JACOBS UNIVERSITY

Study of image preprocessing steps on scrap metal classification using YOLOv5 algorithm

by

Chirag Vaghela
(Mr. No. 30004502)

a thesis for conferral of a Master of Science in Data Engineering

Prof. Dr. Stephen Kettemann
Name and title of thesis supervisor

Dr. Christoph Kirmse
Name and title of external reviewer

Date of Submission:

JACOBS UNIVERSITY

Statutory Declaration

Family Name, Given/First Name	Vaghela, Chirag
Matriculation number	30004502
What kind of thesis are you submitting: Bachelor-, Master- or PhD-Thesis	Master Thesis

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

……………………………………………………………………………………………………
Date, Signature

Abstract

As one of the leading companies in metal industry the SMS group provides solutions to help its customer recycle tons of scrap metals every day. The scrap metals arrive to a manufacturing plant in railway carriages. These scrap metals are sorted manually before it can be processed further to produce the end products. The manual sorting of scrap metals is time consuming, costly and error prone.

This thesis is dedicated to automating the task of classification of scrap metal transported to a sorting facility. The images of scrap metals in railway carriages are used for the classification purpose. The machine learning algorithm YOLOv5 was used to train on image dataset of the scrap metals. The thesis explains the working of YOLOv5 algorithms, the implementation of the scrap metal classification software and additional libraries used for that purpose.

The YOLOv5 algorithm produces promising results when images are of good quality, but the performance of the algorithm gets compromised if images have noise, low contrast, or unexpected shades of colors. Generally, the images taken in scrap yard environment are more prone to produce low quality images because of the dust and lightening conditions. Therefore, the project also focuses on preprocessing the images before training the algorithm on image dataset to improve the image quality.

The results showed that preprocessing operations such as histogram equalization, denoising improved the confidence score of the classification of scrap metals by 20%. The augmentation operations on the images further improved the results when performed on the preprocessed images. The confidence score as accuracy measure of more than 90% was observed for the detection of the scrap metals. The time required for inference of the images was also very low. Hence, the system can also be used on video feeds for real time classification of scrap metals.

Acknowledgments

I would like to acknowledge and give my sincere thanks to my thesis supervisors Dr. Christoph Kirmse and Prof. Dr. Stephan Kettemann without their guidance this work could not have been possible. The constant feedback and suggestions from Dr. Kirmse on each step during the development of this project always kept me on right track and made me learn from my mistakes. His expert opinions and support made this an inspiring experience for me.

I would also like to thank the Metallics Optimizers team members at SMS digital who despite their busy schedule helped me out whenever I found difficulty understanding a complex topic, installing a library or just needed the moral support. I would also like to thank Mr. Maximillian Christ specially for his expert insights and comments on my work progress in our meetings which always encouraged me to explore more than I targeted before starting of this thesis.

Finally, I would like to thank my family for their support, encouragement and having faith in me.

Table of Contents

A. List of Abbreviations

EAF	**E**lectric **A**rc **F**urnace
YOLOv5	**Y**ou **O**nly **L**ook **O**nce **v**ersion **5**
CVAT	**C**omputer **V**ision **A**nnotation **T**ool
GUI	**G**raphical **U**ser **I**nterface
EC2	**E**lastic **C**loud **C**omputing
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etwork
COCO	**C**ommon **O**bjects in **Co**ntext
CNN	**C**onvolutional **N**eural **N**etwork
VOC	**V**isual **O**bject **C**lasses
IOU	**I**ntersection **O**ver **U**nion
BBHE	**B**rightness preserving **Bi**-**H**istogram **E**qualization
QDHE	**Q**uadrant **D**ynamic **H**istogram **E**qualization
CLAHE	**C**ontrast **L**imited **H**istogram **E**qualization
GUI	**G**raphical **U**ser **I**nterface
CVAT	**C**omputer **V**ision **A**nnotation **T**ool
OpenCV	**Open** source **C**omputer **V**ision
GPU	**G**raphical **P**rocessing **U**nit
CPU	**C**entral **P**rocessing **U**nit
Amazon S3	**A**mazon **S**imple **S**torage **S**ervice
DVC	**D**ata **V**ersion **C**ontrol
Amazon EC2	**A**mazon **E**lastic **C**loud **C**omputing
PR curve	**P**recision **R**ecall **curve**
AUC	**A**rea **U**nder the **C**urve

B. List of Figures

C. List of Tables

1. Introduction

1.1 Working Environment

The large-scale enterprise SMS group has expertise in the metal industry and industrial processes. It is a multinational company that values sustainability and social-corporate responsibilities. Consequently, it emphasizes recycling of the metal products turned into scraps after consumption.

The SMS provides solutions for its customers who receives an enormous amount of steel scraps in railway carriages everyday. The received scraps must be checked for quality, sorted manually based on their composition, ensured not withholding any dangerous or unwanted goods, and properly documented to keep track of the received scraps.

Scrap-based Electric Arc Furnaces (EAFs) play a vital role in controlling carbon emissions during the steelmaking process. The sorted scraps based on their shape, size and chemical composition are sent to the EAFs for further processing. Each EAF operates on different temperatures and other parameters based on the type of scrap used in the EAF.

1.2 Motivation

At present, the steel scraps are sorted manually and sent to different EAFs with the help of electromagnetic scrap lifter. This process requires trained persons for monitoring and manual efforts to execute the task. It makes the seemingly simple task of sorting the scraps as a time consuming and expensive process. There are no fine standards defined for the categories of the scraps which makes the classification task error prone and demands skilled person in exceptional cases. The manual involvement in the process also requires safety measures for the employees.

The shortcomings mentioned above are mainly caused by the manual sorting of the steel scraps. It can be overcome by automating the process using machine learning techniques. The automated classification process can perform sorting of the scraps efficiently and promptly. Standardization is a necessary step to automate the classification of steel scraps. When standard properties of each scrap type are well defined, the automated classification task is less likely to make incorrect classifications. The automatic scrap classification is cost-effective and safe for the employees.

A software solution based on machine learning techniques is developed for the automation of scrap classification process. The images of railway carriages carrying the steel scraps from the scrap yard are used as image dataset to train the model. The images are taken from a multiple positions out of which the images from above the railway carriages are used in this project. The solution provides scrap images as output with a label of scrap type, a confidence value and a bounding box assigned to each container in the image.

The solution efficiently classifies the scrap types and removes manual efforts required to perform the task. The rapid classification performed by the solution makes sorting of scraps significantly less time consuming and cheaper than the existing method.

1.3 Objectives

The main objective of this thesis is to automate the classification of the steel scrap types using machine learning algorithm YOLOv5 (You Only Look Once, version 5). The YOLOv5 algorithm is one of the fastest object detection algorithms based on deep learning. The algorithm detects objects in images accurately considering the amount of time it requires in training and inference steps.

Another major emphasis of the thesis is to study the effects on benchmarking performance of the YOLOv5 algorithm after image processing operations are applied to the training image dataset. There are plenty of operations that can be performed on the images. The operations used for this project are histogram equalization and denoising. The algorithm performance was then compared with the original training dataset and preprocessed training dataset.

Like any other supervised machine learning algorithm, the YOLOv5 algorithm uses labelled images as ground truth values for the training. The images are labelled using bounding boxes around the objects to be detected within the image. We also need to specify object names or classes as well. An open-source tool namely Computer Vision Annotation Tool (CVAT) is used for this purpose. CVAT provides a user-friendly Graphical User Interface (GUI) to label the images with many useful features.

We need large amount of training dataset to avoid the overfitting in a deep learning algorithm. We can create new images by transforming the existing images. This process of image transformation diversifies the dataset to avoid the overfitting. The open-source library Albumentation is used for this purpose. This step is performed after the labelling of image dataset. Then algorithm is trained on the diversified image dataset obtained by implementing this process.

1.4 Thesis Outline

Chapter 2 Literature Review

This section covers the idea and purpose for the preprocessing of images. Then the concepts and methods of image annotation is explained in the context of object detection. The augmentation process is also mentioned as a process to overcome overfitting of the algorithm and create more data from the existing dataset. Then this chapter explains the conceptualization details of the machine learning algorithm YOLOv5. The advantages of using this algorithm are also briefly described in this section.

Chapter 3 Methodology

This part of the thesis explains the overall architecture of the project. Then it discusses the approach to utilize the YOLOv5 algorithm to achieve the objectives of the thesis. The details of pre-processing operations, histogram equalization and denoising are covered in this section. It helps to understand the effects of these preprocessing operations performed on the quality of the scrap yard images. Then it mentions the operations performed for annotation purpose using CVAT tool. It also explains the augmentation operations used from the Albumentation library with the motivation for using them on the image dataset.

Chapter 4 Implementation

This chapter provides implementation details of each step mentioned in the software architecture. It briefs about the image data acquisition and cleaning. This section also explains implementation details of the YOLOv5 algorithm and purpose of the additional developed Python scripts. A short description for implementation details of other external libraries and packages used is also included in this section. The configuration of Amazon Elastic Cloud Computing (EC2) instance and Data Version Control (DVC) tool is briefly mentioned to explain management of high computing load and storage space required for the project.

Chapter 5 Results and Discussion

This section discusses the experimental results obtained upon execution of the project and the conclusion drawn from those results. It compares the results when algorithm was trained after applying different augmentation operations and the pre-processing steps on the image data with the results acquired from the original dataset. Also, the insight of their effects on benchmark performance of the algorithm will provide the research aspects of the thesis. The results are represented with certain evaluation parameters for precise comparison.

Chapter 6 Summary and Future Work

This chapter summarizes the work performed to achieve the two major objectives of the thesis, the automation of scrap classification and effects of image preprocessing steps on performance of the algorithm.

The project has tremendous scope for further development. How the project can have more features in future to automate the scrap yard management processes which are manual at present. The project can also be studied in the context of accuracy improvement for the algorithm.

2. Literature Review

2.1 Image Annotation

The supervised machine learning algorithms need ground truth values on which the algorithm can be trained. The ground truth values for the object detection algorithms are the labelled images. These labelled images are used for the algorithm to learn the objects in labels provided by direct observations. The objects directly observed in dataset images are labelled by different shapes based on the computer vision task at hand. This process of manual labelling of objects in the image is called image annotation.

Image annotation is a crucial step for a robust object detection pipeline. Because the annotated images set the norms for a supervised algorithm to detect these annotations in test dataset images after training. So, the algorithm performance gets highly impacted if there is any error in the annotation step. Therefore, the objects need to be annotated precisely with annotation shape tightly wrapping the objects. This brings about the best performance of a trained neural network [1].

There are two ways to annotate the images manually or using an automated annotation tool. In case of auto annotation method, we need to train the algorithm on objects that need to be annotated. This is generally used for image segmentation task, where free shape boundaries are required around objects. This type of annotation is time consuming if done manually. On the other hand, if the algorithm is performing image classification task, then only labels as class numbers are required.

The manual annotation is used for this project because object detection requires standard rectangular shape. The rectangular shape used to annotate objects in an image is called bounding box. A machine learning algorithm expects numerical representation of a bounding box. There are several bounding box representation formats [2]. Some examples of representation formats are provided in table 1.

Format Name	Bounding Box Representation
Pascal VOC	(x_min, y_min, x_max, y_max)
COCO	(x_min, y_min, width, height)
Albumentation	Normalized (x_min, y_min, x_max, y_max)
YOLO	Normalized (x_center, y_center, width, height)

Table 1: Bounding box representation formats

Since we are using YOLOv5 algorithm for this project, let us try to understand the YOLO bounding box representation format. To explain this format, consider an example image from Common Objects in Context (COCO) dataset as shown in Figure 1. This image of a cat in the bounding box has a resolution of 640×480. We can see the pixel at top-left corner of the image has coordinates (0, 0) while bottom-right pixel has (640, 480).

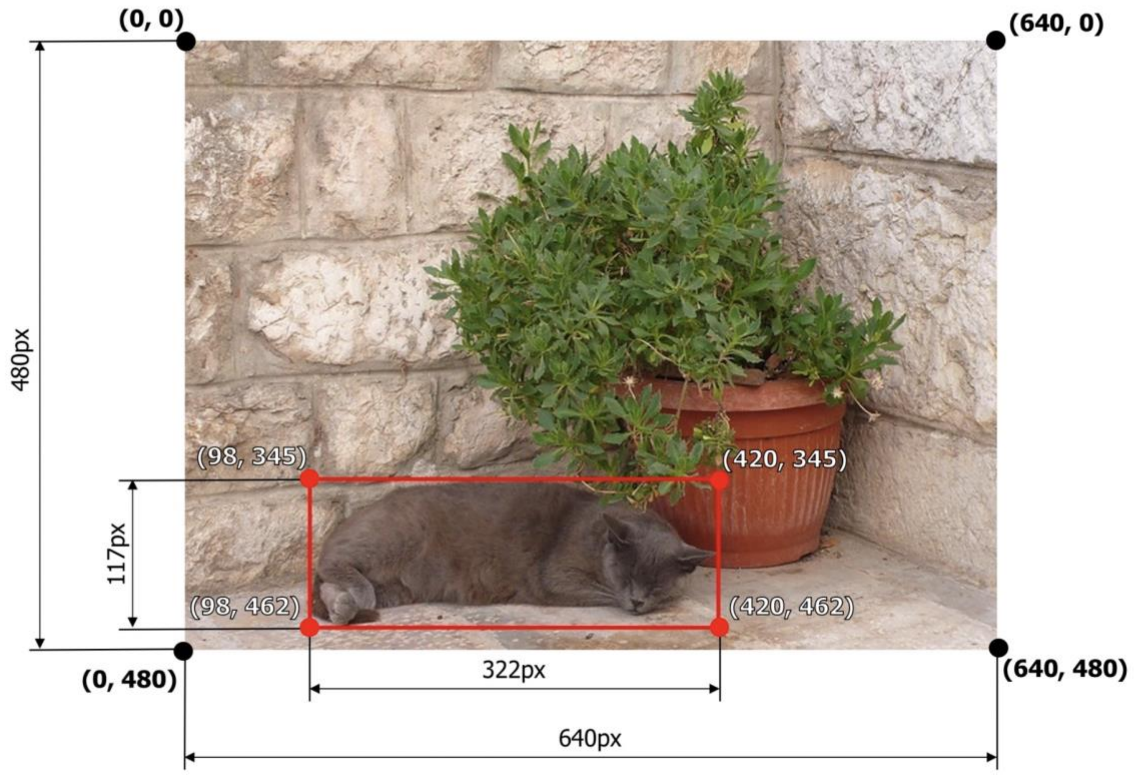


Figure 1: Image of a cat in bounding box from COCO dataset taken from [1]

The width (w_b) and height (h_b) of the bounding box is 322 pixels and 117 pixels respectively. The top-left corner (x_{min}, y_{min}) and the bottom-right corner (x_{max}, y_{max}) of the bounding box has coordinates

$$\begin{aligned}(x_{min}, y_{min}) &= (98, 345)\\ (x_{max}, y_{max}) &= (420, 462)\end{aligned}$$

as shown in figure 1. Then the coordinates of the center of bounding box (x_{center}, y_{center})

$$(x_{center}, y_{center}) = ((98 + 420) / 2, (345 + 462) / 2) = (259, 403.5)$$

Since, the bounding box representation in YOLO format is

$$bounding\ box \equiv normalized(x_{center}, y_{center}, w_b, h_b)$$

$$= \left(\frac{x_{center}}{image\ width}, \frac{y_{center}}{image\ height}, \frac{w_b}{image\ width}, \frac{h_b}{image\ height}\right)$$

$$= \left(\frac{259}{640}, \frac{403.5}{480}, \frac{322}{640}, \frac{117}{480}\right)$$

$$= (0.4046875, 0.840625, 0.503125, 0.24375)$$

2.2 Image Augmentation

The performance of traditional machine learning algorithms increases with training dataset size but eventually it gets saturated. But deep learning algorithms such as YOLOv5 often have large number of parameters. Because of that the complexity of the model increases exponentially. The complex algorithm with large number of parameters requires to

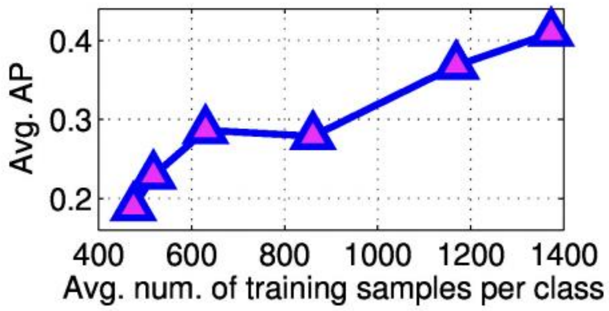


Figure 2: Detection algorithm performance on PASCAL VOC dataset vs training data size taken from [2]

be trained on huge dataset for performance improvement. The correlation of training dataset size and mean Average Precision (mAP) of deep learning based algorithm trained on PASCAL VOC dataset is shown in Figure 2. It shows that the performance of neural network based detectors keeps increasing with the size of training dataset because they are generally complex [3].

There is high probability of overfitting when the deep convolutional neural network is trained on insufficient data. The concept of overfit is not absolute but rather involves comparison. The model is considered overfit when it is more complex than another simpler model which fits equally well [4]. In other words, a complex model should fit better then simpler models to avoid overfitting. The complex convolutional neural network (CNN) based models such as YOLOv5 fits better when trained on more data.

However, it is not always possible to acquire enough training dataset images. There could be several reasons for that. There could be legal restrictions or efforts from the experts required to get healthcare data. It is cost intensive to acquire satellite images when the use case involves training on such dataset images [5].

We also need to label objects after obtaining the sufficient dataset images. For example, we need to assign correct class labels to images for image classification task, draw bounding box around the objects to be detected in images for object detection or assign object classes at pixel level for segmentation task. It demands laborious efforts and domain experts in some cases. There are few service providers for the labelling of objects for example Amazon Sagemaker Ground Truth charges 0.08$ per object for upto 50,000 objects [6]. Therefore, if 40,000 images are needed to train the model, the cost for just labelling the images would be $40{,}000 \times 0.08\$ = 3{,}200\$$. This makes the labelling task expensive as well.

The process of creating new training images from the existing ones is called image augmentation. Image augmentation is commonly used to fulfill the huge image dataset requirement for training of complex machine learning algorithms. The new images are created by performing certain transformation operation on the original image. We can

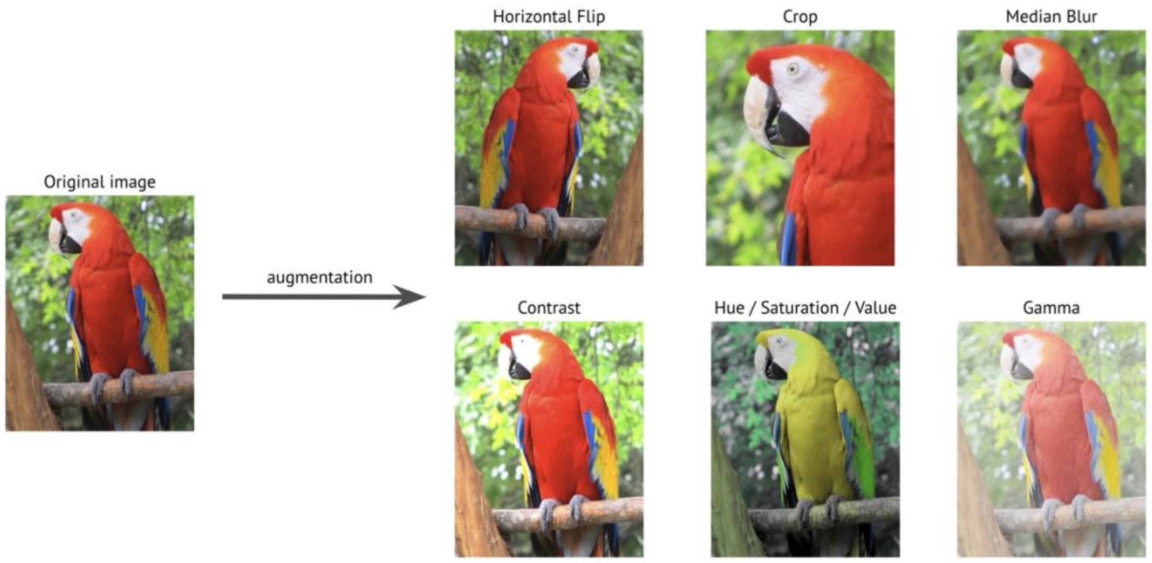


Figure 3: Augmentation transformation examples taken from [3]

perform horizontal flip, vertical flip, crop out some region of the image, blur the image, contrast adjustment, brightness variation are few simple operations that we can perform. We can observe in figure 3 that from one single image we can create six new images by performing some of the augmentation transformation. There are numerous such transformation operations possible. We can also create several images from only single transformation operation as well by changing the transformation operation parameters. For example, we can create multiple images for different values of brightness level from single image. In addition, we can combine two or more transformations and for each permutation of their parameters new image can be created. This allows us to create almost infinite number of images from only one image.

