Preventive Maintenance using Object Detection in Rolling Stock images

Submitted in partial fulfillment of the requirements for the degree

of

Master of Technology

by

Aditya Dandriyal
142202003

Under the guidance of

Dr C.K. Narayanan

IIT PALAKKAD

DEPARTMENT OF DATA SCIENCE

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

IIT PALAKKAD

Department of Data Science
Indian Institute of Technology, Palakkad
India - 678623

CERTIFICATE

This is to certify that we have examined the thesis titled **Preventive Maintenance using Object Detection in Rolling Stock Images**, submitted by **Aditya Dandriyal** (Roll Number:**142202003**) a postgraduate student of **Department of Data Science** in partial fulfillment for the award of degree of **Master of Technology**. We hereby accord our approval of it as a study carried out in **Wabtec Corporation** and presented in a manner required for its acceptance in partial fulfillment for the Post Graduate Degree for which it has been submitted. The thesis has fulfilled all the requirements as per the regulations of **the Institute and that of Wabtec Corporation** and has reached the standard needed for submission.

Dr. CK Narayanan

Internal Guide
Head of Department
Department of Data Science
IIT, Palakkad

Abhishek Dubey

Director - Engineering
Wabtec Corporation

Place: Palakkad
Date: 16th May 2024

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest gratitude to my internal supervisor at IIT Palakkad, **Dr. C.K. Narayanan** and my mentor at Wabtec corporation **Nagul R (Staff Data Scientist, Kinetix MVA Team)**, for their constant guidance, support, and encouragement which was invaluable throughout the duration of this project. I am immensely grateful for their expertise, patience, and dedication, which have significantly contributed to the success of this work.

I would also like to extend my thanks to **Indian Institute Of Technology Palakkad, Department of Data Science**, its faculty members and **Wabtec Corporation**, for providing the necessary resources and conducive environment for research and learning.

Lastly, but certainly not least, I want to extend my heartfelt gratitude to all the individuals who have directly or indirectly played a role in shaping this project. Their insights, feedback, and assistance have really been invaluable.

Aditya Dandriyal
MTech Data Science
IIT Palakkad
Date: 16^{th} May 2024

ABSTRACT

This study explores the improvement of YOLOv8 model for the task of object detection in rolling stock images. It also shines light on the need for using augmentation techniques to generate potential use-cases due to the scarcity of image data for certain components of rolling stock images and further explores the comparizon of various image contrast enchancement technique to know which one best suits the object detection task.

To improve the performance of YOLOv8 for the task of object detection, we incorporate different types of attention modules with the YOLOv8 architecture of varying sizes (nano and small). The attention modules are chosen based on their efficacy in refining the feature representation while still keeping the computational overheads in check. The attention modules chosen were ResBlock + CBAM [1], GAM [2], ECA [3], SA [4].

Upon implementing the attention modules, findings reveal that for the YOLOv8 nano model the attention modules show no improvement over the baseline model, but for the YOLOv8 small model the ResBlock+CBAM and ECA attention module give superior performance as compared to the baseline.

This study also gives a rough insight on how a object detection model is chosen to be put into production for detection of rolling stock components using images.

Table of Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Wabtec is a leading global provider of equipment, systems, digital solutions, and value-added services for the freight and transit rail sectors. With two large business entities in freight and transit rail, the company has supplied more than 300 locomotives to Indian Railways and have an installed base of subsystems in over 2,000 cars in Metro projects.

At Wabtec, the **Kinetix Machine Vision Algorithms Team** works at automated, proactive monitoring of Rolling Stock (Engines and Carriages used by railways) condition, thereby providing data that can be processed to effectively assess rolling stock condition from component level to full train inspection. Understanding component condition in real time, enables maintenance cost savings, helps prevent costly incidents, decreases operational delays, and increases the predictability of long-term maintenance scheduling.

1.1 Objective

Monitoring of the rolling stock components is done using Object Detection Models (YOLOv8) on image data collected using cameras positioned at the required customer site.

Some of the major challenges faced during object detection are as follows: It's a bit difficult to detect components which are a bit smaller in size. Sometimes captured images suffer from challenging lighting conditions. Further the dataset for the

images is vastly limited and also there are none to very few images for defective/faulty cases.

The objective is to improve the performance/detection capability of the Object Detection model (YOLOv8) such that we are able to overcome all or most of the challenges mentioned above.

1.2 Motivation

The motivation behind detecting the objects is to assess the rolling stock component condition by either detecting the presence/absence of particular elements of that component or by using the positional co-ordinates of the elements in the component to deduce the conditon of the component and proceed accordingly. These steps after object detection are done during post processing.

Chapter 2

LITERATURE REVIEW

Attention mechanisms, which enable neural networks to accurately focus on all the relevant elements of the input, have become an essential component to improve the performance of deep neural networks. They have also proven to be a potential means to enhance deep CNNs and thus can improve the performance of a broad range of computer vision tasks, e.g., image classification, object detection, and instance segmentation.

Attention mechanism has obtained excellent results in the field of object detection. With the integration of attention mechanism, object detection models are able to learn better feature representation while suppressing unnecessary information.

There are mainly two attention mechanisms widely used in computer vision tasks, namely spatial attention which captures the inter-spatial (pixel-level pairwise) relationship between features and channel attention which captures the inter-channel dependency between the features. Subsequently the development of the attention modules can be roughly divided into two directions: enhancement of feature aggregation and combination of channel and spatial attention, while also limiting the computatinal overhead as much as possible.

Based on this some of the attention modules considered are as follows:

2.1 CBAM: Convolutional Block Attention Module [Woo et al.,2018]

This paper proposes a network module, "Convolutional Block Attention Module [1]". Based on convolution operations that extract informative features by blending cross-channel and spatial information together, this module extracts meaningful features along the channel and spatial axes.

To enhance performance of CNNs, recent researches have mainly targeted three factors width, depth, and cardinality of the network, while this work focuses on one of the curious facets of human visual system, namely attention.

To achieve this, a channel and a spatial attention module were applied sequentially, so that the network can learn what and where to attend in the channel and spatial axes respectively. Thus, this module within the network learns which information to emphasize or suppress. CBAM module is shown in Fig. 2.1.

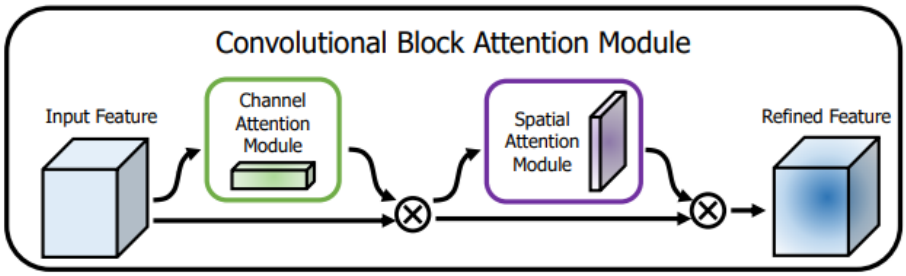

Figure 2.1: An overview of CBAM [1]

This attention module decomposes the process that learns channel and spatial attention, instead of computing the 3D attention map directly, and thus has much lesser computational and parameter overhead, and can therefore be used as a ready to integrate module for pre-existing base CNN architectures.

The channel attention submodule, focuses on "what" is meaningful in an input image, and produces a channel attention map by utilizing the inter-channel relationship of features. For enhanced feature aggregation, the channel attention module makes use

of both max-pooled and average-pooled features simultaneously, which greatly improves the representation power of networks as compared to their independent usage.

First aggregation of the spatial information of a feature map is done using both the pooling operations mentioned earlier, generating two different spatial context descriptors. Both these descriptors are then forwarded to a shared MLP(multi-layer perceptron) network with one hidden layer. After the shared network is applied to each descriptor, the output features are merged using element-wise summation. The channel attention is illustrated in Fig. 2.2.

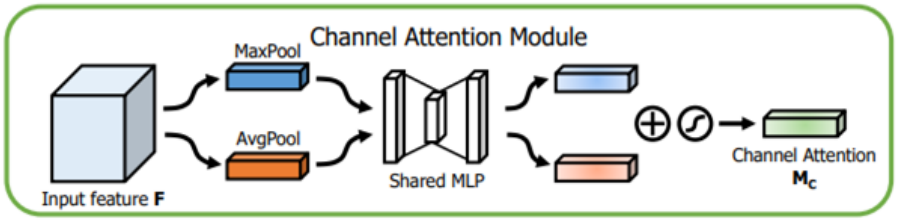

Figure 2.2: Channel Attention module in CBAM [1]

The Spatial attention sub-module, focuses on "where" is an informative part, which complements the channel attention module, and produces a spatial attention map by exploiting the inter-spatial relationship of features. For feature aggregation along the channel axis, this module uses average and max channel pooling, thus generating two maps which are then concatenated and convolved by a standard convolution layer, thus producing a 2D spatial attention map. The spatial attention is illustrated in Fig. 2.3

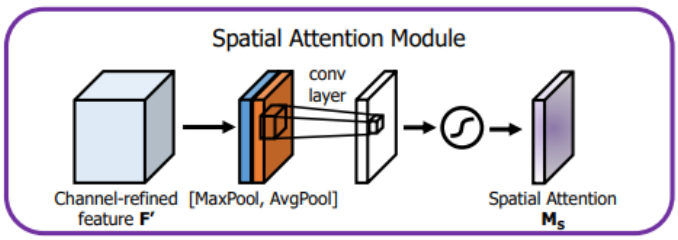

Figure 2.3: Spatial Attention module in CBAM [1]

2.2 Global Attention Mechanism: Retain Information to Enhance Channel-Spatial Interactions [Liu et al., 2021]

The existing attention mechanisms have used both spatial and channel attention mechanisms separately and achieved higher accuracy, but these attention mechanisms ignore the channel-spatial interactions and hence utilize visual representations from a limited receptive field due to information loss and dimension separation and thus lose the global spatial-channel interactions in the process.

To address this limitation, global attention module [2] was proposed which reduces information reduction and is capable of directly capturing the significant features across all three dimensions. GAM adopts the sequential channel-spatial attention mechanism from CBAM [1] as shown in Fig. 2.4 and redesigns the submodules.

