Received 10 May 2023, accepted 4 June 2023, date of publication 7 June 2023, date of current version 16 June 2023.
Digital Object Identifier 10.1109/ACCESS.2023.3283596

RESEARCH ARTICLE

Custom Lightweight Convolutional Neural Network Architecture for Automated Detection of Damaged Pallet Racking in Warehousing & Distribution Centers

MUHAMMAD HUSSAIN AND RICHARD HILL
School of Computing and Engineering, University of Huddersfield, Queensgate Campus, HD1 3DH Huddersfield, U.K.
Corresponding author: Muhammad Hussain (M.Hussain@hud.ac.uk)

ABSTRACT This paper proposes a Convolutional Neural Network–Block Development Mechanism (CNN-BDM) enabling the development of a lightweight deep learning architecture for the detection of damaged pallet-racking, within the manufacturing/warehousing environment. The developed CNN architecture consisted of only 6.5 Million learnable parameters, making it the first custom designed CNN architecture for the pallet racking domain. Architectural training was based on a real dataset collected from various warehouses after implementation of several data modelling strategies for scaling and increasing the variance within the dataset, in a representative manner. Additionally, after achieving a baseline accuracy of greater than 90%, various regularization strategies were applied for further enhancing the performance and generalizability of the network. Dropout at a drop rate of 50% provided the highest performance during training, achieving 99% precision, recall and F1 score. The effectiveness of the proposed methodology was manifested by the fact that the architecture was able to maintain high performance on the test data achieving an overall F1 score of 96%.

INDEX TERMS Architectural complexities, image modelling, damage detection, lightweight footprint, convolutional neural network.

I. INTRODUCTION

Warehouses, distribution and logistics centers are viewed as an integral component within supply chain management (SCM). One of the key functions of these critical entities is for depositing and holding inventory unless and until, shipping is prescribed [1]. The rise of the digital age coupled with recent events such as Covid-19 have shifted consumer behaviors, manifested in the form of increased demand for a variety of products, product information and level of service. As a result, warehousing operations have increased in complexity with the expectation of storing multiple inventories whilst responding effectively to volatile consumer demands [2]. Businesses are striving to meet increased demand and challenges by utilizing technological advancements with the aim to transform their existing industrial locations into smart manufacturing/storage hubs such as the adoption of Artificial Intelligence (AI) [3] and Internet of Things (IoT) [4].

Increased demand within warehousing, implies higher interaction between pallet-racking, forklifts and employees. Each of these entities carries high importance within a warehouse setting. Pallet-racking is used for temporarily holding stock, forklifts are utilized for depositing and picking stock from racking, driven by employees. Hence, a risk arises, that is related to the structural integrity of the pallet racking. The structural integrity of the racking is at risk each time stock is loaded or picked from it. In the case of damage to pallet racking, depending on the severity of the damage or due to damage accumulation over time, racking can collapse, resulting in multiple losses such as loss of lives, financial, operational and reputational. According to [5], 90% of racking failure is a result of an impact administrated by a forklift. The dynamic

The associate editor coordinating the review of this manuscript and approving it for publication was Yongjie Li.

movement coupled with heavy weight of these powered vehicles, means any unwarranted contact with racking can lead to instant damage or damage accumulation over time leading to racking collapse.

To address the issue of racking damage detection and prevention, several methods have been introduced from conventional quality inspection to mechanical-based column protectors. Quality inspection services are provided by companies such as SEMA [6] and DAMOTECH [7], involving human inspection of racking. This however can be a costly and time-consuming option depending on the amount of racking that is to be inspected. Another option is the use of column protectors [8], although this approach can help to dampen the impact, it lacks intelligence to conceive damage and report it. To address the issue of real-time damage detection, A-safe introduced Rackeye [9]. Whilst this sensor-based solution can detect damage in real-time, it's detection range is limited. Also, the sensor must be mounted on the racking legs, hence resulting in significant deployment cost for warehouses with high number of pallet racking.

To address the limitations of the above solutions, deep learning can be utilized in particular computer vision for defect detection [10], [11], [12]. The development of rack damage detection algorithms, coupled with strategic placement of the detection device can lead to an optimized solution in terms of cost, coverage and Realtime detection.

A. LITERATURE

There is a shortage of literature investigating the implementation of Machine Vision for pallet racking inspection. In order to provide a sufficient and systematic review we have widened the scope of the examined literature to include a similar domain known as structural health monitoring (SHM).

Starting with conventional methods, Hong-Hu Zhu et al. [13] examined image processing methodologies for computer vision based structural health monitoring (CV-SHM) applications. Advantages of the evaluated techniques are presented as non-invasive, increased detection distance, electromagnetic inferences and simultaneous classification of multiple targets. With regards to the constraints, authors highlight lack of integration into existing production applications as a major drawback, preventing researchers from validating their architectural performance against ground realities. Whilst approving of the fact, that production-based testing can be difficult due to factors such as resultant down-time, depending on the nature of the business, we address this issue by procuring an real-term dataset from the production floor for our architecture development.

Dong et al. [14] presents and evaluation of CV methodologies for various domains within SHM. The authors segment SHM into one of two categories; local and global. Local category consists of defects such as delamination, loose bolts and crack detection. Whilst the global category compromises of measuring location, structural analysis, vibration and modal identification. Authors agree for CV-SHM to be successful in defect assertion, a large amount of quality data representative of the target application is required. However, the procurement of high quantity, high quality data, that so, from within a production site can be a difficult task. Hence, this paper demonstrates, through our proposed methodology, how a small dataset consisting of less than 100 samples can be domain-modeled via selective augmentations to achieve sufficient samples to train a high performant classifier.

Focusing on the internal architectural infrastructure of computer vision models, it is clear that convolutional neural networks (CNN) is the de facto architecture when dealing with image data. Almost a decade back, Hinton along with his team of researchers presented AlexNet [15], in 2012, along with a graphics processing unit (GPU) for accelerating calculation time. To decrease the time for model convergence, a rectified linear activation (ReLU) function, was proposed. In the subsequent years, a multitude of architectures have been introduced such as GoogleNet [16], VGGNet [17], R-CNN [18], Fast R-CNN [19], Faster R-CNN [20], with the aim to enhance accuracy, decrease inference time and reduce power consumption.

State-of-the-art models such as those mentioned above are deployed across various domains including healthcare, autonomous vehicles and renewable energy [21]. However, due to the high computational demand of these architectures, deployment infrastructure is usually facilitated via a cloud architecture [22]. Although this is suitable for applications where computational requirements is not the top priority, domains such as manufacturing can have stringent data security and power consumption requirements, hence demanding edge device inference close to the data source. The lack of device deployment within production is due to the fact that all algorithms mentioned above demand a significant number of computational resources [22].