The implementation of augmentation involving multiple transformations can be challenging. It becomes more complex if custom transformation operations are used for a particular task in hand. There are many libraries available to support such operations such as OpenCV [7], Pillow [8], TorchVision, TensorFlow, Keras etc. Though these libraries assist with implementation of custom augmentations, the augmentation specific tool such as Albumentation [9] supports custom combination of diverse transformation operations.

The augmentation process transforms the original image. It is inefficient to annotate the transformed ones manually when the original image is already annotated. Therefore, the augmentation specific tool Albumentation provides features to transform the input image as well as the annotation shapes or output masks. This reduces great amount of manual work or expenses requisite to annotate the newly created images. An example image describing how Albumentation library performs transformation on the input image as well as the output mask is shown in figure 4.

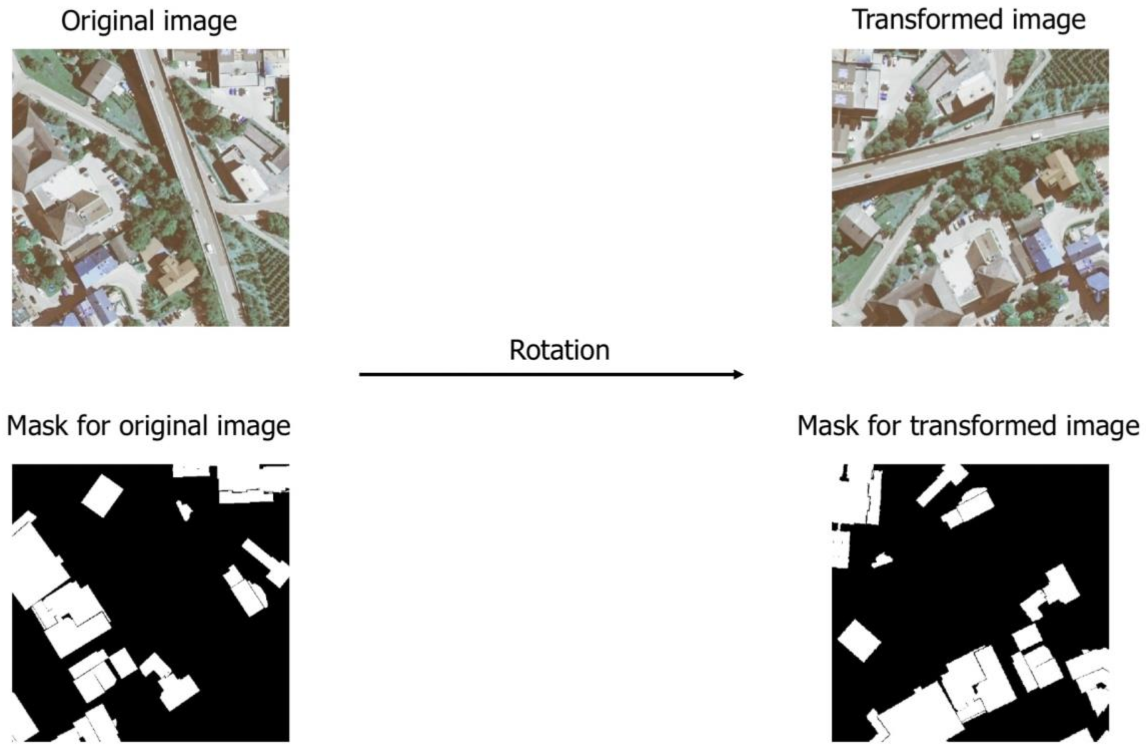


Figure 4: Transformation applied on both input image and output mask taken from [4]

We can randomly apply different augmentation operations to diversify the obtained image dataset resulting in model performance improvement. Also, choosing the appropriate augmentation operation can affect the model performance significantly. For example if rotate operation is used on CIFAR-10 dataset it might improve model performance but if same operation is used on MNIST dataset it can reduce algorithm performance because the algorithm will not be able to distinguish between numbers 6 and 9. There are algorithms such as AutoAugment which determines the augmentation type to be based on model to improve its performance [10].

Conclusively, augmentation creates large dataset to avoid overfitting in the model and increase performance of the neural network. This is important for the computer vision tasks such as object detection required to achieve the objectives of this project.

2.3 The YOLOv5 algorithm

The computer vision is a field of computer science focused on endowing the machines with abilities to mimic human visual system. One of the common tasks of computer vision is to detect real time objects and locate them in the image which is called object detection. There has been great amount of research work invested by the scientific community to come up with the algorithms to detect the objects faster and accurately.

Object detectors usually have neural networks pre-trained on ImageNet. It extracts features from the image initially and have fully connected layers which provide tensor as output. YOLOv5 is a neural network based algorithm having a single convolutional neural network (CNN). The network predicts locations of the objects as well as the probability of having a correct object in each location simultaneously. This unified approach of YOLO algorithm makes it extremely fast. Unlike region proposal based algorithm e.g., R-CNN (Region-based Convolutional Neural Network) and its variants, YOLO encodes contextual information about classes and their appearances for the entire image [11].

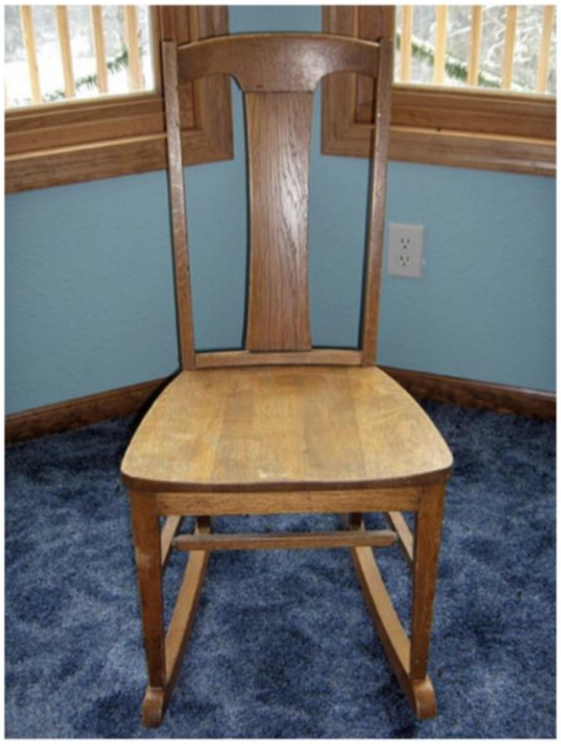


Figure 5: Example image from Pascal VOC dataset taken from [5]

Typically, object detection algorithm takes an input image and provide the same image with bounding box around the object of interest and object name (class) as label for that bounding box. To understand the working of YOLOv5 algorithm consider an image taken from standard Pascal Visual Object Classes (VOC) dataset as shown in figure 5. This image is taken from VOC 2008 challenge where list of 20 classes is provided [12].

During training, YOLO algorithm divides the input image into $S \times S$ grid. Each grid is responsible for detecting the object if center of object lies in that grid. PascalVOC provides an annotation file having bounding box coordinates which must be changed to YOLO representation as explained in chapter 2.1. There are also total $\mathbb{C} = 20$ objects listed in PascalVOC dataset.

Each object class is recognized as an integer (c). So, the name of bounding box containing an object becomes class number of the object. The chair object is listed as class 9 in that list. Therefore, class number for chair object is assigned as $c = 9$ where $c \in [1, 20]$.

Each grid cell predicts B bounding boxes with confidence score $(\hat{c})$ denoting how confident the model is about its prediction for the object. It is defined as intersection over union (IoU) between predicted box and ground truth times the probability of cell containing the object inside it. Therefore,

$$\hat{c} = Pr(object) * IOU_{pred}^{truth}.$$

The confidence score is zero if bounding box does not contain the object because in that case $IOU_{pred}^{truth} = 0$.

Let us assume that our example image is divided into a grid of 7×7 or $S = 7$. Also, assume that each cell predicts two bounding boxes $(B = 2)$ for the detection of chair. Now, consider a grid cell at the center of image which predicts two bounding boxes $\widehat{b_1}$ and $\widehat{b_2}$ which are shown as green and cyan boxes respectively in figure 6 (left).

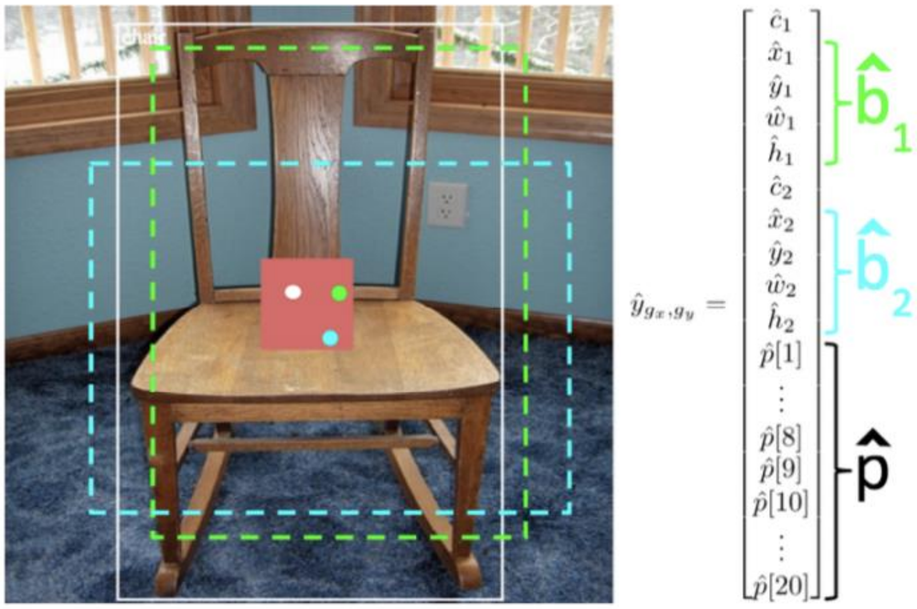

chair
$$\hat{y}_{g_x,g_y} = \begin{bmatrix} \hat{c}_1 \\ \hat{x}_1 \\ \hat{y}_1 \\ \hat{w}_1 \\ \hat{h}_1 \\ \hat{c}_2 \\ \hat{x}_2 \\ \hat{y}_2 \\ \hat{w}_2 \\ \hat{h}_2 \\ \hat{p}[1] \\ \vdots \\ \hat{p}[8] \\ \hat{p}[9] \\ \hat{p}[10] \\ \vdots \\ \hat{p}[20] \end{bmatrix}$$
$\hat{b}_1$
$\hat{b}_2$
$\hat{p}$

Figure 6: Visualize bounding box predictions and its parameters taken from [5]

Each bounding box $\widehat{b_i}$ consists of five parameters: confidence score and the bounding box coordinates

$$\widehat{b_i} = (\hat{c}_i, x_i, y_i, w_i, h_i)$$

Each grid cell predicts $\mathbb{C}$ number of conditional probability values.

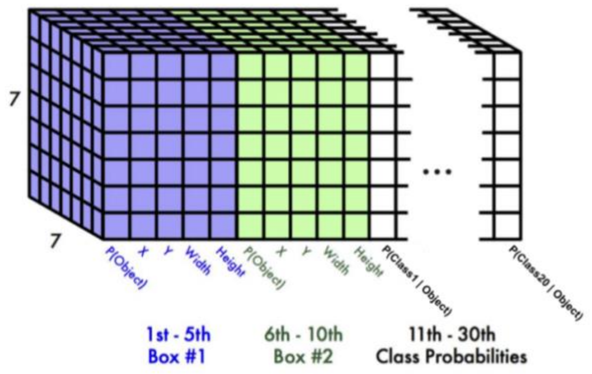
$$Pr(class_i |\ object)\ \forall\ i \in [1, 20]$$

This is regardless of number of bounding boxes predicted by the cell. This conditional probability $Pr(class_i |\ object)$ represents that the detected object in the cell belongs to $class_i$ provided that center of object is inside the cell.

In our example, there are two bounding boxes ($B = 2$) where each box provids five parameters. In addition, 20 conditional probability values belonging to each object class ($\mathbb{C} = 20$). Therefore, we get truth vector of $(2 \times 5) + 20 = 30$ parameters for grid cell at the center of the image as shown in figure 6 (right). We consider all 7×7 grid cells then we get a tensor of $7 \times 7 \times 30$ as shown in figure 7 (a).

Generically, if a grid at position (g_x, g_y) predicts B bounding boxes, then it will provide a truth vector of cardinality $B * 5 + \mathbb{C}$. For the entire image containing all grids the final output tensor will be of dimension $S \times S \times (B * 5 + \mathbb{C})$.

Let $\hat{b}, \widehat{b_1}$ and $\widehat{b_2}$ denote ground truth bounding box and two predicted bounding boxes respectively. Then predicted bounding box is assigned values of ground truth bounding



$$\lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Figure 7: (a) YOLO CNN output interpretation (left) source: medium.com (b) YOLOv5 loss function (right) taken from [5]

box $\hat{b}$ based on which predicted bounding box has the highest IoU. Then object class c is used to set probability value $P[c] = 1$ at index having highest value and set zero to others.

$$\hat{b} = \begin{bmatrix} x \\ y \\ w \\ h \end{bmatrix}, \widehat{b_1} = \begin{bmatrix} x_1 \\ y_1 \\ w_1 \\ h_1 \end{bmatrix}, \widehat{b_2} = \begin{bmatrix} x_2 \\ y_2 \\ w_2 \\ h_2 \end{bmatrix}, c = \max_{\widehat{b_\iota} \in \{\widehat{b_1}, \widehat{b_2}\}} IoU(\hat{b}, \widehat{b_\iota}) \ \dots (i)$$

This is how we get the confidence score obtained from highest IoU with ground truth and the object class obtained from probability values [13]. The calculated truth vector is illustrated in figure 8 (right).

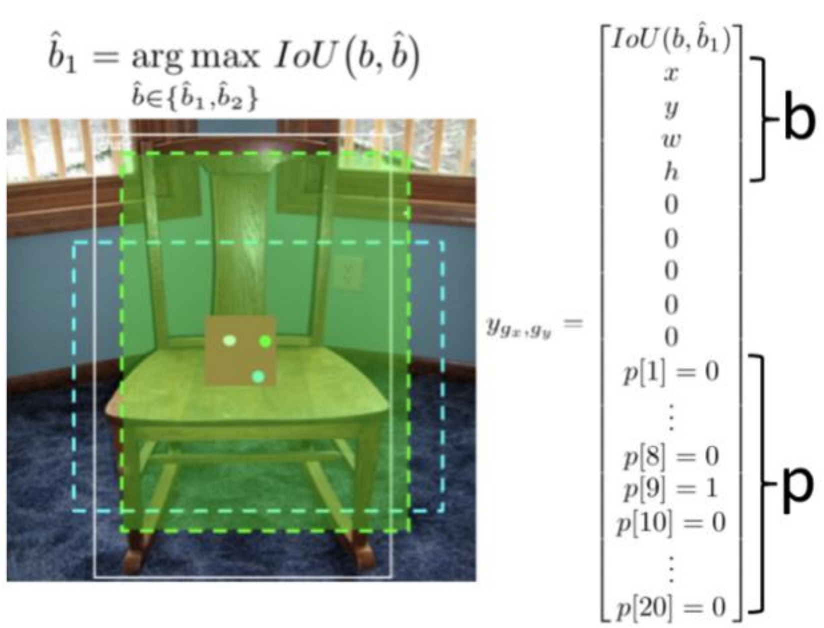


Figure 8: Truth vector y_(gx, gy) after using IoU taken from [5]

Once the predicted object is encoded as $(\hat{b}, c)$ the loss function for position (g_x, g_y) $\mathcal{L}_{g_x,g_y}$ is calculated as weighted linear regression. Two parameters λ_{coord} and λ_{noobj} are used to weight bounding box coordinates and confidence score $\hat{c}_i$ of the bounding box when it does not contain the object respectively. These weights have values $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$ to increase the loss from predicted bounding box coordinates and decrease the loss from predicted confidence values of boxes that does not contain the object. The formulated loss function as shown in figure 7 (b) is optimized during the training.

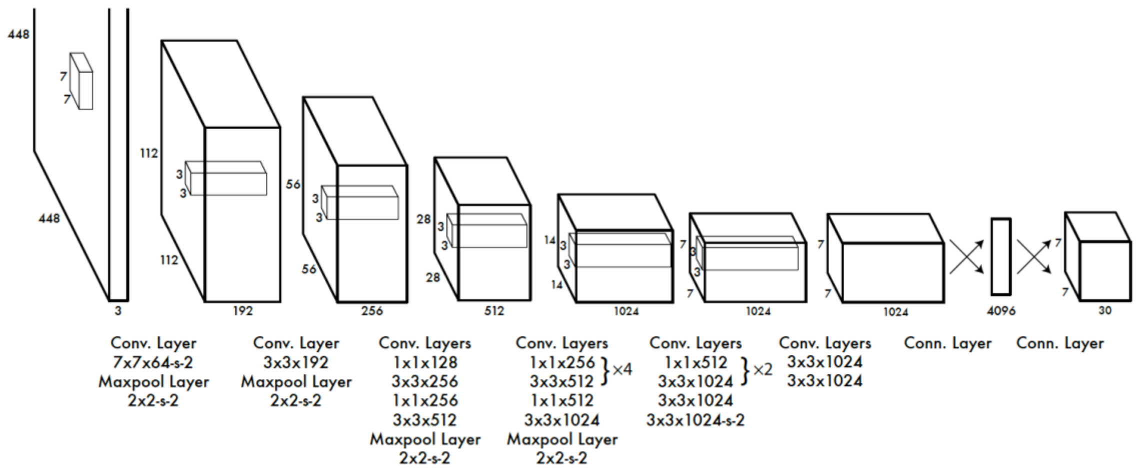

448
448
7
7
3
112
112
3
3
192
56
56
3
3
256
28
28
3
3
512
14
14
3
3
1024
7
7
3
3
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }x4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }x2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 9: Neural network architecture of YOLO model taken from [6]

The model has a single neural network containing 24 convolutional layers and 2 fully connected layers as shown in figure 9. The trained algorithm needs to run neural network only once for the inference phase as well. Faster version of YOLOv5 algorithm uses different number of convolution layers keeping other parameters same [11].

2.4 Image Preprocessing

The image quality plays a vital role in the performance of the object detection algorithms. There can be several factors contributing to a decrease in image quality. It can be flares of sunlight, dust, lightening conditions of the surroundings, or camera-induced noise. The object detectors can efficiently extract features from high-quality images. It also reduces the computation time and resources required to provide results.

The image needs to be preprocessed to improve its quality before it could be used as input to a machine learning model. The objective of processing the images is to either keep accuracy with reduced training time or increase the model performance. Several operations can be performed on images to increase image quality. These operations vary based on the computer vision task at hand.

