An Object Detection Approach for Automated Detection of Groove Line in Tube Yoke

Uday Kulkarni[1], *Kiran* Naregal[1], *Vishal* Farande[1*], *Soumya* Guttigoli[1], *Apoorva* Angadi[1] and *Raunak* Ujwane[2]

[1]KLE Technological University, Hubli, Karnataka, India
[2]Dana Anand India Private Limited, Dharwad, Karnataka, India

Abstract. Deep Neural Networks are designed explicitly for vision- based applications. They have become a standard method for image classification, object detection, real-time processing capabilities, and other computer vision tasks. With the intervention of human beings, it is challenging for businesses to verify that the product is appropriately created without any glitches, as this could result in human errors due to a lack of training, which could further cause losses and several complications. Thus, there is a need for a machine learning-based solution that improves work efficiency and accuracy in identifying processed and unprocessed parts. Machine Learning models have several advantages over human intervention in tasks, including consistency, speed, scalability, cost-effectiveness, and improvement over time. These advantages can help organizations improve their operations and achieve better outcomes. To do this, we propose an approach that involves building a model using an object detection method, You Only Look Once (YOLO), which is then deployed on a Raspberry Pi and integrated with the assembly line to carry out the task of validating the processed parts by achieving 98% accuracy.

1 Introduction

In recent years, the field of Artificial Intelligence (AI) has seen tremendous advancements, particularly in the area of image and video processing. Image processing has become an important aspect of many industries as it plays a critical role in improving product quality, increasing efficiency, and reducing costs. Deep Neural Networks (DNNs) have been widely used in image processing applications due to their ability to extract complex features from images. They are the type of artificial neural network that have multiple layers between the input and output layers. Each layer consists of a set of neurons, and these neurons are connected to neurons in the previous and subsequent layers [9]. DNNs are able to learn complex representations of data by iteratively transforming the input through multiple layers of non-linear transformations [1], resulting in hierarchical representations of the input data.

*Corresponding author: vishalfarande5@gmail.com

This allows them to capture more intricate relationships and patterns in the data than simpler neural networks.The focus of our proposed technique is on the identification of groove lines in a tube yoke using computer vision techniques. Tube yoke is a component of any drive shaft. The tube yoke has a groove line formed due to the grooving process and is considered a processed part in Fig 5. Object detection algorithms divide an image into a grid of cells and predict the bounding boxes using several techniques and class probabilities for each cell during the object proposal generation step [24]. The outcome of each detection contains the bounding box around the interested area with the class name and confidence. Intersection Over Union (IOU) evaluates the correctness of the predicted area compared to the ground truth [25]. The bounding boxes use different colors to give a better inference about the predicted class [11].

The dataset for the tube yoke part is first gathered from the company and the lab, and preprocessing is done. To get the anticipated outcomes, the YOLOv8 object detection algorithm is used. The algorithm is trained on a dataset of tube yoke images and is able to detect groove lines with reasonable accuracy. It is then converted to the proper format and deployed on the Raspberry Pi [20]. On the monitor, the outcome is visible with accurate prediction. YOLOv8 is an improvement over the previous versions of YOLO, which is a popular object detection algorithm. It uses a deep learning model called DarkNet, which is a high-performance object detection model with a smaller size and faster speed Fig 1.

The paper is divided into various sections. The background study of various object detection algorithms is covered in Section 2 of this article. The design and architecture of YOLOv8 are the main topics of section 3[21]. In section 4, we go over our suggested methodology, which outlines the procedures for detecting the groove, format conversion, and Raspberry Pi deployment. The precise details of our dataset are described in section 5, and the experiment's results are listed along with the bounding box- containing images. The project's conclusion and future scope are covered in section 6. This project was conceptualized and Implemented Jointly by KLE Tech and Dana Anand India Private Limited.