Focusing on the limited literature specific to the racking domain, Farahnakian et al. [23], propose a segmentation-based process for automated racking defect detection. The authors propose Mask-RCNN coupled with ResNet-101 backbone for feature extraction. The authors prepare the dataset i.e., ground truth annotations for assisting with the training process. The reported mean average precision based on an intersection-over-union (IoU) of 50% was respectable at 93.45%. looking deeper at the dataset, it was observed that the racking images were taken within a laboratory or other external environments rather than from the production floor. This raises questions on the representativeness of the dataset and the true generalizability of the model. To counter this, our research is based on an actual dataset procured from within manufacturing facilities followed by representative data scaling.

Hussain et al. [24] presented the first edge-based automated pallet racking inspection framework based on MobileNetV2. The authors claimed the architecture to be computationally lightweight and hence deployable onto a constrained edge device such as a raspberry pi [25], achieving a mean average precision of 92.7%. Additionally, the authors claimed the strategic positioning of the edge device i.e., onto the Forklift cage, enabled wider coverage of the racking.

Further focusing on the optimization of automated pallet racking inspection, Hussain et al. [26], proposed a feature mapping mechanism for addressing the issue of data scarcity, during the procurement process of racking images. The proposed mechanism enabled representative data augmentations modelling the production floor variations at various sites. staying with the theme of edge-based deployment, authors subscribed to the object detection domain, selecting Yolov7 as the deployment architecture. The mean average precision for the proposed framework was respectable at 91.1%.

A similar domain to pallet-racking damage detection is steel defect detection. Authors in [27] on Metal casting defects based on the development of custom CNN architectures and comparing the performance against SOTA architectures [ResNet, MobileNet, Inception]. The research takes an interesting approach by borrowing the depth wise convolutional process from MobilNet and implementing it within the custom architectures along with various other optimization strategies including Blurpool, Stochastic Weight averaging, MixUp, Label smoothing and squeeze-excitation. The authors claim their architecture to be superior based on reduced number of parameters however, the accuracy standards at 81.87% whilst Inception achieved 91.48%.

He et al. [28] claim a novel defect detection system, focused on industrial applications using steel surface defect as a use case. Experiments showed that developed architecture achieved 99.67% accuracy for defect classification task and 82.3 mAP for defect detection task. In addition, the system can run at a detection speed of 20 ft/s while keeping the mAP at 70%. The authors explain that due to the localization of faults happening during the end layers of the network, some locational data would be lost from the initial layers. this is due to the fact that shallow features contain rich information but are not discriminative enough, whilst deeper features are semantically robust but more abstract hence detailed information is lost. to address this the authors, propose a Multilevel-feature fusion network (MFN) combining features from all stages into a single rich feature.

Summing the literature review, it is clear that there is a lack of literature focused on automated racking inspection via deep learning. We can say with a high degree of confidence, that our initial work [24] was the icebreaker, presenting a automated racking inspection solution based on the MobileNetV2 architecture along with the first real world racking dataset. This was followed by a YoloV7 implementation [26] focused on real-time performance. However, both solutions are anchored around the same principal, that is object detection which mandates additional data preparation i.e., bounding box generation prior to training. Although this can be ignored when dealing with small datasets but can become an issue when bounding box generation is required for thousands or tens of thousands of images.

Feng et al. [29] look at the automation of Hot rolled steel strip defects due to its impact on various industries including manufacturing. The authors present a RepVGG architecture coupled with the spatial attention mechanism. although on the test data the architecture achieves on average 95.10%, in some defective categories the accuracy is significantly lower reaching (78.95%). It is appreciated that the authors present the computational complexities of their architecture and accept their architecture is relatively large in terms of learnable parameters (83.825Million) and computational complexity (17.892 GMAC's).

Yang et al. [30] propose the implementation of yolov5 for production-based weld steel defect detection based on X-ray images of the weld pipe. The paper is a good example of how single shot detectors can be effectively trained for industrial defect detection compared to the two-stage conventional two-stage detectors such as the once de facto 'Faster-RCNN'. Authors encounter the argument that two-stage detectors by default have better performance in-terms of accuracy by demonstrating how a carefully constructed data augmentation strategy, can lead to the yolov5 performing better in terms of the speed and accuracy when compared to the Faster-RCNN. The authors claim that the trained YoloV5 reached an MAP of 98.7% (IoU-0.5) whilst meeting the real-time detection requirements of steel pipe production with a single image detection rate of 0.12 seconds. However, when looking at the processing setup, the x-ray data is sent to a PC where the processing and inferencing is done via a GPU.

B. CONTRIBUTION AND PAPER ORGANIZATION

Since the utilization of computer vision within the pallet racking domain is at an early stage as evident from the literature, our first contribution is in the form of presenting a development pipeline dubbed as CNN-Block Development Mechanism (CNN-BDM). There were several motives behind the proposal of CNN-BDM. Firstly, it provides a streamlined pipeline enabling researchers within the warehousing domain with limited expertise in CNN development to develop tailored CNN architectures. The operational environment mandates the deployment of the developed architecture to be compatible for implementation on constrained edge devices with limited computational power. This requirement rules out many state-of-the-art (SOTA) architectures due to their architectural depth and computational demand. Hence, CNN-BDM facilitates the development of custom, lightweight CNN architecture by benchmarking it against a SOTA 'look-up table' referencing the number of learnable parameters of several SOTA architectures.

Prior to the development of the CNN architecture, various data modelling strategies are implemented in a streamlined manner, providing justification of operational representativeness as opposed to applying augmentations in an arbitrary fashion. The former may provide high training performance but due to the incompatibility with respect to ground realties may experience detrimental performance post deployment.

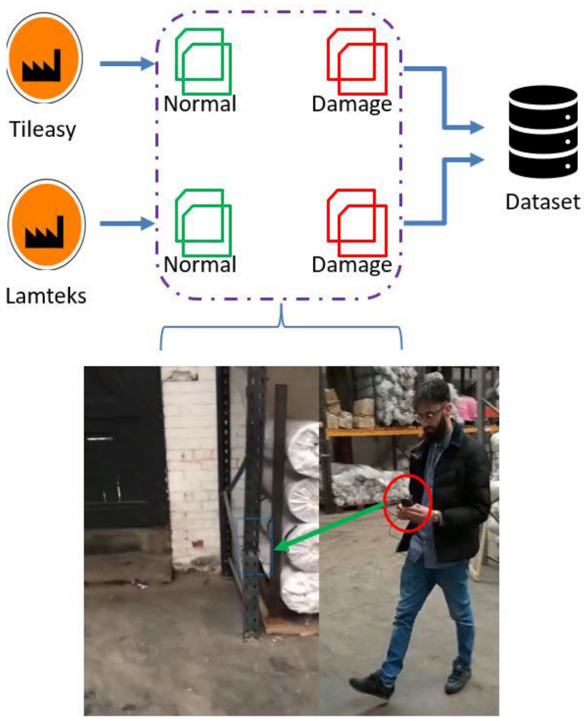


FIGURE 1. Data procurement process.