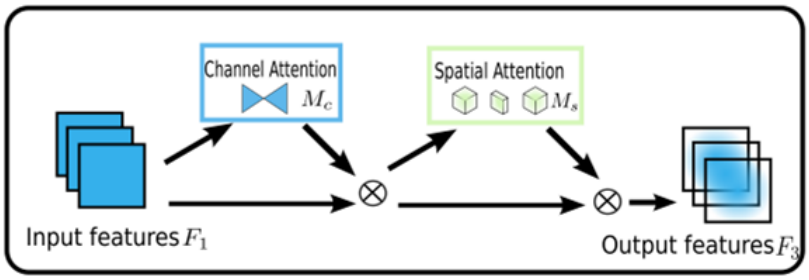

Figure 2.4: An overview of GAM [2]

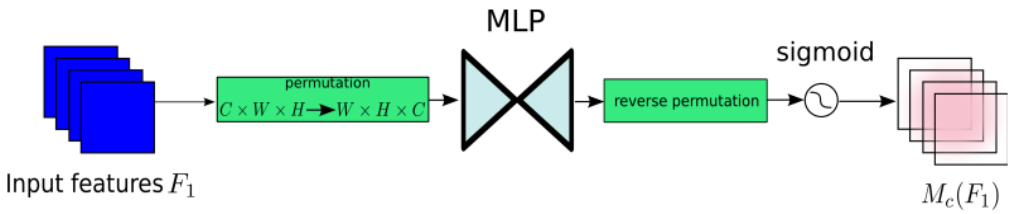

Figure 2.5: Channel Attention module in GAM [2]

The channel attention sub-module uses 3D permutation to rearrange information across three dimensions. Subsequently, it enhances cross-dimension channel-spatial dependencies using a Multi-layer perceptron with two layers. The final attention map after applying reverse permutation is normalized using sigmoid function. Channel attention sub-module is illustrated in Fig. 2.5

The spatial attention sub-module uses two convolutional layers for spatial information fusion and further removes the pooling operation to further retain the feature maps and as a result the spatial attention module sometimes increases the number of parameters significantly. The spatial attention sub-module is illustrated in Fig. 2.6

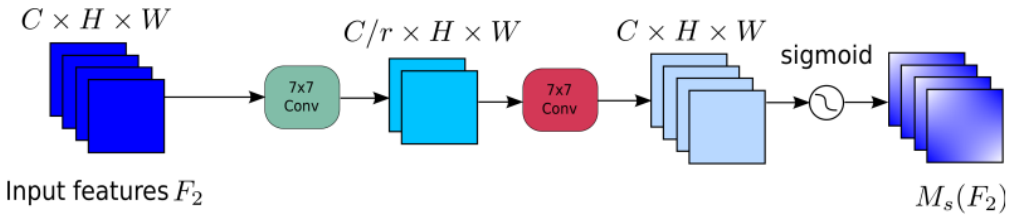

Figure 2.6: Spatial Attention module in GAM [2]

2.3 ECA-Net: Efficient Channel Attention for Deep CNNs [Wang et al.,2020]

Channel attention mechanism offer great potential in improving the performance of computer vision tasks. However, most existing methods develop very sophisticated attention modules for achieving better performance, which in turn increases the model complexity thereby increasing the computational load.

To overcome this trade-off between performance and complexity, Efficient Channel Attention [3] module brings a clear performance gain while involving only a handful of parameters. Efficient Channel Attention module captures cross-channel interaction in an efficient way by avoiding channel dimensionality reduction. As illustrated in Fig. 2.7, after applying channel independent global average pooling without dimensionality reduction, ECA captures the local cross-channel interaction by considering

every channel and its k neighbours. This is efficiently implemented by using fast 1D convolution of kernel size k, where k is determined by an adaptive approach where the kernel size k is proportional to the channel dimension.

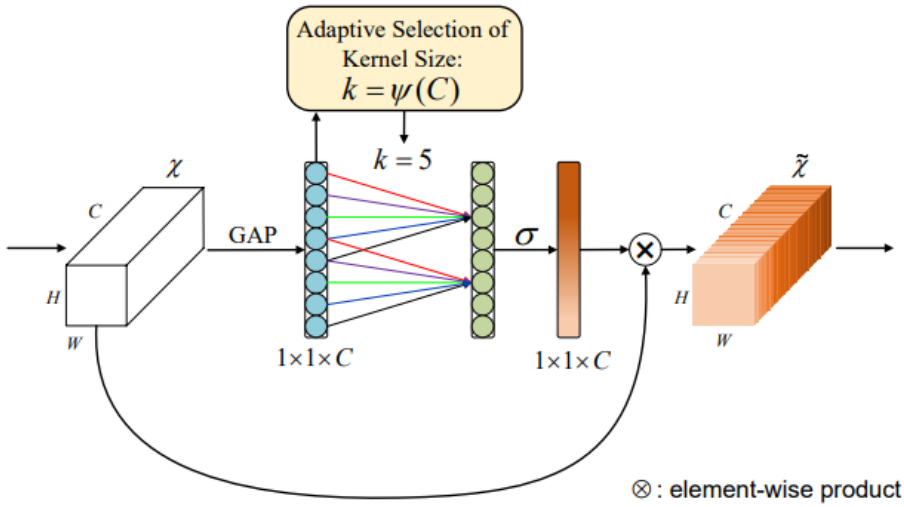

Figure 2.7: An overview of Efficient Channel Attention module [3]

2.4 SA-NET: Shuffle Attentin for Deep CNNs [Zhang et al., 2021]

Fusing spatial and channel attention may achieve better performance than their individual implementations, but it will inevitably increase the computational overhead.

Shuffle Attention [4] is the answer to on how to combine spatial and channel attention modules in a lighter, more efficient manner. Shuffle attention first divides a given feature map into sub-groups along the channel dimension, where each sub-feature gradually captures a specific semantic response in the training process. At the beginning of each attention unit, the input sub-feature is split into two branches along the channel dimension. One branch produces a channel attention map by utilizing inter-channel relationships, while the other branch is used to generate a spatial attention map by exploiting the inter-spatial relationship of features, such that the model can simultaneously focus on the "what" and the "where" is meaningful.

Then the two branches are concatenated to make the number of channels the same as that in the input. After this, all the sub-features are aggregated and a "channel shuffle" operator is used to enable cross-group information flow along the channel dimension. The final output of SA module is the same size as that of the input, which makes it quite easy to integrate in networks. SA is illustrated in the Fig. 2.8

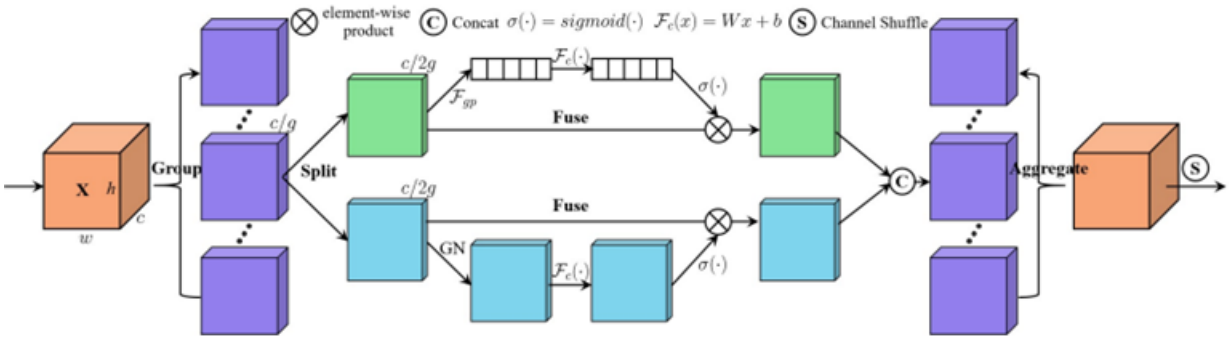

Figure 2.8: An overview of Shuffle Attention module [4]

Chapter 3

METHODOLOGY

As seen from the literature review there are various approaches on how to improve the performance of deep CNN networks using attention mechanism while at the same time limiting the computational overhead. Using YOLOv8 as our baseline model for the task of object detection we proceed further.

3.1 Dataset Preparation

As mentioned before one of the major challenge faced for preventive maintenance using object detection in rolling stock images is the low availability of data or images. Thus, to increase the amount of data available for training of an object detection model (YOLOv8) we use certain data augmentation techniques so as to generate images of potential use-cases that are observed in the setting of images for rolling stock components. The images are augmented by brightening/darkening them, introducing gaussian noise, gaussian blur and rotating the image.

Secondly, the images for rolling stock components generally suffer from poor contrast conditions. So we also make use of certain image contrast enhancement techniques like Histogram equalization and Contrast Limited Adaptive Histogram Equalization (CLAHE). Further we also analyse which contrast enhancement approach is giving the best result for object detection using YOLOv8.

3.2 Incorporating Attention modules in YOLOv8 architecture

The attention modules are used to improve the feature representation of deep CNNs network. Here we are using the attention modules to improve the performance of YOLOv8 for object detection tasks.

YOLOv8 architecture consists of three components namely, the backbone, the neck and the head. The backbone is a deep learning architecture that extracts features at various resolution levels, the neck combines the features extracted by the backbone and the head is responsible for predicting the object classes and the bounding box co-ordinates.

We are adding the attention modules to the neck of the YOLOv8 architecture (as shown in Fig. 3.1) as the neck combines the features from different levels of abstraction and are used to generate the final predictions. By introducing attention mechanisms at this stage, the model can refine the feature representations and enhance the localization accuracy.