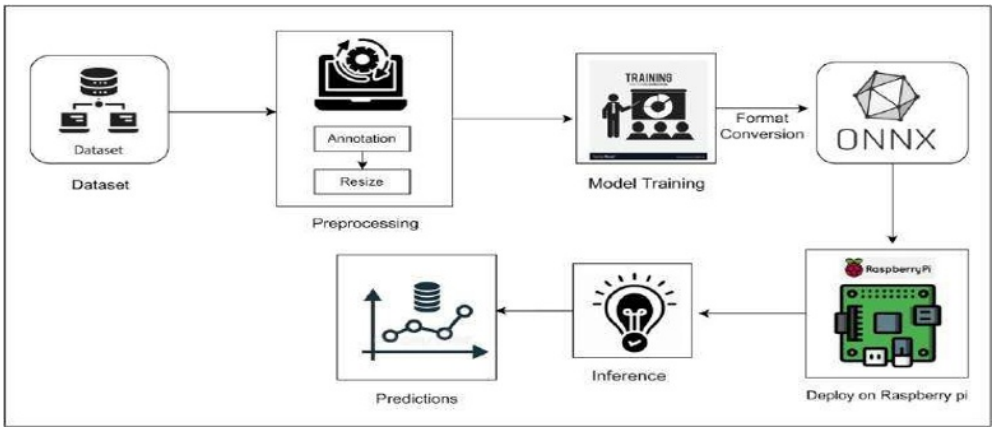

Fig. 1. Workflow of proposed methodology

2 Background Study

Object detection in computer vision can be achieved in two ways, one-step regression and classification-based networks, and two-step regions were proposal-based. Two-step regions proposal- based networks work in two stages [3]. The first step passes input images through a Selective search algorithm. The second step is passing region proposals through a CNN model to classify the object and calculate the bounding box regression [5]. We have three variants of region proposal-based object detectors Region-based Convolutional Neural Networks (RCNN), Fast-RCNN, and Faster-RCNN [22].

RCNN works by applying the selective search in the first step and then passing the region proposals through the CNN model. The selective search algorithm produces region proposals using various similarities like the shapes and colours of an object. Region proposals can be of different sizes, which are then resized. The second stage passes resized region proposals through a CNN model to classify the object and calculate the bounding box regression.

Fast RCNN works by passing the image through CNN and a selective search algorithm at the same instance. Selective search gives region proposals and CNN produces a feature map. Region proposals are mapped onto feature maps. Region proposals can be of different sizes so we use Region of Interest (RoI) pooling which gives the scale-invariant features. Apply a fully connected layer at the end to get a classification and bounding box coordinates [17]. RCNN and Fast RCNN use the normal convention of representing the bounding box w hich is the x, and y coordinates of the top left corner followed by the width and height of the object.

Faster RCNN uses a CNN for selective search known as Region Proposal Network (RPN). The image is directly passed to a CNN model to get feature maps which are then passed through an RPN. Region proposals of RPN are combined with the feature map of CNN by an RoI pooling layer and then passed to a fully connected layer to get class prediction and bounding box regression. Faster RCNN uses the anchor notation of a bounding box in which the center x, and y coordinates of an object are followed by the width and height of an object [11]. Anchor bounding box sizes are predefined and are selected based on the mean size of bounding boxes in the training dataset. The anchor box eliminates the need to verify the bounding box at every point of the input. This enhances the model throughput [6].

The region proposal-based object detection has the advantage of processing the input in two steps. This process helps the model to learn and generalize better on the given data set. This results in better class confidence and bounding box regression performance. The disadvantages are large model sizes and slow image processing speed.

One-stage classification and regression-based network process the input image in one step. Region proposal, classification, and bounding box regression are all combined into a single stage. The network divides the image into smaller girds. Each gird is processed independently to check the presence of an object using anchor boxes of standard sizes, generating dense(overlapping) predictions of the same object.

Applying Non-Max Suppression (NMS) on dense prediction helps to filter out duplicate less confident predictions [18]. The advantages of this process are smaller architecture, model sizes, and less inference time.

It has the advantage of being faster and more efficient than region proposal-based networks, making them more suitable for real-time applications. The smaller architecture and model size also makes it easier to deploy the model on edge devices with limited resources. However, one-stage networks may not perform as well as two-stage networks in detecting smaller objects or objects with low contrast against the background. The dense prediction process of one-stage networks can also lead to a higher number of false positives. To address these limitations, researchers have developed hybrid approaches that combine the advantages of both region proposal-based and one-stage networks. These hybrid models use anchor boxes to generate region proposals and then use a one-stage network to classify and regress the bounding boxes [11]. These models can achieve high accuracy while maintaining faster inference times and smaller model sizes. Additionally, recent advancements in object detection research have focused on improving the robustness and generalization of models, such as using self-supervised learning and domain adaptation techniques.