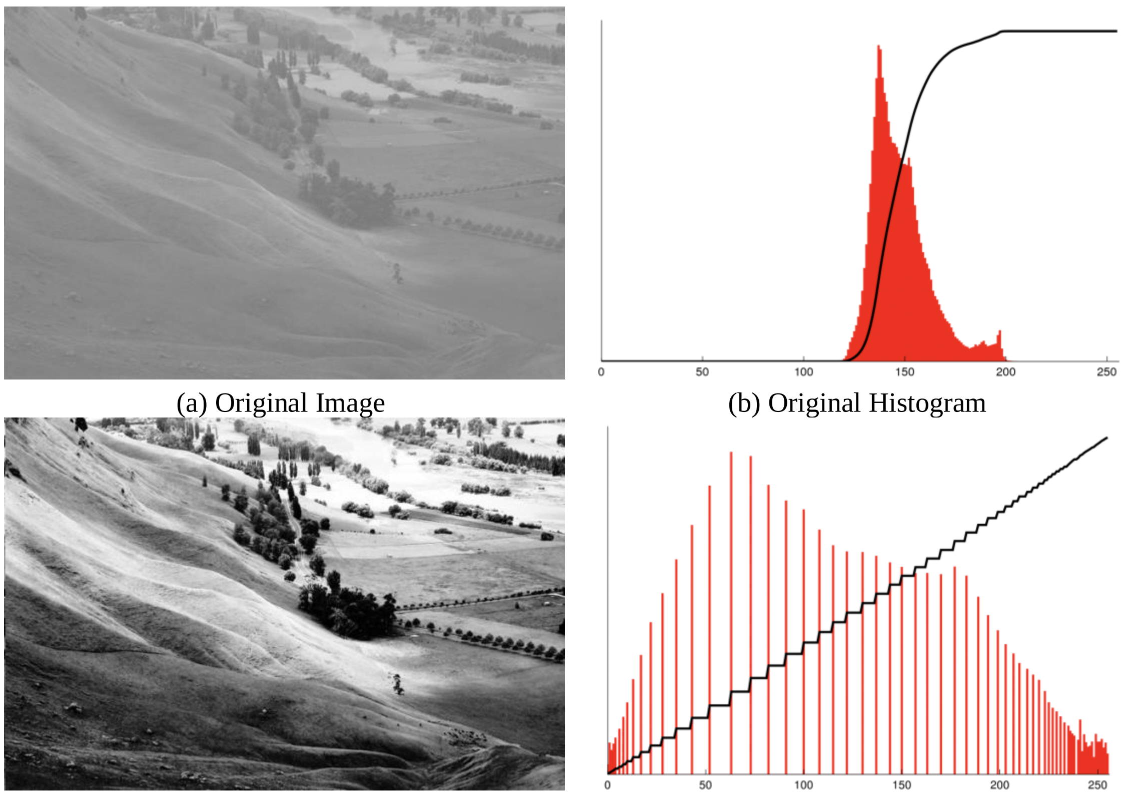
Denoising of an image is one of the most important operations to improve the image quality. The noise gets added while image capturing, data transmission, compression, or illumination. The noise needs to be removed without losing features, smoothing edges, or inserting new artifacts in the image. The classical ways of denoising are spatial domain filtering and variational denoising method [14].

Spatial domain filtering can again be performed in two ways using: linear filters and non-linear filters. Linear filters such as mean filter can be used to remove Gaussian noise, but it smooths out the input image. Wiener filter another approach to overcome this disadvantage, but it also blurs the edges present in the image. Non-linear filters such as median filter, weighted median filter and bilateral filters are widely used as edge preserving filters to reduce noise in the image [15].

Variational denoising methods such as total variation based regularization is based on the fact that original image is smooth locally and image intensity varies gradually in different regions. This method effectively removes noise retaining sharp edges but smoothens texture details and introduce staircase effect in the output image. In another approach non-local regularization uses full information provided in the image and is robust at handling noise [16].

Contrast Enhancement is used to improve the observability of objects in an image. Contrast enhancement techniques try to enhance the difference in visibility of objects and background in the image. This leads to effective extraction of feature from the image for improve training of a machine learning model. Contrast enhancement uses histogram created by the frequency of pixel values. Then histogram equalization methods are applied for contrast adjustments. The effect of histogram equalization on the histogram, cumulative histogram (black curve) and original image is shown in figure 10.

There are various methods to perform histogram equalization. Brightness preserving Bi-Histogram Equalization (BBHE) method uses average intensity value to separate dark and bright area [17]. Later it was found that median intensity value is more effective than the mean value. Another method Quadrant Dynamic Histogram Equalization (QDHE) divides the histogram into four parts using median of original image. Then each sub parts are equalized after normalization [18].



(a) Original Image (b) Original Histogram

(c) Histogram Equalized image (d) Histogram Equalized Histogram

Figure 10: Effect of histogram equalization on image and its histogram taken from [7]

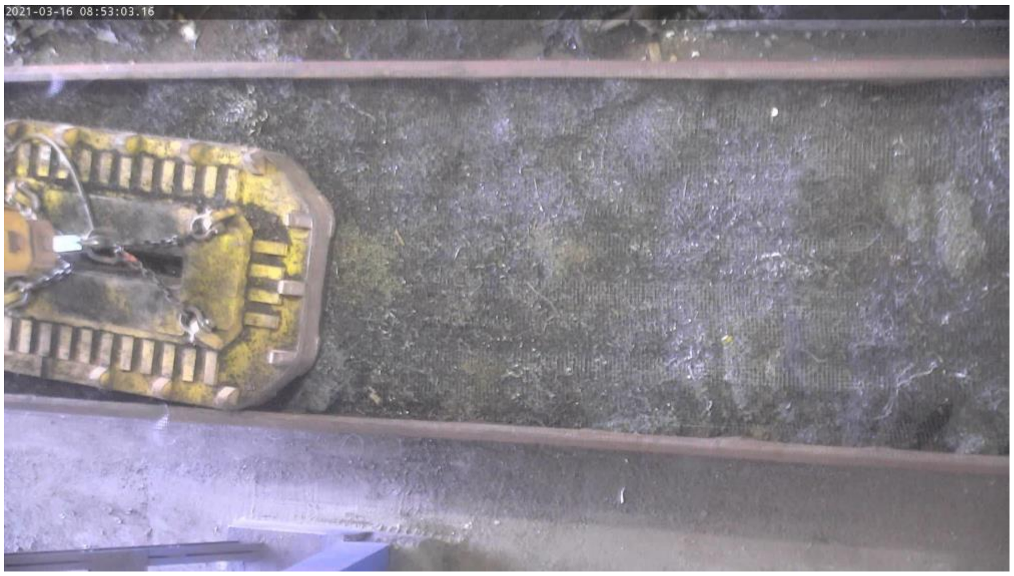
Adaptive Histogram Equalization (AHE) techniques are widely used which differs from classical histogram equalization. It creates many histograms corresponding to each section in the image and preserves local contrast of the image. Contrast Limited Adaptive Histogram Equalization (CLAHE) is an improved version of AHE that limits the contrast values by applying thresholding. It avoids the over amplification of noise that can be possible in AHE [19].

There are plenty of other operations that can be applied to preprocess the images such as image morphological operations which are mainly used to preprocess images before image segmentation [20]. The Fourier Transform can be applied to convert images into frequency domain and based on analysis step the converted images can be processed for denoising, contrast adjustment and more. Then images are converted back into spatial domain after processing using Inverse Fourier Transform [21].

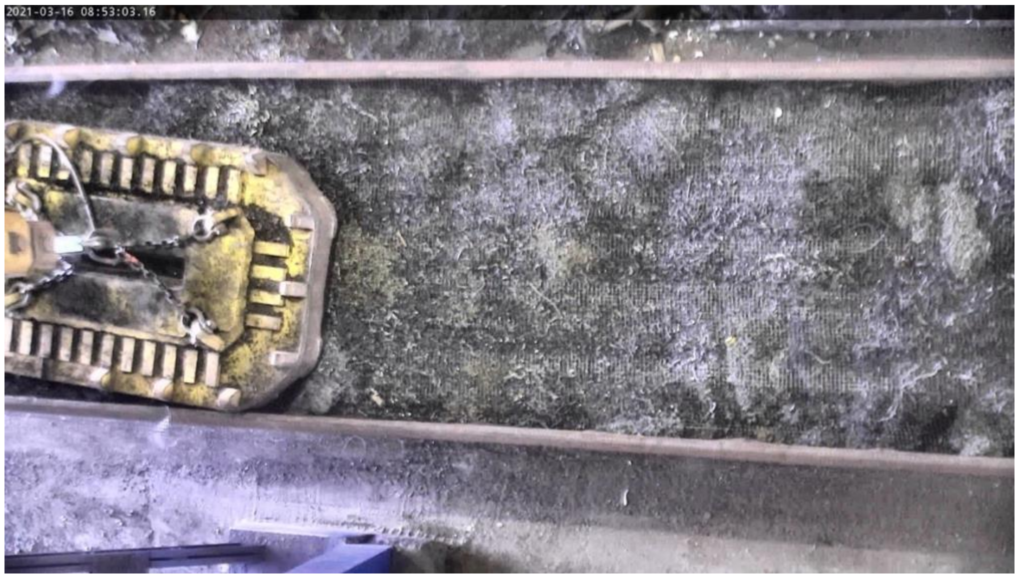
3. Methodology

3.1 Preprocess Image Dataset

The are some operations need to be performed on images to improve its quality. So that a machine learning algorithm can provide its optimum performance when used as input. The



(a) Original Image



(b) Preprocessed using CLAHE operation

Figure 11: Comparison of original and preprocessed (CLAHE operation) dataset image

preprocessing operations used depend on the source of degradation of images. Since the dataset images are taken in the scrap yard, the source of deterioration of the images can be equipment limitations, dust, or the lightening conditions of surroundings.

Sometimes, the images are taken from camera with improper exposure or background and foreground (with objects to be detected) both are too bright or too dark. The adverse lightening conditions in the scrap yard might lead to this situation.

The contrast adjustment is a useful operation in this case. As discussed in chapter 2.4, there are several approaches for contrast adjustment. The CLAHE method is used to perform the histogram equalization in this project. The normal histogram equalization amplifies the noise along with required pixel intensity values. The adaptive histogram equalization is used to overcome this disadvantage by dividing the image into $M \times N$ grids. Then histogram equalization is performed on each grid to get the enhanced image without noise. This method is computationally more intensive than the normal equalization but provides better resulting image. The Open source Computer Vision (OpenCV) library [7] provides a method that expects single channel image to perform CLAHE. Therefore, the image is first converted from BGR to YUV color model. Then the histogram of luminance component (Y component) is equalized to adjust the contrast. The other two components are chrominance components where U is blue project and V is red projection. The figure 11 shows comparison of original dataset image and CLAHE operation performed on the same image.

The denoising operation is another way to enhance the image quality. It is also applied separately on dataset images as well as on the contrast adjusted dataset images for the comparison of model performance. The images are converted to CIELAB (Commission Internationale de l´Eclairage L*, a*, and b*) color space initially. Then denoising is applied on L and AB channels. The fast non-local means denoising algorithm is used for this purpose. The algorithm performs well in terms of keeping texture and tiny image details after removing the noise. This algorithm substitutes a pixel value with the mean of all pixel values of a similar color [22]. The implementation of the algorithm with additional optimizations is available in the OpenCV library.

The processed dataset images are then further analyzed for detection of scrap types. The algorithm performance is compared when it is trained on original dataset and preprocessed dataset. The preprocessing steps would help to understand how standard performance of the algorithm can be improved.

3.2 Annotation Using Computer Vision Annotation Tool (CVAT)

The YOLOv5 algorithm was used to detect scrap types in the images of scrap containers. The algorithm expects input images as well as annotation of bounding boxes around the objects in each image. The annotation can be performed manually or automatically using an annotation tool. The automatic annotation method is used when sizeable image dataset needs to be annotated and manual annotation of the dataset is extremely time consuming or expensive. Automatic annotation needs a machine learning algorithm to be trained on significantly lesser number of annotated images. Then this trained algorithm is used with the help of annotation tool to annotate the large image dataset quickly without any additional cost. But the manual method is chosen to annotate the image dataset used for this project considering the standard size of the dataset and augmentation step to automatically create more dataset images.

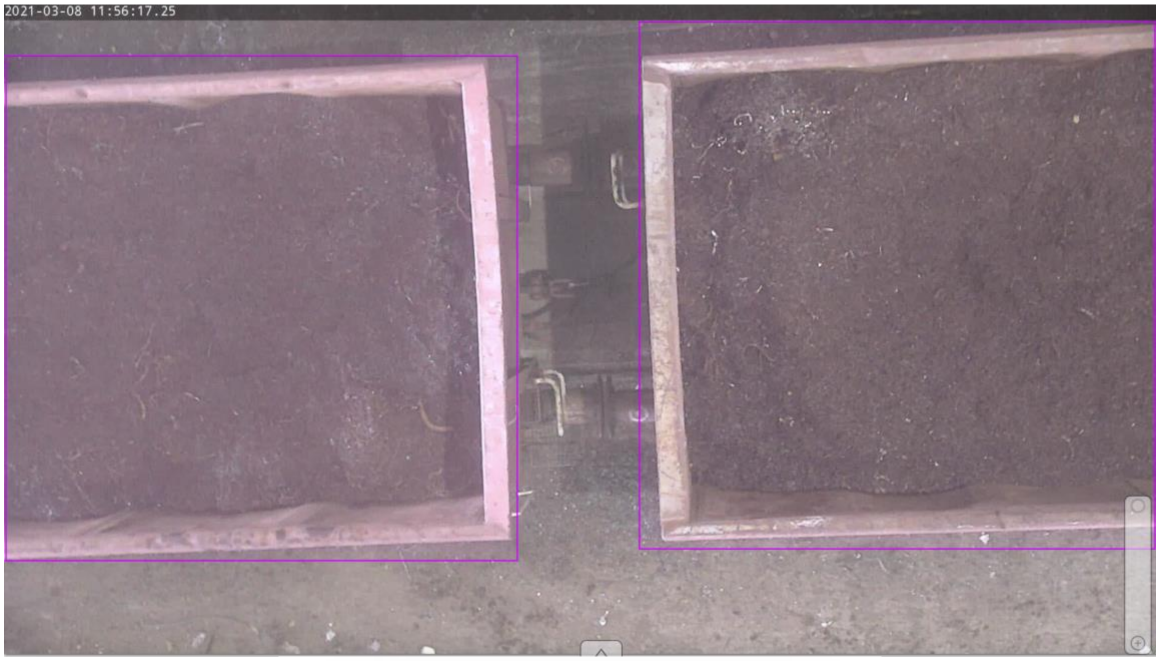
The image can be annotated manually using any annotation tool. The annotation tool is a software application that provide ways to annotate images using simple Graphical User Interface (GUI). It automatically generates the bounding box coordinates as output based on the bounding box drawn around the object to be detected in an image using GUI tools. An annotation tool contains many shapes to annotate the objects in images such as rectangles, ellipses, cuboid, polygons, free style shapes and more. The shape used for annotating the object depends on the computer vision task at hand. For example, we would simply assign a class label when performing image classification, rectangles for object detection and polygons or free style shapes for image segmentation.

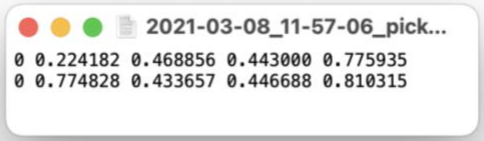
The rectangular shape is used for the task of detecting scrap types in scrap yard images. The rectangular bounding box should properly and tightly enclose the object in image. This step is most crucial for optimum model performance because the algorithm learns the annotated objects. These bounding boxes function as truth values for the algorithm. Poor annotation often affects training and results of the model significantly and provides poor predictions.

There are several annotation tools available for this purpose. The open source Computer Vision Annotation Tool (CVAT) [23] is used to annotate the image dataset of this project. We need to declare different scrap types as labels in CVAT before starting the annotation task. These labels act as object classes while training the algorithm. Then bounding boxes are drawn using the two point rectangular shape available in GUI of CVAT. Once, all the images are annotated, CVAT provides a feature to export the YOLO format of image annotations in text files. It contains a separate line for each bounding box. There are five numbers in each line starting with integer number representing the object class, then four consecutive floating point numbers represent the bounding box coordinates in YOLO format which is $normalised(x_{center}, y_{center}, width, height)$.

Consider an example image of scrap metal inside two containers as shown figure 12 (top). The objects are tightly bounded with two separate bounding boxes. Then their annotations in YOLO format are also shown in figure 12 (bottom). It should be noticed there is separate text file for each image containing the bounding box coordinates of object in that image.

CVAT creates a text file containing annotations for each image with same name. e.g., if image name is scrap.jpg then corresponding annotation file would be scrap.txt. That is how YOLOv5 algorithm can relate annotation to a particular image.





```
0 0.224182 0.468856 0.443000 0.775935
0 0.774828 0.433657 0.446688 0.810315
```

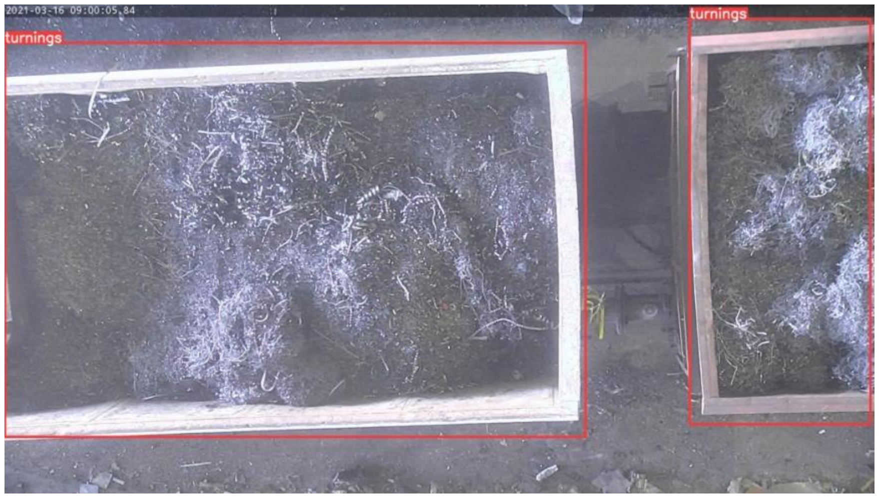
Figure 12: Scrap type annotated in an image (top), Annotations of bounding boxes in YOLO format (bottom)

In this case, there are two lines dedicated for each bounding box. Every line is having an integer as first value denoting the scrap type class which is 0 for both bounding boxes in this image. This integer value is followed by four float values for each line. They represent the bounding box coordinates in YOLO format. Since the bounding box coordinates are normalized so they are always in the range $[0, 1]$.

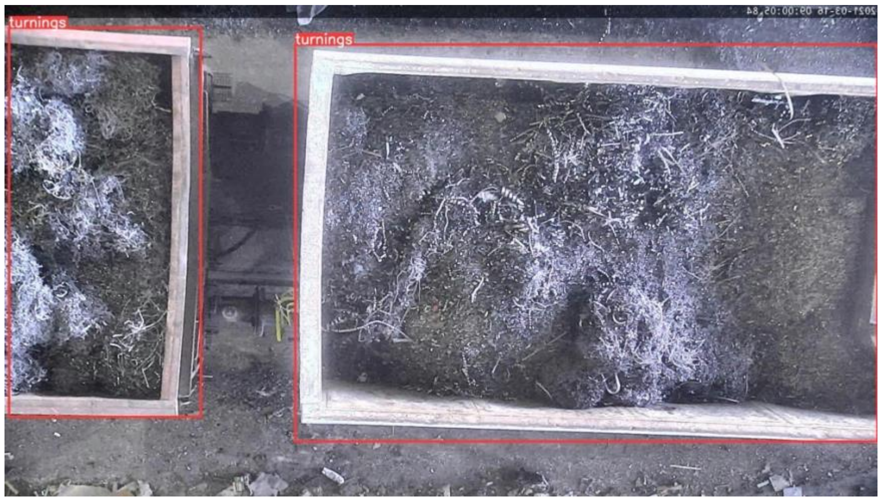
3.3 Augmentation Using Albumentation

We augment the image dataset once the images are annotated to create more images and avoid overfitting of YOLOv5 model. There are lot of libraries available to perform the augmentation operations. These libraries have limited transformation operations available.

