big data and cognitive computing

MDPI

Article

YOLO-v5 Variant Selection Algorithm Coupled with Representative Augmentations for Modelling Production-Based Variance in Automated Lightweight Pallet Racking Inspection

Muhammad Hussain

Department of Computer Science, Centre for Industrial Analytics, School of Computing and Engineering, University of Huddersfield, Queensgate, Huddersfield HD1 3DH, UK; m.hussain@hud.ac.uk

Abstract: The aim of this research is to develop an automated pallet inspection architecture with two key objectives: high performance with respect to defect classification and computational efficacy, i.e., lightweight footprint. As automated pallet racking via machine vision is a developing field, the procurement of racking datasets can be a difficult task. Therefore, the first contribution of this study was the proposal of several tailored augmentations that were generated based on modelling production floor conditions/variances within warehouses. Secondly, the variant selection algorithm was proposed, starting with extreme-end analysis and providing a protocol for selecting the optimal architecture with respect to accuracy and computational efficiency. The proposed YOLO-v5n architecture generated the highest MAP@0.5 of 96.8% compared to previous works in the racking domain, with a computational footprint in terms of the number of parameters at its lowest, i.e., 1.9 M compared to YOLO-v5x at 86.7 M.

Keywords: defect detection; YOLO-v5; pallet racking; smart manufacturing; quality inspection

Citation: Hussain, M. YOLO-v5 Variant Selection Algorithm Coupled with Representative Augmentations for Modelling Production-Based Variance in Automated Lightweight Pallet Racking Inspection. *Big Data Cogn. Comput.* **2023**, *7*, 120. https://doi.org/10.3390/bdcc7020120

Academic Editor: Carson K. Leung

Received: 30 April 2023
Revised: 30 May 2023
Accepted: 12 June 2023
Published: 14 June 2023

1. Introduction

Warehouses play a critical role in supply chain management (SCM), as they are responsible for storing and managing inventory prior to shipment. With the rise of e-commerce, further accentuated by the impact of the recent pandemic (COVID-19), warehouses are experiencing a surge in demand for their services, leading to an increase in operational complexities [1]. To address these challenges, businesses are turning to technological solutions such as artificial intelligence (AI) [2] and the Internet of Things (IoT) [3] with the aim of optimizing their operations [4].

Within the warehouse environment, forklifts, pallet racking, and employees are all seen as critical components, playing a critical role in making sure the inventory is efficiently stored, maintained, and dispatched. However, issues can arise between these components, i.e., forklifts may cause damage to pallet racking, due to several reasons such as driver carelessness, which can lead to serious consequences such as operational downtime, compromised stock quality or complete wastage, reputational damage, and, in severe cases, loss of human life [5]. Currently, the most common approach to preventing pallet damage is through human-led racking inspections, but these can be time-consuming, expensive, and prone to bias or judgement errors.

To address these limitations, various solutions have been proposed in the industrial warehousing market, such as column protectors and sensor-based devices such as Rack-eye [6]. However, these solutions have their own limitations, such as constrained detection range and high installation/maintenance costs. Computer vision, also known as machine vision in the industrial context, can offer an attractive proposition for pallet racking damage detection, with the potential to significantly reduce costs, broaden coverage, and provide real-time inference capabilities. By developing lightweight convolutional neural network

(CNN)-based architectures for racking damage detection and delivering strategical deployment via edge devices, businesses can potentially improve their warehouse operations by ensuring the safety of a critical component, i.e., pallet racking, and ultimately guaranteeing stock preservation prior to shipment.

1.1. Literature Review

Due to automated, non-invasive pallet racking inspection being a relatively new research domain, there is a lack of relevant literature. Therefore, to conduct a thorough review, the research base was broadened to include similar fields such as structural health monitoring (SHM).

In 2012, the computer vision research community experienced a significant advancement in CNNs with the introduction of AlexNet [7], coinciding with the development of graphical processing units (GPU) for accelerating matrix calculations and the introduction of the rectified linear activation function (ReLU). ReLU was designed to optimize network convergence with its simple mathematical composition. In the subsequent years, there was a proliferation of CNN-based architectures focused on both image classification and object detection, such as Google-Net [8], ResNet [9], VGG-Net [10], RCNN, Fast-RCNN [11], and Faster-RCNN [12]. These architectures were developed based on mutual objectives of improving accuracy, lessening inference time, and easing power consumption.

In a study by Chuan-Zi Dong et al. [13], the authors review computer vision (CV) methods for various applications in SHM. They bifurcate SHM into local and global segments. Local SHM involves the detection of defects such as cracks and delamination, while global SHM focuses on location measurement, structural and vibration analysis, and modal identification. The authors argue that for CV-based SHM to work effectively in terms of identifying defects, it is necessary to have large amounts of training data that are also representative in their manifestation of the real-world scenarios for the given domain. However, these requirements can be difficult to achieve in manufacturing facilities due to factors such as production downtime and limited access for data collection. Therefore, the authors propose a method that utilizes a small set of original images and applies purposeful data augmentations and architecture selection for generating representative data samples to create a highly accurate detection network.

Architectures mentioned earlier such as VGG, ResNet, and Google-Net are known as state-of-the-art (SOTA) models trained on large benchmark datasets, and they can be utilized via transfer learning for generalization on specific datasets for use in different sectors such as healthcare, security, and renewable energy. However, the complex internal architecture of these SOTA models means high computational demand, requiring cloud-based infrastructure for deployment. Although cloud deployment may be preferred in certain domains/applications, it is not recommended in certain manufacturing scenarios [14,15] due to constrained capacity, i.e., close-to-the-source data actioning, limited battery life, and security concerns.

He et al. [16] proposed a novel approach for defect detection in industrial settings, explicitly focusing on defect identification in steel surfaces. The authors introduced a multilevel feature-fusion network (MFN), combining features from various aspect ratios, i.e., at different stages of the architecture, aimed at information conservation. The proposed architecture achieved striking results, with the defect identification accuracy reported as 99.67% and 82.3% MAP for defect localization. Moreover, the system was able to operate at a detection speed of 20 ft/s while retaining 70% MAP.

The authors of [17] focused on defect determination in metal castings and proposed a custom CNN architecture. They compared the performance of the developed architecture with state-of-the-art (SOTA) models such as Inception, ResNet, and MobileNet. The authors implemented a depth-wise convolutional process, comparable to the one used in MobileNet, in addition to optimization strategies such as blur-pooling, stochastic weight average, and the squeeze–Excitation (SE) technique. Although the authors argued the proposed architecture was preferred due to the limited number of parameters compared to the

SOTA models, the accuracy of the proposed architecture was significantly lower at 81.87% compared to Inception, outputting an accuracy of 91.48%.

Moving closer to the pallet racking domain, Fahimeh Farahnakian et al. [18] investigated automated damage detection in pallet racking based on the image segmentation mechanism. The authors implemented Mask-RCNN as the architecture with ResNet-101 selected as the backbone for feature extraction. The achieved performance with respect to the mean average precision was reported as 93.45% based on an intersection over union (IoU) of 50%. Looking deeper at the development pipeline, it was observed that the training dataset was not representative of the production floor as it was not captured in a warehouse environment, raising concerns about the true generalization of the proposed architecture. In contrast, our research gives true representation a high degree of importance on by utilizing a real dataset, ascertained from multiple manufacturing facilities, in addition to representative data augmentation strategies for scaling and variance injection purposes.