In addition to CNN-BDM, various regularization strategies are implemented on the constructed architecture aimed at improving the overall generalization of the architecture by reducing the distance between the training and validation accuracies i.e., overfitting suppression.

II. METHODOLOGY

A. DATASET

To the best of our understanding, there is no open-source dataset available for pallet racking. Hence, the initial dataset consisting of less than 100 images of normal and damaged racking was collected from various manufacturing facilities including Tile easy and Lamteks.

The procurement process is presented in Figure 1. An iPhone 8 was used for the capturing of racking images with a 12MP camera. This particular camera specification was selected due to its similarity with a raspberry pi camera with respect to mega-pixels. Another key consideration considered before capturing the images was the operating environment and device placement. The proposed architecture would be deployed onto an edge device which would be strategically placed onto the forklift cage. Hence, to model this scenario, the image would be captured by a person (simulating the forklift), holding out the camera device (simulating the edge device) towards the racking.

Figure 2 presents sample images belonging to both normal and defective classes. For the normal class (Figure 2(A)), the classification can be assumed to be a straightforward process, as the model would need to focus on determining the racking from the background. Whilst with the defective class (Figure 2(B)) it can be observed that the severity of

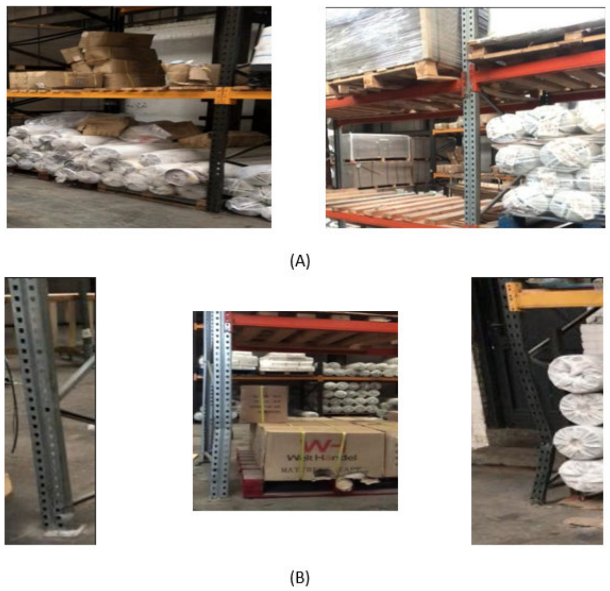


FIGURE 2. Racking data sample internal class inspection (A) normal (B) damaged racking.

damage can vary significantly from case to case. For example, Figure 2 (B) right image, presents major damage easily detected with the human eye. Whilst the centre image in Figure 2(B) is also a manifestation of a damaged racking leg, however this is not easily detectable when compared to the far right and left images. All images contain a background context, this would also need to be considered whilst modelling augmentations, to make sure the environmental context is preserved.

B. DATA MODELING

As mentioned earlier the initial dataset captured directly from within the manufacturing facilities accumulated to less than 100 samples for both classes. This would simply be considered too small for training a CNN architecture to achieve any form of significant generalization. Conventionally, data augmentations are implemented for data scaling, however, applying augmentations without any domain criteria does not guarantee generalization. Hence, our focus was on applying domain specific augmentations, that were representative of the manufacturing floor. The first modelling objective was for the model to be rotationally invariant, as post deployment the captured images for inferencing would be relative to the placement of the edge device. For modelling this variance angle-based rotations were introduced via,

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

A shift component was introduced (f_x, f_y) for pixel level shifting, representative of displacement due to physical contact with loaded pallets,

$$H = \begin{bmatrix} 1 & 0 & f_x \\ 0 & 1 & f_y \end{bmatrix} \tag{2}$$

TABLE 1. Augmented dataset post splitting.

	Augmented Dataset Post Splitting		
Class	Training	Validation	Test
Normal	562	164	80
Damage	633	188	96

The transformation was implemented post translation on matrix H into an array, where z_x = input image, z_y = resultant image and H = translation matrix,

$$z_y\,(x,y) = z_x(H_{11}x + H_{12}x + H_{13}, H_{21}x + H_{22}y + H_{23} \tag{3}$$

The second criteria was based on modelling the light intensity variations found within production floor facilities. This was done via the introduction of pixel darkness/brightness adjustments, for greater generalization when faced with dissimilar LUX intensities. LUX modelling was particularly important for wider deployment prospects, as manufacturing facilities LUX intensities can vary depending on various factors such as location, regulations and the nature of the business.

Global image blurring at various ratios was also introduced for modelling variance caused by differing hardware specifications in particular camera quality as presented in Table 1. Pixel-noise was introduced to cater for extreme cases, where the nature of the business meant increased blurring due to external factors such as dust particles. Table 1 presents the scaled dataset post data augmentations and train, validation and test splits.

C. PROPOSED CNN ARCHITECTURE

The rationale for deciding to create a custom architecture as opposed to subscribing to the transfer learning approach i.e., riding on the shoulder of SOTA architectures was due to the domain requirement for a lightweight architecture that could be deployed onto constrained edge devices, demanding less power consumption, making battery power a feasible option.