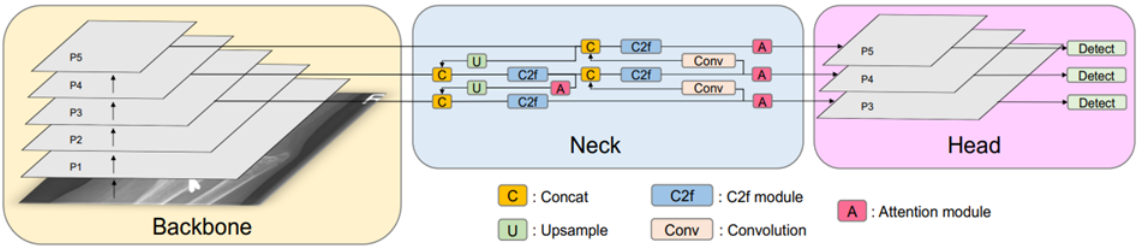

Figure 3.1: YOLOv8 architecture with Attention modules [5]

The attention modules used are:

- ResBlock + Convolutional Block Attention Module [1]
- Global Attention Module [2]
- Efficient Channel Attention [3]
- Shuffle Attention [4]

3.3 Implementing using different YOLOv8 model size

As a real time object detection approach for preventive maintenance using rolling stock images, the processing time and the computational load on the edge devices at the customer site are certain parameters which are very critical for a choice of a model for production, a comparative study is done using YOLOv8 nano and small variant so as to analyse the trade-off between the performance improvement and the increase in the inference time or the computational overhead.

Bigger variants of YOLOv8 (medium, large etc) were not considered as they are too large for production.

Chapter 4

EXPERIMENTAL EVALUATION

4.1 How image data is collected?

For collecting image data of rolling stock components, at the customer site (as shown in Fig. 4.1), along the railway tracks a system comprising of sensors, camera modules and edge devices is setup. When a train approaches the setup, first it passes through the sensors placed on the railway tracks which trigger the camera modules into action and according to the calibration of these modules they capture images of the components that they are assigned to, thus collecting the required data for usage.

Figure 4.1: A site view showing the cameras, sensors and edge devices setup

4.2 Original Dataset Details

The dataset used consists of images of **Extreme Guiding** components. The Extreme guiding components connect the wagon (bogie) with the locomotive (engine). The images are captured from two camera views namely the Control Arm Bolt View (as shown in Fig. 4.2) and the Side view (as shown in Fig. 4.3).

Figure 4.2: Control Arm bolt View

Figure 4.3: Side View

The original dataset consists of a total of **111 images** having **9 object classes** namely, Arm nut, Bolt Thread, Horizontal Nut, Security Plate Bolt, Top Security Plate, Top Security Wire, Vertical Bolt, Vertical Nut and Washer. These classes are depicted in sample images shown in Fig. 4.4 and Fig. 4.5 .

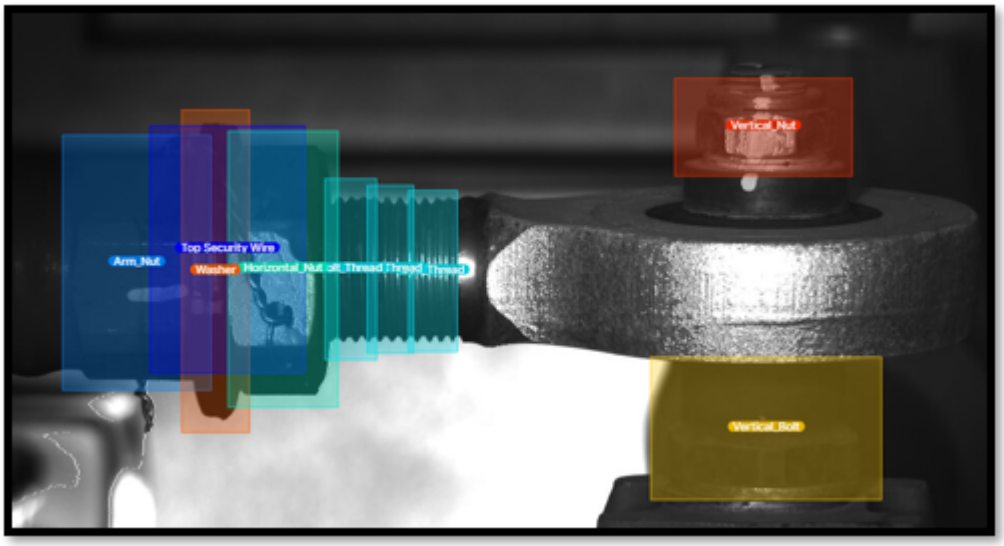

Figure 4.4: Sample image data of control arm bolt view

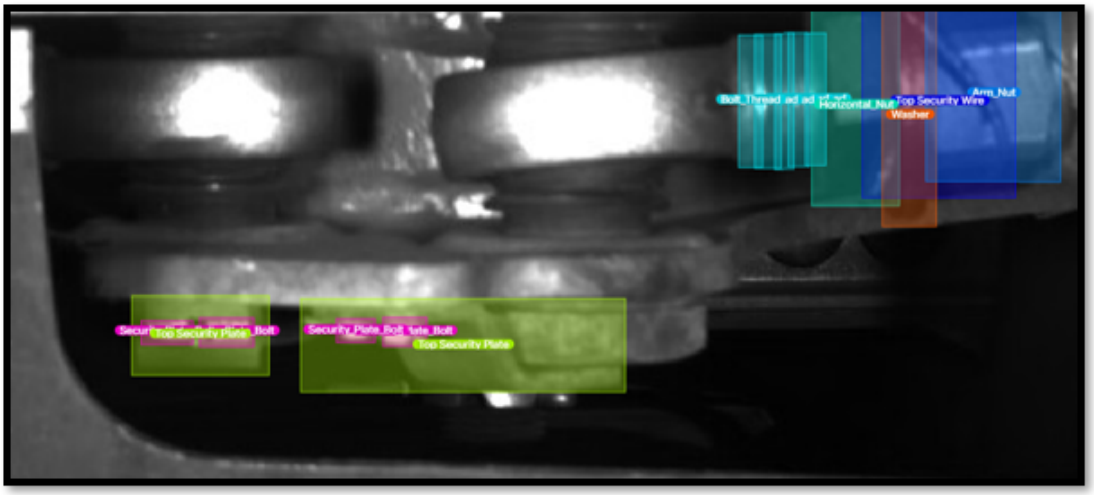

Figure 4.5: Sample image data of side view

4.3 Object Detection using YOLOv8 nano and small models

Object Detection by training the YOLOv8 nano and small model on the original dataset with a 80:20 train, validation split gives the following result as shown in table 4.1 .

YOLO Model	mAP 50	mAP 50-95
Nano	82.4	49.8
Small	83.0	51.4

Table 4.1: YOLOv8 object detection results on original dataset

The class wise performance of YOLOv8 nano and small model is shown using the Precision Recall curve in Fig. 4.6 and Fig. 4.7 .

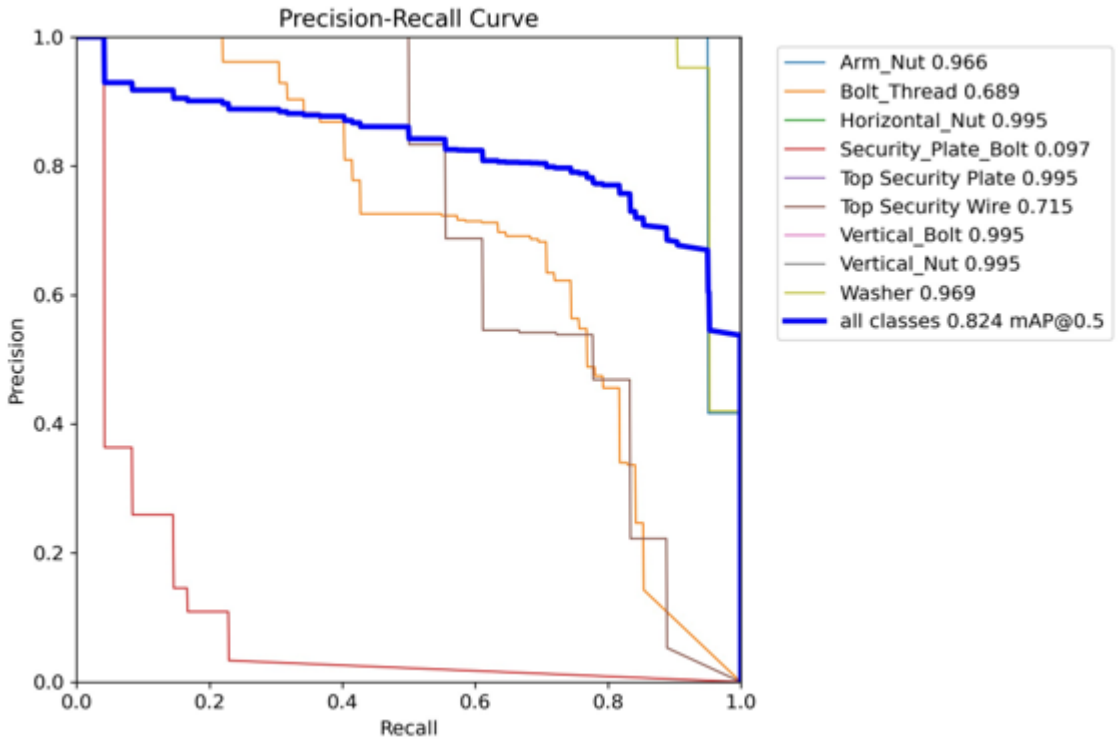

Figure 4.6: Precision-Recall curve for YOLOv8 nano model for original dataset

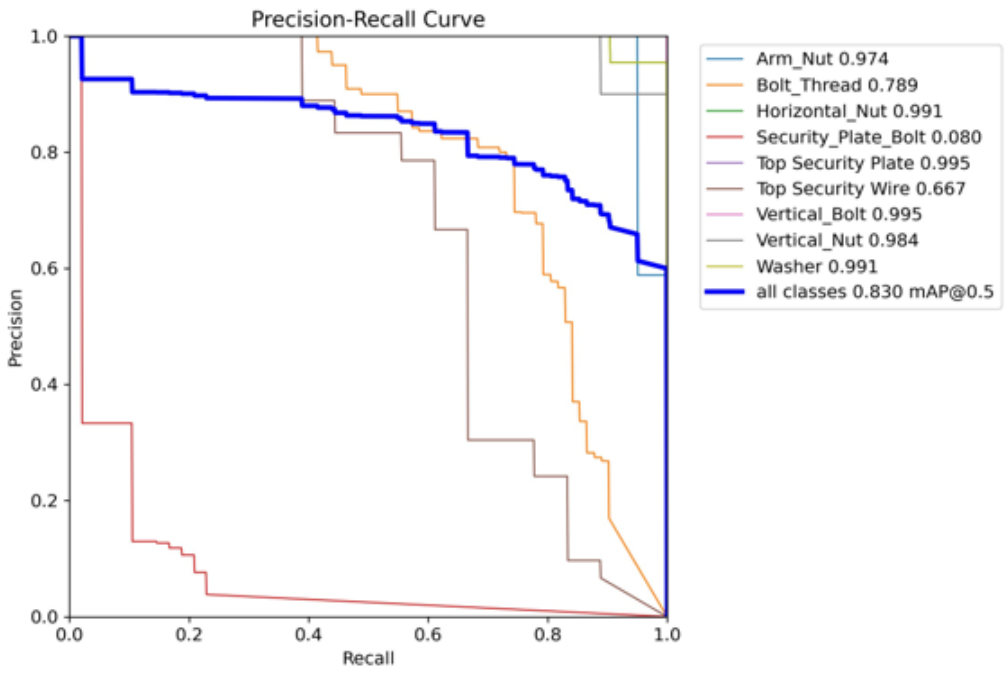

Figure 4.7: Precision-Recall curve for YOLOv8 small model for original dataset

The models are not giving very good overall performance on the original dataset and give extremely poor performance for 3 particular object classes: Bolt Thread, Security Plate Bolt and Top Security Wire.

