A

Project report on

Image classification & segmentation using Deep learning for Industrial purposes

by

Vikas Bhadoria (vikasbadhoria69@gmail.com)

Department of Informatics engineering

Wismar University of Applied Sciences: Technology, Business and Design

Wismar, Germany

ABSTRACT

This project focuses on image classification & segmentation using Deep Learning techniques. Artificial Intelligence and Machine Learning are transforming the manufacturing industry. According to the report released by the World Economic Forum, these technologies will play significant roles in the fourth industrial revolution. Major areas which can be benefited from this are:

- Maintenance Department
- Production Department
- Supply Chain Department

In this work I have successfully used 'Transfer Learning' technique where I used the already pre trained Microsoft's ResNet convolutional neural model to classify the image as defect or normal. The results for this were encouraging and an understanding was developed as to how powerful convolutional neural networks can be when it comes to image classification. I then used the ResUNet model(a deep learning framework for semantic segmentation) to localize defects using imagery data that could significantly improve the production efficiency in the manufacturing industry. I carried out training of the model on a huge dataset and the final predictions made by the model were very encouraging if not 100 percent perfect.

This project report consists of 5 chapters. Chapter 1 introduces the image segmentation and its relevance. Chapter 2 presents the problem statement and the objectives of this project. Chapter 3 is devoted to the related literature survey. Chapter 4 of this report discusses theory and methodology that I have followed. Chapter 5 highlights phase by outcome of the project and concludes the report.

Keywords: Deep learning, Convolutional Neural Networks, Transfer Learning, ResNet, Image segmentation, Image classification, ResUNet.

TABLE OF CONTENTS

1. Chapter 1: Introduction
2. Chapter 2: Problem statement & objectives
3. Chapter 3: Literature survey
    - What is semantic segmentation?
    - Applications
    - Convolutional neural networks for segmentation
    - Skip connections
    - Transfer learning
    - Loss function
    - Choosing the model
4. Chapter 4: Theory & methodology
    - Theory
    - Dataset
    - Methodology
    - Project status
5. Chapter 5: Conclusions

Chapter 1: Introduction

Nowadays, semantic segmentation is one of the key problems in the field of computer vision. Looking at the big picture, semantic segmentation is one of the high-level tasks that paves the way towards complete scene understanding. The importance of scene understanding as a core computer vision problem is highlighted by the fact that an increasing number of applications nourish from inferring knowledge from imagery. Some of those applications include self-driving vehicles, human-computer interaction, virtual reality etc. With the popularity of deep learning in recent years, many semantic segmentation problems are being tackled using deep architectures, most often Convolutional Neural Nets, which surpass other approaches by a large margin in terms of accuracy and efficiency.

In this project, I have discussed how to use deep convolutional neural networks to do image classification & segmentation. The project is practical based and works on classification & segmentation of the industrial images from a steel manufacturing company. The project is developed in Python using the deep neural framework of Keras & Tensorflow and is divided into four phases.

- Phase 1 is about exploring & visualization of the dataset .
- Phase 2 Using transfer learning to deploy Microsoft's ResNet model to observe the power of CNN for image classification. Here the model will classify which images are normal & which are with defects.
- Phase 3 is about image segmentation using the ResUNet model. Where I will try to localize the defects in a defective image.
- Phase 4 describes how good the model's predictions are.

Chapter 2: Problem Statement and Objectives

To leverage the power of Deep learning and build a model that can be used for classifying & detecting defects in industrial images.

The objectives of this project are:

1. To use Transfer Learning technique and deploy 'ResNet' deep learning architecture on the dataset to classify which images are having defects and which are normal.

2. Once the images with defects are classified with high accuracy the second step is to localize the defects. That means I will be trying to point out where exactly the defects in an image are.

3. The final goal is to test the model on the images which it has never seen before. By achieving a good accuracy on test data, this model can be deployed in real world industrial applications.

Chapter 3: Literature Survey

Pixel-wise image segmentation is a well-studied problem in computer vision. The task of semantic image segmentation is to classify each pixel in the image. Deep learning and convolutional neural networks (CNN) have been extremely ubiquitous in the field of computer vision. CNNs are popular for several computer vision tasks such as Image Classification, Object Detection, Image Generation, etc. Like for all other computer vision tasks, deep learning has surpassed other approaches for image segmentation.

What is semantic segmentation?

Semantic image segmentation is the task of classifying each pixel in an image from a predefined set of classes. In the following example, different entities are classified.

Fig1. Example of image segmentation.

In the above example, the pixels belonging to the bed are classified in the class “bed”, the pixels corresponding to the walls are labeled as “wall”, etc.

In particular, our goal is to take an image of size W x H x 3 and generate a W x H matrix containing the predicted class ID’s corresponding to all the pixels.

Input

Semantic Labels

Usually, in an image with various entities, we want to know which pixel belongs to which entity, For example in an outdoor image, we can segment the sky, ground, trees, people, etc.

Semantic segmentation is different from object detection as it does not predict any bounding boxes around the objects. We do not distinguish between different instances of the same object. For example, there could be multiple cars in the scene and all of them would have the same label.