Hence a ‘bottom-up’ approach was taken for the CNN development starting with a single convolutional block containing 5 filters, followed by a single fully connected layer containing 25 nodes. Max-pooling was introduced for the elimination of positional dependencies post convolutional process. Prior to pooling, ReLu was applied as the activation function converting the weighted input sum to the non-linear output representation. Although various activations functions exist such as Sigmoid and TanH, ReLu was selected due to its simplistic computational nature, mathematically represented as,

$$f(x) = \mathrm{Max}(0, x) \tag{4}$$

where x= input, when x is a negative/zero value it is represented as a zero. When x is greater than zero, the value (v) for x is retrieved,

$$f'(x) = \begin{cases} v, & \mathit{for}\ x \geq 0 \\ 0, & \mathit{for}\ x < 0 \end{cases} \tag{5}$$

During the training process a loss function was required for calculating the error between the actual (y) and predicted labels (P(y)). Due to binary output nature of the task i.e., normal or damaged, binary cross entropy was implemented, mathematically represented as,

$$K_z = [ylogP\,(y) + (1-y)\log(1-p\,(y))] \tag{6}$$

Stochastic gradient descent with momentum (SGD-M) was selected as the optimizer for facilitating the backpropagation process during the training phase. Plain SGD takes the derivative of the weights (dW) and the bias (dB) for every training epoch,

$$\mathrm{W} = \mathrm{W} - \eta \times dW, \quad B = B - \eta \times dB \tag{7}$$

Additionally, SGD-M introduces momentum (M) as the moving mean for the gradients. The moving mean between 0 and 1, when computing dW and dB on the current batch is represented as,

$$\begin{aligned} M_{dW} &= \beta \times M_{dW} + (1-\beta)dW, \\ M_{dB} &= \beta \times M_{dB} + (1-\beta)dB \end{aligned} \tag{8}$$

where β was utilized as a hyperparameter for controlling the exponentially weighted means.

The objective was to design a high performant CNN by iteratively increasing the number of convolutional blocks and the internal filters within each block, whilst leaving the supporting infrastructure untouched i.e., activation function, pooling. After each design iteration, the resultant CNN architecture was compared against a SOTA lookup table, presented in Table 2 containing the learnable parameters of SOTA CNN architectures for image classification such as ResNet. If the number of parameters of the developed CNN were greater than those within the lookup table, the internal filters were sent back to the design stage for filter refinement. In the case, the parameters of the resultant architecture were less than those within the lookup table, the architecture was forwarded for training.

Post training, the validation accuracy was evaluated against a checkpoint requiring 90% accuracy. In the case of achieving this, the architecture was forwarded to the regularization stage for reducing overfitting and enhancing the performance. Conversely, if the architecture failed to achieve the 90% benchmark on the validation accuracy, the architecture was referred back to the design stage for internal block development, hence restarting the process presented in the figure. An arbitrary value of 5 filters was selected for the initial CNN design containing a single convolutional block.

D. ARCHITECTURAL REGULARIZATION

Regularization techniques were introduced into the selected architecture post the CNN design stage. The aim was to enhance the performance of the architecture whilst suppressing overfitting.

Batch normalization was implemented with the aim to address internal covariance in particular for the defective

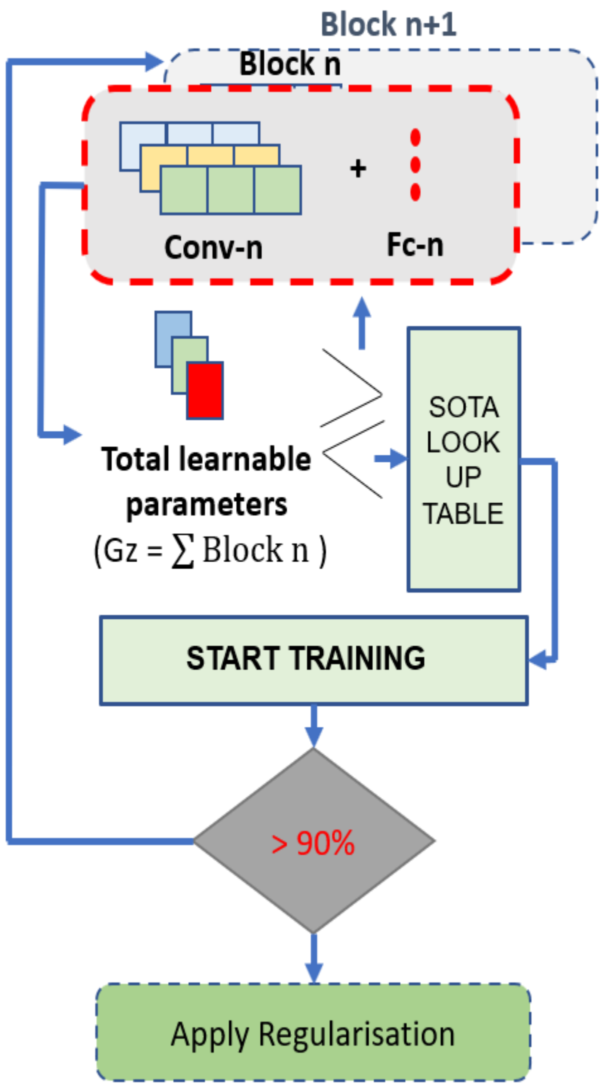


FIGURE 3. CNN-block development mechanism (CNN-BDM).

TABLE 2. SOTA look-up table.

Architecture	Parameters (M)
VGG-16	134.70
VGG-19	143.67
AlexNet	61.0
GoogleNet	13.0
ResNet-18	11.69
ResNet-34	21.50
ResNet-50	23.90
ResNet-101	42.8
ResNet-152	58.5

class, as evident from the data inspection, the location and severity of the damage can vary significantly between images. The concept is mathematically presented in [equation number], where o represents neuron outputs, o^n equals normalized output for neuron, m_o equates to the mean of the neurons and s_o refers to the standard deviation of the neurons output.

$$o^n = (\frac{o - \mathrm{m}_o}{\mathrm{s}_o}) \tag{9}$$

Devoid of batch normalization, input layer, $a^{[l-1]}$ passes through a reserved transform prior to passing through an activation function ($g^{[l]}$), the activation function ($a^{[l]}$) is then presented as,

$$a^{[l]} = g^{[l]}(w^{[l]}a^{[l-1]} + b^{[l]} \tag{10}$$

Post batch normalization via a transform (BN), the output is presented as (14),

$$a^{[l]} = g^{[l]}(BN\left(w^{[l]}a^{[l-1]}\right)) \tag{11}$$

Noting the fact batch normalization brings two additional parameters, β & γ, for each unit. In the case of $\beta = \mu$ and $\gamma = \sqrt{\gamma}^2 + \varepsilon$, then $o^n = $ o, i.e., an identity function.

Another regularization technique implemented for reducing overfitting was dropout. Dropout is similar to the concept of bagging, utilized widely in Machine learning algorithms, where several models are trained on different subsets of the same training data. Dropout was implemented at within all internal blocks of the CNN, barring the input block, with the aim to reduce co-adaptation.

