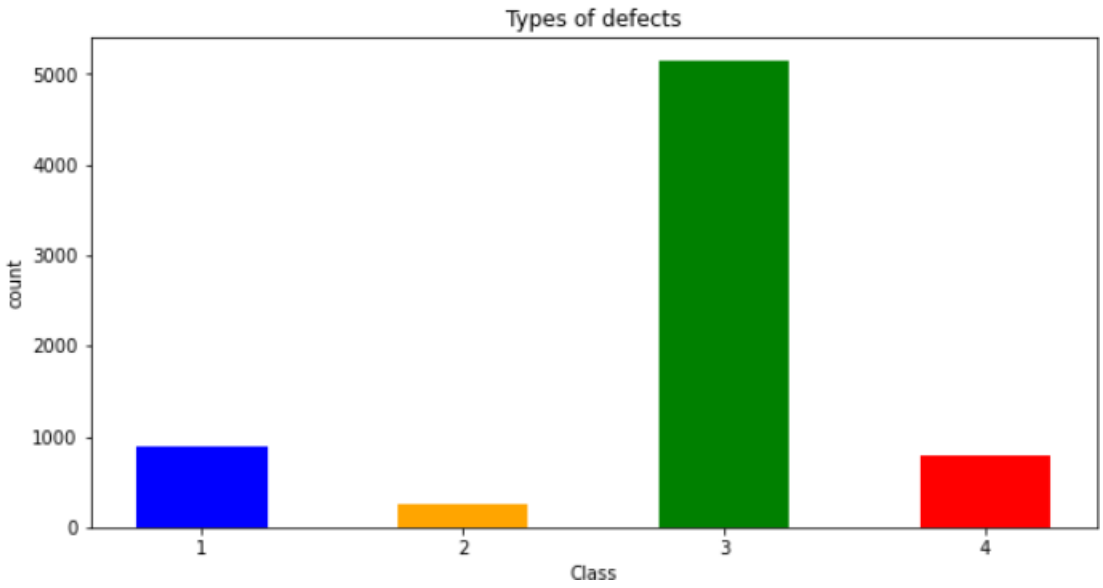


Fig 1: Number and frequency of defect classes.

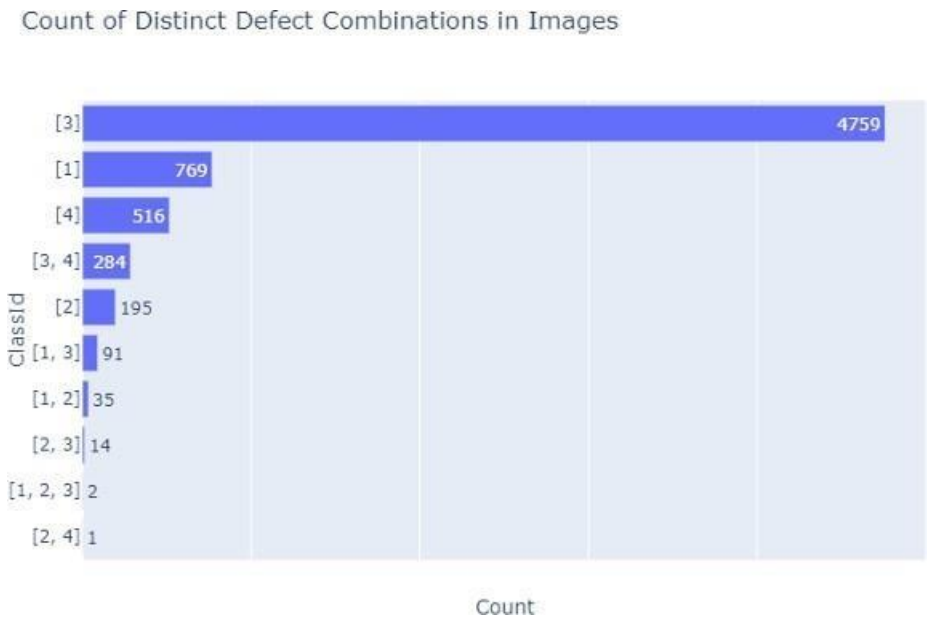


Fig 1.1: Count of Distinct Defect Combinations in Images.