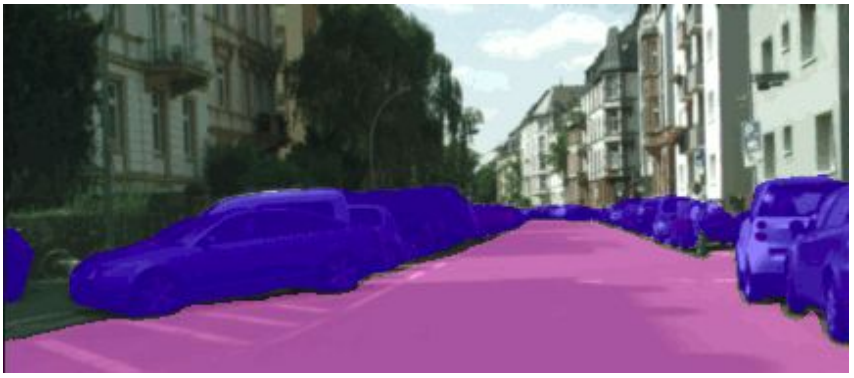


Fig3. Example of image segmentation.

In order to perform semantic segmentation, a higher level understanding of the image is required. The algorithm should figure out the objects present and also the pixels which correspond to the object. Semantic segmentation is one of the essential tasks for complete scene understanding.

Applications

There are several applications for which semantic segmentation is very useful.

Medical images

Automated segmentation of body scans can help doctors to perform diagnostic tests. For example, models can be trained to segment tumors.

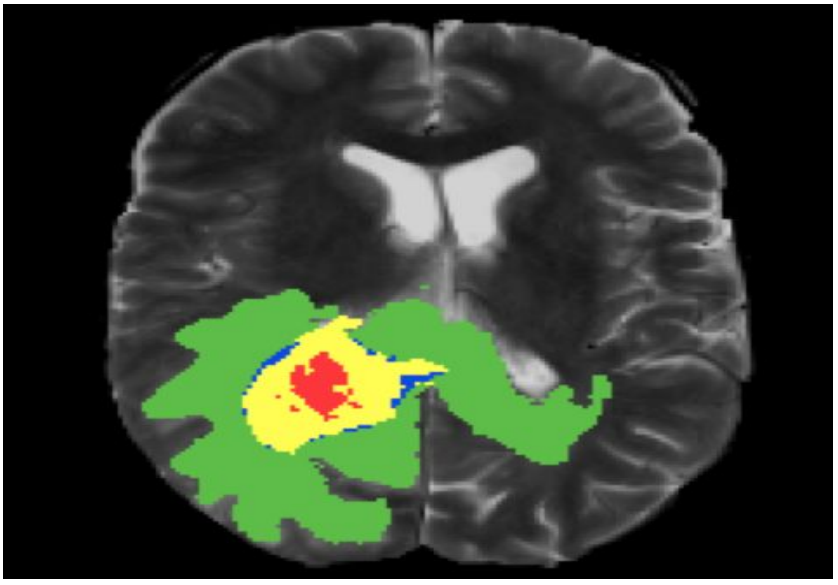


Fig 4. Brain MRI segmentation.

Autonomous vehicles

Autonomous vehicles such as self-driving cars and drones can benefit from automated segmentation. For example, self-driving cars can detect drivable regions.

Fig 5.Image segmentation for self driving cars.

Satellite image analysis

Aerial images can be used to segment different types of land. Automated land mapping can also be done.

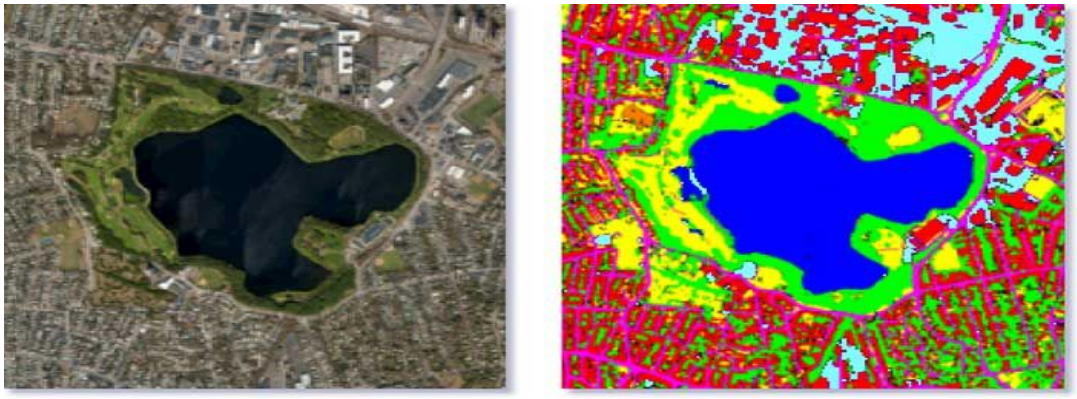


Fig 6. Image segmentation for satellite images.