As a product for object detection in extreme guiding components of rolling stock, the model should give good performance for all of the object classes.

4.4 Data Preparation Pipeline

As our dataset is very small, to extend the dataset, augmentation techniques were used to generate potential use cases that are observed in images of rolling stock components. Also, the augmentation of training and validation set is done separately so as to avoid any data leakage.

Further to find the optimal pre-processing technique we generate 3 sets of data, normal, histogram equalized and contrast limited adaptive histogram equalized.

The Testing set is taken from a batch of data different from the original dataset, obtained at a later time. The data preparation pipeline is shown in Fig. 4.8

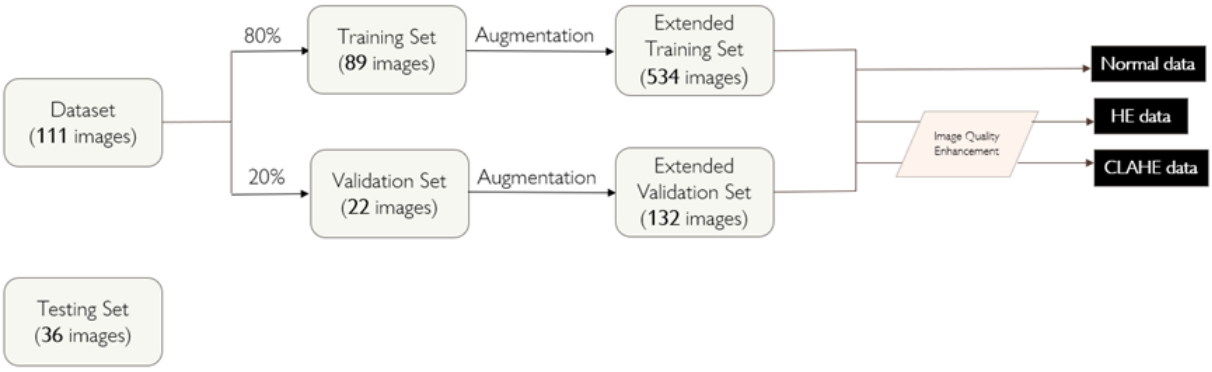

Figure 4.8: Data preparation pipeline

4.5 Extended Dataset Details

The extended dataset's train, validation and test split is shown in table 4.2 and the object class data distribution is shown in the Fig. 4.9

Extended Dataset	Training	Validation	Testing
666 + 36	534	132	36

Table 4.2: Train, validation, Test split for extended dataset

4.6 Finding the Optimal Image Contrast Enhancement Technique

Since the images for the rolling stock generally suffer from poor contrast, thus for pre-processing contrast enhancement techniques are being considered to improve the object detection performance of YOLOv8 model.

Two techniques were considered namely Histogram Equalization and Contrast Limited Adaptive Histogram Equalization (CLAHE) and the performance of YOLOv8

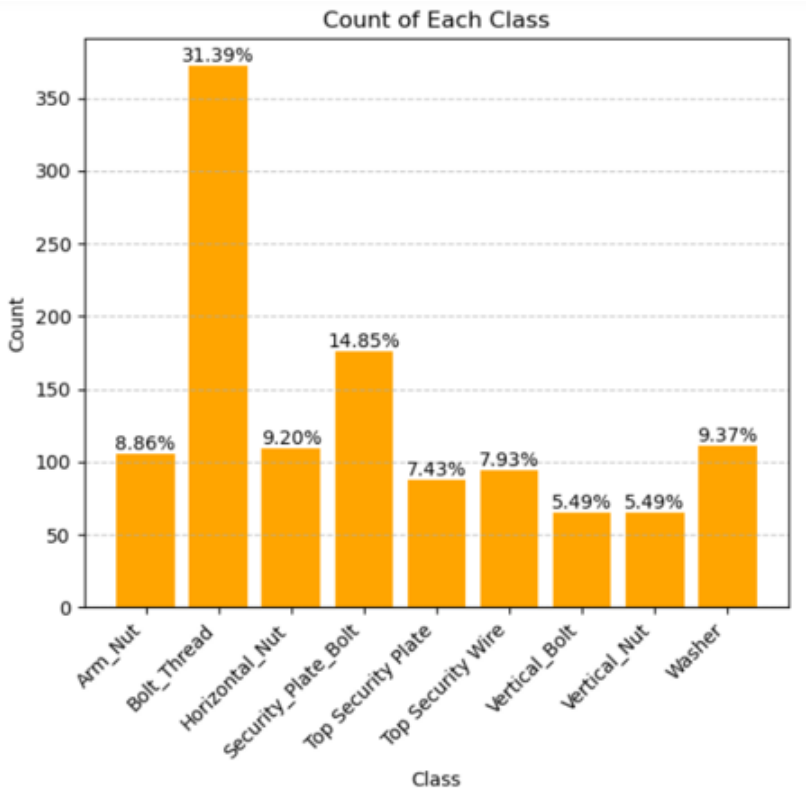

Figure 4.9: Object class distribution

object detection model both nano and small, were compared on all 3 generated dataset (one normal and two enhanced).

4.6.1 Histogram Equalization

Images with low contrast may have their pixel intensities concentrated in a narrow range. This results in a dull or washed-out image. Histogram Equalization works by spreading out these intensities across a wider range thus making the pixel intensity histogram relatively flat. This increases the contrast in the image.

4.6.2 CLAHE

It stands for contrast limited adaptive histogram equalization and it addresses some limitation of histogram equalization technique, particularly in preserving local context. It is useful in scenarios where there is a wide range of intensity variations across different regions of an image.

First it divides the image into smaller regions or tiles, then histogram equalization is applied independently to each tile. CLAHE also limits the contrast enhancement for each tile by clipping the histogram of each tile which in turn prevents noise over amplification.

The results for YOLOv8-nano on the 3 datasets is shown in table 4.3 along with the class wise performance shown using precision recall curves in figures 4.10, 4.11 and 4.12 for normal, histogram equalized and clahe dataset respectively.

Datasets	**mAP 50**	**mAP 50-95**
Normal	91.5	56.3
Histogram Equalized	93.3	56.0
CLAHE	91.0	55.3

Table 4.3: YOLOv8 nano object detection results

The results for YOLOv8-small on the 3 datasets is shown in table 4.4 along with the class wise performance shown using precision recall curves in figures 4.13, 4.14 and 4.15 for normal, histogram equalized and clahe dataset respectively.

Datasets	**mAP 50**	**mAP 50-95**
Normal	94.9	59.0
Histogram Equalized	95.2	59.5
CLAHE	94.4	58.9