The augmentation transformations can be divided into two categories (a) pixel level augmentation (b) spatial level augmentation. The original image is transformed when pixel level transformation is applied on the image. The bounding box and output mask does not get transformed in that case. e.g., brightness change, contrast adjustment, Gaussian blur are examples of the pixel level transformations. The original image as well as bounding boxes or output masks are also transformed when spatial level transformations are applied. e.g., crop, flip, mirroring and rotate are examples of spatial level transformations.



(a) Original Image



(b) Image after pixel level and spatial level transformations

Figure 13: Comparison of images and bounding boxes before and after pixel level and spatial level transformations

The new images obtained from spatial level transformation need the bounding box to be transformed as well. Consider an example image shown in figure 13 (b) with both pixel level and spatial level transformations applied to the original image in figure 13 (a). It can be observed that pixel level transformation namely CLAHE does not need the bounding box to be transformed but spatial level transformation horizontal flip needs to transform the bounding boxes as well.

The augmentation specific open-source library Albumentation [9] is used for the augmentation transformation purpose in this project. This library provides robust methods to transform the images in addition to automatic transformation of the bounding box or output masks based on the type of transformation used. This reduces the extra effort required to annotate the new images obtained from augmentation. The library also provides easy to use interface to implement transformation with probabilities.

The deep learning algorithms are generally executed on Graphical Processing Units (GPUs) because of their parallel computing capabilities while the augmentation operations are carried on Central Processing Units (CPUs) using multiprocessing capabilities. Recent developments of GPUs are outperforming CPUs in terms of processing time. The Albumentation library tries to overcome that challenge by wrapping augmentation functions on the other library functions with fastest implementation. This comes at a cost of having dependencies over several other libraries and using their optimum implementations. But the library comes bundled up with various generic and domain specific transformations as an advantage [24].

The vertical flip, horizontal flip and image invert operations are used as spatial level augmentation and blur, color to greyscale conversion, image compression and random change in brightness and contrast operations are used as pixel level transformations. We also try to keep the original dataset by using no change operation.

The Albumentation library allows to use probability values to randomly apply different augmentation operations. The probability value provided to a transformation operation defines the likelihood of applying that operation on input images. Most of the library methods for transformation operations accept probability as parameter that helps to increase the diversity in new obtained images. It results in improvement of machine learning model performance when new obtained images together with original image dataset is used as training dataset. The probability must be assigned value in range [0, 1].

Generally, multiple transformations are applied altogether instead of using them individually. For example, two different transformations brightness change and mirroring are applied at once on the image shown in figure 13. Sometimes, the transformations are needed to be applied randomly and their magnitude must be defined as well. The Albumentation library provides simple interface to implement the augmentation pipeline fulfilling these requirements.

There are two ways to perform augmentations on training dataset. One approach is to augment the original dataset before training starts and then train the algorithm on original dataset together with augmented dataset. This approach is useful when training is to be performed on very large dataset and number of object classes to be detected images are large in number. Another approach which is used in this project is to augment the original dataset before starting of each epoch during the training. This way new training data is generated for each epoch. This method generates diversified augmented dataset for the model to train upon. It results in improved confidence values during object detection and rapid convergence of the algorithm during training. This method was chosen to study the effect of augmentation on benchmarking performance of the YOLOv5 algorithm.

3.4 Training and Inference of YOLOv5 Algorithm

The non-local denoising algorithm and CLAHE operations are performed on original dataset. Then the algorithm was trained separately on resulting datasets obtained after performing these operations. The annotation and augmentation of each training dataset is automatically handled by Albumentation library. The training of YOLOv5 algorithm is performed on these annotated and augmented datasets. Each training run is executed separately for original dataset and preprocessed datasets described in Table 2. The algorithm runs mentioned in table 2 are studied into 3 groups to facilitate visualizations of graphs and performance parameters. The grouping is based on operation types performed on the dataset.

Training Run	Group	Operations performed on training dataset
1		Original Dataset
2	1	Dataset with only pixel level augmentations performed
3	1	Dataset with only spatial augmentations performed
4	1	Dataset with both pixel level and spatial level augmentations performed
5	2	Dataset preprocessed with only CLAHE operation performed
6	2	Dataset preprocessed with only denoising operation performed
7	2	Dataset preprocessed with both denoising and CLAHE operations performed
8	3	Dataset preprocessed with CLAHE operation and all augmentation operations performed
9	3	Dataset preprocessed with denoising operation and all augmentation operations performed
10	3	Dataset preprocessed with both denoising and CLAHE operations and with all augmentation operations performed

Table 2: Operations performed on training dataset

Thus, we get distinct trained instances of the algorithm by combining and permutating different image processing and augmentation transformations. It helps to study the effects of image preprocessing operations on the performance of YOLOv5 algorithm. These trained instances are then used for detection of the scrap types in scrap yard images and their performance is analyzed and compared. This contributes for insights about improving the standard algorithm performance by enhancing the quality of images before using them for algorithm training.

3.5 Project Architecture

The software application to classify scrap metal has mainly two phases: training of the YOLOv5 algorithm and detection or inference of scrap type in scrap yard images. The training phase train the machine learning model on scrap yard images.

The scrap yard images of the shipping containers can be acquired from different angles but the top view images of containers having scrap metal inside them is used to achieve the objectives of this project.

Therefore, the images from different angles of containers are collected and only the top view images are filtered using python script. Once the top view images of the containers having different types of scrap metal is acquired, these dataset images are uploaded to cloud service Amazon Simple Storage Service (S3) [25] bucket using Data Version Control (DVC) [26] to make it available to other instances of the project.

Then preprocessing operations non-local means denoising and CLAHE are used to generate four different copies of dataset images as mentioned in table 3.

Sr. No.	Dataset type
1	Original dataset images
2	Dataset images with only denoising operation performed
3	Dataset images with only CLAHE operation performed
4	Dataset images with CLAHE and then denoising operations performed

Table 3: Training datasets generated using preprocessing operations

The original dataset images are then uploaded to onsite server of SMS digital. The annotation tool CVAT is hosted on this server. The images are annotated using CVAT and the annotation files in YOLO format are produced. The annotations for the other copies of datasets remains same because preprocessing operations does not alter the location of the object in any way. Therefore, same set of annotation files can be used for other dataset copies. The dataset copies and annotation files are also uploaded to the S3 bucket.

The operations performed so far could be easily performed on local machine, but training of the YOLOv5 algorithm is computationally intensive. Therefore, the YOLOv5 algorithm is trained on the Amazon Elastic Cloud Computing (EC2) instance. The EC2 instance reduce the training time significantly by leveraging the parallel computing on GPUs. The EC2 instances can be considered as a remote machine with same setup available on a local machine. The difference between a local machine and EC2 instance is that latter has high computational resources.

Training of scrap metal classification software

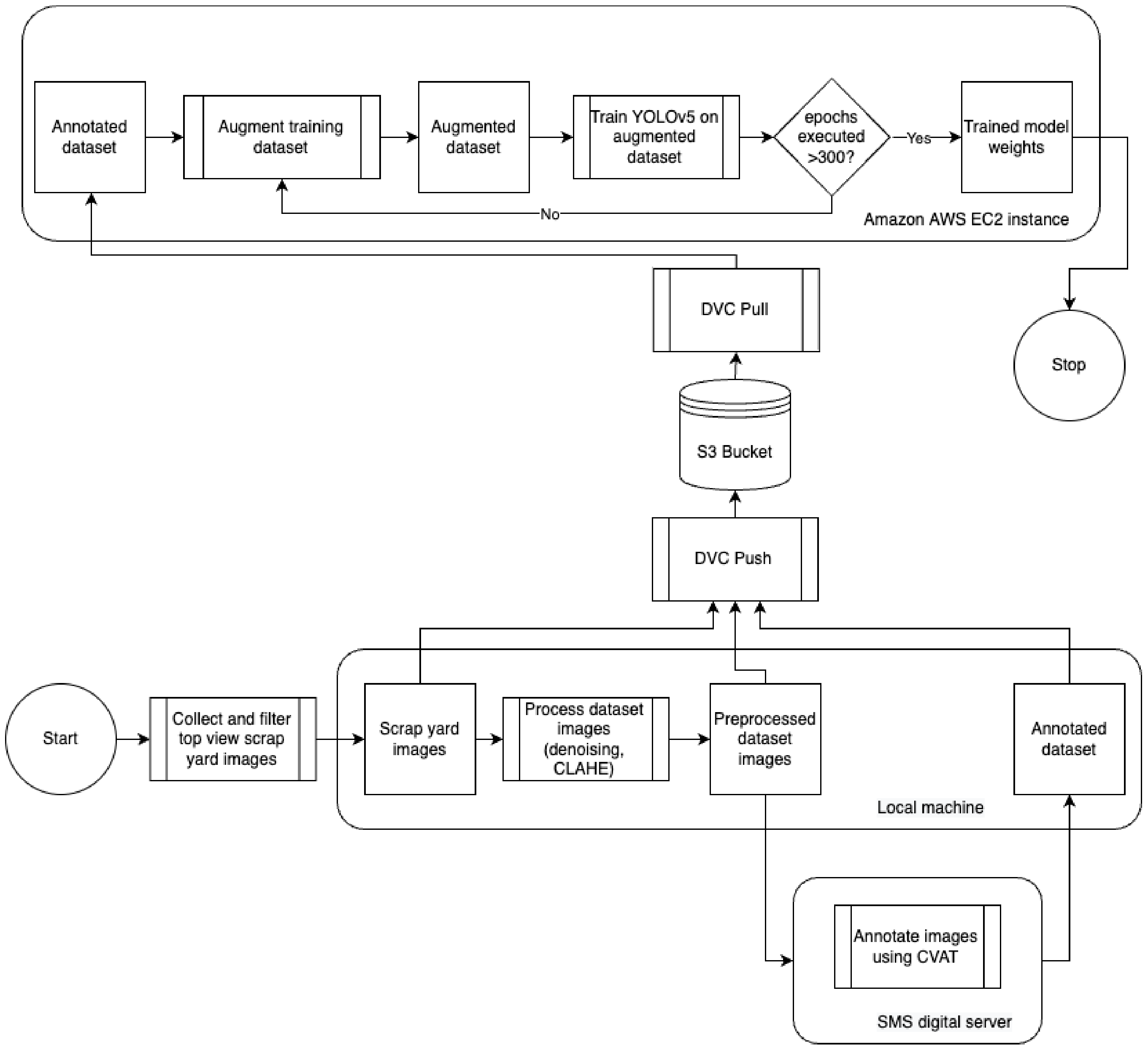


Figure 14: Project architecture diagram for the training phase

A project instance is created in the EC2 instance using version control system. The annotated images available on S3 bucket are also downloaded to EC2 instance. Then the YOLOv5 algorithm is trained using four different copies of the dataset. The dataset images are also augmented before start of each epoch. Consequently, the algorithm is trained multiple times based on the preprocessing operations used and augmentation functions applied on dataset images as explained in chapter 3.4. The augmentation transformations and their magnitude values are applied randomly. This results in generating new training images upon start of each epoch for the algorithm to train on. The separate model weights are obtained for each training run. The architecture of this entire process of training is illustrated in figure 14.

These model weights obtained after training are used for the detection of the scrap types in test dataset images. There are different model weights obtained from multiple training run of the algorithm. Because algorithm was trained differently on distinct preprocessed and augmented dataset images. This makes algorithm to assign different weights in neural network in each run.

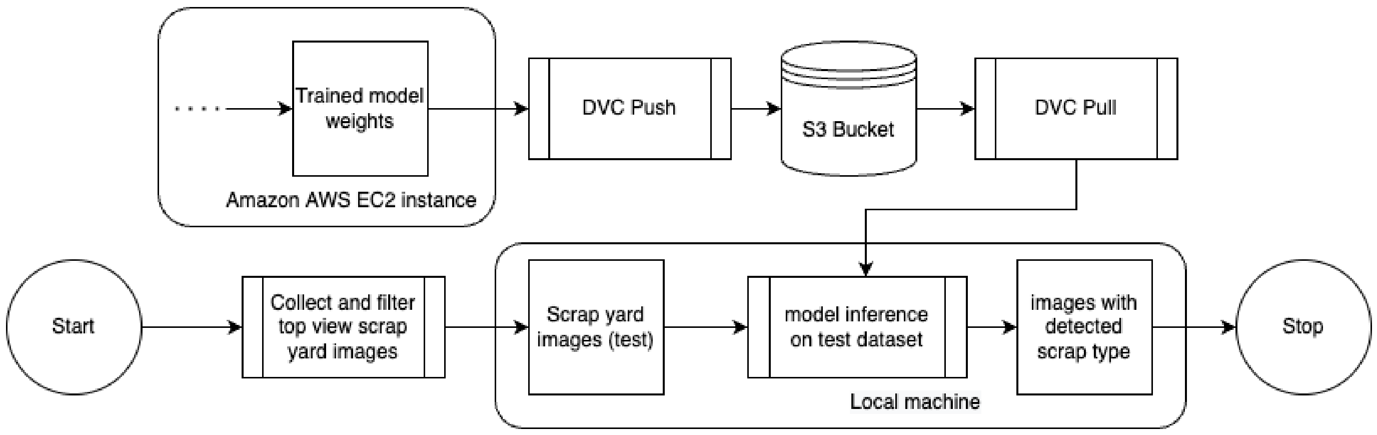


Figure 15: Project architecture diagram for the inference phase

All these different model weights are used to detect scrap types on same test dataset. The different model weights provide different results on the same test dataset images.

The algorithm generates a bounding box around detected scrap types in images, confidence scores and class label denoting scrap types for each image in test dataset. The algorithm performance is then studied for different train runs based on certain parameters. It provides insights about the effects of image preprocessing steps on the performance. The steps involved in detection of scrap types in scrap yard images or in other words inference phase is shown in figure 15.

4. Implementation

4.1 External Libraries Required

There are some external libraries and tools required apart from the YOLOv5 algorithm to develop the software solution to detect scrap metal types in scrap yard images. This part of the thesis explains which major external libraries are required and their utilization to achieve the task of scrap type classification. The list of libraries and tools required is ordered in a way they are utilized in step by step execution of the project. Which is as follows:

- ***Git:*** Git is a version control tool for the project code. The project would be having multiple instances e.g., one instance on local machine and another one on cloud. This tool is required during the project setup for code management between these instances. This tool also helps reviewer to keep track of development progress of the project. The hosting service GitHub was used for this project.
- ***OpenCV:*** The dataset images are first processed using image processing operations. There is an OpenCV library that contains the optimized implementation of non-local means denoising algorithm and CLAHE operation. There are four different datasets generated from original dataset with the help of these library functions.
- ***Docker:*** It creates an isolated virtual space called containers with only required dependencies preinstalled for intended software to run. This tool is required to run CVAT in the container. This tool was setup on the on-premises server of SMS group.
- ***CVAT:*** This open-source tool provides simple Graphical User Interface (GUI) to annotate images. CVAT supports several annotation shapes and annotation formats to export. It is used for the annotation of scrap yard images using four point rectangular shape.
- ***Amazon S3:*** It is convenient to store the image data on cloud service. The Amazon S3 service provides scalable storage space, easy access, security, and access management features for the data. The dataset images are stored in Amazon S3 buckets.
- ***DVC:*** It is evident from its name Data Version Control that it is a version control tool for data files. This tool is equivalent to what Git is for the source code. This tool is useful to manage the image data when it gets added, deleted, or altered. It is used to synchronize image data in project instance (on local or cloud) with S3 bucket. This way, if image data is altered on local machine its changes can be made available to the project instance on cloud.
- ***Albumentation:*** This library is used to augment the training dataset images. The Albumentation library provides various in-built augmentation transformations. The pixel level and spatial level both types of operations are used to augment the images. It is exceedingly simple to implement the augmentation pipeline using this library. This library provides important feature to transform the bounding boxes according to the transformation operation used.
- ***Amazon EC2:*** It is a service that provides virtual machine called EC2 instance on cloud with scalable computing resources. The project copy is made available on EC2 instance for training of the algorithm. This trains the YOLOv5 algorithm rapidly that

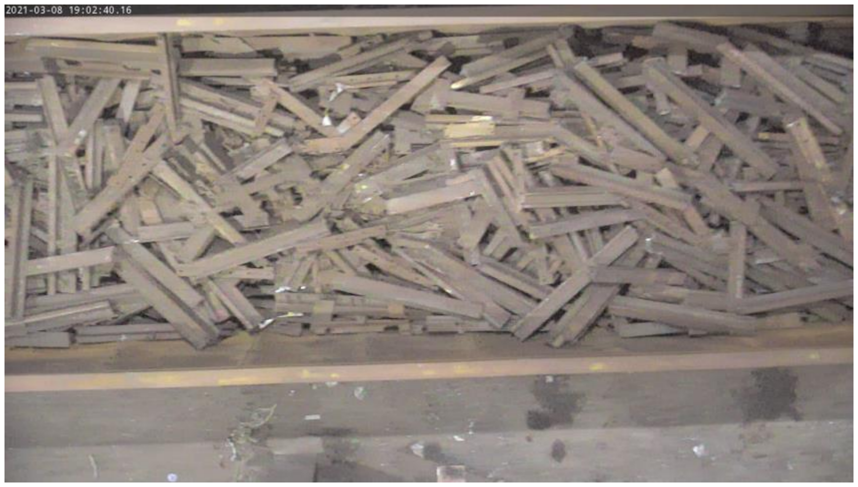
would otherwise require considerable amount of time if local machine was used for this purpose.

- ***wandb:*** This is a visualization tool to compare the model performances, generate performance graphs and track hyperparameters of the model. The web interface of this tool is used to generate performance graphs and loss curves for different training runs of the algorithm.

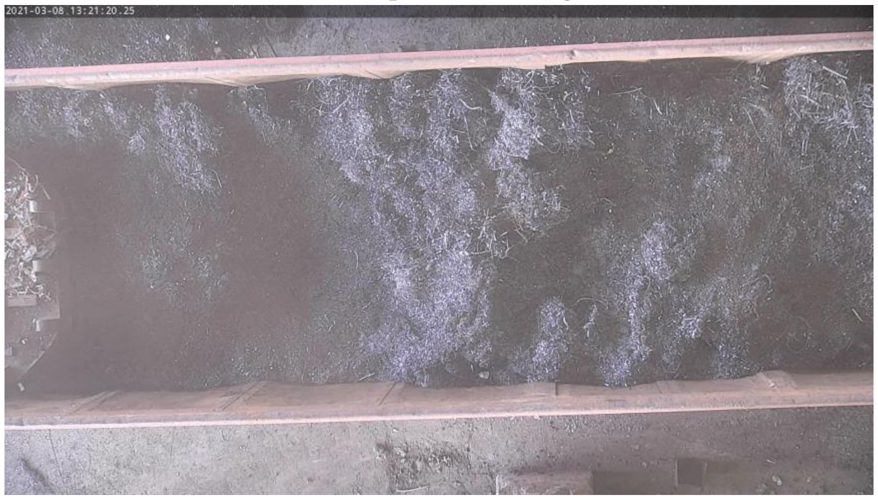
There are some other external libraries like TensorFlow, Keras, PyTorch, Seaborn and more required as dependencies for the YOLOv5 algorithm.

4.2 Data Acquisition and Data Cleaning

The railway carriages arrive filled with scrap metals at scrap yard facility. The images of these carriages are taken from different angles by cameras installed at fixed positions. The obtained images contain different views of railway carriages but for this project we only



(a) Sample Rails Image



(b) Sample Turnings Image

Figure 16: Sample images of scrap types

use images taken from above the carriages with scrap metals. The image file names have naming convention as

{production_time}_pick-{pick_id}_cam-{cam_id}_crane-{crane_id}{file ending}

e.g., *'2021-03-08_00-00-40_pick-017482_cam-Par1CarB_crane-PR13.jpg'*. The top view images are filtered out using Python script based on *cam_id* defined for the top view images.