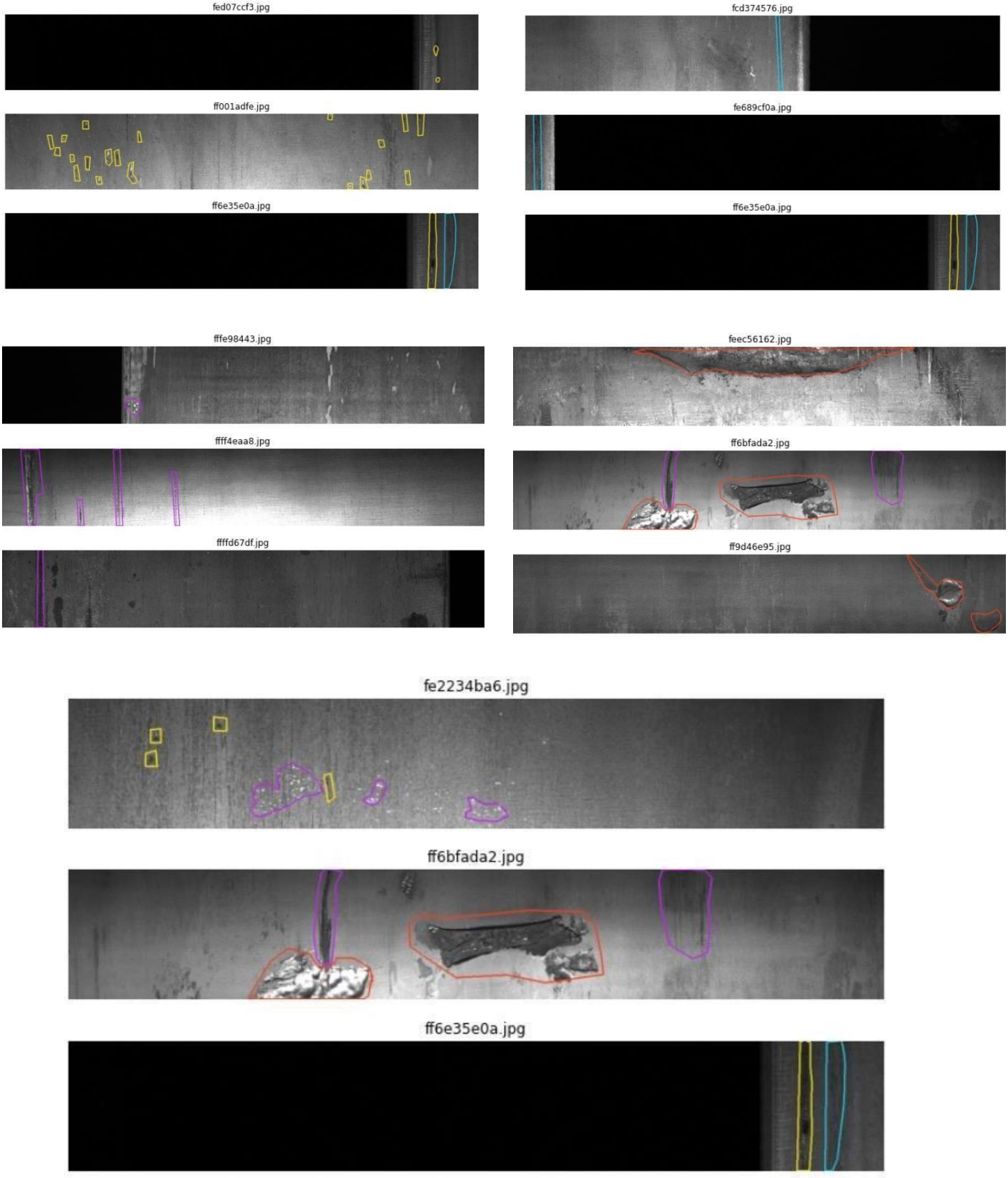


Fig 1.2: Sample image with instance-level defects.