Co-adaptation leads to bestowing increased predictive capabilities to certain neurons which could lead to reduced performance in the case of false generalization. Traditional regularization strategies such as L1, L2 would not suffice as they are based upon predictive capabilities of different neurons. Implementation of dropout is presented in the equations below based on the dropout probability p of 50%, where z^l is the input vector, y^l is representative of the output for a hidden layer, i is the hidden neuron with $w_i^{(l)}$ as weight and $b_i^{(l)}$ as bias, f represents the applied activation function. A vector $r^{(l)}$ is applied to $y^{(l)}$ with the $r^{(l)}$ vector elements as 1 with probability of p and 0 with probability of $1 - p$.

$$r_j^{(l)} \sim Bernoulli(p) \tag{12}$$
$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)} \tag{13}$$
$$z_i^{(l+1)} = w_i^{(l+1)}\tilde{y}^{(l)} + b_i^{(l+1)} \tag{14}$$
$$y_i^{(l+1)} = f(z_i^{(l+1)}) \tag{15}$$

E. INTERNAL BLOCK-WISE COMPUTATION

The internal block-wise computational payload for the proposed CNN is presented in Table 3. The advent of batch normalization within the convolutional blocks, resulted in an increase in the computational parameters, however, this increase was not significantly high. Sigmoid was implemented as the output layer activation function. Comparing the computational payload of the proposed CNN to the SOTA look-up table presented in Table 2, it is evident that the proposed CNN was significantly more lightweight with a margin of 5.19 Million parameters compared to ResNet-18.

III. RESULTS

A. HYPER-PARAMETERS

To evaluate the performance of the proposed CNN architecture along with the impact of various regularization

TABLE 3. Block-wise internal computation.

Layer	Output Shape	Learnable Parameters
Input	3, 224×224	-----
Conv-1	5, 222×222	140
Batch norm	5, 222×222	10
ReLu	5, 222×222	-----
Max-pool	5, 111×111	-----
Conv-2	25, 109×109	1,150
Batch norm	5, 109×109	50
ReLu	5, 109×109	-----
Max-pool	5, 54×54	-----
Fc-1	90 neurons	6,561,090
ReLu	-----	-----
Fc-2	45 neurons	4,095
ReLu	-----	-----
Output	2 neurons	92
Total Learnable Parameters		6.5 Million

TABLE 4. Hyperparameters.

Hyperparameters	
Batch Size	32
Epochs	40
Learning Rate	0.02
Loss Type	Cross Entropy
Optimizer Type	SGD-M

strategies presented in the previous section, global training parameters were defined, enforcing performance evaluation integrity. The globally defined hyperparameters are presented in Table 4. Google Collaboratory was selected as the platform for facilitating all training and evaluation due to its free GPU allowance. However, GPU allowance was limited, hence the training epochs were capped at 40.

B. INITIAL ARCHITECTURE

The initial architecture as per the proposed CNN-BDM consisted of a single convolutional block and a fully connected layer. Due to the singular network structure, the architecture passed the parameter checkpoint with respect to the SOTA look-up table, hence the architecture was trained on the racking dataset based on the hyperparameters defined within Table 4.

Figure 4 presents the training and validation accuracies of the initial architecture. It is clear from Figure 4; the architecture lacked the basic capacity for providing the 90% accuracy benchmark set in CNN-BDM. It may be argued, the poor performance was due to the small number of epochs, however, viewing both training and validation accuracies it is evident that a plateau has been reached as early as the 15th epoch, hence further increasing of the training epochs would not improve the performance.

C. MODIFIED (DEPTH INCREASE) ARCHITECTURE

The poor performance of the initial architecture was attributed to the lack of capacity within the internal architectural blocks. Hence the next iteration involved the introduction of an additional convolutional block, with the number of filters equal to the square of filters within the initial block i.e., 25. Additionally, and additional fully connected layer was introduced containing twice the number of neurons with respect to the initial layer, as per the proposed CNN-BDM. The motive behind the depth increase was to achieve the required capacity for generalization, however and implication of this was increased computational parameters. Hence, the iterated architecture was compared against the SOTA look-up table to confirm the number of parameters were less than referenced SOTA architectures. The architecture was well within the computational parameters remit, with only 6.5 Million parameters compared to ResNet-18 at 11.69 Million. Thus, the modified architecture was forwarded for training based on hyperparameters defined in Table 4. The performance of the architecture experienced significant improvement with the training accuracy optimal result and the validation accuracy at 98.9%. Table 5 presents a granular inspection of the CNN based on three metrics; precision, recall and F1 score, endorsing the effective generalization of the architecture with an overall F1 score of 98.8%.

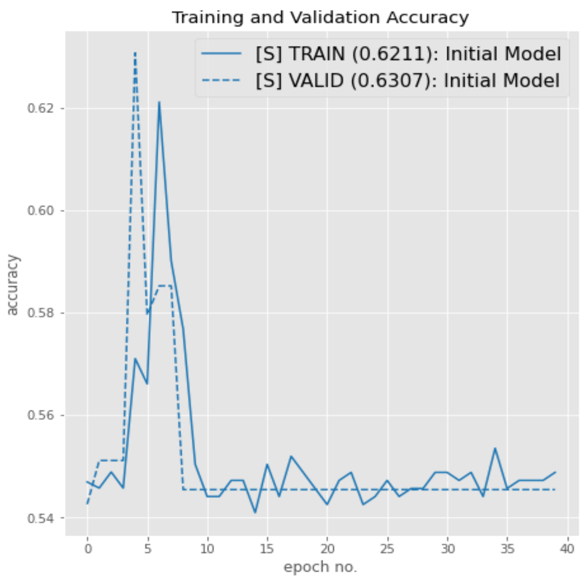


FIGURE 4. Accuracy graph for developed CNN classifier.

TABLE 5. Performance evaluation modified architecture.

Performance Evaluation Modified Architecture	
Precision	99.0%
Recall	97.0%
F1 score	98.0%

Figure 5 further compliments the results in Table 5, via the confusion matrix. Class breakdown shows 5 out of 188 samples were incorrectly classified as normal, whilst 1 out of 164 samples were incorrectly classified as defective.

D. INTERNAL BATCH NORMALIZATION

The performance of the modified architecture presented in Table 5 was impressive. The application of batch normalization was to investigate if the performance can be further

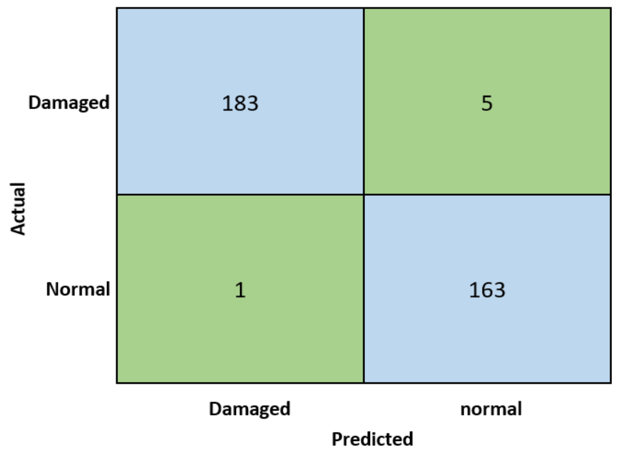


FIGURE 5. Confusion matrix for developed CNN classifier.