3 YOLO Architecture Description

YOLO is one of the most popular and state-of-the-art model architectures for object detection, best known for its accuracy and performance. It achieves high accuracy with less data and a small training time.

YOLO architecture consists of three main blocks backbone, neck, and head [16]. The backbone is CNN- based DarkNet architecture with Cross-stage Partial Connections (CSP) which helps with repeated gradient problems and decreases the floating point operations. The backbone extracts most of the image features and passes them to the neck and head. Neck improves the information flow by constructing the Feature Pyramid Network (FPN)[13]. It helps in focusing on the low-level features which are important. Head generates the features maps of three different sizes of m x m, where m can be 20, 40, and 80, which helps the system to identify objects of different scales.

The Fig 2 represents consolidated YOLOv8 tiny architecture. Each CONV sub-block in the above diagram is a block of standard convolution layer, layer normalization followed by SiLU (a variant of ReLU) activation. C2f is a CSP bottleneck that consists of two convolutional layers with few Bottleneck layers. Spatial Pyramid Pooling – Fast (SPPF) block contains two convolutional layers, each followed by max pooling [10][19]. The concat layer performs tensor concatenation.

Detect sub-block has six convolution layers in total. Three layers of which are used for classification and the other three for regression.The backbone of YOLO consists of CONV followed by CONV and C2f as a block repeated four times. We then pass the feature maps through the Spatial Pyramid Pooling – Fast (SPPF) layer as a part of the backbone. Downsampling followed by upsampling helps to detect small objects as we have more deep feature maps at the end when compared to the initial layers.

The neck contains a upsample layer, concatenation followed by C2f, which is repeated twice. The neck helps combine the features learned by many layers of the backbone, which helps detect small objects. The head consists of two sub-blocks. The first sub-block has CONV, concat followed by C2f repeated twice. The second sub-block is detected.