CHPTER 4: 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.

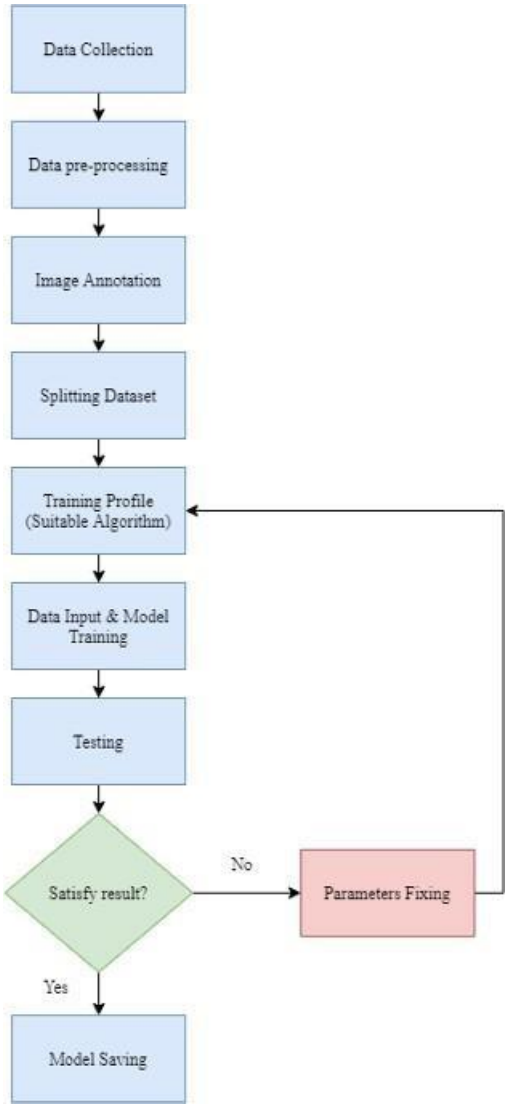


Fig 2: Implementation Process.

4.1 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.

4.2 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.

4.3 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.

4.4 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.

4.5 Algorithm to convert the Annotations:

4.5.1 Color palette is used to distinguish between the different class types.

4.5.2 In the dataset EncodedPixel is provided which is nothing but the RLE (run length Encoding).

4.5.3 As a first step is to convert RLE to mask and can be done using utility function.

4.5.4 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.

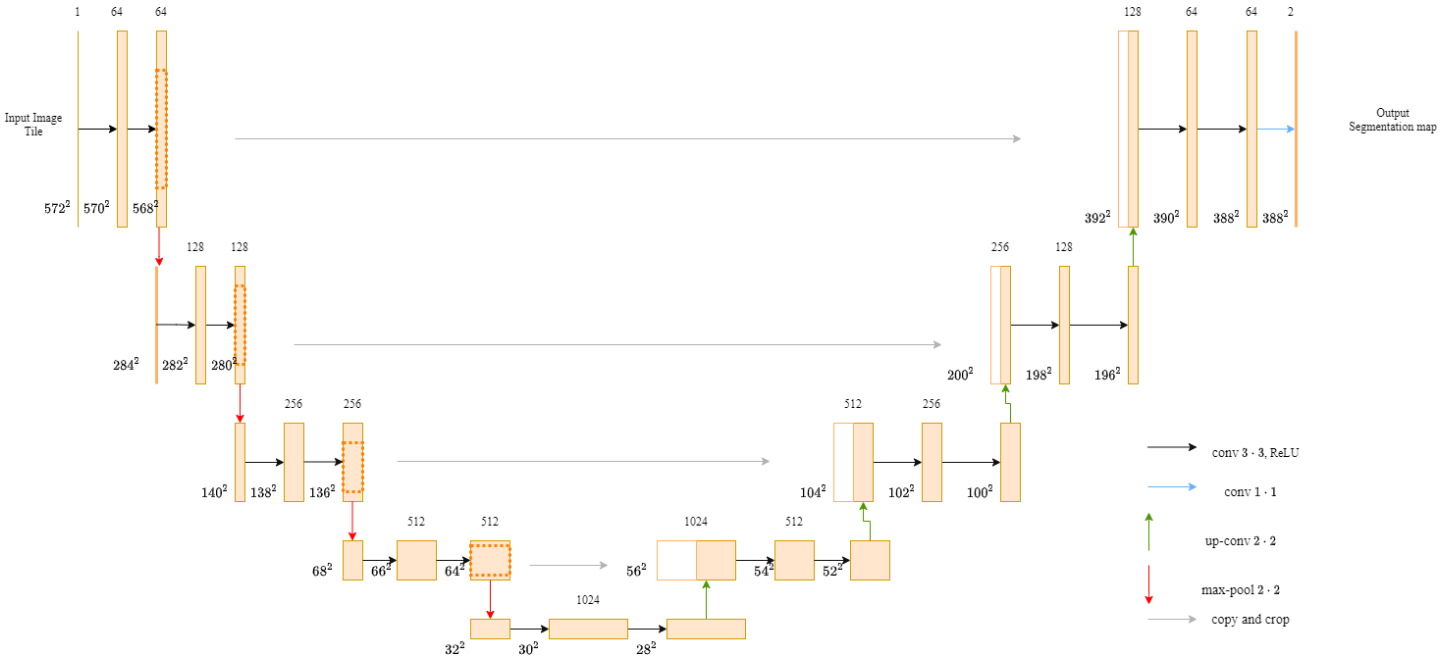
4.5.5 Hence in next step mask is converted into contour.