Hussain et al. [19] proposed a strategically placed edge device solution for pallet racking inspection that was based on MobileNetV2 base architecture. The authors justified MobileNetV2 selection as depth-wise utilization within the architecture resulted in a lightweight architecture with reduced internal convolutions, achieving a respectable mean average precision of 92.7%. The developed architecture was aimed for deployment onto constrained hardware such as a Raspberry Pi [20]. Extending their work further, Hussain et al. [21] further improved their research, proposing a rack feature modelling pipeline, with the goal of overcoming data scarcity. For this, the authors selected Yolov7 as the single-stage architecture due to its lightweight internal composition and real-time inference capabilities. The reported mean average precision was 91.1%, i.e., less compared to their preceding performance with MobileNetV2, with a decrease in MAP of 1.6%.

In summary, pallet racking damage prevention and inspection can be split into three categories: mechanical, sensor-based, and machine vision. Purely mechanical solutions are based on damage prevention via rackgurads and lack any intelligence for reporting damage. Sensor-based solutions such as Rackeye [6] have limited intelligence, i.e., report damage exceeding a certain threshold with respect to an accelerometer; however, they are commercially expensive, as a sensor must be placed on every racking leg. The latter, i.e., machine vision, is an emerging area within pallet racking inspection, seen as a non-invasive method for pallet racking inspection strategically placed on the forklift, hence reducing cost and providing broader coverage. Based on the works presented in the literature review, it can be confidently concluded that there is a lack of research directly focused on the implementation of machine vision for automated pallet racking inspection. One of the reasons for the slow adoption of AI for this domain is due to the absence of open-source pallet racking datasets, along with the difficulties of attaining image data from within manufacturing facilities due to operational constraints and access restrictions. Without sufficient training samples of pallet racking, researchers have been unable to generate meaningful research output for the automated inspection of defective pallet racking.

1.2. Paper Contribution

This research was focused on delivering an automated pallet inspection architecture with two key objectives, i.e., high performance and computationally light weight. As mentioned earlier, the automated racking domain is a developing field and, hence, the procurement of racking images with the aim of collating a racking dataset can be a cumbersome task. Hence, the first contribution was the proposal of several augmentations that were selected based on modelling production floor conditions within warehouses. The proposed augmentations had to be representative of the real-world application in order to provide high model generalization, leading to high accuracy.

Secondly, as in addition to high accuracy, the proposed architecture selection mechanism would need to have a lightweight footprint in order for it to be deployable on the edge in constrained environments, the variant selection algorithm was proposed, starting with extreme-end analysis and providing a protocol for selecting the optimal architecture

with respect to accuracy and lightweight footprint. Our proposed methodology resulted in the selection and development of the optimal architecture, achieving the highest reported MAP@0.5 of 96.8% compared with other works in this domain with the computational complexity at its lowest, i.e., number of parameters: 1.9 M compared to 86.7 M for YOLO-v5x.

2. Methodology

2.1. Dataset

From the reviewed literature, we ascertained that there was an absence of open-source data set for pallet racking. As a result, we collected our initial dataset from various manufacturing facilities, including Tile Easy, Lamteks, and Welt Handel. The data collection process involved utilizing a smartphone (iPhone 8) with a 12 MP camera for capturing images of racking. The rationale for selecting a 12 MP camera was based on the camera specification's similarity to the potential deployment hardware, i.e., Raspberry Pi camera.

The angle of capturing images was carefully considered to simulate the end-point device deployment, i.e., forklift adjustable brackets. Hence, to simulate this scenario, the user captured images by holding out the phone at waist height with the camera facing towards the racking and walking towards the racking. This enabled representative samples, simulating the deployment environment. Figure 1 shows normal and defective samples. It is evident from Figure 1 that variance exists at a global and internal class level. For global variance, we can observe various color variations within the image frames due to different racking colors and the layout of stock placed near the racking. For normal images, i.e., Figure 1A, global variance plays a pivotal role, whilst for the defective racking in Figure 1B, it can be observed that in addition to global factors, there are varying degrees of defects that need to be taken into account, i.e., minor damage may be misclassified as normal.

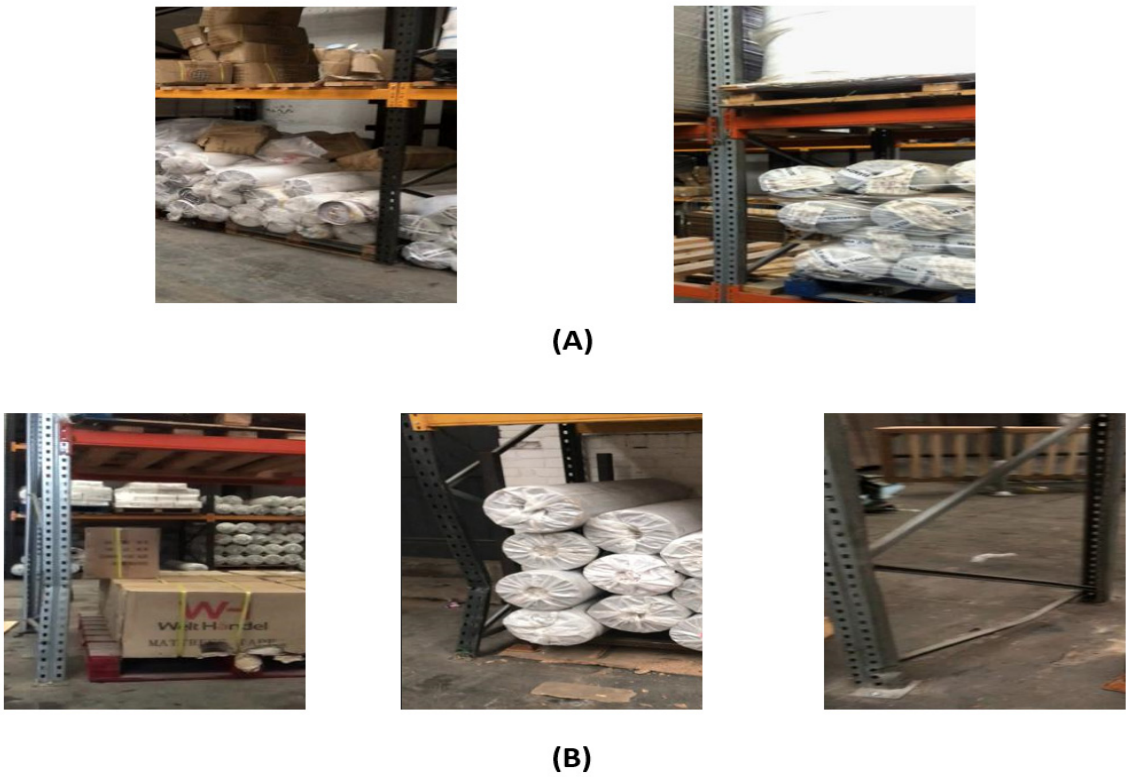

Figure 1. Original dataset. (**A**) Normal and (**B**) damaged racking.