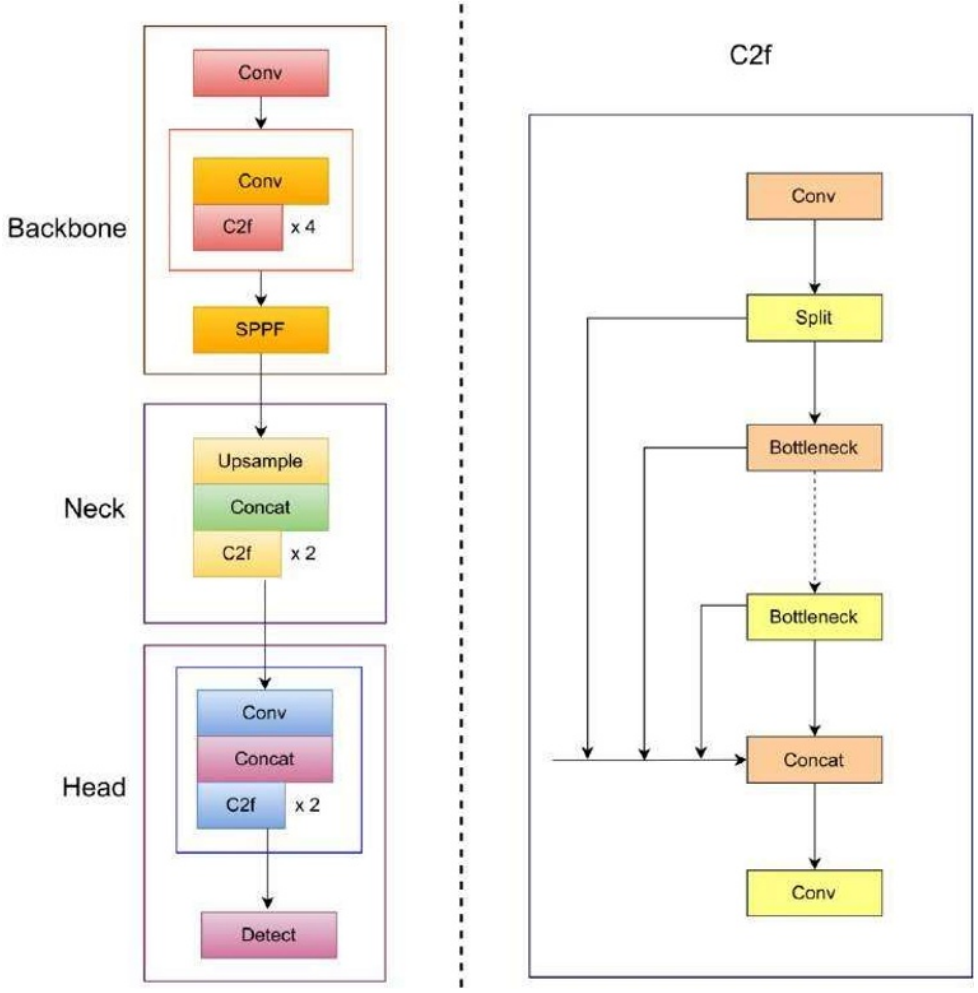

Fig. 2. YOLO Architecture

3.1 Data augmentation

Scaling, colour space adjustments, and Mosaic augmentation are some of the data augmentation techniques used. Scaling is changing the input dimensions (height and width) and adjusting the colour in images. Mosaic augmentation is combining at most four images of various scales into a single image. This improves the prediction of small objects in an image [12]. Mosaic augmentation helps model training in the initial stages as training progresses, affecting the model performance. YOLOv8 is trained without mosaic augmentation in the last few epochs, which is not the case in previous versions.

3.2 Anchor free detection

Anchor boxes have a standard set of predefined sizes. This helps in the fast prediction of bounding boxes. The format of the anchor box used in the YOLO algorithm is as follows

$$O = \begin{bmatrix} C \\ X_c \\ Y_c \\ O_h \\ O_w \end{bmatrix} \tag{1}$$

The matrix (1) represents the single object in an image. Contents of the ma- trix are explained as C is a class index, X_c and Y_c are the x and y coordinates of the object center. O_h and O_w are height and width of the object.

The anchor box in the previous YOLO version is calculated based on the deviation from a known anchor of the training set, which can be complicated and does not perform well on custom datasets. When a pre-trained base is used, the anchor box distribution used can represent the benchmark anchor distribution. This may lead to conflict between benchmark and custom anchor box distribution. Instead of calculating the deviation from the known anchor, YOLOv8 predicts the centre of the image directly, which reduces the number of bounding boxes. Early versions of YOLO uses Three detect head, whereas YOLOv8 uses one detect head.

3.3 Loss Function

YOLO classifies an object and calculates the bounding box simultaneously. The total loss function used is the combination of classification loss and regression loss. Classification loss is standard cross entropy loss and for bounding box regression loss we use mean squared error.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{2}$$

$$CE = -\sum_{i=1}^{n}[y_i \; log(\hat{y_i})] \tag{3}$$

In eqn.(2) MSE is a mean squared error and in eqn.(3) CE is cross entropy loss of n number of samples. In both equations, yi is the ground truth label for i-th sample, and y predicted label for i-th sample.

$$L_{total} = L_{CE} + L_{MSE} \tag{4}$$

In eq (4) the LCE is the cross entropy loss for the classification task and LMSE is a mean square error for the bounding box. Ltotal is the total loss function.

3.4 Model complexities of YOLOv8

YOLOv8 offers five different network architecture sizes: tiny, small, medium, large, and extra-large. These models can be used as per the need for a trade-off between accuracy and inference time. We trained tiny and small architecture. This is because of the constraints on model size, computational resources, and minimum inference time required [21].

4 Proposed Methodology

One-step classification and regression-based object detection networks are more compatible with quality control tasks in the industry when compared to two-step region proposal networks. One-step classification and regression networks have the advantage of smaller model size, less training time, high accuracy with fewer data, less inference time, and performance well with limited resources on edge devices. Single Shot Detector (SSD), SqueezeDet, DetectNet, and YOLO are some examples of one-step object detection systems. YOLO is the most popularly used object detection algorithm. It was originally implemented in the C programming language. Continued development of this algorithm with architecture variations and implementation language changes led to various versions. YOLOv8 is the most recent and best-performing version that has been chosen for implementation.