4.5.6 Third stage is to enlarge a mask

4.5.7 The second and third together put blank space between defect and mask contour for bettervisualization.

CHAPTER 5: MODEL SELECTION (UNET)

Unet is currently the most popular semantic segmentation model. It adopts the structure of the encoder and decoder. Encoder is simply a conventional stack of maximum pool layer and convolution layer, whereas the decoder utilizes the transposed convolution to achieve accurate positioning. As a result, it is the end-to-end convolutional network.



Unet Architecture

5.1 Model Implementation

In this part, according to the dataset and model used, some part need to be changed to fit the dataset.

ResUnet

Since ResUnet is composed of ResNet and Unet, the model can be divided into these two parts. The difference between ResNet 18 and ResNet 50 are the identity blocks. The former one is called basic block and the later one is called bottleneck. The implementation and parameters of the ResUnet model used are presented in the below listings:

Algorithm 3 ResNet architecture

```
procedure RESNETENCODER(resnet)
    layers ⟵ intended layers
    if resnet == 'resnet18' then                          ▷ block: BasicBlock
        layers ⟵ zero padding 3 × 3
        layers ⟵ convolution 7 × 7
        layers ⟵ batch normalization
        layers ⟵ ReLu
        layers ⟵ maximum pooling
        layers ⟵ conv block 64 × 64
        layers ⟵ 1 identity block 64 × 64
        layers ⟵ conv block 128 × 128
        layers ⟵ 1 identity block 128 × 128)
        layers ⟵ conv block 256 × 256
        layers ⟵ 1 identity block 256 × 256
        layers ⟵ conv block 512 × 512
        layers ⟵ 1 identity block 512 × 512
    else if resnet == 'resnet50' then                     ▷ block: Bottleneck
        layers ⟵ zero padding 3 × 3
        layers ⟵ convolution 7 × 7
        layers ⟵ batch normalization
        layers ⟵ ReLu
        layers ⟵ maximum pooling
        layers ⟵ conv block 64 × 64 × 256
        layers ⟵ 2 identity blocks 64 × 64 × 256
        layers ⟵ conv block 128 × 128 × 512
        layers ⟵ 3 identity blocks 128 × 128 × 512
        layers ⟵ conv block 256 × 256 × 1024
        layers ⟵ 5 identity blocks 256 × 256 × 1024
        layers ⟵ conv block 512 × 512 × 2048
        layers ⟵ 2 identity blocks 512 × 512 × 2048
    output layers
```