2.2. Domain-Inspired Augmentations

The initial dataset consisted of approximately 100 images comprising normal and defective pallet racking instances. This sample size was simply too small for any purposeful training as the architecture would not have enough samples from which it can learn and

provide optimal weight convergence, i.e., network generalization. Data augmentations are traditionally used to scale image datasets both in terms of size and variance, but applying 'blanket' augmentations without domain logic rationale does not guarantee true generalization. Therefore, we focused on applying domain representative augmentations that were demonstrative of the manufacturing floor conditions with reference to pallet racking. Our first objective was to ensure the network would be rotationally invariant to facilitate the dynamic placement tolerance of the edge device and the resultant variations in the captured images. This variance was modelled by implementing 'angled' rotations via a matrix formula, expressed as (1)

$$A = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{1}$$

To factor in potential physical contact and resultant device displacement due to loaded pallets, pixel-shifting components (i_x, i_y) were integrated into the dataset augmentation mechanism (2). This would facilitate better network convergence on images captured on the production floor where the edge device may experience slight displacement due to pallet movement.

$$A = \begin{bmatrix} 1 & 0 & i_x \\ 0 & 1 & i_y \end{bmatrix} \tag{2}$$

The translation matrix A was then transformed into an array consisting of u_x, representative of the input image, u_y, representative of the resultant image, and A as the translation matrix, expressed as (3)

$$u_y(x, y) = u_x(A_{11}x + A_{12}x + A_{13}, A_{21}x + A_{22}y + A_{23} \tag{3}$$

Figure 2 presents a sample augmentation result for the displacement of 11° left and right, which may be the result of the edge device being offset due to unwarranted contact with loaded pallets.

Figure 2. Device displacement modeling.

To increase the model's ability to generalize and adapt to the different levels of light intensities commonly found within different production facilities, the second criteria focused on adjusting pixel-level intensities resulting in brighter and darker augmented images. These were ascertained by introducing pixel-level brightness manipulation for simulating variations in LUX intensity. Modeling varying LUX intensities was focal for broader deployment potential as different manufacturing facilities have varying LUX levels due to multiple factors, i.e., location, regulatory requirements, and the nature of the stored inventory. The mathematical expression for brightness augmentation, aimed at adjusting pixel values for the given sample image to increase or decrease the brightness, is expressed as (4)

$$z_x(i, j, k) = \min\big(\max\big(\alpha \times z_y(i, j, k) + \beta, 0\big),\ 255\big) \tag{4}$$

where z_x signifies the resultant image, z_y represents the input image, α is the brightness factor, β equates to the bias, i and j represent the pixel coordinates, and k signifies the color channel, i.e., red, green, or blue. The min and max functions were utilized to ensure that the resulting pixel values were clipped within the valid range of 0 to 255.

To account for variations in camera quality, pixel blurring was introduced at multiple degrees. Pixel-level noise was also injected to model extreme scenarios such as dust particles clogging the camera lens causing distorted samples. The mathematical expression for image blurring augmentation can be represented as (5)

$$i(x, y) = \Sigma\_(i = -k)^{(k)} \Sigma\_(j = -k)^{(k)} w(i, j) \times J(x + 1, y + 1) \quad (5)$$

where $i(x,y)$ is the blurred image, $J(x,y)$ is the original image, and $w(i,j)$ is a convolution kernel that determines the amount and direction of blurring. The size of the kernel is represented by k, and larger values of k result in more blur.

The resultant datasets after applying the representative data augmentations discussed above are presented in Table 1.

Table 1. Transformed datasets.

Dataset	Samples
Training	1905
Validation	129

2.3. Proposed Architecture Selection Mechansim

The selection criteria for the architecture were based on domain-specific deployment requirements. As mentioned earlier, due to the constrained nature of the deployment environment, the architecture would need to satisfy multiple requirements such as being lightweight in order to be hostable on an edge device, along with reduced computational load for energy conservation and real-time inference capacity. In addition to the stringent deployment requirements, the architecture would need to provide high accuracy in order to justify its efficacy compared to similar works.

The preceding requirements ruled out two-stage detectors such as Faster-RCNN due to their architectural depth and computational load, making them practically infeasible for deployment onto an edge device such as Raspberry Pi. As a single-stage detector, based on these requirements of a single-stage object detector being preferred over a two-stage detector, YOLOv5 was selected as the base architecture. Interestingly, it contains multiple variants within its arsenal (YOLOv5-N/S/M/L/X), as shown in Table 2, permitting developers to select the most suitable variant for their application based on the performance with respect to accuracy, computational load, and real-time inference capacity.

Table 2. YOLO-v5 variant comparison.

Model	Average Precision (@50)	Parameters	FLOPs
YOLO-v5s	55.8%	7.5 M	13.2B
YOLO-v5m	62.4%	21.8 M	39.4B
YOLO-v5l	65.4%	47.8 M	88.1B
YOLO-v5x	66.9%	86.7 M	205.7B

YOLOv5 is essentially an evolution of its preceding variants with incremental improvements, primarily the introduction of modules such as Mosaic, BottleneckCSP, Focus, Spatial Pyramid Pooling (SPP) [22], and Path Aggregation Network (PANet) [23], as shown in Figure 3. Internally, the architecture comprised firstly the backbone required for feature extraction purposes from the input images, neck module, and three separate heads for

detection. Additionally, cross-stage partial networks (CSPs) were implemented in coordination with the backbone for suppressing the computational load. Furthermore, the implementation of the focus layer aimed to reduce the number of parameters, i.e., FLOPs, along with CUDA memory, resulting in an improved forward–backward propagation rate. YOLOv5 also integrates the SPP module, aimed at expanding the receptive field whilst performing the segregation of significant contextual features.