4.1 Model Building

4.1.1 Training

The training dataset is prepared by annotating the objects of interest in the images. The annotations include the object's class and its bounding box coordinates. The input images are pre-processed by resizing the images into 640 X 640 while maintaining the original image's aspect ratio. The image is then normalized so that all of its pixel values lie between 0 and 1[15].

YOLOv8 uses the modified version of darknet architecture as the backbone. The pre-processed images are fed into neural networks to train the model, which is pre-trained on the COCO dataset. Object detection systems generally predict more than one bounding box for a single object. To tackle this problem, Non-Max Suppression (NMS) is used. This technique uses the Intersection over Union (IoU) to reduce the number of predictions and bounding boxes[18]. NMS algorithm discards a prediction if the IOU value crosses some threshold over other predictions. IOU reduces the prediction error by comparing the ground truth and the predicted area.

4.1.2 Model evaluation

Once you have trained the model, you can evaluate its performance by running it on a set of test images different from the ones used in training. Choosing the right metrics for evaluating object detection can be a challenging task. Most of the systems use precision, recall, f1 score, and mean average precision(mAP) as metrics. Precision represents a fraction of positive predictions that are correctly detected as the positive class. It is represented in eq. (5).

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{5}$$

The recall is a metric that quantifies the ability to correctly identify positive instances among all the instances that are actually positive. It is represented in eqn.(6).

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{6}$$

$$F1\ Score = 2 \times \frac{\text{Precision} \times Recall}{\text{Precision} + Recall} \tag{7}$$

Both precision and recall are inversely proportional to each other [4]. Model's capability of predicting a positive class input as positive class decreases as the number of the positive class prediction increases. The F1 score is a combination of both precision and recall. Higher F1 score better performance in eq. (7). The average Precision metric is the weighted mean of Precision scores achieved at each precision-recall curve threshold, with the increase in recall from the previous threshold used as the weight.

$$AP = \sum_{k=0}^{n-1} [Recall(k) - Recall(k+1)] \times \text{Precisions}(k) \tag{8}$$

The Average Precision in eqn.(8) is explained as n number of thresholds, calculating the difference between the current and the next recall values and multiplying the weight by the current precision value. Mean Average Precision that takes into account the balance between precision and recall, while also accounting for both false positives and false negatives. This makes mAP a well-suited evaluation measure for a wide range of detection applications.

$$mAP = \frac{1}{n} \sum_{k=1}^{n} AP_k \tag{9}$$

The mAP is calculated by finding Average Precision (AP) for each class and then averaging over a number of classes. It is represented in eq. (9).

In the inference stage, live feed from the camera is taken as input using OpenCV which is an open- source computer vision library [7]. The preprocessing of the video frames remains the same as the pre- processing explained in training. The video frames are passed to the model to get a dense prediction. Apply NMS and project the results on the screen.

4.2 Format Conversion

Most machine learning models are enormous in size and take a lot of time to compute the end results because of their complex computations. They are trained on dedicated hardware to accelerate the process. YOLO is one of the state of the art architecture well known for its speed, but deploying this on an IoT device(Raspberry Pi) can be challenging.

The problem is the model format used in YOLOv8. Pytorch is one of the most popular machine-learning frameworks used for computer vision and Natural Language Processing (NLP). Pytorch saves the model in .pt or .pth file format, which has all the information needed to load, train and make inferences using models, which makes file format heavy[2].

ONNX is another file format that runs on ONNX runtime to make inferences. While converting the model format to ONNX, we replace the computation with constant results to a constant. This helps the model to save time while running inference on machines with low computational power.