5.2 Algorithm Implementation

Dice Coefficient

The dice coefficient algorithm was also implemented in Python by using the mathematical formula as described, resulting in the below algorithm.

```
1: procedure DICE(pred, mask)
2:     batch_size ⟵ length(pred)
3:     change pred dimension
4:     change mask dimension
5:     pred ⟵ float elements in pred if elements>0.5
6:     mask ⟵ float elements in mask if elements>0.5
7:     let smooth ⟵ 0.0001
8:     intersection ⟵ sum(pred × mask)
9:     dice_pos ⟵ (2. × intersection + smooth) / (sum(pred) + sum(mask) + smooth)
10:    intersection ⟵ sum(element if element in (pred + mask) == 0)
11:    dice_neg ⟵ (2. × intersection + smooth) / ( sum(element if element in pred ==
   0) + sum(element if element in mask == 0) + smooth)
12:    dice ⟵ (dice_pos + dice_neg) / 2.0
   output dice
```

5.3 Controlled Experiment with training data

After applying data preprocessing to increase the data availability for the training process, independent variables are set in this controlled experiment, the training data is transformed to horizontally flipped images. The following parameters are considered for this project.

- Batch size = 16
- Epochs = 30
- Learning rate = 10^{-3}

- After completing through the 30 Epochs it is found that the best values for dice_coef and val_loss is obtained as 0.6461 and 0.0145 respectively at 29th Epoch.
- So, the best weighted values are considered for testing the model.
- And training time took nearly 4 hours.

5.4 Testing

- In order to test the model the dataset is given with the sample_submission.csv file which consists of ImageId, the model will consider a bunch of images as a batch size and starts predicting the EncodedPixel which determines the defective region and also the model classifies the defect type of that particular image.
- And it is observed that model has identified the defective region along with the defect class type pretty accurately.

5.5 Comparison of Different Algorithm Performance

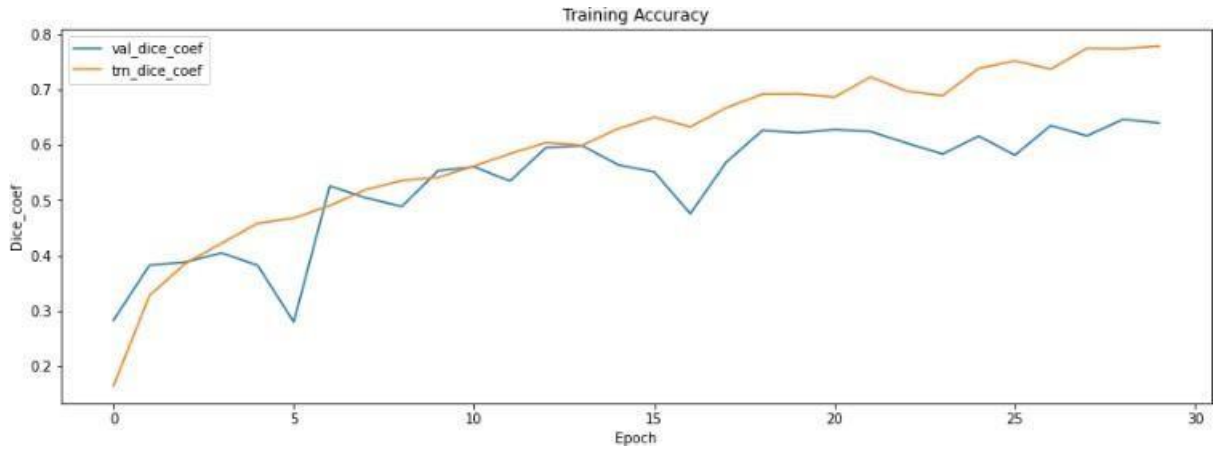
Table below shows the brief results of dice coefficient (best value achieved) of different models. It is obviously demonstrated that there is the highest dice coefficient score of improved mask R-CNN (0.81), but it takes the longest training time (12 hours), while the worst performance is Deeplab v3 plus.

	Basic Mask R-CNN	ResUnet	Deeplab v3 plus	Improved Mask R-CNN
Dice Coefficient	0.80891874	0.79928034	0.77426002	0.81071570
Training Time	around 9 hours	around 7 hours	around 7 hours	12 hours

CHAPTER 6: RESULTS

- **Unet**

The below figure reveals the training accuracy over Epochs that can be visualized using a histogram.