Image segmentation using deep learning

Like most of the other applications, using a CNN for semantic segmentation is the obvious choice. When using a CNN for semantic segmentation, the output is also an image rather than a fixed length vector.

Convolutional neural networks for segmentation

Usually, the architecture of the model contains several convolutional layers, non-linear activations, batch normalization, and pooling layers. The initial layers learn the low-level concepts such as edges and colors and the later level layers learn the higher level concepts such as different objects.

At a lower level, the neurons contain information for a small region of the image, whereas at a higher level the neurons contain information for a large region of the image. Thus, as we add more layers, the size of the image keeps on decreasing and the number of channels keeps on increasing. The downsampling is done by the pooling layers.

For the case of image classification, we need to map the spatial tensor from the convolution layers to a fixed length vector. To do that, fully connected layers are used, which destroy all the spatial information.

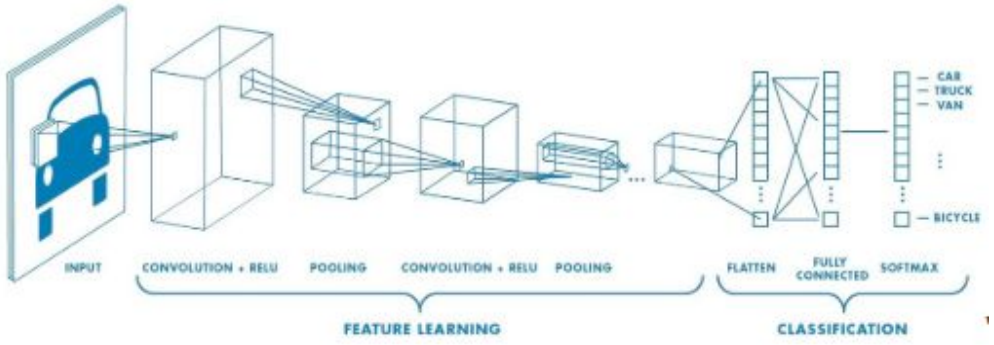


Fig 7. Convolutional Neural Network.

For the task of semantic segmentation, we need to retain the spatial information, hence no fully connected layers are used. That's why they are called fully convolutional networks. The convolutional layers coupled with downsampling layers produce a low-resolution tensor containing the high-level information.

Taking the low-resolution spatial tensor, which contains high-level information, we have to produce high-resolution segmentation outputs. To do that we add more convolution layers coupled with upsampling layers which increase the size of the spatial tensor. As we increase the resolution, we decrease the number of channels as we are getting back to the low-level information.

This is called an encoder-decoder structure. Where the layers which downsample the input are the part of the encoder and the layers which upsample are part of the decoder.

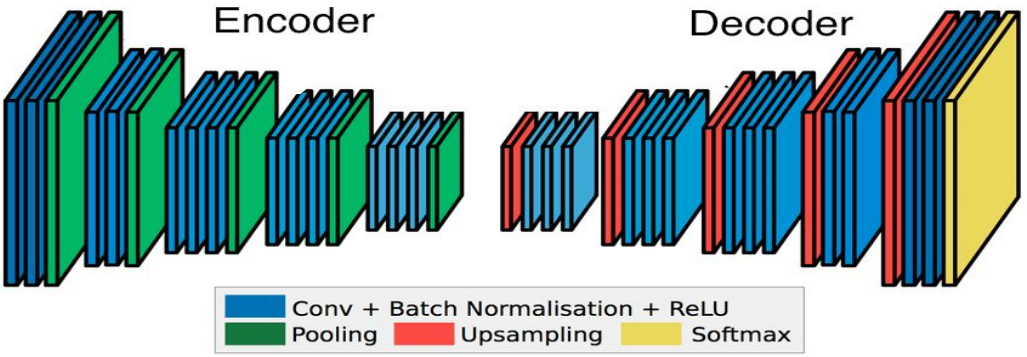


Fig 8. Image segmentation model architecture.

When the model is trained for the task of semantic segmentation, the encoder outputs a tensor containing information about the objects, and its shape and size. The decoder takes this information and produces the segmentation maps.

Skip connections

If we simply stack the encoder and decoder layers, there could be loss of low-level information. Hence, the boundaries in segmentation maps produced by the decoder could be inaccurate.

To make up for the information lost, we let the decoder access the low-level features produced by the encoder layers. That is accomplished by skip connections. Intermediate outputs of the encoder are added/concatenated with the inputs to the intermediate layers of the decoder at appropriate positions.