There are several types of scrap metals available in these images. The scrap types namely turnings and rails are used for this project. Example images of these scrap types are shown in figure 16. The images of these two scrap types are manually selected from all top view images to create datasets. The description of obtained dataset is provided in table 4.

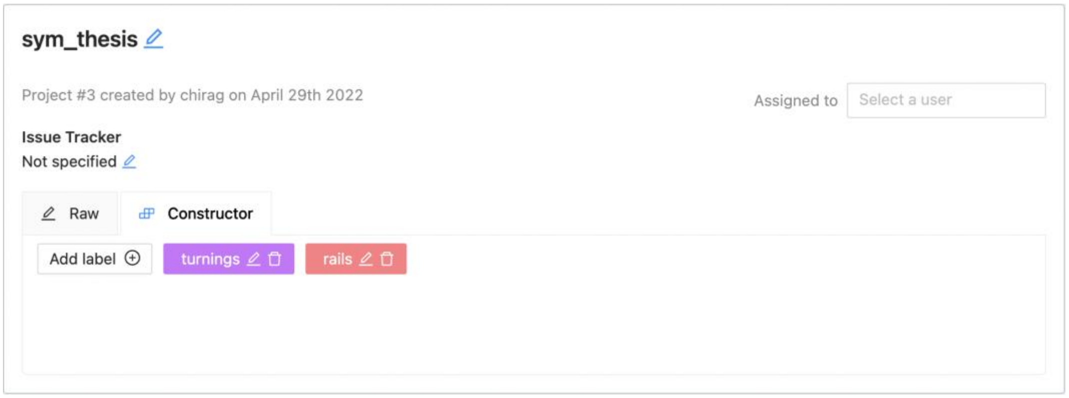
Dataset	Turnings Images	Rails Images	Total Images
Training set	114	97	211
Validation set	20	20	40
Test set	71	80	151

Table 4: Dataset description

The number of images having turnings scrap type are kept approximately same as rails scrap type to avoid biasing during the training of YOLOv5 model. The Python script are pushed to GitHub repository and the dataset images are stored in S3 bucket using DVC.

4.3 Preprocess Data using Image Processing and Annotations

The denoising operation using non-local means denoising algorithm is performed on training set images once the dataset images are generated as mentioned in chapter 4.2. It creates denoised version of all training set images. Similarly, one set of histogram equalized training images are produced using CLAHE operation and one set made from applying both these operations. The implementations of these operations available in


sym_thesis

Project #3 created by chirag on April 29th 2022

Assigned to: Select a user

Issue Tracker

Not specified

Figure 17: Labels of bounding boxes for annotation

OpenCV library are used for this purpose. This step creates four training sets according to table 3.

After the multiple training sets are generated, the original training set and validation set images are imported to CVAT tool hosted on the server of SMS group. The two labels 'turnings' and 'rails' are created for two scrap types while creating new project as shown Figure 17. Two jobs in CVAT are created for two sets of images. Image annotation can be started using the list of shapes available in the jobs created for each set.

Since YOLOv5 algorithm only considers rectangular bounding boxes. We use two point drawing method to annotate the railway carriages containing the scraps. While choosing the shape for annotation the scrap type can assigned to bounding box using the labels defined earlier. The snapshot of CVAT interface for annotation can be seen in Figure 18.

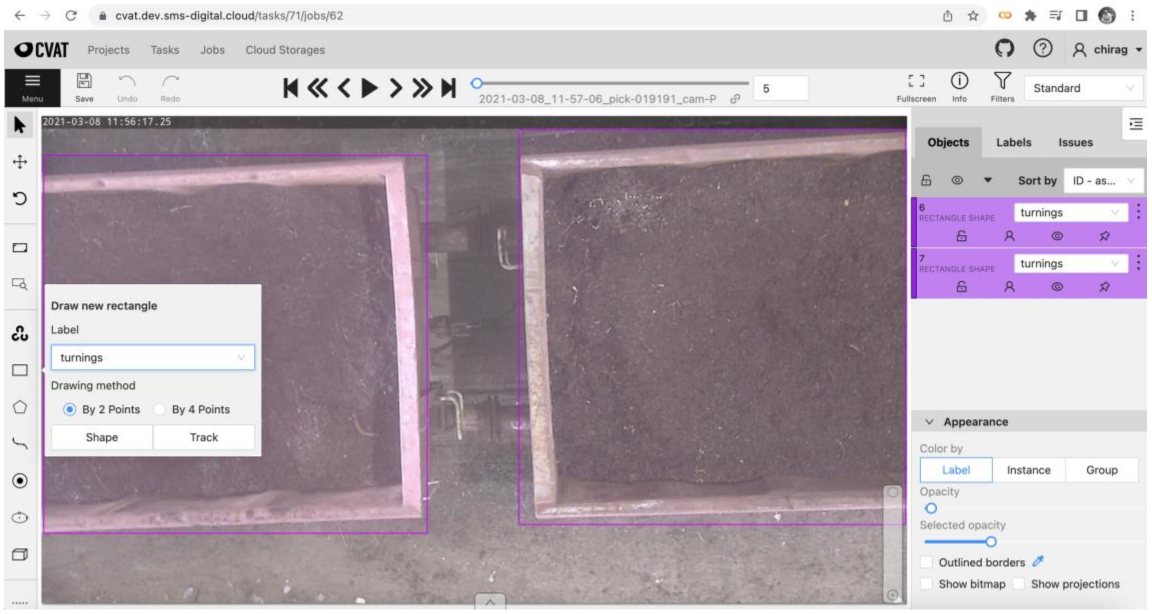


Figure 18: Interface of CVAT for image annotation

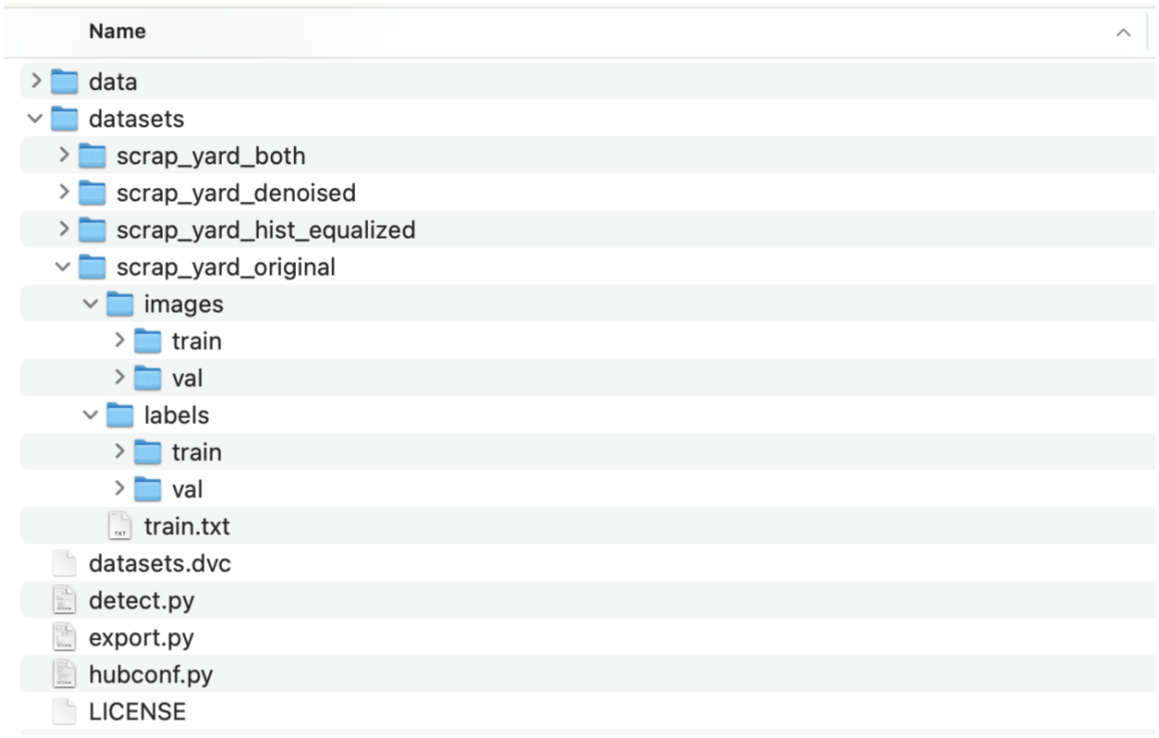
The image annotations in YOLO format can be acquired in text files after the bounding boxes are drawn for all images in training set and validation set. CVAT generates one text file for all instances of scraps in one image. Each instance of scrap in the image is represented by a line of 5 values. The first integer value is an object class enumerated from labels declared initially. The remaining four float values represent the bounding box for that object.

The other versions of training set are created by performing image processing operations. Those operations do not alter the locations of the scraps in the images. Therefore, the text files with annotations of original training set can be used with these datasets as well.

4.4 Implementation of YOLOv5 Training

The project is setup by cloning the GitHub repository of the YOLOv5 algorithm and custom python scripts for filtering top view images and image processing operations are added in $utils$ package of the project.

Then images and annotation files must be arranged in a certain order. The $/datasets/$ directory which is created inside project root directory. Every dataset as mentioned in table 3 should have training and validation set. For example, /scrap_yard_original/ should contain /images/ and $/labels/$ directories dedicated for images and annotation files respectively. Each of these directories should have $/train/$ directories. Because the algorithm will look for training images inside $/image/train/$ and their annotations inside $/labels/train/$. Similarly, $/val/$ directories are ordered for the validation set. The filename of an image and its annotation file generated by CVAT are same. That is how the algorithm related image to its annotation file. The other dataset directories are ordered likewise. This dataset images and annotation files arrangement is illustrated in Figure 19.



```
Name
> data
v datasets
  > scrap_yard_both
  > scrap_yard_denoised
  > scrap_yard_hist_equalized
  v scrap_yard_original
    v images
      > train
      > val
    v labels
      > train
      > val
      train.txt
  datasets.dvc
  detect.py
  export.py
  hubconf.py
  LICENSE
```

Figure 19: Directory order for dataset images and annotation files

Then a configuration file (YAML file) is created in $/data/$ directory for custom training of YOLOv5 algorithm. This file should contain path to one of four datasets created in addition to path to images and labels. This also mentions number of classes or scrap types to be detected with their names. This value is 2 for this project because the project classifies turnings and rails scrap types.

The custom Python scripts, configuration file and standard YOLOv5 algorithm related code are uploaded to GitHub repository. The dataset images and annotation files are uploaded to S3 bucket using DVC. Then the project was cloned from GitHub and data files from S3 bucket using DVC on EC2 instance. The execution time needed to train the YOLOv5 algorithm for each run is reduced from average 38 hours on local machine to average 1 hour and 30 minutes when EC2 instance is used.

The YOLOv5 algorithm is trained on EC2 instance for 350 epochs with batch size of 16. The YOLOv5 algorithm has five different versions namely nano, small, medium, large and xlarge. The small version configured in $yolov5s.yaml$ is used for the training purpose considering the size of training dataset used in the project. The algorithm keeps updating output console with progress updates in P value, R value, mAP value, object loss, class loss, box loss and some other parameters for each epoch. These progression updates are received in output files (exp_name.out) for each experiment and stored in $/outputs/$ directory. The wandb library is also used for live visual tracking of these parameters.

The critical point to notice is that before starting of each epoch the training dataset gets augmented using Albumentation. Each time after applying augmentation transformations the training images get changed. Therefore, the model weights get updated on new training set for each epoch. This makes the effective training samples for the model as

$$Effective\ Training\ images = \ Input\ training\ images \times number\ of\ epochs ... (ii)$$

The input training images are 211 as mentioned in table 4. Therefore,

$$Effective\ Training\ images = \ 211 \times 350 \ = \ 73850$$

In other words, we could generate 73,850 data samples for the model from just original 211 data points. In addition, the Albumentation library also transforms the bounding boxes if it is needed for certain transformations.

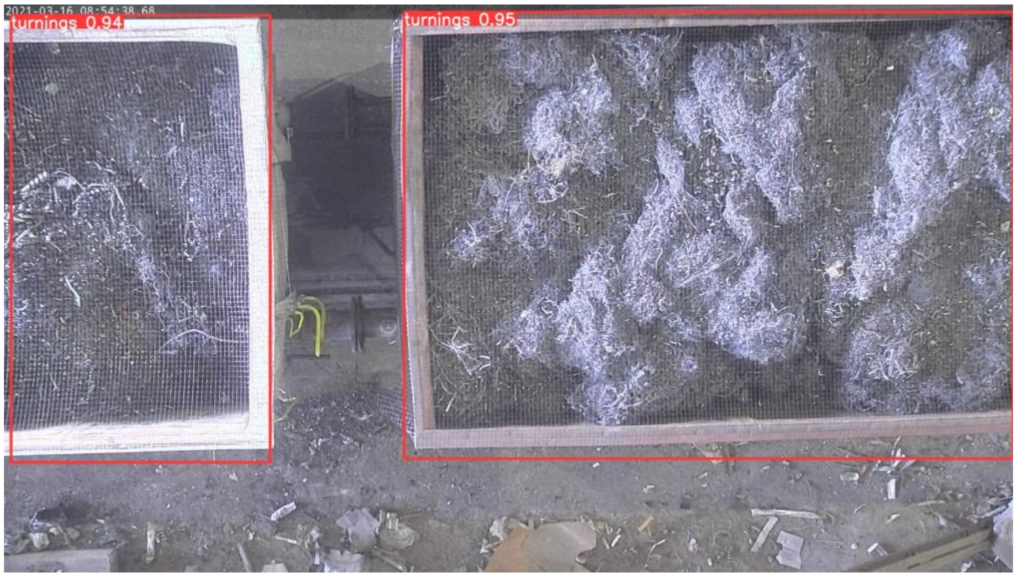
Once the training phase is over after 350 epochs, the algorithm produces trained model weights along with other performance metrices such as Precision curve, Recall curve, PR curve, graphs of loss functions for training and validation set, mean Average Precision (mAP) graphs, labels correlogram, confusion matrix and some other files useful for the performance evaluation.

This process of training the algorithm is executed 10 times separately as mentioned in Table 2. The model weights and performance evaluation related files for each run are uploaded to S3 bucket using DVC. So that these files can be accessed from other project instances.

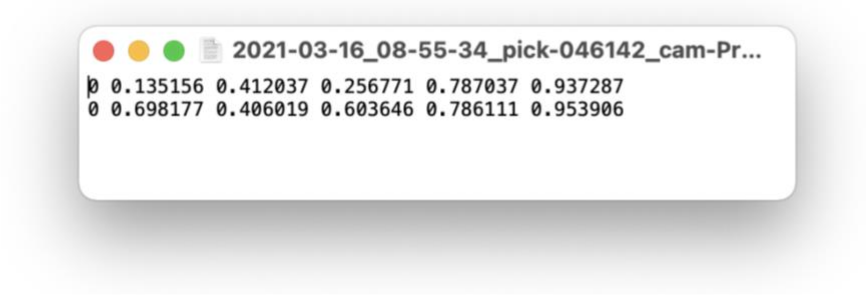
4.5 Implementation of YOLOv5 Inference

The trained model weights and other performance related files are obtained from S3 bucket for all runs using DVC on local machine. The algorithm is executed using trained model weights on test dataset to classify scrap types in the images.

The inference experiments are performed for 10 different model weights available after the training phase. The confidence threshold value of 0.5 was used for the detection. That means the algorithm would show a bounding box around a detected scrap only when it is at least 50% sure if that scrap belongs to one of the declared scrap types. The results obtained are also saved in text form inside $/runs/detect/exp/labels/$ directory in addition to presenting them in images.



(a)



```
0 0.135156 0.412037 0.256771 0.787037 0.937287
0 0.698177 0.406019 0.603646 0.786111 0.953906
```

(b)

Figure 20: (a) Image from test dataset with predicted bounding boxes, scrap type and confidence score (b) Text file containing labels for the same image

The output images produced by each experiment contain bounding boxes enclosing the detected scrap type. There is a class label associated with every bounding box denoting the name of the scrap type and a confidence score presenting the assertiveness of the algorithm in the prediction of scrap type. The example of scrap type detected in test dataset image can be seen in figure 20 (a).

The results saved in text files contain coordinates of predicted bounding boxes with class labels and a confidence score. The representation of bounding boxes for detected objects is the same as for the bounding boxes in annotated input images. The difference between annotated labels and predicted labels is that the coordinates of each bounding box are followed by a floating point confidence score as shown in figure 20 (b). These text files are then used for results analysis once the inference phase is over.

5. Results and Discussion

5.1 Result parameters description

There are certain parameters needed to be discussed to better understand the performance of YOLOv5 model. Moreover, certain result parameters are used in a way to facilitate the comparison of different model runs for training and inference.

The scrap types are classified by scrap type name assigned to detected scrap in an image as a label. There are four possible cases as follows when predicted labels and their actual value or ground truth is considered.

1. True Positive (TP): The label is predicted by the model as well as it matches with ground truth.
2. True Negative (TN): The label is not predicted by the model as well as not available in ground truth.
3. False Positive (FP): The label is predicted by the model, but it is not available in ground truth.
4. False Negative (FN): The label is not predicted by the model, but it is available in ground truth.

Confusion matrix is a way to concisely visualize above attributes in a tabular form.

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Table 5: Confusion matrix

Precision (P) is a way of determining how many correct predictions are made (TP) out of all predictions (TP + FP). Therefore,

$$Precision = \frac{TP}{TP + FP} \quad \ldots(iii)$$

Similarly, Recall (R) is a way of determining how many objects are correctly detected (TP) out of all available objects (TP + FN). Therefore,

$$Recall = \frac{TP}{TP + FN} \quad \ldots(iv)$$

The table 5, equation (iii) and (iv) show interpretation of confusion matrix for 2 classes. The definition of the confusion matrix can be generalized for more than 2 classes [27].

Intersection over Union (IoU) is a ratio of common area of two bounding boxes over the area of their union. This is generally used to quantify the accuracy of predicted bounding box. If there are two bounding boxes $pred$ and $truth$ then mathematically IoU_{truth}^{pred} is

$$IoU_{truth}^{pred} = \frac{common\ area\ of\ pred\ and\ truth}{union\ area\ of\ pred\ and\ truth}$$

Generally, it is tried to find a trade-off between Precision and Recall. The Precision-Recall (PR) curve is drawn by plotting P versus R values when different values of IoU threshold is used for the algorithm. The average precision (AP) is calculated by averaging the P values for all corresponding R values. Mathematically, AP is

$$AP = \frac{1}{|\{Recall_i\}|} \sum_{Recall_i} Precision(Recall_i) \quad \ldots(v)$$

where $Recall_i$ denotes the *i*th R value and $Precision(Recall_i)$ as P value corresponding to that R value. If PR-curve can be plotted as continuous function, then the area under the PR-curve is calculated to find the value of AP.

The mean Average Precision (mAP) is calculated by averaging the AP for each class.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \quad \ldots(vi)$$

where, N denotes number of object classes and AP_i is AP value for *i*th object class. The value ranges between [0, 1]. The higher mAP value means more correct detections are predicted by the model without making false predictions [28].

Another term to find the trade-off between P and R is F1-score. It combines P and R by taking their harmonic mean. F1-score is given by,

$$F1\ score = \frac{2 * (P * R)}{P + R} \quad \ldots(vii)$$

The value also has range of [0, 1] and higher F1 score indicates more balance between P and R values [29].

The loss functions are measure of deviation in predicted values with respect to the actual values for one or more features. There are three loss functions utilized by the YOLOv5 algorithm to optimize the performance. The performance metrics generated by the algorithm using training and validation datasets are box loss, object loss and class loss. A short description of these loss functions is provided as follows

1. *box loss*: It implies how precisely predicted bounding box covers the actual bounding box.

2. *object loss*: objectivity shows the probability of bounding box containing the object.
3. *class loss*: Class loss is measure of correctly classifying the class label

The model tries to minimize these loss function values with each epoch during training phase [30].