Conclusion

The main purpose of this project is to investigate the application of the deep learning approach to surface anomaly. The dataset is used for the research, including 5902 images with defects and 6666 distinct images. Amongst them, there exist four classes of defects. The models output the pixel-wise defected area with corresponding labels. One of the objective is aimed to explore the deep learning for image processing. After gaining the relevant knowledge including semantic segmentation and instance segmentation, we are capable to conduct the Unet model. Unet has performed well by considering the value of dice coefficient and training time compared to other models. Hence for semantic segmentation Unet is most popular. The research uses only a large dataset and it is imbalanced, which causes some problems

when training the models. More extensive work could be done by training more datasets, where other samples may improve the model.

Future Work

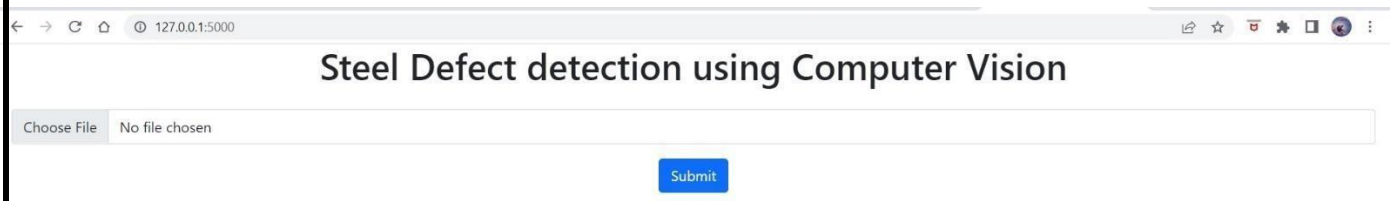
First of all, since it is an imbalanced dataset between various labels, some other kinds of approaches of data augmentation can be applied for the scarce anomaly classes on the input data, including but not limited to scaling, cropping, and increasing Gaussian noise. Furthermore, the new model also can be improved to get a much higher dice coefficient at least 90 by doing more deep experiments on the design part. On the other hand, efforts can be made to help steel industries to evaluate the severity of damage to each type of anomaly on the steel plate in order to determine whether to dispose of this steel plate via the information of the severity.