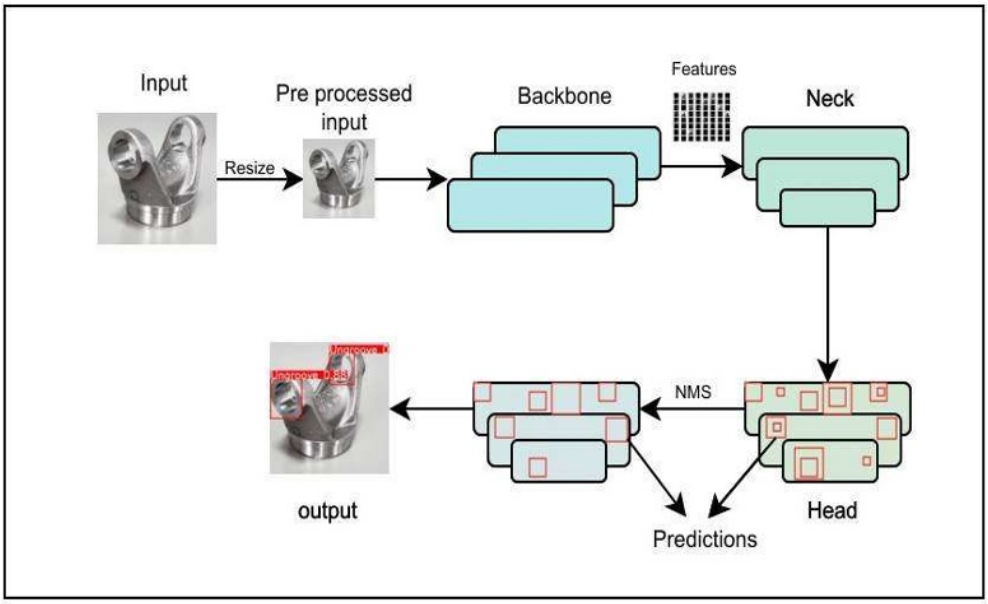

Fig. 3. Model Workflow of Methodology

4.3 Deploying on Rasberry pi

The Raspberry Pi is a relatively affordable single-board computer. It has many input/output ports that help to interact with other devices. It also has a set of General Purpose Input/Output(GPIO) pins that can be used to control electronic devices for physical computing and investigate the Internet of Things (IoT)[8].

The main component of it is Central Processing Unit(CPU) which is responsible for executing instructions and data processing. Its architecture is based on ARM Processor, which is frequently used in embedded systems and mobile devices and is created for low power consumption. System on a Chip(SoC) approach is used to design Raspberry's architecture. It has two types of memory: non-volatile storage and Random Access Memory(RAM)[23]. The storage capacity of the system can be expanded using external storage devices like SD cards.

Raspberry Pi runs different types of operating systems like Ubuntu, Raspbian, etc. In the proposed system, we have used Debian 'bullseye' OS to run Raspberry Pi 4. The formatted model has been ported to a Raspberry Pi 4 for inference. It is challenging to interact with hardware components like a conveyor belt using a CPU. GPIO pins of Raspberry Pi help in easy interaction with hardware components. When the model detects an unprocessed part, it sends a signal through the GPIO pins of the Raspberry Pi 4, which are in turn connected to other hardware components like the conveyor belt. The conveyor belt stops when the GPIO pin of the Raspberry Pi is high[14]. We have to remove the unprocessed part from the conveyor manually. The system then detects the changes and resumes the inspection of processed parts.

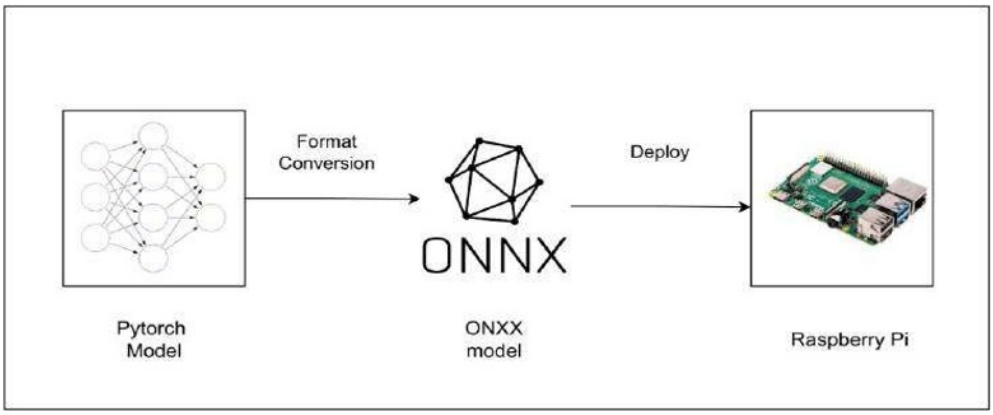

Fig. 4. Format conversion of model

5 Results

5.1 Dataset Description

5.1.1 Classes

The dataset contains two classes, processed (groove) and non-processed (ungrooved) parts, which are the target labels for the classification task. Each image in the dataset contains a set of objects. Each object is assigned one of these two class labels based on whether it contains a groove or not. The classification task aims to learn a function that can accurately classify objects in a new image into one of these two classes based on the patterns and features in the image.