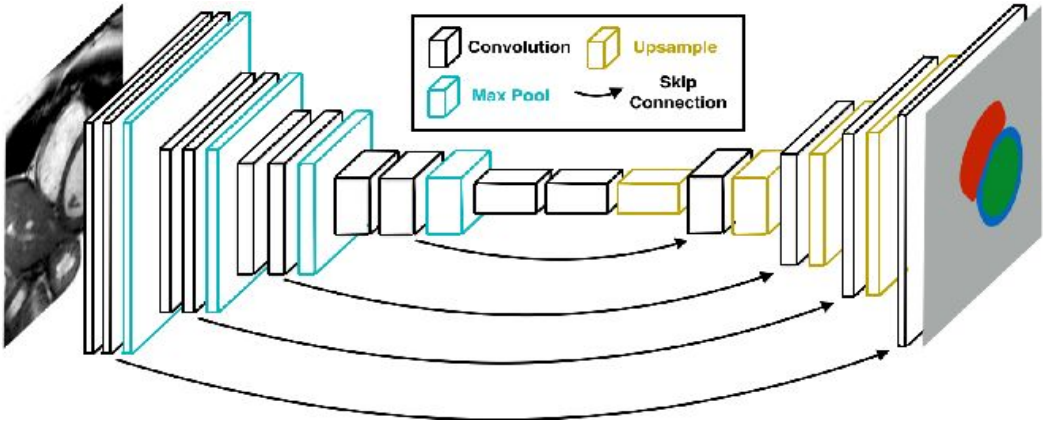


Fig 9. Skip connection architecture.

The skip connections from the earlier layers provide the necessary information to the decoder layers which is required for creating accurate boundaries.

Transfer learning

The CNN models trained for image classification contain meaningful information which can be used for segmentation as well. We can re-use the convolution layers of the pre-trained models in the encoder layers of the segmentation model. Using Resnet or VGG pre-trained on the ImageNet dataset is a popular choice.

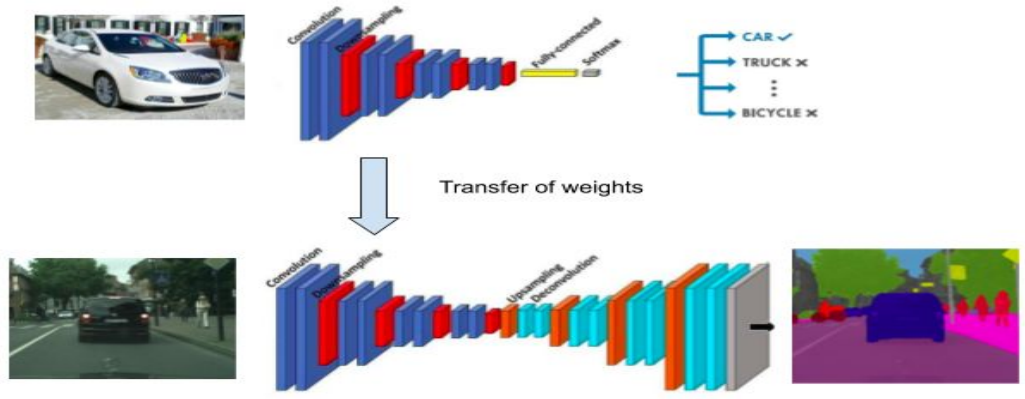


Fig 10. Transfer Learning process.

Loss function

There are two different tasks to accomplish in this project. The first is image classification and later segmentation. For task 1 I have used the 'binary cross entropy loss' function. But for task 2 I have used a custom loss function which is 'tversky loss function'.

Choosing the model

There are several models available for semantic segmentation. The model architecture shall be chosen properly depending on the use case. There are several things which should be taken into account:

1. The number of training images
2. Size of the images

3. The domain of the images

Usually, deep learning based segmentation models are built upon a base CNN network. A standard model such as ResNet, VGG or MobileNet is chosen for the base network usually. Some initial layers of the base network are used in the encoder, and the rest of the segmentation network is built on top of that. For most of the segmentation models, any base network can be used.

Choosing the base model

For selecting the segmentation model, our first task is to select an appropriate base network. For many applications, choosing a model pre-trained on ImageNet is the best choice.

ResNet: This is the model proposed by Microsoft which got 96.4% accuracy in the ImageNet 2016 competition. ResNet is used as a pre-trained model for several applications. ResNet has a large number of layers along with residual connections which make it's training feasible.

VGG-16: This is the model proposed by Oxford which got 92.7% accuracy in the ImageNet 2013 competition. Compared to Resnet it has fewer layers, hence it is much faster to train. For most of the existing segmentation benchmarks, VGG does not perform as well as ResNet in terms of accuracy. Before ResNet, VGG was the standard pre-trained model for a large number of applications.

MobileNet: This model is proposed by Google which is optimized for having a small model size and faster inference time. This is ideal to run on mobile phones and resource-constrained devices. Due to the small size, there could be a small hit in the accuracy of the model.

Custom CNN: Apart from using an ImageNet pre-trained model, a custom network can be used as a base network. If the segmentation application is fairly simple, ImageNet pre-training is not necessary. Another advantage of using a custom base model is that we can customize it according to the application.

If the domain of the images for the segmentation task is similar to ImageNet then ImageNet pre-trained models would be beneficial. For input images of indoor/ outdoor images having common objects like cars, animals, humans, etc ImageNet pre-training could be helpful. The pre-trained model can also be trained on other datasets depending on the type of input images for the task.

The classification model that I have selected for my project is the ResNet model as I have a big dataset. The accuracy of ResNet is high and I have worked with ResNet before.

Going ahead with the segmentation model. There are some popular models which can be discussed.