The algorithm produces two parameters in performance metric mAP_0.5 and mAP_[0.5, 0.95] which can be defined as follows

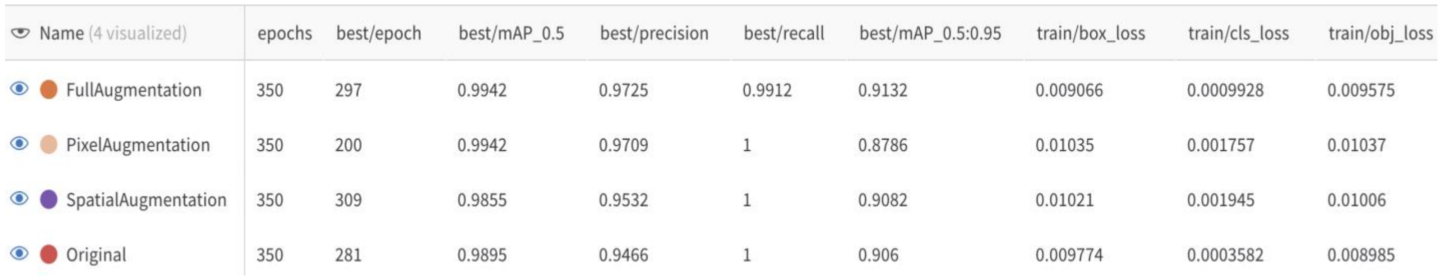
$$mAP_{0.5} = mean\ Average\ Precision\ for\ IoU\ =\ 0.5 \quad \ldots (viii)$$
$$mAP_{[0.5,0.95]} = mean\ Average\ Precision\ \forall\ IoU\ \in\ [0.5, 0.55, .., 0.95] \ldots (ix)$$

5.2 Results analysis and discussion

There are different experiments performed on YOLOv5 algorithm as described in table 2. It produces different trained models and then the object detection is performed on test dataset using these models to study the algorithm performance.

The algorithm runs grouped in Group1 according to table 2 provides insights about the effects of augmentation operations applied on dataset in different ways. Similarly, Group2 and Group3 involve the runs with image preprocessing operations and preprocessing as well as augmentation operations performed on the dataset respectively.

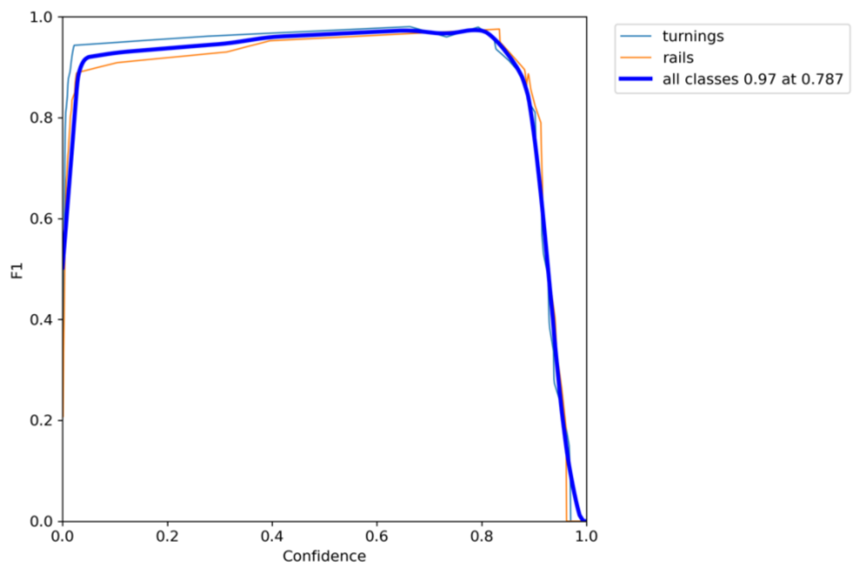
The *wandb* visualization tool is used to compare the performance of different runs. The performance parameters comparison of the runs belonging to Group1 are presented in table 6. We can observe that the $mAP_{0.5}$, $mAP_{[0.5,0.95]}$, Precision and Recall values are improved when augmented input images are used compared to the original input images for the training phase of YOLOv5 algorithm. Precisely, 0.47%, 0.79% and 2.73% of improvement is observed for $mAP_{0.5}$, $mAP_{[0.5,0.95]}$ and Precision values for the algorithm runs when dataset having both pixel level and spatial level augmented images (FullAugmentaion as shown in Table 6) are used compared to the original dataset.



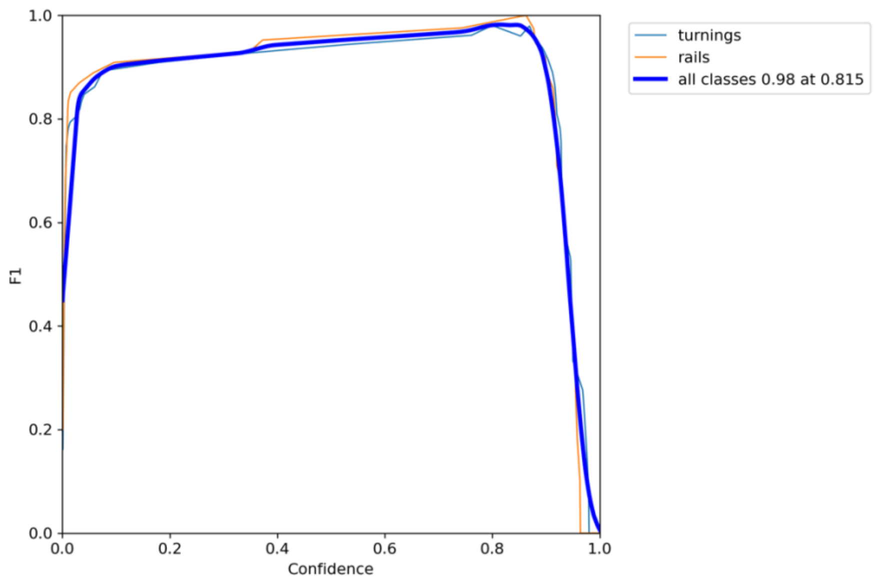
Name (4 visualized)	epochs	best/epoch	best/mAP_0.5	best/precision	best/recall	best/mAP_0.5:0.95	train/box_loss	train/cls_loss	train/obj_loss
● FullAugmentation	350	297	0.9942	0.9725	0.9912	0.9132	0.009066	0.0009928	0.009575
● PixelAugmentation	350	200	0.9942	0.9709	1	0.8786	0.01035	0.001757	0.01037
● SpatialAugmentation	350	309	0.9855	0.9532	1	0.9082	0.01021	0.001945	0.01006
● Original	350	281	0.9895	0.9466	1	0.906	0.009774	0.0003582	0.008985

Table 6: Performance parameter values for group1 runs

For FullAugmentaion run, the Recall gets decreased, but this increases the Precision value resulting in improvement of F1-score from 0.97 to 0.98 as shown in figure 21. This is because Precision and Recall values 0.9725 and 0.9912 are more balanced for dataset corresponding to figure 21 (b).



(a) F1-curve for original dataset

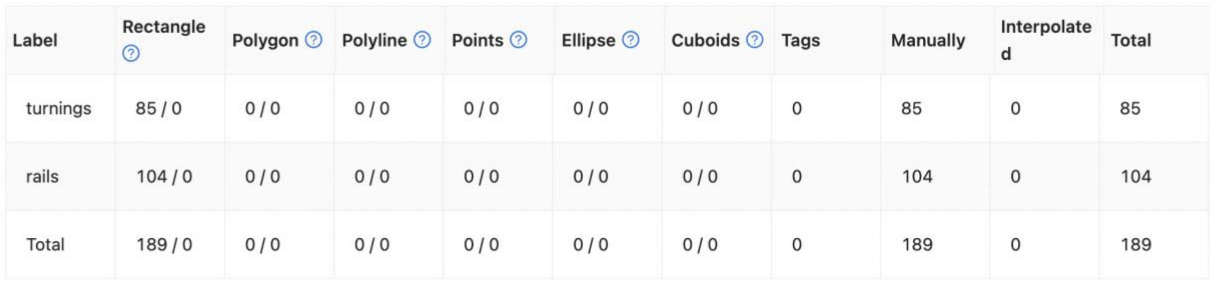


(b) F1-curve for dataset augmented with pixel and spatial level transformations

Figure 21: Comparison of F1-score for original dataset and dataset augmented with pixel and spatial level transformations

The stand-alone pixel level augmentation causes early stopping of the algorithm at 300 epochs when patience value of 100 epochs was used. Although in terms of loss functions there is no regular improvement observed for this run. This significantly reduces the training time and computational cost required to train the algorithm.

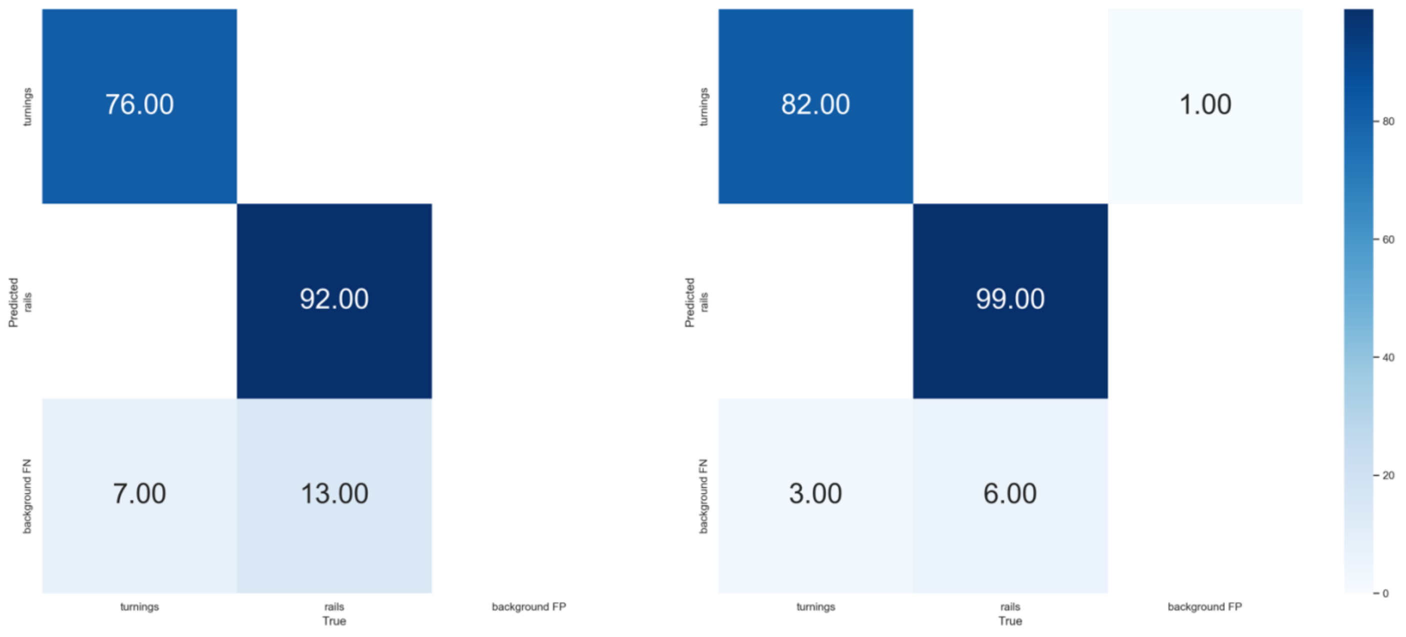
The training of YOLOv5 is improved slightly when pixel level or spatial level transformations are applied as a stand-alone operation, but it improves significantly when both kinds of operations are applied on the dataset images. This is evident with the inference performed on the test dataset.



Label	Rectangle ⓘ	Polygon ⓘ	Polyline ⓘ	Points ⓘ	Ellipse ⓘ	Cuboids ⓘ	Tags	Manually	Interpolated	Total
turnings	85 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0	85	0	85
rails	104 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0	104	0	104
Total	189 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0	189	0	189

Table 7: Annotation Statistics

There is total 189 scrap object instances present in test dataset out of which 85 are turnings type and 104 are rails type as shown in Table 7.



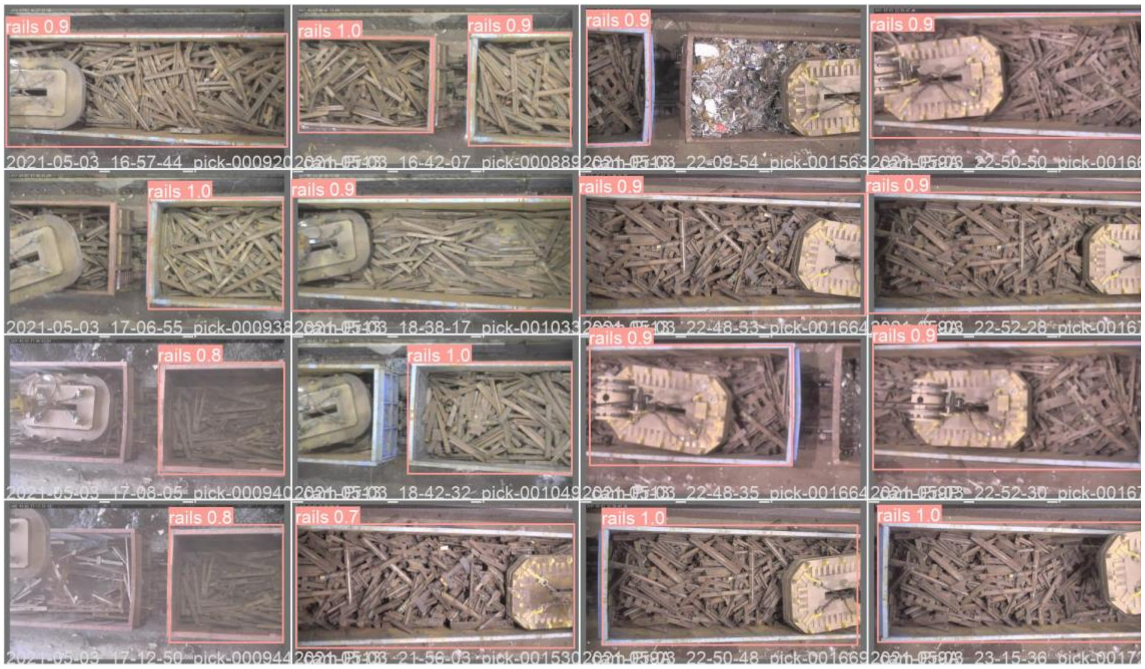
(a) Original dataset (b) augmented dataset

Figure 22: Comparison of confusion matrix for original dataset and augmented dataset

The confusion matrix is obtained from inference using model weights trained on original dataset and augmented (from now meaning both pixel and spatial level transformations) dataset. The comparison of resulting confusion matrix is shown in figure 22.

There were 7 and 13 instances of turnings and rails respectively which were not detected by the algorithm when it was trained on original dataset. This gets reduced to only 3 and 6 instances of turnings and rails when algorithm is trained on augmented images. Therefore, we can deduce that augmentation improves the algorithm performance in terms of detecting both true positive (TP) cases and false negative (FN) cases. There is no misclassification in determining the scrap type observed in both cases.

The comparison of predictions in batch by both cases is shown in figure 23. It illustrates that how algorithm trained on original dataset miss to detect scrap types in bottom 3 images of column 1 in figure 23 (a). While in the other case, these instances are detected with high confidence value as shown in figure 23 (b). This explains that the augmentation decreases the FN cases observed in confusion matrix comparison.


(a) original dataset

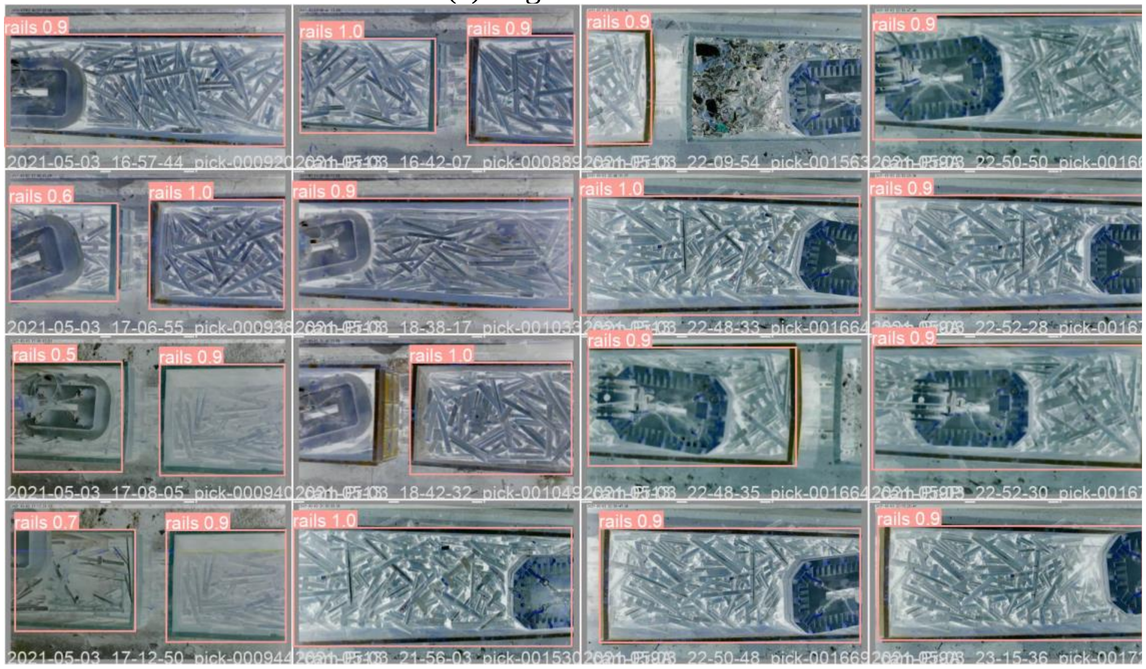
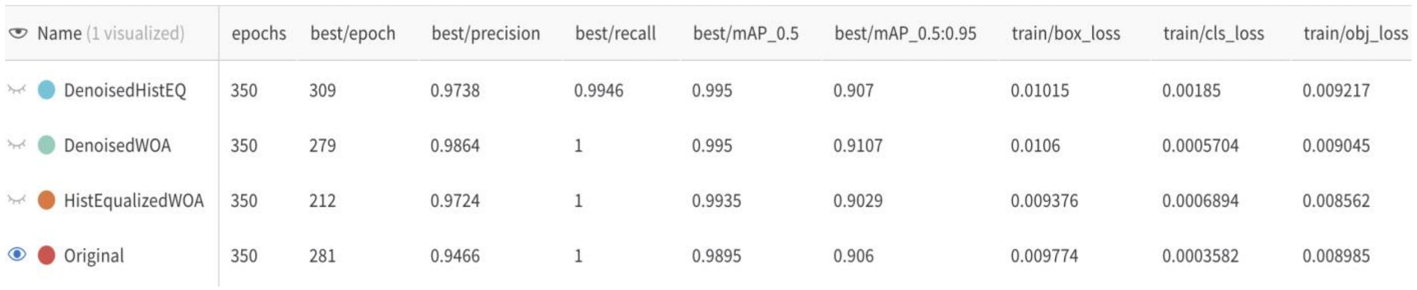

(b) augmented dataset

Figure 23: prediction comparison of models trained on original and augmented dataset

The insights discussed above proves that augmentation contributes to the improvement of algorithm performance parameters such as F1-score, mAP value, FN cases.