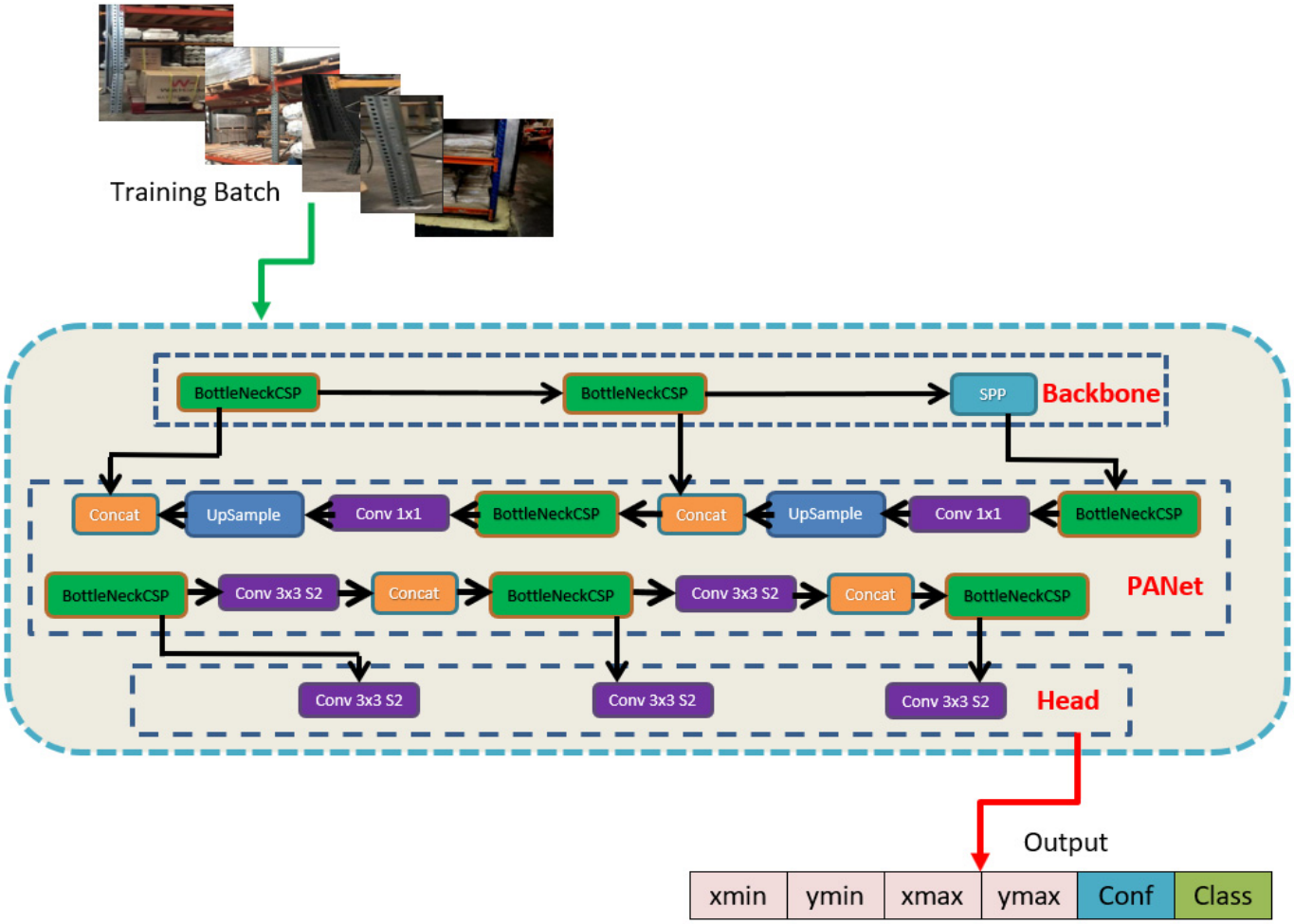

Figure 3. YOLO-v5 internal architecture.

Finally, the head maintains the same implementation logic deployed by its predecessors, i.e., YOLO-v3 and YOLO-v4, comprising three convolutional layers for predicting bounding box locations, classes, and scores. However, the principal expression for compiling object coordinates in YOLO-v5 has been altered compared to the preceding variants, which were expressed as (6)–(9)

$$b_x = \sigma(t_x) + c_x \tag{6}$$

$$b_y = \sigma(t_y) + c_y \tag{7}$$

$$b_h = p_h \times e^{th} \tag{8}$$

$$b_w = p_w \times e^{tw} \tag{9}$$

The altered formula for computing respective object coordinates in YOLO-v5 is expressed as (10)–(13)

$$b_x = (2 \times \sigma(t_x) - 0.5) + c_x \tag{10}$$

$$b_y = (2 \times \sigma(t_y) - 0.5) + c_y \tag{11}$$

$$b_h = p_h \times (2 \times \sigma(t_h))^2 \tag{12}$$

$$b_w = p_w \times (2 \times \sigma(t_w))^2 \tag{13}$$

As mentioned earlier, YOLO-v5 generates three outputs for the detection task: object class, bounding box, and object-ness scores. To calculate the loss function, two types of losses were fused: binary cross entropy for calculating the class and object-ness loss, and complete intersection over union (CIoU) for ascertaining loss with respect to the location. Hence, the formula for computing the loss function is expressed as (14)

$$\text{loss } = \lambda L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \tag{14}$$

The method by which the object-ness loss was calculated was dependent on the prediction layer size. YOLO-v5 deploys a tertiary prediction layer mechanism, i.e., layers of varying dimensions, with each of these layers assigned different weights. The default balance weights for each layer can be calculated via (15)

$$L_{obj} = 4.0 \times L_{obj}^{small} + 1.0 \times L_{obj}^{medium} + 0.4 \times L_{obj}^{large} \tag{15}$$

2.4. Proposed YOLO-v5 Variant Selection Mechanism

Table 3 presents the proposed algorithm for selecting criteria to ensure the optimal variant with respect to the domain application, i.e., high accuracy and less computational payload. The aim was to train and evaluate the extreme-end variants, i.e., YOLO-v5-nano and YOLO-v5-xtra, and incrementally select the next variant from the small model in the case of it having a smaller MAP@0.5 compared to the larger variant by a margin of 5%. Hence, the train function takes in two YOLO-v5 models (YOLOv5nano and YOLOv5xtra), a training dataset (T_d), and a validation dataset (V_d), as presented in Table 1. It trains both models on T_d and evaluates their MAP@0.5 scores on V_d. If the MAP@0.5 score of YOLO-v5xtra is greater than the MAP@0.5 score of YOLO-v5nano by more than 5%, the function enters a loop to increment YOLO-v5nano to the next variant, i.e., YOLO-v5medium, and evaluates its MAP@0.5 score on V_d. If the MAP@0.5 score of the new model is greater than the MAP@0.5 score of YOLO-v5xtra by more than 5%, the function selects the new model as the best model and exits the loop. Otherwise, the function continues to increment YOLO-v5nano to the next variant and evaluate its MAP@0.5 score until a compatible model is found. If the MAP@0.5 score of YOLOv5xtra is not greater than the MAP@0.5 score of YOLOv5nano by more than 5%, the function selects YOLOv5nano as the best model and returns it. The next variant function takes in a YOLOv5 model and returns the next variant in the sequence. The train function uses this function to increment YOLO-v5nano to the next variant. The loop in the train function exits if YOLO-v5nano is equal to YOLO-v5xtra or if one of the variants other than YOLO-v5xtra has an MAP@0.5 difference of less than 5% compared to YOLO-v5xtra.

Table 3. Pseudo algorithm.

```
train(YOLOv5nano, YOLOv5xtra, T_d, V_d):
    YOLOv5nano.train(T_d)
    YOLOv5xtra.train(T_d)
    nano_map = YOLOv5nano.evaluate(V_d)
    xtra_map = YOLOv5xtra.evaluate(V_d)
    while True:
        if xtra_map—nano_map > 0.05:
            YOLOv5nano = YOLOv5medium.train(T_d)
            new_map = YOLOv5nano.evaluate(V_d)
            if new_map—xtra_map > 0.05:
            YOLOv5nano = YOLOv5medium
            break
        else:
            xtra_map = new_map
    else:
        best_model = YOLOv5nano
        break
    if YOLOv5nano == YOLOv5xtra:
        break
    YOLOv5nano = next_variant(YOLOv5nano)
    nano_map = YOLOv5nano.evaluate(V_d)
  return best_model

next_variant(model):
  if model == YOLOv5nano:
    return YOLOv5medium
  elif model == YOLOv5medium:
    return YOLOv5large
  elif model == YOLOv5large:
    return YOLOv5xtra
  else:
    return model
```

3. Results

3.1. Hyperparameters

To ensure transparency in the reported results, global training parameters or hyperparameters were defined as presented in Table 4, before initiating any training. These hyperparameters were used for training all YOLO-v5 variants under experimentation. Google Collaboratory, part of the Google Cloud Platform (GCP), was selected as the training and evaluation platform due to free GPU access, facilitating faster training. However, GPU access was time-bound, and, hence, the number of epochs was capped at 40, whilst pretrained weights based on the IMAGENET dataset were utilized to increase the accuracy of the model.

Table 4. Hyperparameters/configuration.