FCN : FCN is one of the first proposed models for end-to-end semantic segmentation. Here standard image classification models such as VGG and AlexNet are converted to fully convolutional by making FC layers 1x1 convolutions. At FCN, transposed convolutions are used to upsample, unlike other approaches where mathematical interpolations are used. The three variants are FCN8, FCN16 and FCN32. In FCN8 and FCN16, skip connections are used.

SegNet : The SegNet architecture adopts an encoder-decoder framework. The encoder and decoder layers are symmetrical to each other. The upsampling operation of the decoder layers use the max-pooling indices of the corresponding encoder layers. SegNet does not have any skip connections. Unlike FCN, no learnable parameters are used for upsampling.

UNet : The UNet architecture adopts an encoder-decoder framework with skip connections. Like SegNet, the encoder and decoder layers are symmetrical to each other.

PSPNet : The Pyramid Scene Parsing Network is optimized to learn better global context representation of a scene. First, the image is passed to the base network to get a feature map. The feature map is downsampled to different scales. Convolution is applied to the pooled feature maps. After that, all the feature maps are upsampled to a common scale and concatenated together. Finally another convolution layer is used to produce the final segmentation outputs. Here, the smaller objects are captured well by the features pooled to a high resolution, whereas the large objects are captured by the features pooled to a smaller size.

For images containing indoor and outdoor scenes, PSPNet is preferred, as the objects are often present in different sizes. Here the model input size should be fairly large, something around 500x500.

For the images in the medical domain, UNet is the popular choice. Due to the skip connections, UNet does not miss out the tiny details. UNet could also be useful for indoor/outdoor scenes with small size objects.

For a fairly large dataset like I have, I have preferred to use UNet architecture for this task.

Chapter 4: Theory and Methodology

Theory:

Images received from the dataset are separated into training, validation and testing dataset. The training dataset is used to train the model followed by validation dataset which is used to validate the model's performance.

In task one I have used the ResNet CNN model to classify if the images are having defects or not. All the images then detected with defects are used for image segmentation. The goal of image segmentation is to understand the image at the pixel level. It associates each pixel with a certain class. The output produced by the image segmentation model is called a "mask" of the image. Masks can be represented by associating pixel values with their coordinates. To represent this mask we have to first flatten the image into a 1-D array. This would result in something like [255,0,0,255] for the mask. Then, we can use the index to create the mask. Finally we would have something like [1,0,0,1] as our mask.

RUN LENGTH ENCODING (RLE):

Sometimes it is hard to represent a mask using index as it would make the length of mask equal to the product of height and width of the image. To overcome this we use lossless data compression technique called Run-length encoding (RLE), which stores sequences that contain many consecutive data elements as a single data value followed by the count. For example, assume we have an image (single row) containing plain black text on a solid white background. B represents black pixel and W represents white:

WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWWWWW
WWWWWWWWWWWWBWWWWWWWWWWWWW

- Run-length encoding (RLE):

12W1B12W3B24W1B14

this can be interpreted as a sequence of twelve Ws, one B, twelve Ws, three Bs, etc.

My basic motivation is to train my deep learning model in a way that it can classify between defective and normal images. In the second step it should take the defective images and point out the exact location of the defect on the image.

Dataset:

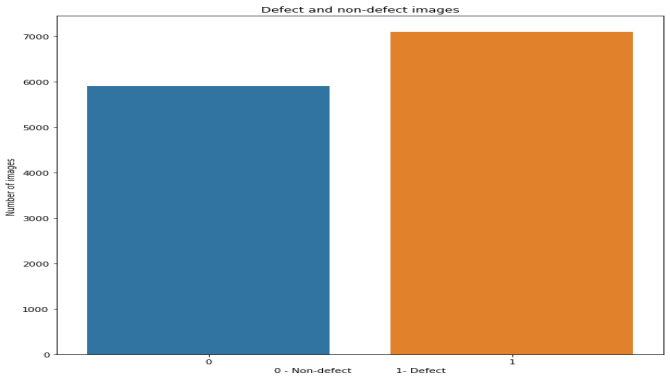
The dataset used is downloaded from kaggle which contains 12600 images that contain 4 types of defects, along with their location in the steel surface. The location is nothing but the mask with the exact location of the defect. The masks of images are encoded using Run-length encoding and for this project I will be using a helper function to convert RLE to a mask which is of the exact size of the image.

Methodology:

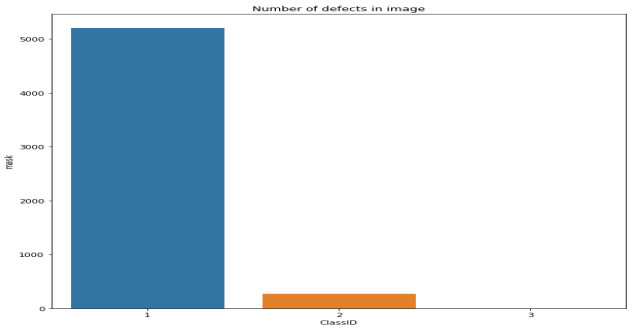
The project work was divided into the following 4 phases:

Phase 1: After looking for the dataset and finding the data it was really important to explore the dataset and visualize it. This is a crucial step in any project related to deep learning as it gives a lot of information about the data such as missing values, imbalance data, unique values and so on. I did some visualizations to explore the dataset.