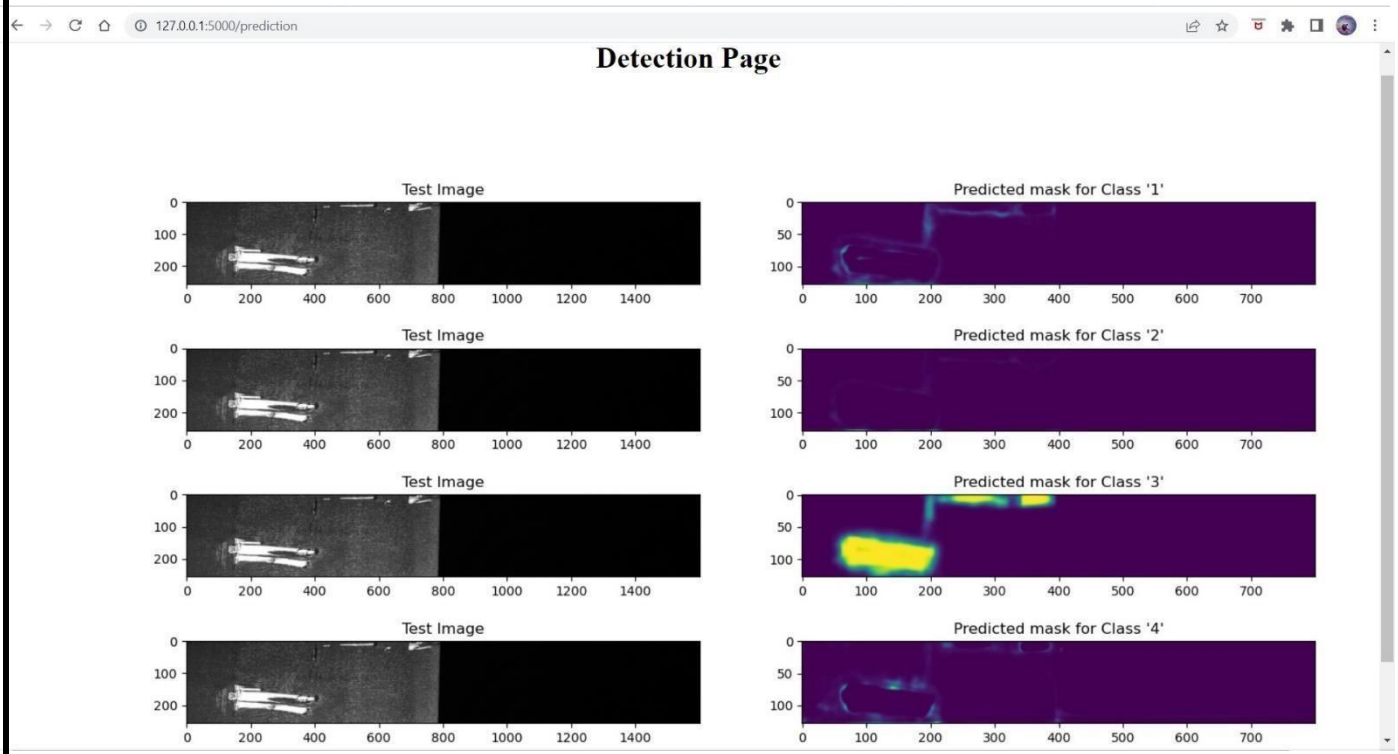
DEPLOYMENT

For deployment there were many Frameworks that are available but I used Flask framework which I felt beginner friendly.

Steps followed in Deployment:

- Firstly, I created a Virtual Environment specifically for deploying the ML model that was developed in stage 2.
- Then I installed the Packages (Libraries) required to run this Model.
- After installing all the necessary packages, started importing all of them in Spyder IDE.
- Then I loaded the saved model onto Spyder IDE by passing the Custom Objects and defining the Custom object function.
- Then before actually deploying I just tested whether HTML pages are working as intended, for this I first tried with printing Hello World in HTML page and got Positive result.
- Then started coding for deploying the original Model, getting Predictions out of it and displaying it to the user.
- I created 2 simple HTML web page one to take user input i.e. image of steel Image which may or may not have defect. Another to show the Prediction to the user.
- So, once the user uploads the image in HTML page and clicks submit it is passed onto the model. And model predicts whether there is a defect or not. If defect is detected then the model identifies the Defect region and also classifies it to the class it belongs to.
- And this is what expected at the end of this project.


127.0.0.1:5000
Steel Defect detection using Computer Vision
Choose File
No file chosen
Submit


127.0.0.1:5000/prediction
Detection Page
Test Image
Predicted mask for Class '1'
Test Image
Predicted mask for Class '2'
Test Image
Predicted mask for Class '3'
Test Image
Predicted mask for Class '4'

References:

[1] F. H. Xiaojie Duan, Fajie Duan, “Study on Surface Defect Vision Detection System for Steel Plate Based on Virtual Instrument Technology,” in 2011 International Conference on Control, Automation and Systems Engineering (CASE). IEEE, 2011, p. 1–4. [Online]. Available: https://ieeexplore-ieee-org.proxy.lnu.se/document/5997625

[2] N. N. Gagan Kishore Nand, Noopur, “Defect detection of steel surface using entropy segmentation,” in 2014 Annual IEEE India Conference (INDICON). IEEE, 2014, p. 1–6. [Online]. Available: https://ieeexplore-ieee-org.proxy.lnu.se/ document/7030439

[3] W. L. Z. Yichi and L. Xuedong, “Defects Detection of Cold-roll Steel Surface Based on MATLAB,” in 2011 Third International Conference on Measuring Technology and Mechatronics Automation. IEEE, 2011, pp. 827–830 [Online]. Available: https://ieeexplore-ieee-org.proxy.lnu.se/document/5720911