Epochs	40
Image Size	640
Cache	RAM
Device Type	GPU
Pretraining	IMAGENET

3.2. Performance Evaluation Metrics

Multiple metrics were utilized for the purpose of evaluating the YOLO-v5 variants under experimentation, including precision, recall, and F1 scores, in addition to intersection over union (IoU). IoU was employed because the nature of the application was focused on defect localization as opposed to image classification. The Jaccard index, an interchangeable

term for IoU, enabled the processing of similarity calculations between the predicted K_g bounding boxes and the ground truth K_g, expressed as

$$\text{IoU} = \frac{area(K_p \cap K_g)}{area(K_p \cup K_g)}$$

The values of K_p and K_g were set from the onset as 0.5, which meant there must be an overlap of 50% between the predicted and the ground truth bounding box coordinates for the prediction to be considered. The mean average precision (MAP) was used because it considers the sensitivity of the predictions. Initially, precision, recall, and F1 score were calculated for a confidence threshold of 50%. Then, the MAP was computed using the average precision (AP) for each class, where S was representative of the number of classes, expressed as

$$\text{MAP} = \frac{1}{S} \sum_{i=1}^{S} AP_i$$

3.3. YOLO-v5 Extreme End Analysis

As per the proposed training algorithm, the first experimental iteration was based on training and evaluating the extreme-end variants of YOLO-v5, i.e., nano and extra. Training was carried out on the transformed pallet racking dataset presented in Table 1 based on the proposed augmentations. For integrity, all training was marshalled by the hyperparameters, outlined in Table 4.

Figure 4 presents the performance of the two variants with reference to MAP @ IoU (0.5). It is evident from Figure 4 that both variants showed impressive performance upon reaching the final epochs, with YOLO-v5x having the upper hand (99.4%) compared to YOLO-v5n, reaching 96.8%. Additionally, from the convergence analysis in Figure 4, it can be concluded that YOLO-v5x had reached convergence by the 20th epoch, whilst YOLO-v5n at the same time reached an MAP of 63.3%.

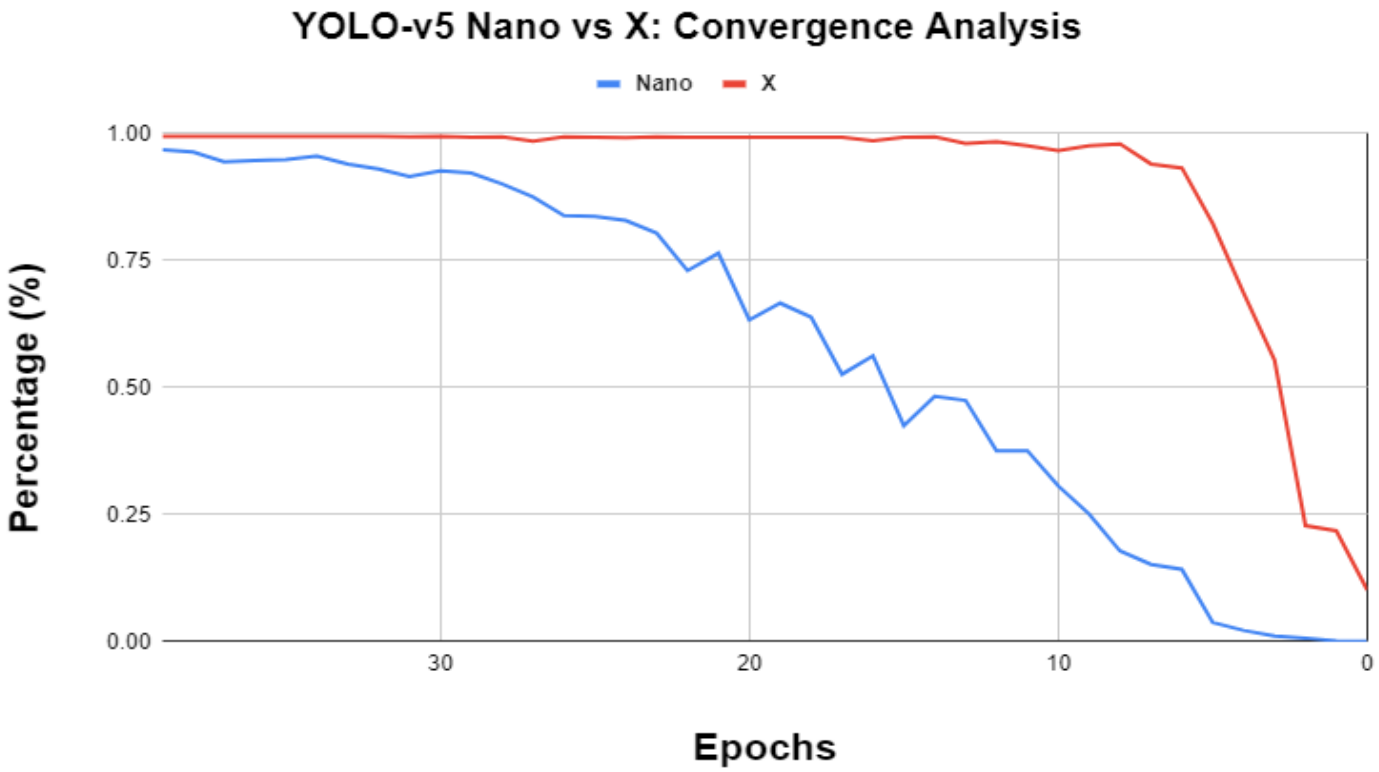

Figure 4. YOLO-v5 convergence analysis.

Based on the results present in Figure 4, YOLO-v5x is the dominant variant; however, as per the proposed training algorithm, the difference in MAP between the two variants is less than 5%, i.e., 2.6%. Hence, the next variant, i.e., YOLO-v5s, would not replace YOLO-v5n for retraining and rather the computational complexities of the two trained architectures would be taken into account.

Figure 5 presents the computational complexity comparison for the two variants based on (A) the number of parameters and (B) the number of FLOPs. Starting with the number of parameters, it is evident from Figure 5A that YOLO-v5n was more efficient in terms of being lightweight, containing 1.9 million parameters, compared to YOLO-v5x, comprising 86.7 million parameters.