Table 4.4: YOLOv8 small object detection results

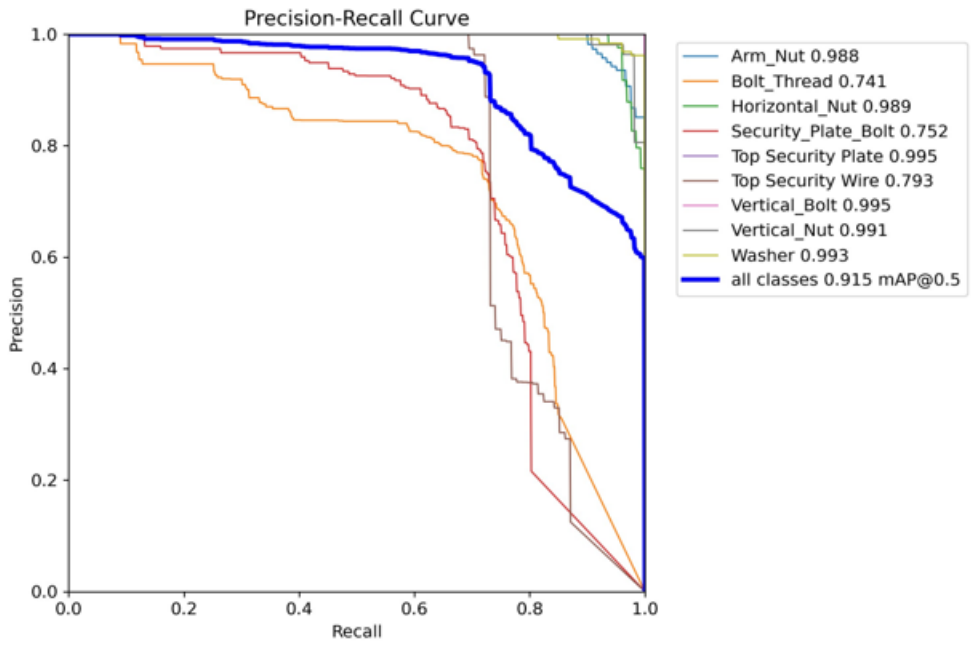

Figure 4.10: Precision Recall curve for YOLOv8 nano on normal dataset

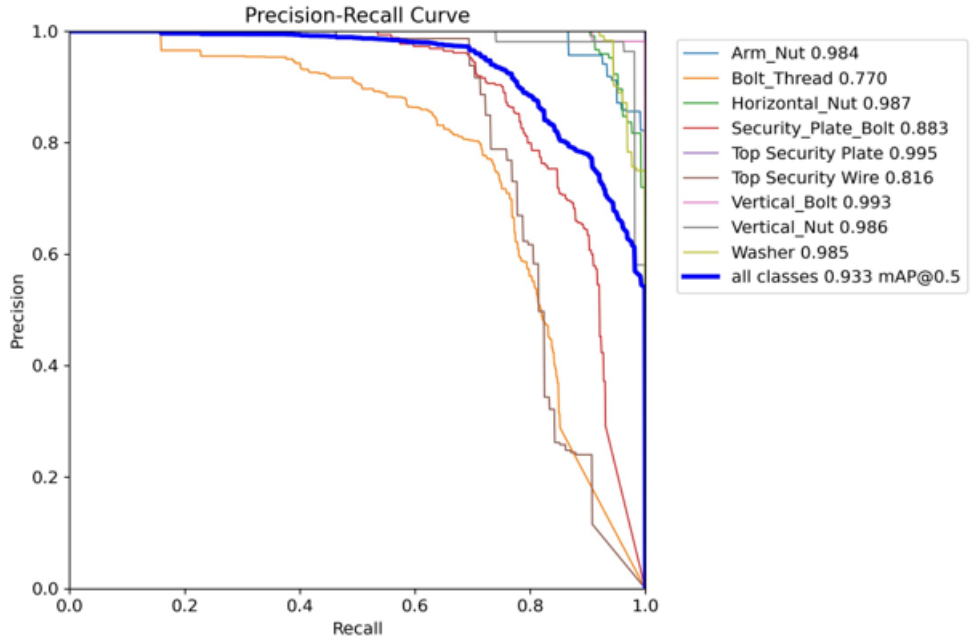

Figure 4.11: Precision Recall curve for YOLOv8 nano on HE dataset

Upon analysing it is seen that both YOLOv8 nano and small models produce best overall and class wise results for histogram equalized dataset. Thus, for the extreme

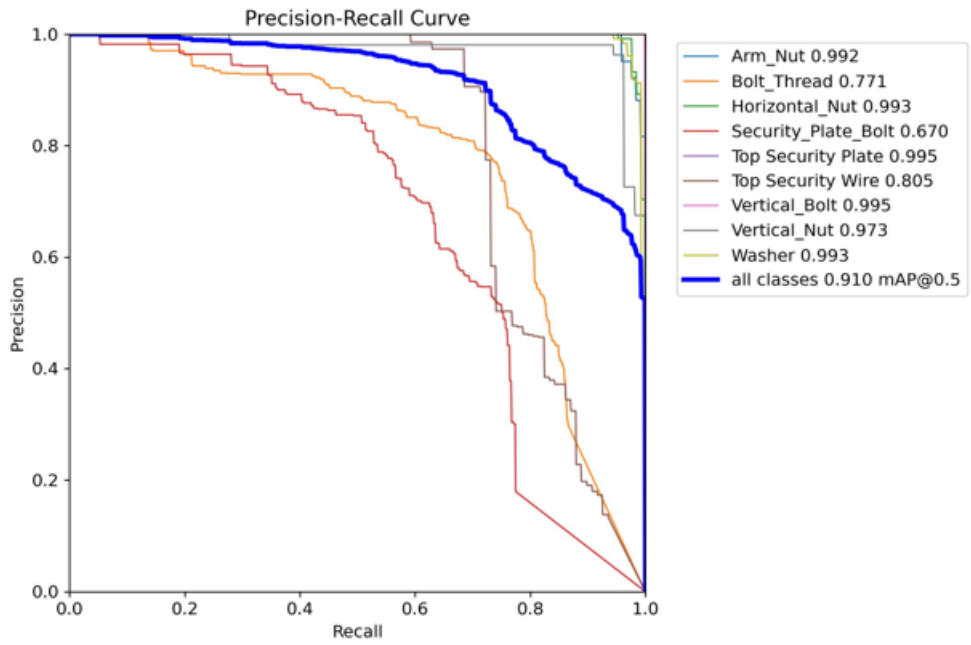

Figure 4.12: Precision Recall curve for YOLOv8 nano on CLAHE dataset

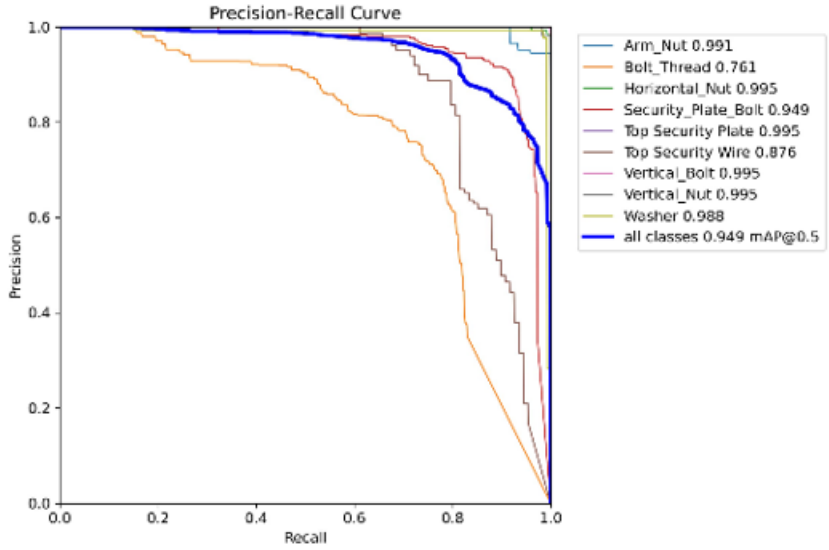

Figure 4.13: Precision Recall curve for YOLOv8 small on normal dataset

guiding dataset histogram equalization is the best image pre-processing technique.

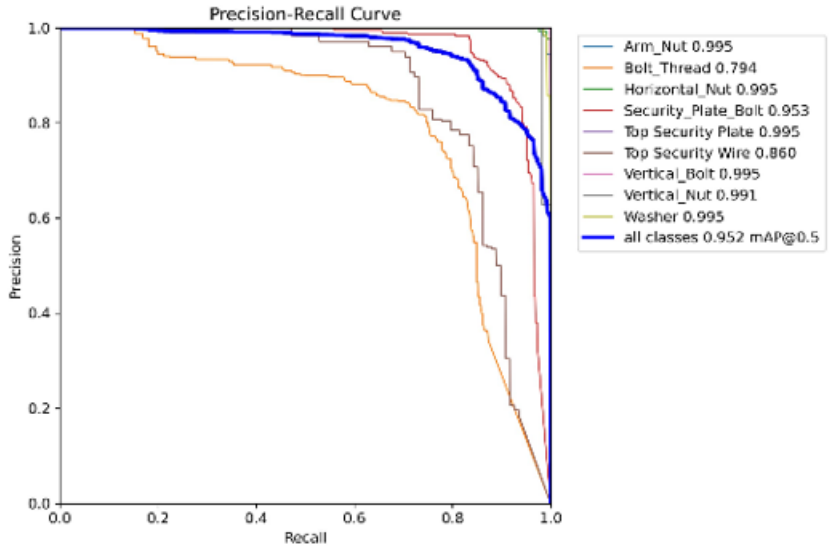

Figure 4.14: Precision Recall curve for YOLOv8 small on HE dataset

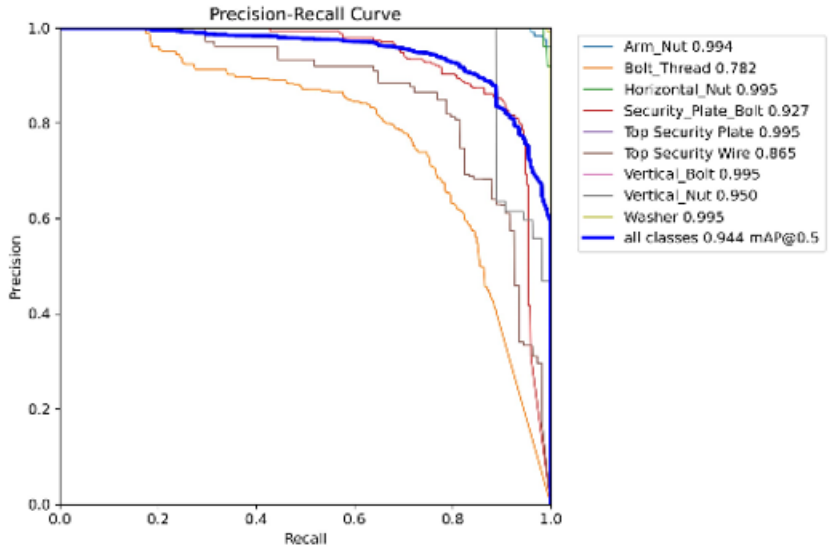

Figure 4.15: Precision Recall curve for YOLOv8 small on CLAHE dataset

4.7 Quantitative Analysis of YOLOv8 Nano + Attention modules

This analysis is done using a histogram equalized dataset with the following tuned hyperparameters: Optimizer- Adam, Epochs- 150, Learning Rate- 0.001, Image-size- 960.

The table 4.5 showcases the performance of YOLOv8 nano model with different attention modules. We can see that YOLOv8 nano model with attention modules does not show any improvement over the baseline.

Attention Module	**Precision**	**Recall**	**mAP 50**	**mAP 50-95**	**Params (M)**	**Inference time(ms)**	**Inference time on test set(ms)**
None	96.2	87.8	93.3	56	3.0	7.9	30.3
ResBlock + CBAM [1]	89.9	86.2	89.6	54.1	4.24	9.2	32.4
GAM [2]	93.0	84.6	90.4	55.2	3.68	14.6	35.1
ECA [3]	89.2	87.5	90.5	53.3	3.0	8	30.2
SA [4]	92.8	85.2	91.6	55.2	3.0	8.3	31.3

Table 4.5: Object detection results for YOLOv8 nano with attention modules

The class-wise performance of the model without attention module and with Shuffle Attention [4] module is shown in fig. 4.16 and 4.17 respectively

4.8 Quantitative Analysis of YOLOv8 Small + Attention modules

This analysis is done using a histogram equalized dataset with the following tuned hyperparameters: Optimizer- Adam, Epochs- 150, Learning Rate- 0.001, Image-size- 960.