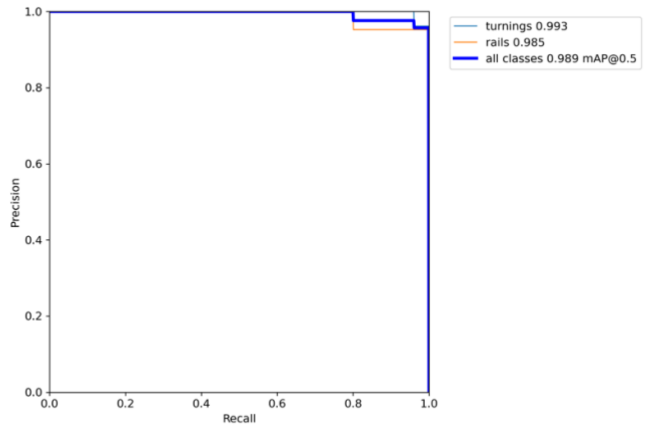
The algorithm runs in Group2 used training datasets when either denoising or histogram equalization operation or both operations were performed on dataset images. The algorithm



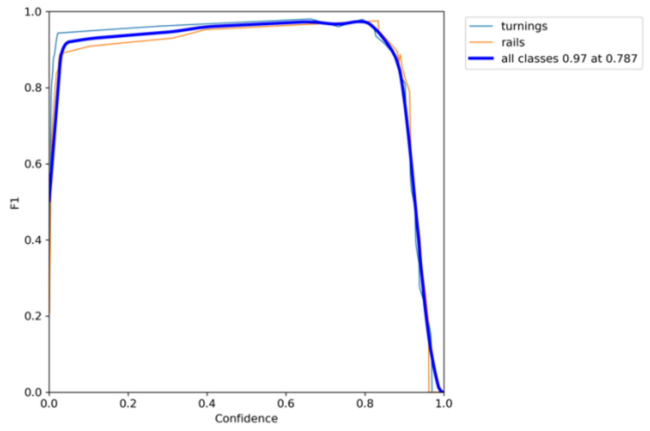
Name (1 visualized)	epochs	best/epoch	best/precision	best/recall	best/mAP_0.5	best/mAP_0.5:0.95	train/box_loss	train/cls_loss	train/obj_loss
● DenoisedHistEQ	350	309	0.9738	0.9946	0.995	0.907	0.01015	0.00185	0.009217
● DenoisedWOA	350	279	0.9864	1	0.995	0.9107	0.0106	0.0005704	0.009045
● HistEqualizedWOA	350	212	0.9724	1	0.9935	0.9029	0.009376	0.0006894	0.008562
● Original	350	281	0.9466	1	0.9895	0.906	0.009774	0.0003582	0.008985

Table 8: Performance parameter values for group2 runs

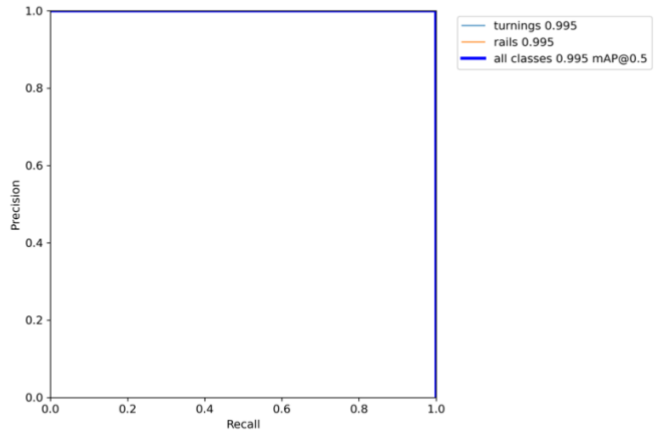
performs better when compared to the original dataset. The algorithm converges at 312 epochs (patience = 100 epochs) when histogram equalized images are used as input. This makes histogram equalization operation suitable for faster training of the algorithm compa-



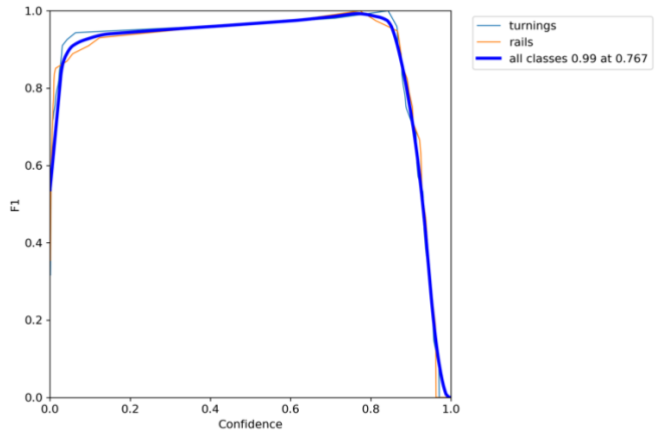
(a) PR curve for original dataset



(c) F1-curve for original dataset



(b) PR curve for denoised dataset

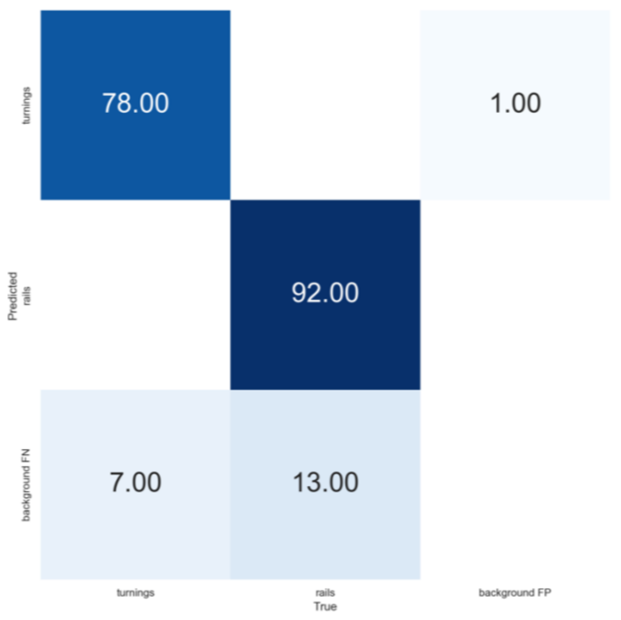


(d) F1-curve for denoised dataset

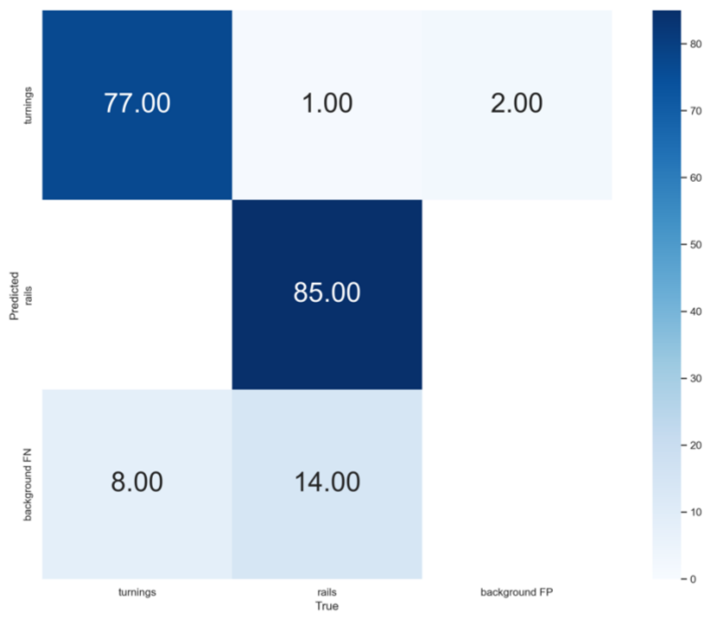
Figure 24: Comparison of PR-curve and F1-curve for algorithm training on original and denoised dataset

-red to using unprocessed images as input. However, this trend is not observed for when denoising is performed on the input images.

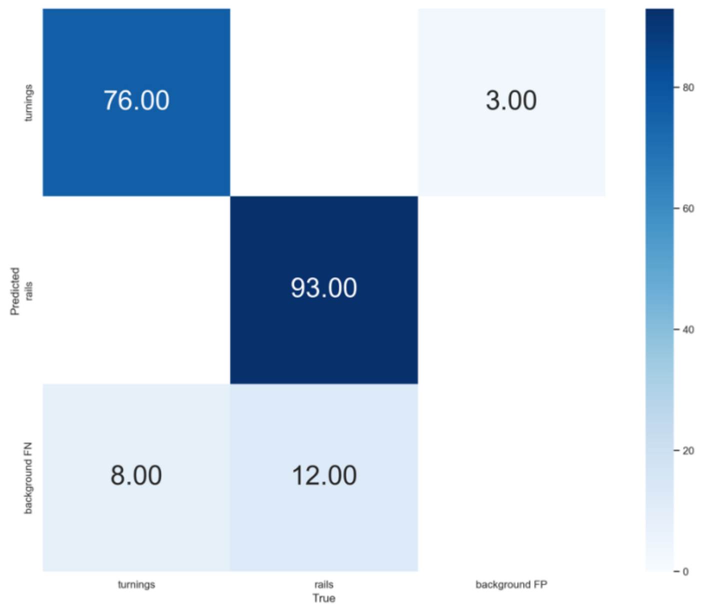
The stand-alone denoising operation improves the performance parameters precision, $mAP_{0.5}$, $mAP_{[0.5,0.95]}$ and F1-score by 2.87%, 0.55%, 0.11% and 2.06% respectively compared to original dataset used during training. It is better than the stand-alone histogram equalization and histogram equalization with denoising as shown in table 8. The improvement in Area Under the Curve (AUC) of PR-curve is shown in figure 24 (a), (b) while F1-score values can be observed in figure 24 (c), (d).



(a) denoised dataset

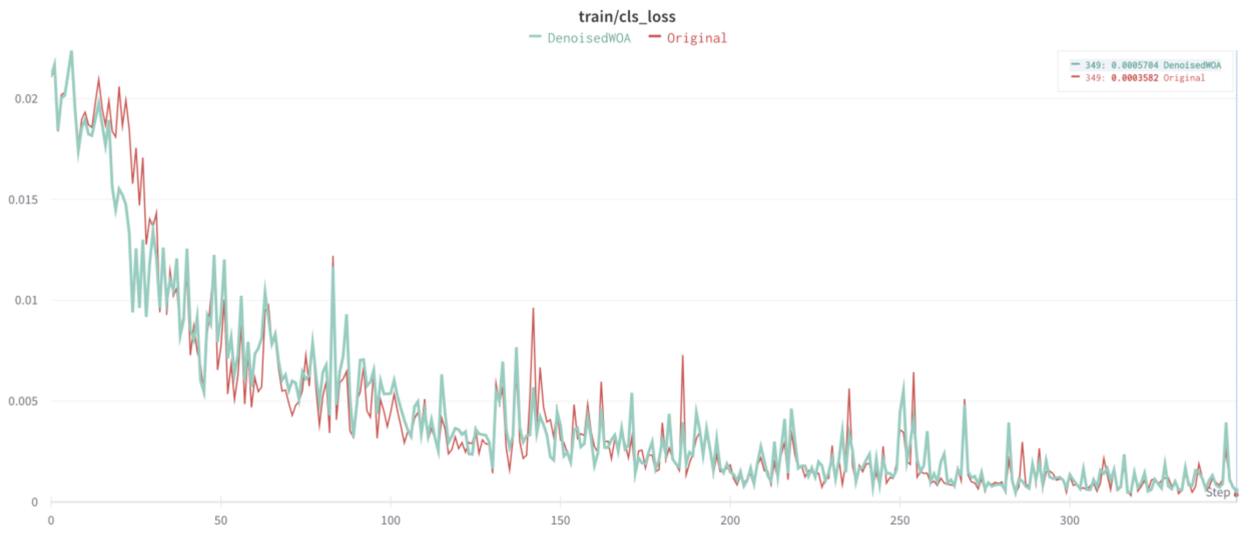


(b) histogram equalized dataset

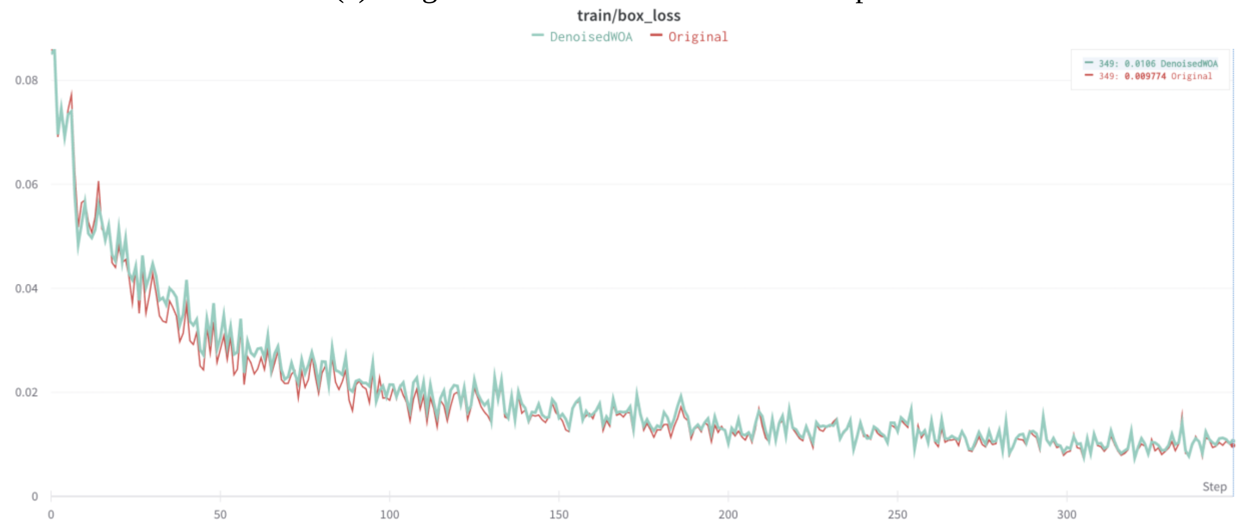


(c) denoised and histogram equalized dataset

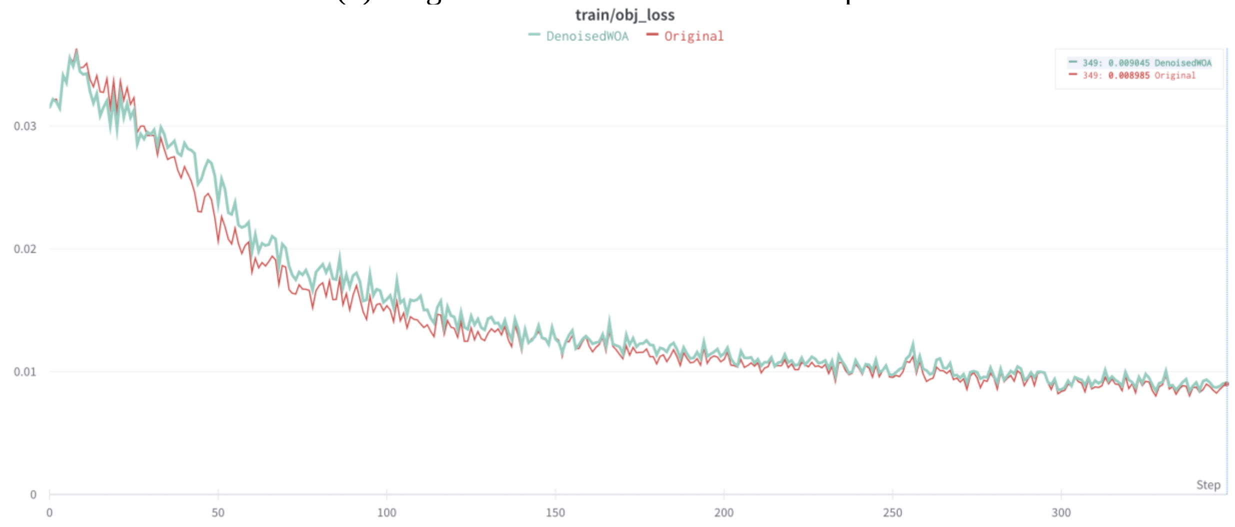
Figure 25: Comparison of confusion matrix for denoised, histogram equalized and their combination on dataset



(a) Progression of class loss value with epoch



(b) Progression of box loss value with epoch



(c) Progression of object loss value with epoch

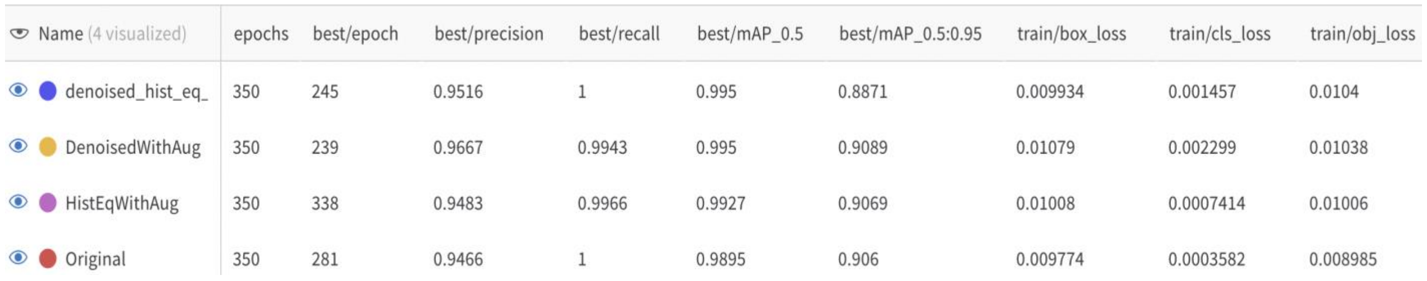
Figure 26: Comparison of loss functions for original dataset and denoised dataset

The comparison of confusion matrix obtained by performing inference on test dataset for denoised, histogram equalized and denoised as well as histogram equalized training datasets is presented correspondingly in figure 25 (a), (b) and (c). When these confusion matrices are compared with figure 23 (a), we can observe that there is no significant improvement in terms of detecting scrap types. But the model trained on denoised dataset provides more TP cases compared to two other cases.

The progression of loss values namely class loss, box loss and object loss are compared between original dataset and denoised dataset as shown in figure 26 (a), (b) and (c) respectively. The graph plots almost overlap each other and shows that the loss function is almost same for both cases.

It can be concluded from above observations that only denoising operation improves the detection of scrap types not significantly but by a small margin. The probable reason of no significant observable gain is the smaller number of instances of scrap types available in test dataset. Another conclusion that can be drawn is that the denoising operation improves the algorithm performance better than the CLAHE operation. In addition, the denoising operation improves image quality better than when it is coupled with CLAHE operation.

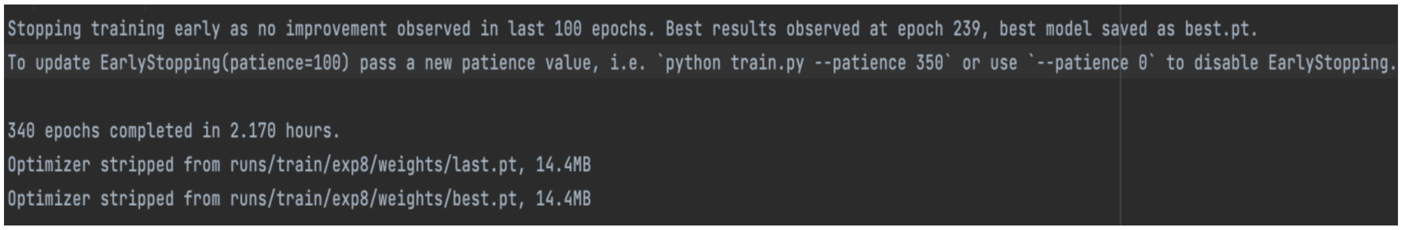
In Group3 set of training datasets, the performance of YOLOv5 algorithm was analysed when the augmentation combined with the image processing operations are performed on input images.