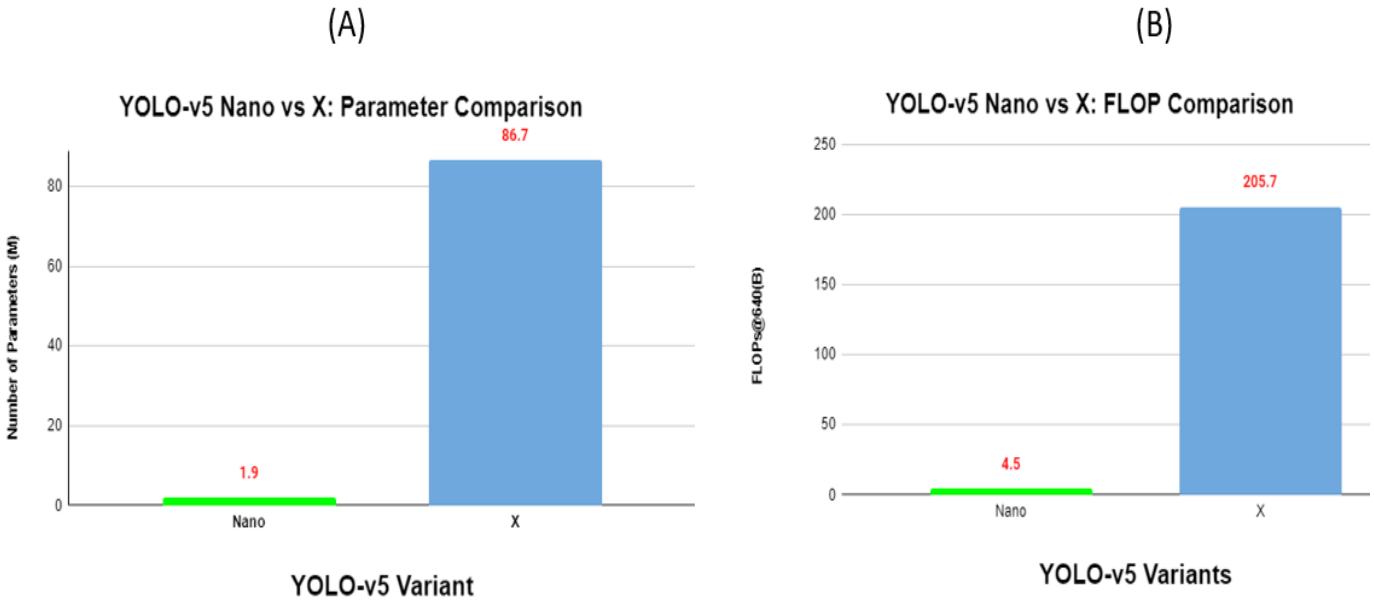

Figure 5. YOLO-v5 computational analysis. (**A**) Parameter; (**B**) FLOP Comparison.

Similarly, for the FLOP comparison, presented in Figure 5B, YOLO-v5n was more effective compared to YOLO-v5x with a difference of 201.2B. Observing the complete set of results in terms of accuracy and computational performance, presented in Table 5, and keeping in mind the deployment environment, YOLO-v5n was selected. The rationale for this was due to the fact that although YOLO-v5x provided higher MAP, the difference was small (2.6%), whilst computationally, YOLO-v5x was significantly more expensive compared to YOLO-v5n. As the deployment environment mandated a computationally lightweight architecture due to constrained deployment hardware, YOLO-v5n was the preferred choice.

Table 5. YOLO-v5: nano vs. x performance comparison.

Model	MAP-IoU@0.5	Parameters	FLOPs
YOLO-v5n	96.8%	1.9 M	4.5B
YOLO-v5x	99.4%	86.7 M	205.7B
Difference	2.6%	84.8	201.2B

The importance of the above evaluation is manifested in Figure 6, presenting the memory size requirements for the two YOLO-v5 variants under experimentation. For both deployment frameworks, OpenVino and ONNX, YOLO-v5n was more effective by a significant margin.

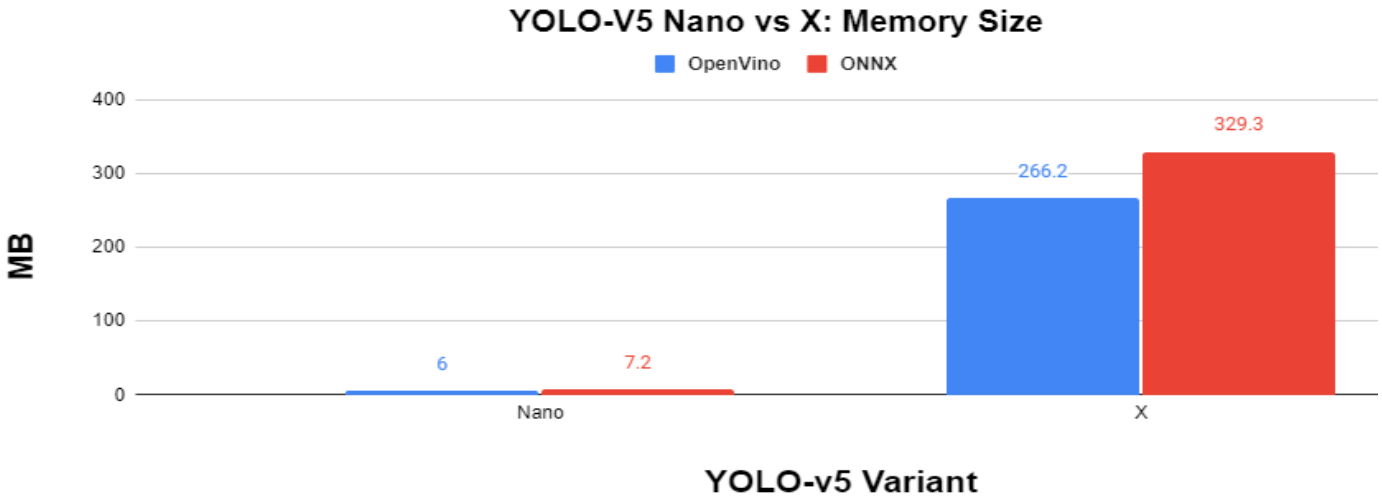

Figure 6. YOLO-v5 variant memory analysis.

3.4. YOLO-v5n Detailed Performance Evaluation

After selecting YOLO-v5n as the optimal architecture due to high accuracy along with high computational efficacy, the trained architecture was evaluated in more depth, focusing on additional metrics in addition to MAP such as precision and recall. Figure 7 presents the performance breakdown across a range of metrics for YOLO-v5n. The evaluated architecture provided impressive results for precision (89.6%), recall (92.7%), and MAP@0.5 (96.8%).

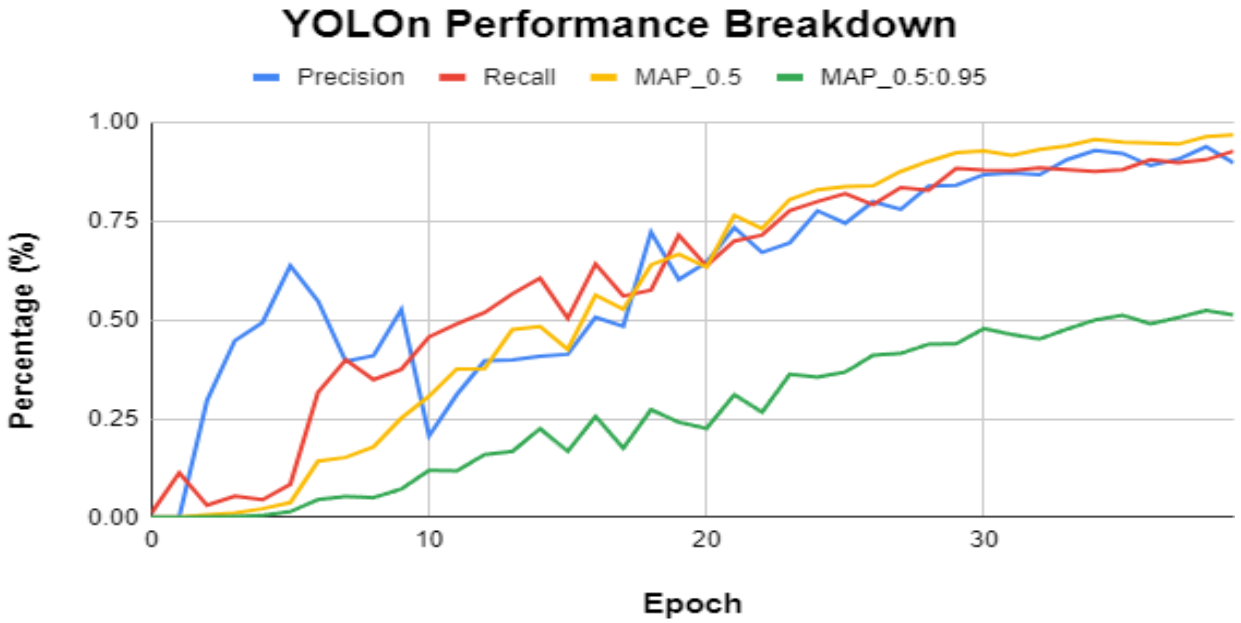

Figure 7. YOLO-v5n performance analysis.