The table 4.6 showcases the performance of YOLOv8 small model with different attention modules. We can see that the ResBlock + CBAM [1] and the ECA [3] attention modules are giving better overall mAP than the baseline model.

The class-wise performance of the model without attention module, with ResBlock+CBAM [1] and ECA [3] attention module is shown in fig. 4.18, 4.19 and 4.20 respectively

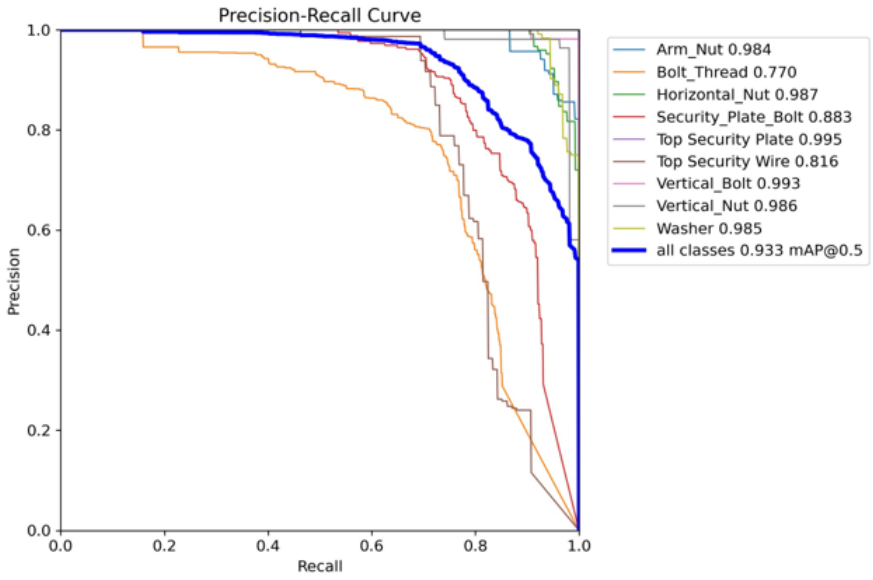

Figure 4.16: Precision Recall curve for YOLOv8 nano without attention

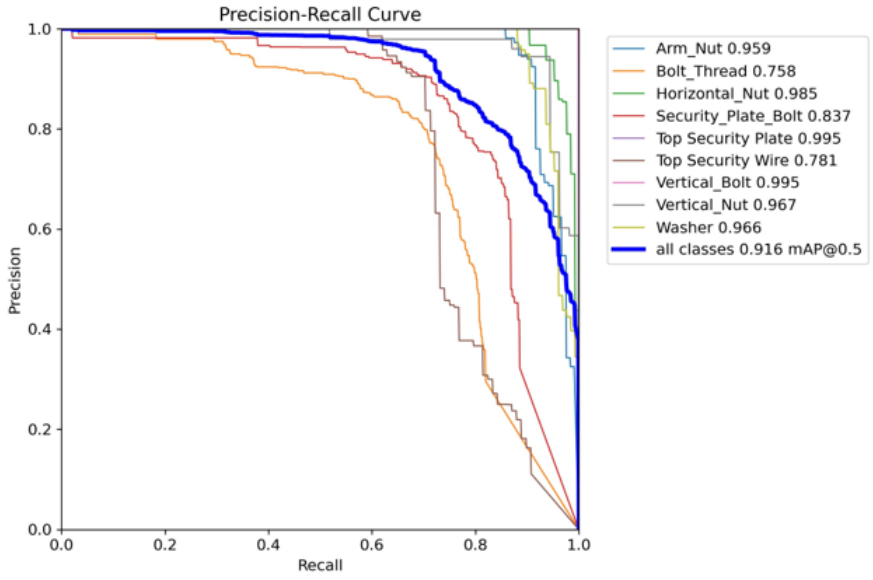

Figure 4.17: Precision Recall curve for YOLOv8 nano with SA

4.9 Qualitative analysis on test set using YOLOv8 small model

Here we compare the qualitative performance of the YOLOv8 small baseline model and the model utilizing the ResBlock+CBAM [1] attention module. Specifi-

Attention Module	Precision	Recall	mAP 50	mAP 50-95	Params (M)	Inference time(ms)	Inference time on test set(ms)
None	93.6	92.1	95.2	59.5	11.13	17.0	38.0
ResBlock + CBAM [1]	95.6	91.3	95.7	60.4	16.05	19.5	43.0
GAM [2]	94.1	91.6	95.1	59.4	13.84	33.4	46.7
ECA [3]	94.0	91.3	95.3	59.6	11.13	17.0	39.2
SA [4]	93.3	90.3	94.0	57.5	11.13	17.8	40.6

Table 4.6: Object detection results for YOLOv8 small with attention modules

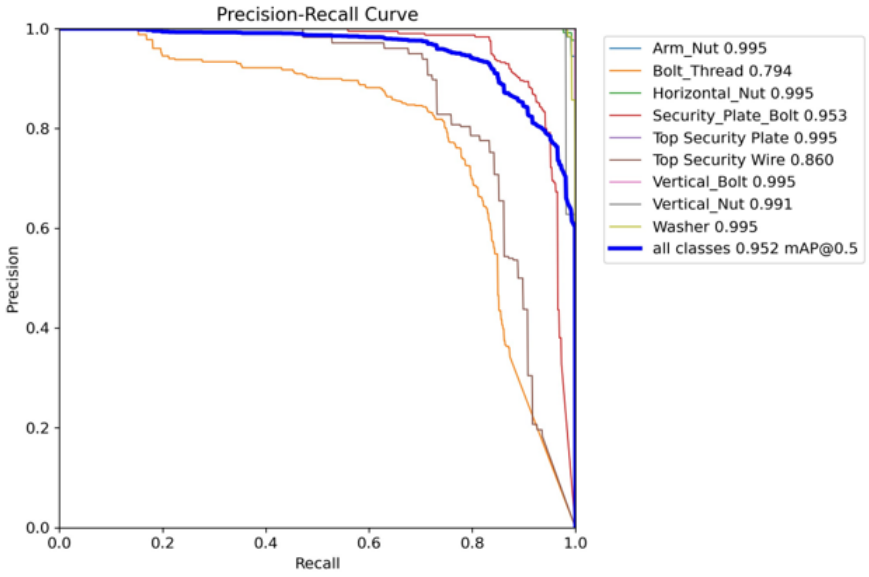

Figure 4.18: Precision Recall curve for YOLOv8 small without attention

cally we compare their performance on the 3 object classes namely Bolt thread, Top Security wire and security Plate Bolt, which were initially very difficult to detect and accurately localize. The qualitative results of the baseline model are shown in fig. 4.21 and that with the attention module is shown in fig. 4.22.

From the figures we can see that even though the baseline model is able to detect all the 3 object classes but it is not able to do so with good localization accuracy as we can see that it gives multiple bounding boxes for the same object. On the other

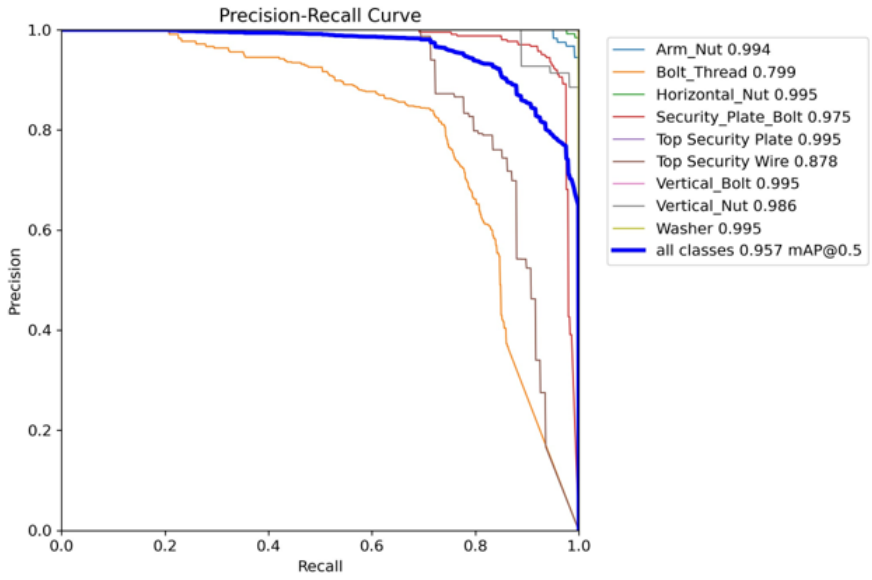

Figure 4.19: Precision Recall curve for YOLOv8 small with ResBlock + CBAM

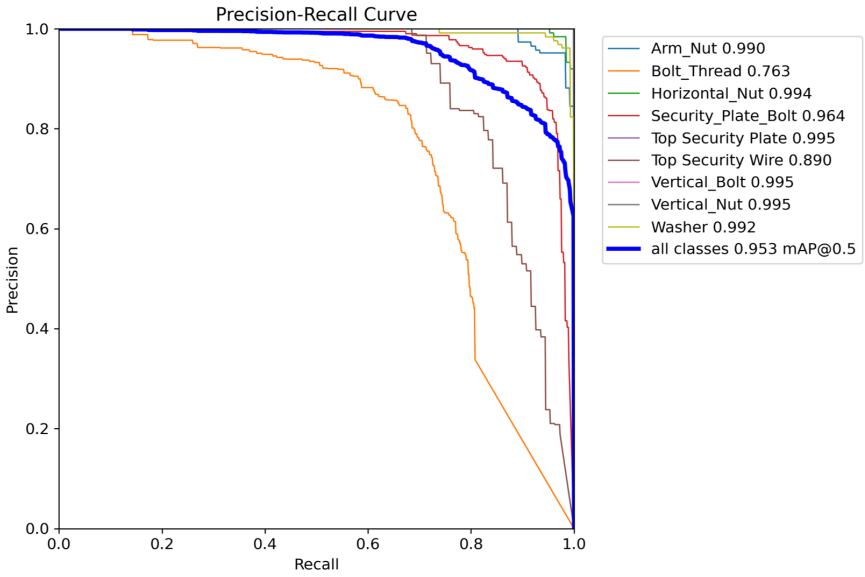

Figure 4.20: Precision Recall curve for YOLOv8 small with ECA

hand the YOLOv8 model with the attention module gives better confidence score and shows far greater object localization accuracy.