Name (4 visualized)	epochs	best/epoch	best/precision	best/recall	best/mAP_0.5	best/mAP_0.5:0.95	train/box_loss	train/cls_loss	train/obj_loss
● denoised_hist_eq_	350	245	0.9516	1	0.995	0.8871	0.009934	0.001457	0.0104
● DenoisedWithAug	350	239	0.9667	0.9943	0.995	0.9089	0.01079	0.002299	0.01038
● HistEqWithAug	350	338	0.9483	0.9966	0.9927	0.9069	0.01008	0.0007414	0.01006
● Original	350	281	0.9466	1	0.9895	0.906	0.009774	0.0003582	0.008985

Table 9: Performance parameter values for group3 runs

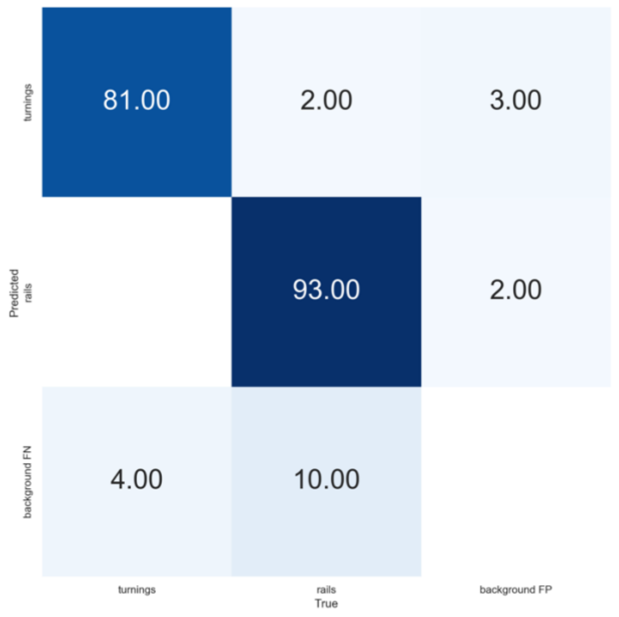
The best algorithm run in Group3 is observed when denoising operation is combined with augmentation operations. The performance parameters precision, $mAP_{0.5}$, $mAP_{[0.5,0.95]}$ and F1-score are improved by 2.12%, 0.55%, 0.32% and 1.06% respectively compared to the original dataset as shown in table 9. This improvement is comparatively better than the other two cases of Group3 which are histogram equalization with augmentation and denoised as well as histogram equalized in combination with augmentation.



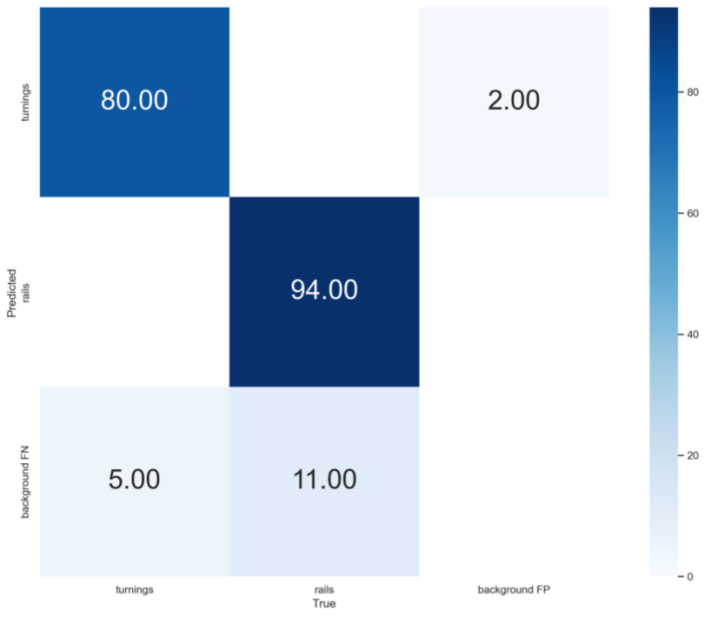
```
Stopping training early as no improvement observed in last 100 epochs. Best results observed at epoch 239, best model saved as best.pt.
To update EarlyStopping(patience=100) pass a new patience value, i.e. `python train.py --patience 350` or use `--patience 0` to disable EarlyStopping.

340 epochs completed in 2.170 hours.
Optimizer stripped from runs/train/exp8/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp8/weights/best.pt, 14.4MB
```

Figure 27: Early stopping during training for denoised and augmented training dataset

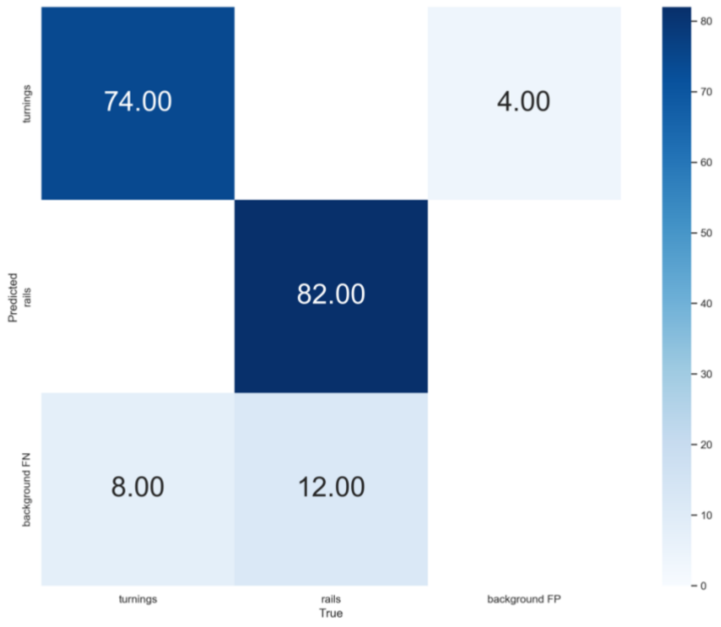
The early stopping was observed for training at 340 epochs (patience=100 epochs) as for denoising with augmentation case as shown figure 27. It is also observed that the denoising



(a) denoised and augmented dataset



(b) histogram equalized and augmented dataset



(c) histogram equalized, denoised and augmented dataset

Figure 28: Comparison of confusion matrix for augmented dataset pre-processing with denoising, histogram equalization and their combination

operations performed training dataset before augmentation improves the scrap type detection. The original dataset detects the turnings scrap type with benchmarking accuracy of 89.41% which gets increased to 95.29 % and for the rails scrap type it improves from 88.46% to 89.42% as shown in figure 28 (a).

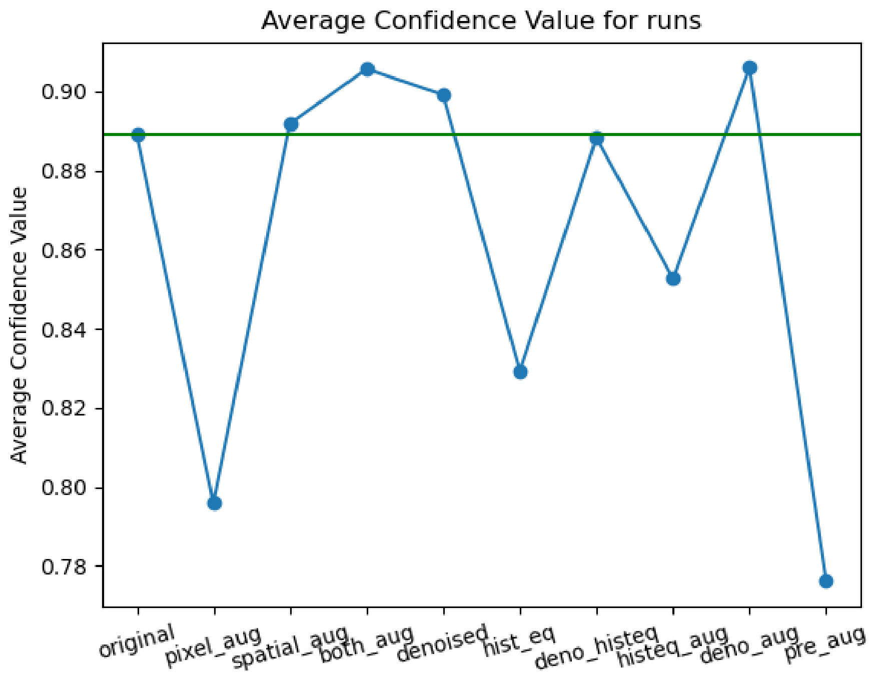


Figure 29: Average confidence value for different runs

The average confidence value was calculated for TP cases of detected scrap types for each run using custom Python script. The average confidence value is plotted for each run as shown in figure 29. The meaning of labels used on X-axis in figure and absolute values of average confidence for each run can be found in table 10.

Operation performed on dataset	Label	Average Confidence Value
Original Dataset	original	0.8890
Pixel level augmentation	pixel_aug	0.7959
Spatial level augmentation	spatial_aug	0.8918
Both types of augmentation	both_aug	0.9056
Denoised	denoised	0.8991
Histogram equalised	hist_eq	0.8292
Denoised and histogram equalised	deno_histeq	0.8882
Denoised with augmentation	deno_aug	0.9060
Histogram equalised with augmentation	histeq_aug	0.8526
Histogram equalised and denoised with augmentation	pre_aug	0.7760

Table 10: Absolute values of average confidence for different runs and their labels used in plot

The green line in figure 29 shows the benchmarking average confidence value for the algorithm trained on original dataset. The plot concludes our analysis so far that augmented dataset (pixel and spatial level), denoised dataset and denoising with augmentation provides better results than the benchmarking performance of the algorithm.

Out of these three runs the algorithms performs best in case of denoised images and performing augmentation during training before starting of each epoch. The algorithm improves in terms of training time as early stopping was observed for this case. It improves performance parameters like Precision, Recall, mAP, F1-score as already discussed. The accuracy is increased from 88.88% to 92.06% as observed in confusion matrix the detection of scrap types gets increased as well. Therefore, conclusively the overall performance in terms of training time and accuracy the algorithm increases in this case.

6. Summary and Future Work

The metal scraps are received in scrap yard in the railway carriages every day. These railway carriages contain different types of scrap metals which needs to be sorted so that they could be sent to appropriate EAF for further processing. This process of sorting is performed manually at present. There are lots of disadvantages because of the manual efforts involved. Therefore, it is necessary to automate the process of scrap type classification to reduce human errors and required time.

We introduce a software application as a solution for this problem. This software automates the task of classification of scrap types using computer vision and machine learning algorithm YOLOv5. Practically, there are many scrap metal types received in the scrap yard, but only two types of scrap metals are chosen namely turnings and rails for the simplicity of the project. The images of the scrap metal in railway carriages are taken from different angles. But the images from above railway carriages are used as input for the algorithm in this project. The number of images used of both scrap types in the datasets are also balanced to avoid any biasing in the training of the algorithm.

The images taken in scrap yard get degraded because the industrial environment does not provide ideal environment for taking pictures. The input images are preprocessed initially using image processing operations before using them for training of the algorithm. The non-local means denoising operation and CLAHE operations are used to improve the input image quality so that the existing algorithm performance can be improved. The implementations of these operations available in OpenCV library are used for this purpose.

The images are then annotated to utilize them as truth values for the algorithm. The open-source tool CVAT hosted on the server of SMS digital is used to annotate the images. The rectangular bounding box is used as annotation shape and YOLO annotation format is used to generate label files. The dataset images and their label files both work as input for the YOLOv5 algorithm.

The input images are augmented as well during the training for each epoch to diversify the dataset and avoid overfitting. This operation significantly reduces the requirement of using large training dataset images as well as annotating them. The augmentation operation transforms the original training dataset containing only 211 images into effectively 73,850 distinct images for the training of YOLOv5 algorithm. The augmentation specific library Albumentation is used for transforming images as well as annotations. This removes the extra effort required to annotate the newly created images. There are two types of augmentation operations namely pixel level and spatial level transformations used in this project. The pixel level transformations used are blur, median blur, conversion to greyscale, and random brightness or contrast. While spatial level transformations used are vertical flip, horizontal flip, invert image operations. These operations are used in way that any one of these operations from each type is used at a time.

The codebase of the project is maintained on GitHub and the image dataset was synchronized on S3 bucket. This helped with the maintenance of the project instances between local machine and on the cloud. The algorithm is trained on accelerated computing version of EC2 instance for 350 epochs. The algorithm could generate trained models rapidly compared to the execution time on local machine.

Then different combination of pre-processing operations and augmentation operations based on their type are used to create different set of training images. There are total 10 different sets of training images made as shown in table 2. Then a comparative study of the algorithm performance is conducted. This showed us that the algorithm accuracy gets increased by roughly 4% for the task of classification of scrap types, when input images are denoised and then augmentation of both types is performed before using them for the training of the algorithm. The algorithm converges earlier when this set of images are used compared to the original images. This reduces the training time and computational cost required to train the algorithm.

This project has tremendous scope in terms of further developments. The initial step of improving the quality of images can be a topic of research for this use case of scrap type classification. We can perform comparative study for more image processing operations suitable for this use case. The Albumentation library has several augmentation operations available. The choice of transformation operation depends on the computer vision task at hand. This also significantly affect the algorithm performance. There is a possiblity to improve the performance parameters further by trying other transformations.

This project uses only two scrap types at present, but more scrap types can be added to further develop the project for industrial purposes. In addition, the project currently focuses on just the classification of the scrap types but there are other manual processes currently in practice for scrap yard management. This concept of automating the scrap type classification using computer vision and machine learning could be used to detect the quality of the scrap metals. The project can include a feature to detect the unwanted goods mixed in scraps. The logging of the received scrap metals can be automated and that data can later be analyzed to take informed business decisions.

7. References

[1] "Image Annotation: Definition, Use Cases & Types" V7 Labs. retrieved on 16 May 2022.

[2] "Different Annotation Formats" Albumentations.ai. Retrieved on 13 May 2022.

[3] Xiangxin Zhu, Carl Vondrick, Charless Fowlkes, Deva Ramanan (2015). " Do We Need More Training Data?". arXiv:1503.01508.

[4] Hawkins, D. M (2003). "The Problem of Overfitting". J. Chem. Inf. Comput. Sci. 2004, 44, 1−12. DOI: 10.1021/ci0342472.

[5] "What is image augmentation and how it can improve the performance of deep neural networks". https://albumentations.ai/docs/introduction/image_augmentation/. Retrieved on 21 May 2022.

[6] "Amazon SageMaker Data Labelling Pricing". Amazon SageMaker. Retrieved on 7 July 2022.

[7] Bradski, G. "The OpenCV Library". Dr. Dobb's J. Softw. Tools 2000. https://www.drdobbs.com/open-source/the-opencv-library/184404319. Retrieved on 21 February 2022.

[8] Clark, A. 2010. "Pillow". https://python-pillow.org. Retrieved on 21 February 2022.

[9] "Albumentations". https://albumentations.ai. Retrieved on 21 May 2022.

[10] Cubuk, Ekin D. and Zoph, Barret and Mane, Dandelion and Vasudevan, Vijay and Le, Quoc V. (2018). "AutoAugment: Learning Augmentation Policies from Data". arXiv:1805.09501.

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi (2016). "You only look once: Unified, real-time object detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. arXiv:1506.02640.

[12] Everingham, M. and Van Gool, L. and Williams, C. K. I. and Winn, J. and Zisserman, A. "The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results". http://host.robots.ox.ac.uk/pascal/VOC/voc2008/index.html.

[13] Hsin C. "Yolo Object Detectors: Final Layers and Loss Functions". Article on medium.com. Retrieved on 14 May 2022.

[14] Alisha P B, Gnana Sheela K (2016). "Image Denoising Techniques-An Overview". IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834. Volume 11, Issue 1, Ver. I (Jan. - Feb .2016), PP 78-84.

[15] Win, Ni & Kyaw, Khin & Win, Thu & Aung, Phyo. (2019). "Image Noise Reduction Using Linear and Nonlinear Filtering Techniques". International Journal of Scientific and Research Publications (IJSRP). 9. p92113. 10.29322/IJSRP.9.08.2019.p92113.

[16] Fan, L., Zhang, F., Fan, H. et al. Brief review of image denoising techniques. Vis. Comput. Ind. Biomed. Art 2, 7 (2019). https://doi.org/10.1186/s42492-019-0016-7.

[17] Yeong-Taeg Kim, "Contrast enhancement using brightness preserving bi-histogram equalization," IEEE Trans. Consum. Electron., vol. 43, no. 1, pp. 1–8, 1997. doi: 10.1109/30.580378.

[18] C. H. Ooi and N. A. Mat Isa, "Quadrants dynamic histogram equalization for contrast enhancement," in IEEE Transactions on Consumer Electronics, vol. 56, no. 4, pp. 2552-2559, November 2010, doi: 10.1109/TCE.2010.5681140.

[19] S. K. Shome, S. Ram, and K. Vadali (2011). "Enhancement of Diabetic Retinopathy Imagery Using Contrast Limited Adaptive Histogram Equalization," Int. J. Comput. Sci. Inf. Technol., vol. 2, no. 6, pp. 2694–2699.

[20] C. Diya, P. Tanvi, S. Joshi and P. Ghanshyam (2015). "Image Segmentation using Morphological Operations". International Journal of Computer Applications. 117. 16-19. 10.5120/20654-3197.

[21] Das, Apurba. (2015). Interpretation and Processing of Image in Frequency Domain. 10.1007/978-3-319-14172-5_3.

[22] A. Buades, B. Coll, J. M. Morel (2011). "Non-local Means Denosing", Img. Proc. on Line (IPOL). https://doi.org/10.5201/ipol.2011.bcm_nlm

[23] "Computer Vision Annotation Tool: A Universal Approach to Data Annotation". https://www.intel.com/content/www/us/en/developer/articles/technical/computer-vision-annotation-tool-a-universal-approach-to-data-annotation.html. Retrieved on 27 February 2022.

[24] Buslaev, A.; Iglovikov, V.I.; Khvedchenya, E.; Parinov, A.; Druzhinin, M.; Kalinin, A.A. "Albumentations: Fast and Flexible Image Augmentations". Information 2020, 11, 125. https://doi.org/10.3390/info11020125.

[25] "Amazon S3". https://aws.amazon.com/s3/. Retrieved on 21 March 2022.

[26] "Versioning data and models". https://dvc.org/doc/use-cases/versioning-data-and-model-files. Retrieved on 25 March 2022.

[27] Manliguez, Cinmayii. (2016). "Generalized Confusion Matrix for Multiple Classes". 10.13140/RG.2.2.31150.51523.

[28] P. Henderson, V. Ferrari (2017). "End-to-end training of object class detectors for mean average precision". https://doi.org/10.48550/arXiv.1607.03476.

[29] Z. C. Lipton, C. Elkan, B. Naryanaswamy (2014). "Thresholding classifiers to maximize F1-score". arXiv:1402.1892v2.

[30] Kasper-Eulaers M, Hahn N, Berger S, Sebulonsen T, Myrland Ø, Kummervold PE. "Short Communication: Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5". *Algorithms*. 2021; 14(4):114. https://doi.org/10.3390/a14040114.

Bibliography

[1] Image of a cat in bounding box from COCO dataset. Retrieved on 22 May 2022. <https://albumentations.ai/docs/getting_started/bounding_boxes_augmentation/>

[2] Detection algorithm performance on PASCAL VOC dataset vs training data size. Xiangxin Zhu, Carl Vondrick, Charless Fowlkes, Deva Ramanan (2015). "Do We Need More Training Data?". arXiv:1503.01508.

[3] Augmentation transformation examples. Retrieved on 21 May 2022. <https://albumentations.ai/docs/introduction/image_augmentation/>

[4] Transformation applied on both input image and output mask. Retrieved on 22 May 2022.<https://albumentations.ai/docs/introduction/why_you_need_a_dedicated_library_for_image_augmentation/>

[5] "Yolo Object Detectors: Final Layers and Loss Functions". Article on medium.com. Retrieved on 14 May 2022. <https://medium.com/oracledevs/final-layers-and-loss-functions-of-single-stage-detectors-part-1-4abbfa9aa71c>

[6] Neural network architecture of YOLO model. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi (2016). "You only look once: Unified, real-time object detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. arXiv:1506.02640.

[7] Effect of histogram equalization on image and its histogram. Retrieved on 14 May 2022. <https://en.wikipedia.org/wiki/Histogram_equalization>