5.1.2 Size

The dataset contains 610 images, with 542 images for training and 68 images for testing. This split is common in machine learning, where a subset of the available data is used to train the classifier, and the rest is used to evaluate its performance on new, unseen data. The size of the dataset is relatively small, which can pose a challenge for building a robust and generalized classifier.

(a) (b)

Fig.5. (a) Processed part and (b) Unprocessed part

The images in the dataset have a large dimension of 2622 x 2622 pixels. This high resolution can be useful for capturing fine details in the images, such as the shape, texture, and pattern of the grooves. However, it also means that the images may take longer to process and require more memory to store, which can affect the performance and scalability of the classifier. To address this challenge, it may be necessary to preprocess the images, such as by resizing them to a smaller size or applying a compression algorithm.

5.1.3 Color channels

The dataset comprises RGB images, which means that each image has three colour channels (red, green, and blue). These colour channels can be used to extract features and patterns from the images that can help the classifier distinguish between the two classes. For example, the classifier can look for differences in the intensity and colour of the pixels in the processed and unprocessed regions, and use this information to make a decision. The choice of features and how they are extracted from the images can have a significant impact on the performance of the classifier and may require careful experimentation and tuning.

5.1.4 Experimental results

Training YOLOv8 tiny and small models on a custom dataset for 1000 epochs with early stop. YOLOv8 small stopped after 278 and tiny after 219 epochs this is due to the small data set[21]. Validating small and tiny models in both the file formats on the custom validation set, results are as follows.
The results of our experiments show that the YOLOv8 algorithm can detect grooves in tube yokes with high accuracy and precision. All four models have approximately the same performance in terms of precision, recall, and mean average precision. The difference between these four models can be seen in terms of inference time taken after deploying the model on an IoT device[20]. Both PyTorch models take more than 2 seconds to process a single image on an IoT device.

Table 1. Results of various models

Model	Model Size	Precision	Recall	mAP 0.5	f1 score
Pytorch tiny	5.9	99.3	96.1 MB	98.9	48.83
Pytorch small	21.5	99.3	95.8 MB	98.9	48.75
ONXX tiny	11.7	99.6	97.5 MB	98.9	49.26
ONXX small	42.6	98.4	95.8 MB	98.9	48.54

The ONNX file format performs better despite the large size when compared to PyTorch models. ONNX small model takes 1 second, and the tiny model takes 0.4 seconds to process a single image. The less time taken for inference is due to the replacement of computations with constant results.

Fig. 6. (a) Ground truth labels of a validation batch and (b) Prediction of above validation batch

6 Conclusion

The detection of groove lines in tube yokes is a critical task in the manufacturing of driveshafts. This research paper explores various object detection algorithms and proposes YOLOv8 as one of the best algorithms that can be implemented for quality control in industries. The model is deployed on a raspberry pi and later is integrated with a conveyor belt. We need to minimize the inference time with some compromises in accuracy. The relatively small model size and format conversions of YOLOv8 help to achieve this accuracy time trade-off. YOLOv8 object detection algorithm gave significant results in the lab as well as in the company's environment. In the future, we will focus on integrating the proposed system with other systems, such as robotic arms or automated sorting systems, to improve the efficiency of quality control. The model can be further developed to detect groove lines in tube yokes of different sizes with significant accuracy and less inference time.

References

1. Mihaela Andrei, Simona Moldovanu, and Nicusor Nistor. Nonlinear transforma- tions applied to image processing on the rgb led display. In 2018 IEEE 24th Inter- national Symposium for Design and Technology in Electronic Packaging(SIITME), pages 319–322, 2018.
2. Pritul Dave, Arjun Chandarana, Parth Goel, and Amit Ganatra. An amalgama- tion of yolov4 and xgboost for next-gen smart traffic management system. PeerJ Computer Science, 7:e586, 06 2021.
3. Lixuan Du, Rongyu Zhang, and Xiaotian Wang. Overview of two-stage object detection algorithms. Journal of Physics: Conference Series, 1544:012033, 05 2020.
4. Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.
5. Reagan L. Galvez, Argel A. Bandala, Elmer P. Dadios, Ryan Rhay P. Vicerra, and Jose Martin Z. Maningo. Object detection using convolutional neural networks. In TENCON 2018 - 2018 IEEE Region 10 Conference, pages 2023–2027, 2018.