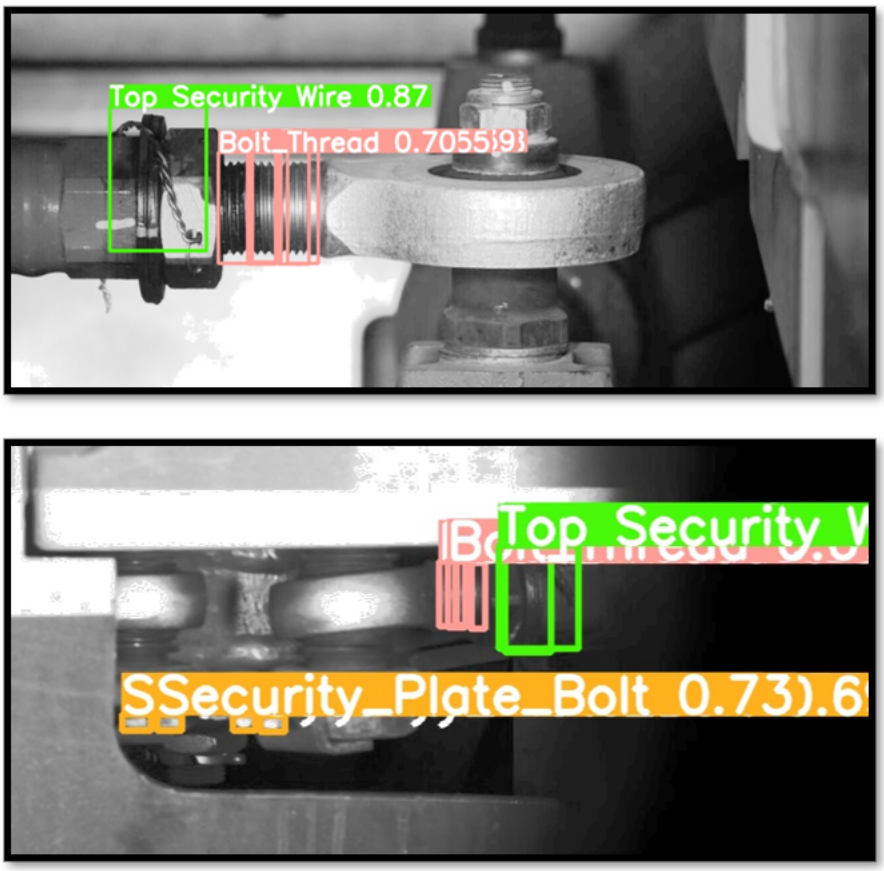

Figure 4.21: Qualitative results using YOLOv8 baseline

4.10 How to choose a model for production?

Choosing a model for production depends on customer requirements which are as follows:

- High mAP, Precision and Recall value
- Higher Precision/Recall
- Total processing time limit

A higher mAP value signifies that the model is quite proficient in accurately localizing a object in the image. Since there is a trade-off between Precision and Recall, according to the requirement whether a customer considers a lower FP or a lower FN as critical to the product's output, a higher Precision or a higher Recall is preferred by the customer respectively. As these products are deployed for usage during real

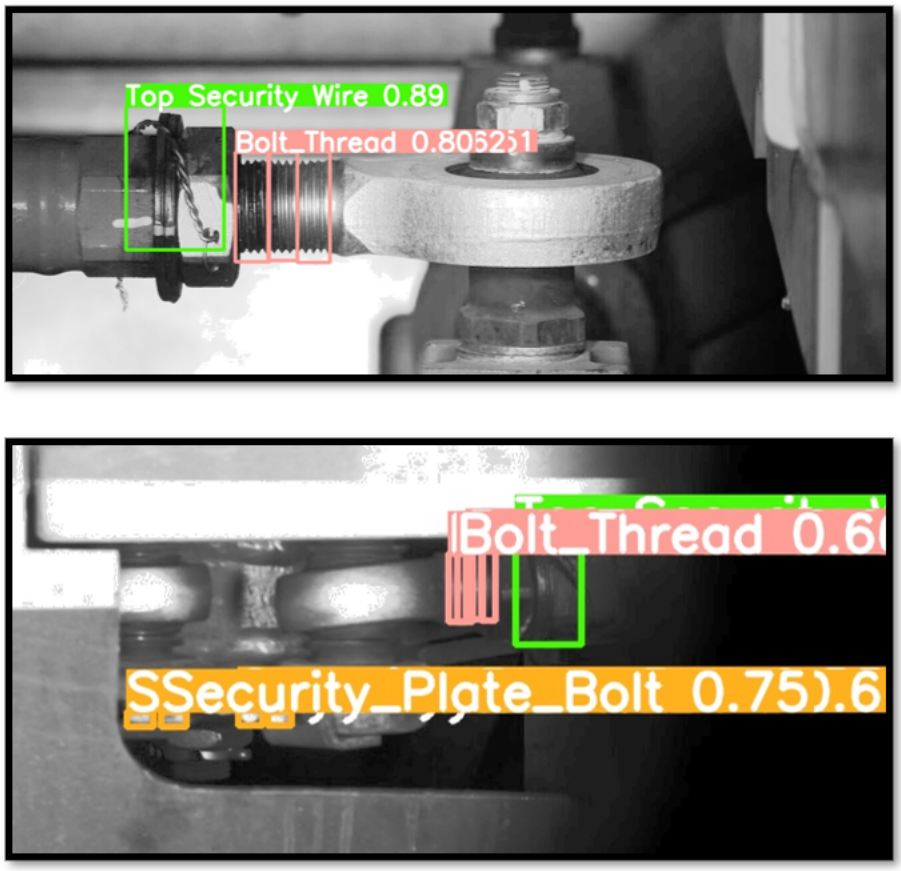

Figure 4.22: Qualitative results using YOLOv8 with ResBlock+CBAM attention

time, the Total Processing time limit is very crucial for a model in production.

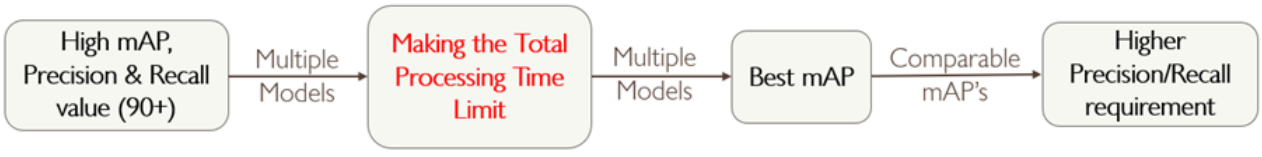

Figure 4.23: Steps in choosing a model for production

The pipeline for choosing a model for production is shown in Fig. 4.23 and it goes as follows: Firstly, the models should have a high enough value for mAP, precision and recall. If multiple models satisfy this, then the first priority is to make the processing time limit. If still we have multiple models in the pipeline, we choose the one having

the best mAP value if the difference in the mAP value is considerable. If the mAP values for multiple models are comparable then we choose the model which satisfies the higher precision/recall requirement of the customer.

Chapter 5

CONCLUSION

- Histogram Equalization is the best image contrast enhancement technique for the extreme guiding dataset for object detection task.
- Adding attention modules to the YOLOv8 nano model didn't result in any improvement over the baseline model.
- As for the YOLOv8 small model, two attention modules ResBlock + CBAM [1] and ECA [3] showed better performance than the baseline model.
- The attention modules not showing any improvement for the nano model could be due to the model not being deep enough to make the most out of the feature extraction.
- We also saw that though using attention modules improved the performance of the model but it also caused a increase in the inference time. Thus, there exists a trade-off between performance enhancement and reducing the inference time and subsequently the processing time of the model, and thus the choice of a particular model depends on the requirements of the customer.

Chapter 6

FUTURE SCOPE

- As the dataset is quite limited in variety, apart from augmentation techniques that have already been used we can use generative models to generate defective/faulty cases which are none to very few in number and also generate data for scenarios where we have challenging weather conditions like rain, fog, snow etc.
- Further we can also use FastSAM [6], a zero-shot segmentation model with YOLOv8 variants for producing segmentation masks which could be helpful in certain use-cases during post processing. FastSAM [6] can take a bounding box as a prompt (Output from the YOLO model) and produce a segmentation mask. The advantage of using FastSAM [6] in this manner is that it would require no extensive retraining and also there will be no need for pixel level annotation (we only need to do bounding box annotations for training YOLO models).

Chapter 7

APPENDIX

This appendix contains the detailed output for the object detection task on the extended histogram equalized dataset for the YOLOv8 model and its variants.

1. YOLOv8 nano baseline in fig. 7.1

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8n summary (fused): 168 layers, 3007403 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        132       1512      0.926      0.878      0.933       0.56
               Arm_Nut        132        120      0.946      0.925      0.984      0.649
           Bolt_Thread        132        492      0.776      0.718       0.77      0.316
        Horizontal_Nut        132        126      0.973      0.913      0.987      0.756
    Security_Plate_Bolt        132        288      0.908      0.726      0.883      0.366
     Top Security Plate        132        144      0.988          1      0.995      0.721
      Top Security Wire        132        108      0.809      0.731      0.816      0.495
          Vertical_Bolt        132         54      0.982      0.986      0.993      0.758
           Vertical_Nut        132         54      0.964      0.978      0.986      0.383
                 Washer        132        126      0.992      0.926      0.985      0.593
Speed: 0.5ms preprocess, 6.3ms inference, 0.0ms loss, 1.1ms postprocess per image
```

Figure 7.1: Detailed Output for YOLOv8 nano

2. YOLOv8 nano with ResBlock+CBAM [1] attention in fig. 7.2
3. YOLOv8 nano with GAM [2] attention in fig. 7.3
4. YOLOv8 nano with ECA [3] attention in fig. 7.4
5. YOLOv8 nano with SA [4] attention in fig. 7.5
6. YOLOv8 small baseline in fig. 7.6

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8n_ResBlock_CBAM summary: 244 layers, 4240627 parameters, 0 gradients, 10.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        132       1512      0.899      0.862      0.896      0.541
               Arm_Nut        132        120      0.948      0.917      0.986      0.666
           Bolt_Thread        132        492      0.769      0.745      0.793      0.326
        Horizontal_Nut        132        126          1      0.954      0.993      0.771
   Security_Plate_Bolt        132        288      0.699      0.552      0.615       0.23
    Top Security Plate        132        144      0.967          1      0.995      0.708
     Top Security Wire        132        108      0.731      0.676      0.708      0.453
         Vertical_Bolt        132         54          1      0.992      0.995      0.747
          Vertical_Nut        132         54          1      0.997      0.995      0.375
                Washer        132        126      0.975      0.923      0.982      0.591
Speed: 0.3ms preprocess, 7.4ms inference, 0.0ms loss, 1.5ms postprocess per image
```