1. Number of defective images(orange) v/s normal images(blue) in the dataset.

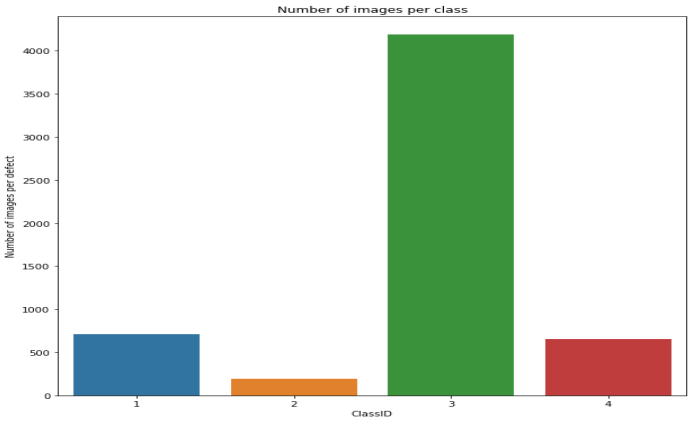


2. Number of defects in a single image(most of the images are with single defect)

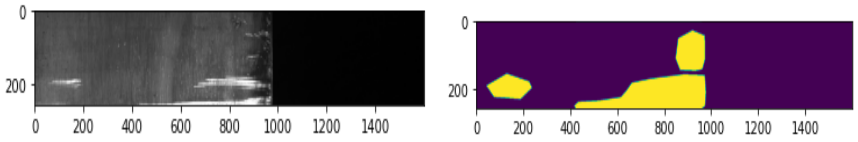


3. Number of images per type of defect.

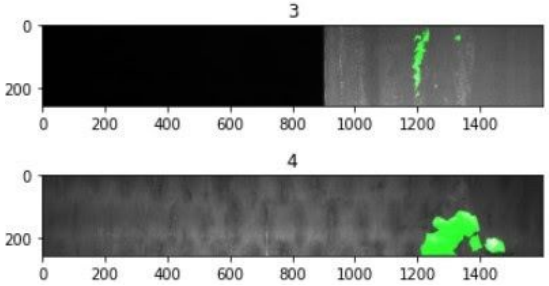
   Clearly the images with type 3 defect are high in number. This can cause an imbalance in the data.



4. A defective image and its corresponding mask indicating the exact location of the defect.



5. Two defective images of defect type 3 & 4 respectively, combined along with their respective masks.



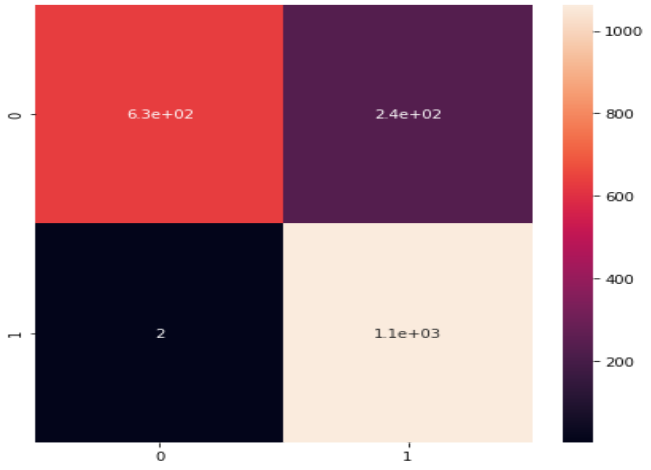
Phase 2: Once done with data exploration and visualization. It is now time to train the CNN model to classify which images are defective and which are normal. For this task the ResNet CNN model has been used. Residual Network (ResNet) is a Convolutional Neural Network (CNN) architecture which was designed to enable hundreds or thousands of convolutional

layers. While previous CNN architectures had a drop off in the effectiveness of additional layers, ResNet can add a large number of layers with strong performance.

ResNet was an innovative solution to the “vanishing gradient” problem. Neural networks train via the backpropagation process, which relies on gradient descent, moving down the loss function to find the weights that minimize it. If there are too many layers, repeated multiplication makes the gradient smaller and smaller, until it “disappears”, causing performance to saturate or even degrade with each additional layer.

The ResNet solution is “identity shortcut connections”. ResNet stacks up identity mappings, layers that initially don’t do anything, and skips over them, reusing the activations from previous layers. Skipping initially compresses the network into only a few layers, which enables faster learning. Then, when the network trains again, all layers are expanded and the “residual” parts of the network explore more and more of the feature space of the source image.

After training the model for 40 epochs an accuracy of 87% on the test images was achieved. Below is the confusion matrix of the result after testing the model on new images.