However, from Figure 7, it can be observed that there is an outlier in the form of MAP@0.95. This is presented in a more focused view in Figure 8. MAP@0.95 can be interpreted as a more stringent requirement for accepting a predicted box, i.e., requiring 95% overlap with respect to the ground truth. Hence, from Figure 8, we can observe that when experimenting with MAP@0.95, the architecture only achieved 51.2%, whilst 96.8% was achieved with MAP@0.5. Although MAP@0.95 may be required in certain applications, in our particular application, MAP@0.5 was sufficient, as any positive inference from the architecture would be followed by a visual inspection for verification purposes by the forklift driver or floor manager.

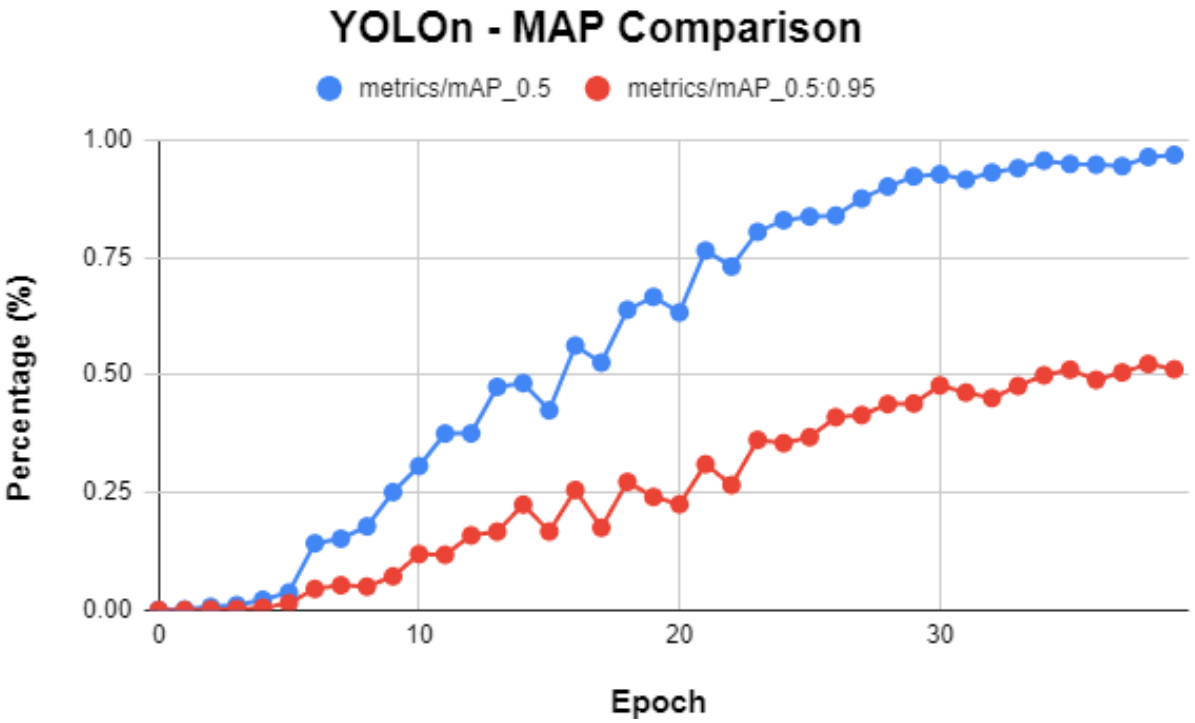

Figure 8. YOLO-v5n MAP analysis.

Figure 9 presents the validation loss for YOLO-v5n for the three key benchmarks. The class loss was the most effective, reaching a loss of 0.004, followed by the object loss at 0.025, with the box loss, referring to the bounding boxes, reaching 0.035. The presented validation losses further endorsed the high efficacy of the trained architecture.

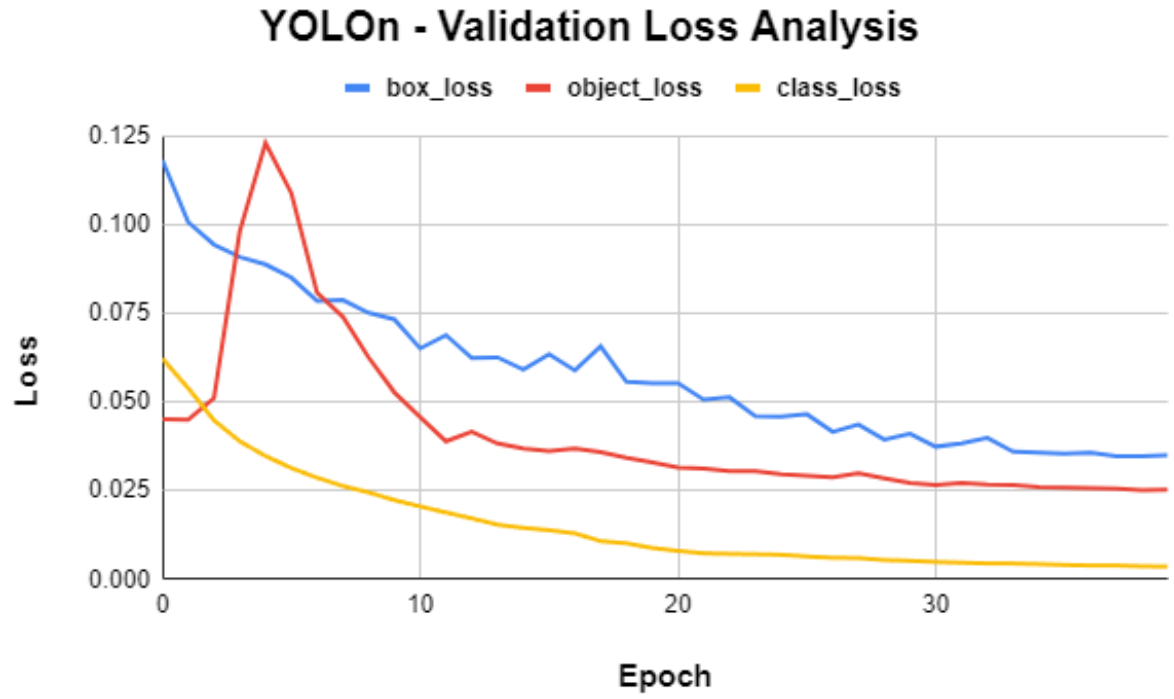

Figure 9. YOLO-v5n validation loss analysis.