Figure 7.2: Detailed Output for YOLOv8 nano with ResBlock+CBAM attention

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8n_GAM summary (fused): 212 layers, 3688683 parameters, 0 gradients, 9.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        132       1512       0.93      0.846      0.904      0.552
               Arm_Nut        132        120      0.972        0.9      0.961       0.68
           Bolt_Thread        132        492      0.778      0.707      0.743       0.31
        Horizontal_Nut        132        126          1      0.914      0.987      0.755
   Security_Plate_Bolt        132        288      0.856       0.59       0.77      0.316
    Top Security Plate        132        144      0.991          1      0.995      0.724
     Top Security Wire        132        108       0.81      0.713      0.771      0.496
         Vertical_Bolt        132         54      0.991          1      0.995      0.724
          Vertical_Nut        132         54       0.98      0.886      0.938      0.352
                Washer        132        126      0.991      0.903      0.973      0.613
Speed: 0.4ms preprocess, 12.8ms inference, 0.0ms loss, 1.4ms postprocess per image
```

Figure 7.3: Detailed output for YOLOv8 nano with GAM attention

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8n_ECA summary (fused): 184 layers, 3007415 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        132       1512      0.892      0.875      0.905      0.533
               Arm_Nut        132        120      0.969       0.95       0.99      0.646
           Bolt_Thread        132        492      0.716       0.75       0.75      0.303
        Horizontal_Nut        132        126      0.925      0.992      0.993      0.729
   Security_Plate_Bolt        132        288      0.835      0.562      0.695      0.279
    Top Security Plate        132        144      0.966          1      0.995      0.713
     Top Security Wire        132        108      0.726      0.685      0.783      0.445
         Vertical_Bolt        132         54      0.984          1      0.995      0.748
          Vertical_Nut        132         54      0.954      0.963      0.956      0.352
                Washer        132        126      0.953       0.97      0.986      0.584
Speed: 0.5ms preprocess, 6.3ms inference, 0.0ms loss, 1.2ms postprocess per image
```

Figure 7.4: Detailed output for YOLOv8 nano with ECA attention

7. YOLOv8 small with ResBlock+CBAM [1] attention in fig. 7.7

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8n_SA summary (fused): 184 layers, 3007619 parameters, 0 gradients, 8.1 GFLOPs
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                  all        132       1512      0.928      0.852      0.916      0.552
              Arm_Nut        132        120      0.947      0.896      0.959       0.66
          Bolt_Thread        132        492      0.803      0.697      0.758      0.317
       Horizontal_Nut        132        126      0.966      0.909      0.985      0.759
   Security_Plate_Bolt        132        288      0.908      0.688      0.837      0.355
    Top Security Plate        132        144      0.992          1      0.995      0.698
     Top Security Wire        132        108      0.835      0.705      0.781      0.492
         Vertical_Bolt        132         54      0.998          1      0.995      0.729
          Vertical_Nut        132         54      0.969       0.87      0.967      0.378
                Washer        132        126      0.935      0.905      0.966      0.582
Speed: 0.2ms preprocess, 6.5ms inference, 0.0ms loss, 1.6ms postprocess per image
```

Figure 7.5: Detailed output for YOLOv8 nano with SA attention

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8s summary (fused): 168 layers, 11129067 parameters, 0 gradients, 28.5 GFLOPs
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                  all        132       1512      0.936      0.921      0.952      0.595
              Arm_Nut        132        120          1      0.927      0.995      0.702
          Bolt_Thread        132        492      0.814      0.745      0.794      0.328
       Horizontal_Nut        132        126      0.949          1      0.995      0.765
   Security_Plate_Bolt        132        288      0.902      0.889      0.953      0.473
    Top Security Plate        132        144      0.992          1      0.995      0.743
     Top Security Wire        132        108      0.806      0.759       0.86      0.534
         Vertical_Bolt        132         54      0.993          1      0.995      0.775
          Vertical_Nut        132         54      0.986      0.981      0.991      0.378
                Washer        132        126      0.984      0.986      0.995      0.657
Speed: 0.2ms preprocess, 15.5ms inference, 0.0ms loss, 1.3ms postprocess per image
```

Figure 7.6: Detailed output for YOLOv8 small baseline

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8s_ResBlock_CBAM summary: 244 layers, 16052723 parameters, 0 gradients, 38.1 GFLOPs
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                  all        132       1512      0.956      0.913      0.957      0.604
              Arm_Nut        132        120      0.975      0.959      0.994      0.704
          Bolt_Thread        132        492      0.816      0.734      0.799       0.32
       Horizontal_Nut        132        126      0.966          1      0.995       0.77
   Security_Plate_Bolt        132        288      0.967      0.911      0.975      0.538
    Top Security Plate        132        144      0.997          1      0.995      0.751
     Top Security Wire        132        108      0.903      0.722      0.878      0.535
         Vertical_Bolt        132         54      0.994          1      0.995      0.768
          Vertical_Nut        132         54      0.991      0.889      0.986      0.397
                Washer        132        126      0.993          1      0.995      0.656
Speed: 0.5ms preprocess, 17.6ms inference, 0.0ms loss, 1.4ms postprocess per image
```

Figure 7.7: Detailed output for YOLOv8 small with ResBlock+CBAM attention

8. YOLOv8 small with GAM [2] attention in fig. 7.8

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8s_GAM summary (fused): 212 layers, 13848427 parameters, 0 gradients, 34.0 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        132       1512      0.941      0.916      0.951      0.594
              Arm_Nut         132        120      0.959      0.979      0.991      0.699
          Bolt_Thread         132        492      0.817       0.74      0.789       0.34
       Horizontal_Nut         132        126      0.984      0.984      0.994      0.764
   Security_Plate_Bolt        132        288       0.93      0.829      0.937      0.425
    Top Security Plate        132        144      0.995          1      0.995       0.76
     Top Security Wire        132        108      0.849      0.731      0.868      0.522
         Vertical_Bolt        132         54      0.992          1      0.995      0.769
          Vertical_Nut        132         54      0.975          1      0.995       0.44
                Washer        132        126      0.964      0.984      0.994      0.629
Speed: 0.1ms preprocess, 30.1ms inference, 0.0ms loss, 3.2ms postprocess per image
```

Figure 7.8: Detailed output for YOLOv8 small with GAM attention

9. YOLOv8 small with ECA [3] attention in fig. 7.9

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8s_ECA summary (fused): 184 layers, 11129079 parameters, 0 gradients, 28.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        132       1512       0.94      0.913      0.953      0.596
              Arm_Nut         132        120      0.953      0.933       0.99      0.694
          Bolt_Thread         132        492      0.753       0.72      0.763      0.326
       Horizontal_Nut         132        126      0.992      0.958      0.994      0.786
   Security_Plate_Bolt        132        288      0.933      0.899      0.964      0.453
    Top Security Plate        132        144      0.994          1      0.995      0.759
     Top Security Wire        132        108      0.868      0.759       0.89      0.536
         Vertical_Bolt        132         54       0.99          1      0.995      0.774
          Vertical_Nut        132         54          1      0.975      0.995      0.413
                Washer        132        126      0.976      0.968      0.992       0.62
Speed: 0.4ms preprocess, 14.9ms inference, 0.0ms loss, 1.7ms postprocess per image
```

Figure 7.9: Detailed output for YOLOv8 small with ECA attention

10. YOLOv8 small with SA [4] attention in fig. 7.10

```
Ultralytics YOLOv8.0.147  Python-3.8.18 torch-2.2.0 CUDA:0 (Quadro P4000, 8192MiB)
YOLOv8s_SA summary (fused): 184 layers, 11129499 parameters, 0 gradients, 28.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        132       1512      0.933      0.903       0.94      0.575
               Arm_Nut        132        120      0.934       0.95      0.987      0.685
           Bolt_Thread        132        492      0.788      0.728      0.794       0.32
        Horizontal_Nut        132        126          1      0.992      0.995      0.767
    Security_Plate_Bolt        132        288      0.842      0.817      0.863      0.411
     Top Security Plate        132        144      0.969          1      0.994      0.744
      Top Security Wire        132        108      0.912      0.768       0.88      0.516
          Vertical_Bolt        132         54      0.992          1      0.995      0.723
           Vertical_Nut        132         54       0.96      0.894       0.96      0.372
                 Washer        132        126          1      0.979      0.995      0.638
Speed: 0.5ms preprocess, 15.7ms inference, 0.0ms loss, 1.6ms postprocess per image
```

Figure 7.10: Detailed output for YOLOv8 small with SA attention

REFERENCES

[1] J.; Lee J. Y.; Kweon I. S. Woo, S.; Park. Cbam: Convolutional block attention module. *European conference on computer vision*, 2018.

[2] Z.; Hoffmann N. Liu, Y.; Shao. Global attention mechanism: Retain information to enhance channel-spatial interactions. 2021.

[3] B.; Zhu P.; Li P.; Zuo W.; Hu Q. Wang, Q.; Wu. Eca-net: Efficient channel attention for deep convolutional neural networks. *IEEE/CVF conference on computer vision and pattern recognition*, pages 11534–11542, 2020.

[4] Y. B. Zhang, Q. L.; Yang. Sa-net: Shuffle attention for deep convolutional neural networks. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2235–2239, 2021.

[5] Kuang-Yi Chou Chun-Tse Chien, Rui-Yang Ju and Jen-Shiun Chiang. Yolov8-am: Yolov8 with attention mechanisms for pediatric wrist fracture detection. 2024.

[6] Yongqi An Yinglong Du Tao Yu Min Li Ming Tang Jinqiao Wang Xu Zhao, Wenchao Ding. Fast segment anything. 2023.