TABLE 6. Performance evaluation with batch normalization.

Performance Evaluation With Batch Normalisation	
Precision	99.0%
Recall	96.0%
F1 score	98.0%

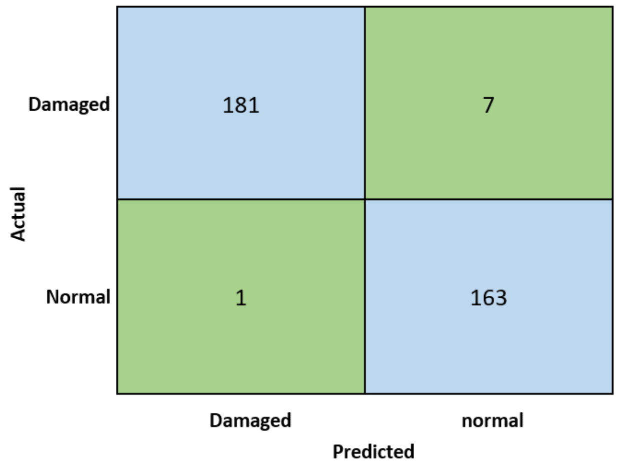


FIGURE 6. Confusion matrix for batch normalised architecture.

enhanced i.e., improving the recall rate. Additionally, the aim of introducing batch normalization was the removal of any co-adaptation which may have occurred within the internal blocks during training. Hence batch normalization was introduced into both convolutional blocks prior to the execution of the activation function, ReLu. Post training, it is clear from Table 6, the application of batch normalization did not improve the overall performance, in fact the recall experience a small drop in performance.

The degradation in performance can also be seen from the confusion matrix, Figure 6, with the number of misclassifications for the damaged class increasing to 7.

E. APPLYING INTERNAL DROPOUT

The next regularization strategy aimed at enhancing the generalization of the proposed CNN on the racking dataset was dropout. Dropout was implemented with a dropout

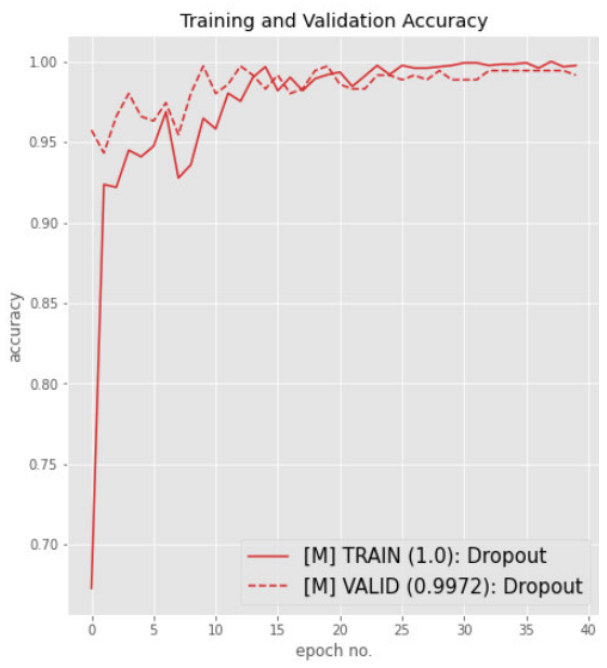


FIGURE 7. Accuracy graph post dropout.

TABLE 7. Performance evaluation with dropout @50%.

Performance Evaluation With Dropout @ 50%	
Precision	99.0%
Recall	99.0%
F1 score	99.0%

probability rate of 50%. Looking at the training and validation accuracies, presented in Figure 7, it can be seen that the distance between the two respective accuracies had decreased, indicating towards further suppression of overfitting as compared to Figure 4. This was due to the disabling of random neurons, discouraging increased weightage for certain connections, leading to improved generalization.

Table 7, demonstrates the improved performance of the proposed architecture post applying of dropout. It can be observed that the recall rate improved by 3% compared to batch normalization, reaching 99%, with an overall F1 score of 99%.

Figure 8, presents the confusion matrix manifesting the improved class-wise performance with only a single damaged racking being misclassified, whilst 2 instances of normal racking were misclassified as damaged.

F. REGULARISATION COUPLING

The proposed architecture provided impressive performance in all three experiments presented above. An outstanding experiment based on intuition was the merging of both batch normalization and dropout, implemented into the proposed architecture with the aim to observe the impact of the overall performance.

Before applying this, dropout was experimented for the second time with reduced dropout rate from 50% to 25%.

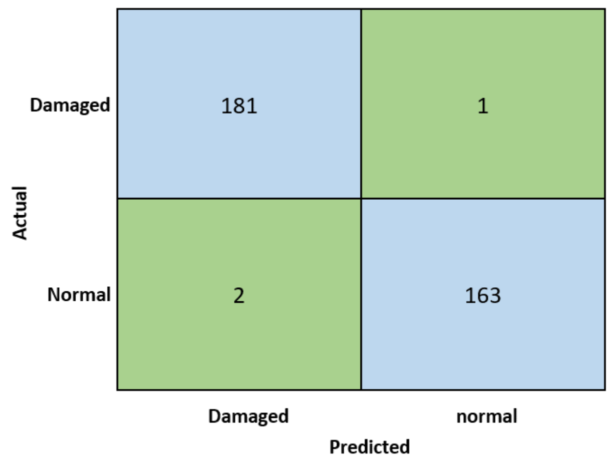


FIGURE 8. Confusion matrix for dropout.

TABLE 8. Network performance comparison modified architecture.

Precision	99.0%
Recall	97.0%
F1 score	98.0%
With Batch Normalisation	
Precision	99.0%
Recall	96.0%
F1 Score	98.0%
With Dropout @ 50%	
Precision	99.0%
Recall	99.0%
F1 Score	99.0%
With Dropout @ 25%	
Precision	99.0%
Recall	98.0%
F1 Score	99.0%
Batch Normalisation with Dropout @ 50%	
Precision	99.0%
Recall	97.0%
F1 Score	98.0%

Table 8 demonstrates, although decreasing the dropout rate to 25%, provided high performance there was a degradation in the recall rate compared to 50% dropout rate of 1%.