4. Discussion

The results section demonstrated the evaluation processes for selecting the YOLO-v5 variant for the given application. After its selection, YOLO-v5n was evaluated on broader metrics and demonstrated its high efficacy both in terms of accuracy and being computationally lightweight. In order to truly manifest the effectiveness of the proposed training algorithm, augmentations, and generalization capacity of the architecture, Table 6 presents a comparison of our work with other works focused on the same domain, i.e., automated pallet racking inspection.

Table 6. Comparison of studies.

	Our Research	**Research by [18]**	**Research by [19]**	**Our Research [21]**
Approach	Object Detection	Segmentation	Object Detection	Object Detection
Dataset Size	2034	75	19,717	2094
Classes	2	1	2	5
Detector	YOLO-v5n	Mask-RCNN	MobileNetV2	YOLOv7
MAP@0.5(IoU)	96.8%	93.45%	92.7%	91.1%

It is evident from Table 6 that the proposed architecture provided the highest performance with respect to MAP@0.5 of 96.8%. In doing so, the proposed approach resulted in better performance than [21], which utilized YOLO-v7 with a higher number of images. The research presented in [19] provided the second-highest performance, based on MobileNet-V2 architecture. However, it required 19,717 images for training, compared to the 2034 images used in our research.

Hence, it can be said with confidence that the proposed augmentations and variant selection algorithm were sufficient for proposing a highly efficient architecture that was able to generalize on the domain dataset whilst generating a lightweight footprint, making it suitable for deployment onto constrained edge devices.

5. Conclusions

In conclusion, this research presented YOLO-v5n as the preferred architecture for automated pallet racking inspection, achieving an MAP@0.5 of 96.8%, higher than all other works presents in this domain, as evident from Table 6. The proposed methodology played a critical role in providing a robust architecture in terms of generalization, accuracy, and lightweight footprint.

Firstly, the high accuracy is a manifestation of the fact that the proposed augmentations were representative of the domain application and contributed towards generating a highly generalized architecture. In order to meet the lightweight requirement necessary for edge deployment, the proposed variant selection algorithm made sure the trade-off between accuracy and computational complexity was efficiently addressed. As a result, YOLO-v5n was selected even though YOLO-v5x provided higher accuracy due to the significant computational load baggage required for the latter.

As a future direction, the authors will look to include a larger variety of faults found within pallet racking, along with segmenting faults into various classes such as vertical damage, support damage, base defects, and horizontal beam damage. The proposed methodology has the potential to be manipulated and integrated into other domains [24] with similar requirements around CNN-based detections requiring lightweight architectures for deployment in constrained environments [25].

Funding: This research received no external funding.

Data Availability Statement: Dataset not yet publicly available.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Aamer, A.M.; Islam, S.S. Distribution center material flow control: A line balancing approach. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *505*, 012078. [CrossRef]
2. Malik, H.; Chaudhary, G.; Srivastava, S. Digital transformation through advances in artificial intelligence and machine learning. *J. Intell. Fuzzy Syst.* **2022**, *42*, 615–622. [CrossRef]
3. Alsboui, T.; Hill, R.; Al-Aqrabi, H.; Farid, H.M.A.; Riaz, M.; Iram, S.; Shakeel, H.M.; Hussain, M. A Dynamic Multi-Mobile Agent Itinerary Planning Approach in Wireless Sensor Networks via Intuitionistic Fuzzy Set. *Sensors* **2022**, *22*, 8037. [CrossRef] [PubMed]
4. Chaouchi, H.; Bourgeau, T. Internet of Things: Building the New Digital Society. *IoT* **2018**, *1*, 1–4. [CrossRef]
5. CEP, F.A. 5 Insightful Statistics Related to Warehouse Safety. Available online: https://www.damotech.com/blog/5-insightful-statistics-related-to-warehouse-safety (accessed on 11 March 2023).
6. Warehouse Racking Impact Monitoring | RackEyeTM from A-SAFE. A-SAFE. Available online: https://www.asafe.com/en-gb/products/rackeye/ (accessed on 12 March 2023).
7. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
8. Ran, H.; Wen, S.; Shi, K.; Huang, T. Stable and compact design of Memristive GoogLeNet Neural Network. *Neurocomputing* **2021**, *441*, 52–63. [CrossRef]
9. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
10. Yang, Z. Classification of picture art style based on VGGNET. *J. Phys. Conf. Ser.* **2021**, *1774*, 012043. [CrossRef]
11. Gajja, M. Brain Tumor Detection Using Mask R-CNN. *J. Adv. Res. Dyn. Control Syst.* **2020**, *12*, 101–108. [CrossRef]
12. Liu, S.; Cui, X.; Li, J.; Yang, H.; Lukač, N. Pedestrian Detection based on Faster R-CNN. *Int. J. Perform. Eng.* **2019**, *15*, 1792. [CrossRef]
13. Dong, C.-Z.; Catbas, F.N. A review of computer vision–based structural health monitoring at local and global levels. *Struct. Health Monit.* **2020**, *20*, 692–743. [CrossRef]
14. Naseer, T.; Burgard, W.; Stachniss, C. Robust Visual Localization Across Seasons. *IEEE Trans. Robot.* **2018**, *34*, 289–302. [CrossRef]
15. Hussain, M.; Al-Aqrabi, H.; Hill, R. PV-CrackNet Architecture for Filter Induced Augmentation and Micro-Cracks Detection within a Photovoltaic Manufacturing Facility. *Energies* **2022**, *15*, 8667. [CrossRef]
16. He, Y.; Song, K.; Meng, Q.; Yan, Y. An End-to-End Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 1493–1504. [CrossRef]
17. Lal, R.; Bolla, B.K.; Sabeesh, E. Efficient Neural Net Approaches in Metal Casting Defect Detection. *Procedia Comput. Sci.* **2022**, *218*, 1958–1967. [CrossRef]
18. Farahnakian, F.; Koivunen, L.; Makila, T.; Heikkonen, J. Towards Autonomous Industrial Warehouse Inspection. In Proceedings of the 2021 26th International Conference on Automation and Computing (ICAC), Portsmouth, UK, 2–4 September 2021. [CrossRef]
19. Hussain, M.; Chen, T.; Hill, R. Moving toward Smart Manufacturing with an Autonomous Pallet Racking Inspection System Based on MobileNetV2. *J. Manuf. Mater. Process.* **2022**, *6*, 75. [CrossRef]
20. Official Raspberry Pi Products | The Pi Hut. Available online: https://thepihut.com/collections/raspberry-pi/products/raspberry-pi-4 (accessed on 1 May 2023).
21. Hussain, M.; Al-Aqrabi, H.; Munawar, M.; Hill, R.; Alsboui, T. Domain Feature Mapping with YOLOv7 for Automated Edge-Based Pallet Racking Inspections. *Sensors* **2022**, *22*, 6927. [CrossRef] [PubMed]
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
23. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
24. Ansari, M.A.; Crampton, A.; Parkinson, S. A Layer-Wise Surface Deformation Defect Detection by Convolutional Neural Networks in Laser Powder-Bed Fusion Images. *Materials* **2022**, *15*, 7166. [CrossRef] [PubMed]
25. Verstraete, D.; Ferrada, A.; Droguett, E.L.; Meruane, V.; Modarres, M. Deep Learning Enabled Fault Diagnosis Using Time-Frequency Image Analysis of Rolling Element Bearings. *Shock Vib.* **2017**, *2017*, 5067651. [CrossRef]