6. Ross Girshick. Fast r-cnn. In Proceedings of the 2015 IEEE International Confer- ence on Computer Vision (ICCV), ICCV '15, page 1440–1448, USA, 2015. IEEE Computer Society.
7. Zheng Guo-rong, Xiong Chang-Zhen, and Zhang Yan.A method of embedded video surveillance based on opencv. In 2011 International Conference on E- Business and E-Government (ICEE), pages 1–4, 2011.
8. Nikunj Gupta, Steve R. Brandt, Bibek Wagle, Nanmiao Wu, Alireza Kheirkhahan, Patrick Diehl, Felix W. Baumann, and Hartmut Kaiser. Deploying a task-based runtime system on raspberry pi clusters. In 2020 IEEE/ACM Fifth International Workshop on Extreme Scale Programming Models and Middleware (ESPM2), pages 11–20, 2020.
9. Donghyeon Han, Dongseok Im, Gwangtae Park, Youngwoo Kim, Seokchan Song, Juhyoung Lee, and Hoi-Jun Yoo. A dnn training processor for robust object de- tection with real-world environmental adaptation. In 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS), pages 501– 501, 2022.
10. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pool- ing in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9):1904–1916, 2015.
11. Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In Pro- ceedings of the ieee/cvf conference on computer vision and pattern recognition, pages 2888–2897, 2019.
12. Parvinder Kaur, Baljit Singh Khehra, and Er. Bhupinder Singh Mavi. Data aug- mentation for object detection: A review. In 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), pages 537–543, 2021.
13. Tsung-Yi Lin, Piotr Dolla´r, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 936–944, 2017.
14. Zehra Ozkan, Erdem Bayhan, Mustafa Namdar, and Arif Basgumus. Object de- tection and recognition of unmanned aerial vehicles using raspberry pi platform. In 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), pages 467–472, 2021.
15. Bandi Narasimha Rao and Reddy Sudheer. Surveillance camera using iot and raspberry pi. In 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), pages 1172–1176, 2020.
16. Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 779–788, 2015.
17. BS Rekha, Athiya Marium, GN Srinivasan, and Supreetha A Shetty. Literature survey on object detection using yolo. Int. Res. J. Eng. Technol., 7(6):3082–3088, 2020.
18. Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. Non-maximum sup- pression for object detection by passing messages between windows. In Asian Conference on Computer Vision, 2014.
19. Mingyu Sheng, Hongfan Zeng, Jingnan Li, and Wenbo Sun. Pooling and convo- lution layer strategy on cnn for melanoma detection. In 2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), pages 153–161, 2021.
20. Pertiwang Sismananda, Maman Abdurohman, and Aji Gautama Putrada. Perfor- mance comparison of yolo-lite and yolov3 using raspberry pi and motioneyeos. In 2020 8th International Conference on Information and Communication Technology (ICoICT), pages 1–7, 2020.
21. Viswanatha V, Chandana R K, and Ramachandra A. C. Real time object detection system with yolo and cnn models: A review, 2022.

22. Shenghui Wang, Leilei Niu, and Nan Li. Research on image recognition of insula- tors based on yolo algorithm. In 2018 International Conference on Power System Technology (POWERCON), pages 3871–3874, 2018.
23. Zhihe Zhao, Kai Wang, Neiwen Ling, and Guoliang Xing. Edgeml: An automl framework for real-time deep learning on the edge. In Proceedings of the Interna- tional Conference on Internet-of-Things Design and Implementation, IoTDI ’21,
24. page 133–144, New York, NY, USA, 2021. Association for Computing Machinery.
25. Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. IEEE Transactions on Neural Networks and Learning Systems, 30(11):3212–3232, 2019.
26. Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In 2019 International Conference on 3D Vision (3DV), pages 85–94, 2019.