Hence the coupling of batch normalization was implemented with dropout at a drop rate of 50%. Interesting this further diminished the recall rate to 97%. This showed that although the coupling of the two strategies (batch normalization and dropout) provided higher performance than batch normalization by itself, the performance was not optima. This could be attributed towards the fact that strictly speaking batch normalization is a regularization strategy focused on removal of internal covariance, hence is more inclined towards providing faster convergence, whilst dropout, directly interferes with the neurons, to enforce generalization.

Figure 9 presents the loss and accuracy graphs for dropout @50% and coupling of batch normalization with dropout. From the accuracy graph (bottom) it can be seen that coupling resulted in more instability during early stages of training as well as a difference of 1.09% between the training and validation accuracies. More clearly, it can be observed from the loss graph (top), validation loss starts to increase after only 15 epochs.

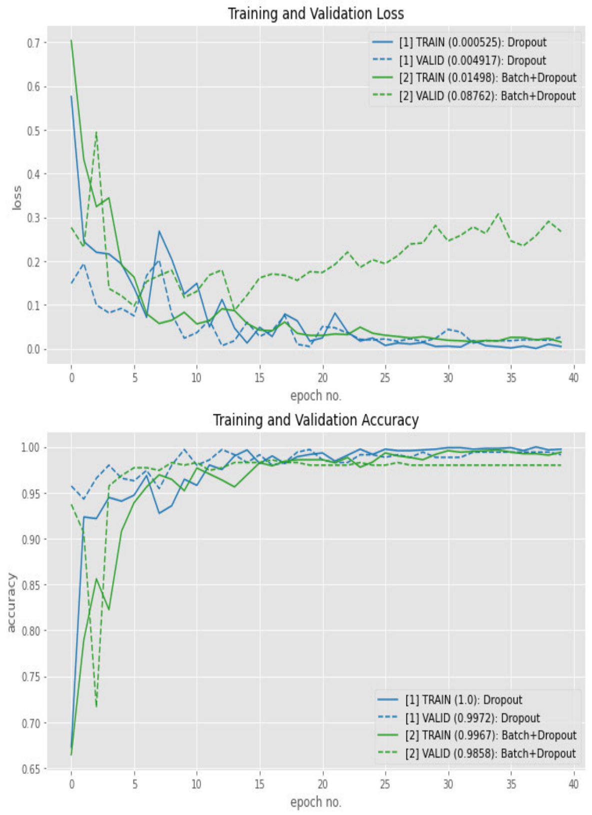


FIGURE 9. Coupling graph comparison.

TABLE 9. Performance evaluation test data.

Performance Evaluation Test Data	
Precision	96.0%
Recall	97.0%
F1 score	96.0%

G. TEST DATA EVALUATION

Upon completion of the development, regularization and training process, the final phase involved testing of the proposed architecture on the test dataset presented in Table 1. For this, the training and validation datasets used in the previous sections for model training were merged into a single training dataset and performance evaluation of the trained architecture was done on the test data.

Table 9 presents the performance of the proposed architecture on the test dataset. Although, a slight drop in the overall F1-score is seen (3%), the overall performance was impressive at 96%, endorsing the high efficacy of the proposed CNN-BDM in effectively designing a custom lightweight architecture for the racking domain.

The results presented in Table 9 are complimented via the class-wise confusion matrix presented in Figure 10. As per the confusion matrix, 3 out of 96 samples belonging to the

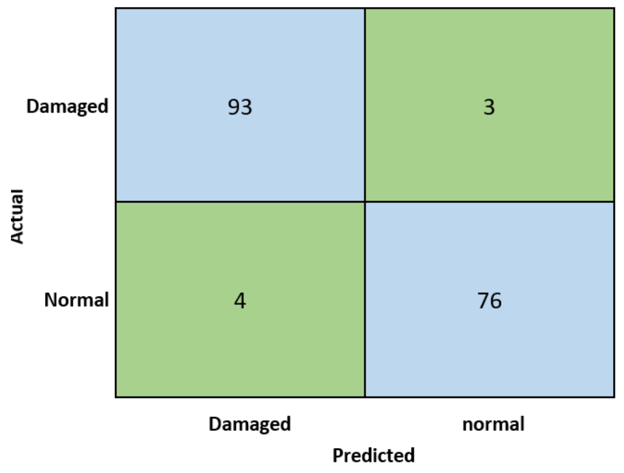


FIGURE 10. Confusion matrix for test data.

TABLE 10. Racking domain research comparison.

Racking Domain Research Comparison				
Research	Domain	Dataset size	Detector	Accuracy
23	Segmentation	75	Mask RCNN	93.45%
24	Object detection	19,717	Mobile Net V2	92.7%
26	Object detection	2094	YoloV7	91.1%
Proposed	Image classification	1723	Custom	96%

damaged class were misclassified as normal. Whilst 4 out of 80 normal instances were misclassified as damaged.

H. SOLUTION COMPARISON

Table 10 presents a comparison of the proposed research against the three distinct research out available on automated racking inspection. Starting with [23], although authors demonstrate high performance (93.45%) based on a small dataset, the selected architecture is not compatible with edge deployment constraints. That is, Mask-RCNN is a computationally demanding architecture based on ResNet-101 as its backbone, containing 44.5 Million learnable parameters, compared to 6.5 Million for the proposed architecture. Prior research done by [24], [25] also provides respectable performance, however, in addition to requiring large amounts of training data, both are based on the object detection domain. Although, this is a computationally more lightweight option as compared to segmentation, it does require additional data preparation, i.e., the labelling of the ground truths (bounding boxes), which may be affected by human bias. Alternatively, our approach is attributed to the image classification domain, hence eliminating the requirement of bounding box determination, providing a more robust processing pipeline. The effectiveness of the proposed methodology is evident from Table 10, achieving the highest accuracy at 96%.

IV. DISCUSSION

The effectiveness of the proposed methodology has been presented in an iterative manner, finally reaching an F1 score of 96% on the test dataset. The inception of the research was focused on comprehension and modelling of the initial normal and damaged racking images. The motive behind this was to generate a more representative dataset, containing the key characteristics founding within different warehouse settings with respect to pallet racking.

The proposed CNN-BDM presented a template for developing a custom CNN architecture via two checkpoints. The first checkpoint required benchmarking the parameters of the designed architecture against the SOTA look-up table, to justify the creation of the architecture. The next checkpoint was based on training the developed architecture on the racking dataset, on globally defined hyperparameters. As per the second checkpoint, the architecture required a validation accuracy of 90% or greater. As demonstrated by the initial architecture achieving a validation accuracy of 63.07%, passing checkpoint one did not guarantee high accuracy as the architecture may still lack the baseline capacity.