The classification report gave really good results for precision and recall.

	precision	recall	f1-score	support
0	1.00	0.73	0.84	872
1	0.82	1.00	0.90	1064
accuracy			0.88	1936
macro avg	0.91	0.86	0.87	1936
weighted avg	0.90	0.88	0.87	1936

Phase 3: After the classification has been done the first task is achieved. In this phase the focus is on task 2 which is 'image segmentation'. The defect in the defective images will be localized. For this task the model used is ResUNet. In phase 2 when I applied CNN for image classification I had to convert the image into a vector and add a classification head at the end. However, in the case of Unet, we convert (encode) the image into a vector followed by up sampling (decode) it back again into an image. In case of Unet, the input and output have the same size so the size of the image is preserved.

- For classical CNNs: they are generally used when the entire image is needed to be classified as a class label.
- For Unet: pixel level classification is performed. U-net formulates a loss function for every pixel in the input image.

Softmax function is applied to every pixel which makes the segmentation problem work as a classification problem where classification is performed on every pixel of the image.

Unet architecture is based on Fully Convolutional Networks and modified in a way that it performs well on segmentation tasks.

ResUNet consists of three parts:

1. Encoder or contracting path
2. Bottleneck
3. Decoder or expansive path

1. Encoder or contracting path consist of 4 blocks:

- First block consists of 3x3 convolution layer + Relu + Batch-Normalization
- Remaining three blocks consist of Res-blocks followed by Max-pooling 2x2.

2. Bottleneck:

- It is in-between the contracting and expanding path.
- It consists of a Res-block followed by up sampling conv layer 2x2.

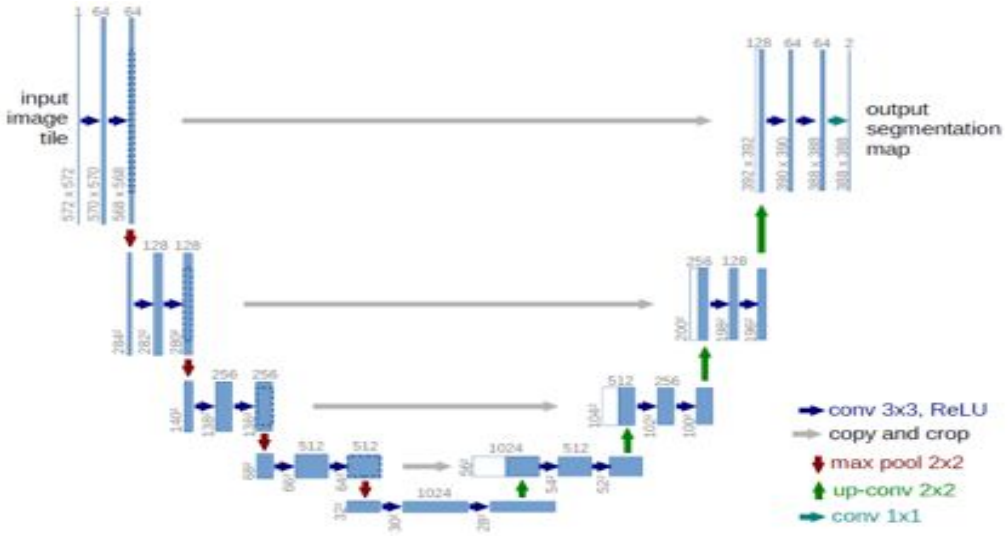


Fig 11. ResUNet architecture.

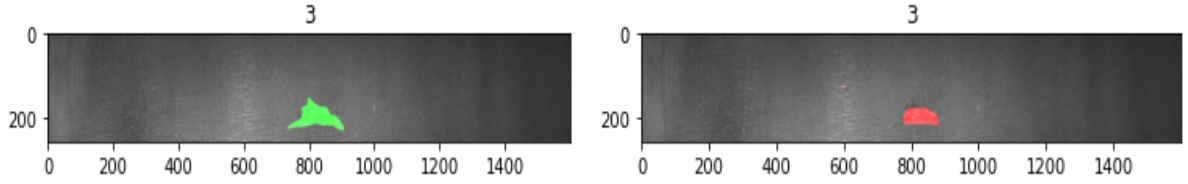
3. Expanding or Decoder path consist of 4 blocks:

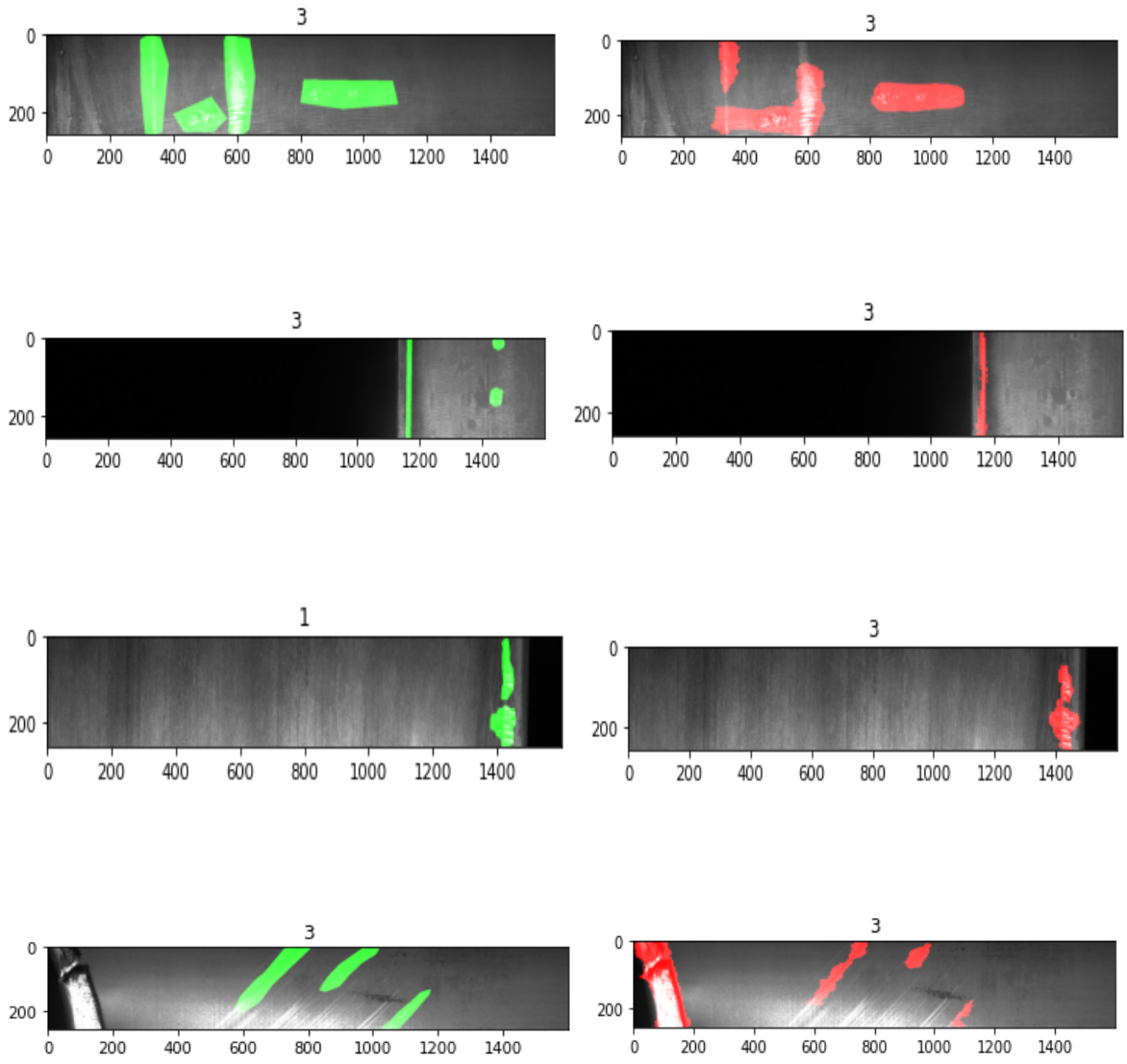
- 3 blocks following bottleneck consist of Res-blocks followed by up-sampling conv layer 2 x 2
- Final block consists of a Res-block followed by a 1x1 conv layer.

The ResUNet model has been trained on 40 epochs.

Phase 4: In this phase the model built and trained is put to test on images which it has never seen before. The test images are random images from the dataset. The task of the trained model is to classify these images as defective or normal and then take all the defective images and mark the defect using its predicted mask.

The prediction of the model can be seen below. On the left side are the real images(green) and on the right side are the images predicted(red) using my trained model.





In most of the cases the trained model is performing really good to localize the defect. But there are few errors as seen in the 4th prediction. The type of error predicted by the model is 3 but in reality its type 1. This can be due to the imbalance dataset which included defect type 3 images the most.

Project status:

Sr.no	Project Phase	Status	Problems
1	Phase 1	complete	--------
2	Phase 2	complete	--------
3	Phase 3	complete	--------
4	Phase 4	The predictions can be made more accurate.	The segmentation model is a little more biased to predict defect type 3 due to the imbalance dataset.

Chapter 5: Conclusions

In this project in order to meet the intended objectives I first used the dataset from kaggle. I did some exploratory data analysis to visualize the data. After getting an insight on the dataset, I used transfer learning technique and used the ReNet deep learning model to classify between normal and defected images. Once the images were classified with an accuracy of 87% I then used the state of art ResUNet architecture for image segmentation and predict the accurate mask that can locate the defect in an image. In the last part I created a pipeline for the entire project which can take in the input and classify the image as normal or defected and later the defected images are put as an input to the segmentation model which predicts the location of the defect.

The overall results can be concluded in 3 points:

1. The model works pretty good to classify the images as normal or defective.

2. The model is well accurate to detect the mask of defective image which locates the defects accurately.

3. The model is producing some inappropriate results in case of detecting the type of defect. As there are 4 types of defects in the dataset. The model is more biased to predict type 3 defects. This can be due to the imbalance dataset as images with type 3 defects are the most in number. With a well balanced dataset this issue can be solved. This can be achieved via image augmentation.