The second iteration of CNN-BDM provided impressive performance achieving an F1 score of 98%. Hence, various regularization techniques were applied for further enhancement, with dropout at drop rate of 50% providing the highest performance of 99% for precision, recall and F1 score. The proposed architecture was finally evaluated on the test dataset, although a slight decrease in F1 score was seen, the performance was nevertheless impressive at F1 score of 96%, validating the proposed methodology.

V. CONCLUSION

In conclusion, the proposed CNN-BDM was successful in developing a custom CNN architecture, containing only 6.5 Million learnable parameters, for the detection of defective pallet racking with an overall F1 score of 96%.

To the best of our understanding, this is the first work focused on the development of a lightweight custom CNN architecture aimed at the racking industry. As racking defect detection via deep learning is a new research field, the proposed CNN-BDM can play a critical role in propelling research in the development of lightweight CNN architectures to address the warehousing environmental and operational constraints such as limited computational power.

The proposed CNN-BDM provides researchers with an alternative to transfer learning. Transfer learning can be a lucrative option for many researchers with limited expertise in CNN internal architecture development, however, transfer leaning is usually based on SOTA architectures that can be computationally very demanding, hence making the less feasibly for deployment in constrained environments.

As an extension of this research, staying in line with the lightweight industrial requirements for edge deployment, additional strategies can be integrated into the CNN-BDM for further compression of the learnable parameters without significantly impacting the performance. Some of the compression techniques that can be experimented include model quantization, compression and pruning applicable to not only pallet racking but also other domains requiring computationally lightweight architectures. With respect to the pallet racking domain, future research will look to include more fault types including racking base plate damage along with horizontal supports with the aim to provide an inclusive racking defect detection architecture.

REFERENCES

[1] A. M. Aamer and S. S. Islam, ''Distribution center material flow control: A line balancing approach,'' *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 505, no. 1, May 2019, Art. no. 012078, doi: 10.1088/1757-899X/505/1/012078.

[2] C. J. Langley, R. A. Novack, B. Gibson, and J. J. Coyle, *Supply Chain Management: A Logistics Perspective*, 11th ed. Boston, MA, USA: Cengage Learning, 2020.

[3] H. Malik, Gopal, and S. Srivastava, ''Digital transformation through advances in artificial intelligence and machine learning,'' *J. Intell. Fuzzy Syst.*, vol. 42, pp. 1–8, Feb. 2021, doi: 10.3233/jifs-189787.

[4] H. Chaouchi and T. Bourgeau, ''Internet of Things: Building the new digital society,'' *IoT*, vol. 1, no. 1, pp. 1–4, Jun. 2018, doi: 10.3390/iot1010001.

[5] FA CEP. *5 Insightful Statistics Related to Warehouse Safety*. [Online]. Available: https://www.damotech.com and https://www.damotech.com/blog/5-insightful-statistics-related-to-warehouse-safety

[6] Storage Equipment Manufacturer's Association. *Storage Equipment Manufacturers Association | SEMA*. Accessed: Jan. 5, 2023. [Online]. Available: https://www.sema.org.uk/

[7] Damotech. *Damotech | Pallet Racking Inspection, Protection & Repair Solutions*. Accessed: Jan. 5, 2023. [Online]. Available: https://www.damotech.com and https://www.damotech.com/

[8] The Rack Group. *Rack Armour*. [Online]. Available: https://therackgroup.com/product/rack-armour/

[9] A-SAFE. *Warehouse Racking Impact Monitoring | RackEyeTM from A-SAFE*. [Online]. Available: https://www.asafe.com/en-gb/products/rackeye/

[10] M. Hussain, H. Al-Aqrabi, M. Munawar, R. Hill, and S. Parkinson, ''Exudate regeneration for automated exudate detection in retinal fundus images,'' *IEEE Access*, early access, Sep. 12, 2022, doi: 10.1109/access.2022.3205738.

[11] I. Konovalenko, P. Maruschak, J. Brezinová, O. Prentkovskis, and J. Brezina, ''Research of U-Net-based CNN architectures for metal surface defect detection,'' *Machines*, vol. 10, no. 5, p. 327, Apr. 2022, doi: 10.3390/machines10050327.

[12] F. Gao, T. Liu, G. Shao, Y. Su, J. Yang, and J. Ruan, ''A fast surface-defect detection method based on Dense-Yolo network,'' *SSRN Electron. J.*, 2022, doi: 10.2139/ssrn.4224315.

[13] H.-H. Zhu, F. Dai, Z. Zhu, T. Guo, and X.-W. Ye, ''Smart sensing technologies and their applications in civil infrastructures 2016,'' *J. Sensors*, vol. 2016, pp. 1–2, Jan. 2016, doi: 10.1155/2016/8352895.

[14] C.-Z. Dong and F. N. Catbas, ''A review of computer vision–based structural health monitoring at local and global levels,'' *Struct. Health Monitor.*, vol. 20, no. 2, pp. 692–743, Mar. 2021, doi: 10.1177/1475921720935585.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[16] H. Ran, S. Wen, K. Shi, and T. Huang, ''Stable and compact design of memristive GoogLeNet neural network,'' *Neurocomputing*, vol. 441, pp. 52–63, Jun. 2021, doi: 10.1016/j.neucom.2021.01.122.

[17] Z. Yang, ''Classification of picture art style based on VGGNET,'' *J. Phys., Conf. Ser.*, vol. 1774, no. 1, 2021, Art. no. 012043, doi: 10.1088/1742-6596/1774/1/012043.

[18] M. Gajja, ''Brain tumor detection using mask R-CNN,'' *J. Adv. Res. Dyn. Control Syst.*, vol. 12, no. SP8, pp. 101–108, Jul. 2020, doi: 10.5373/jardcs/v12sp8/20202506.

[19] S. Liu, ''Pedestrian detection based on faster R-CNN,'' *Int. J. Performability Eng.*, vol. 15, p. 1792, 2019, doi: 10.23940/ijpe.19.07.p5.17921801.

[20] L. Fu, Y. Majeed, X. Zhang, M. Karkee, and Q. Zhang, ''Faster R–CNN–based apple detection in dense-foliage fruiting-wall trees using RGB and depth features for robotic harvesting,'' *Biosyst. Eng.*, vol. 197, pp. 245–256, Sep. 2020, doi: 10.1016/j.biosystemseng.2020.07.007.

[21] M. Hussain, H. Al-Aqrabi, and R. Hill, ''PV-CrackNet architecture for filter induced augmentation and micro-cracks detection within a photovoltaic manufacturing facility,'' *Energies*, vol. 15, no. 22, p. 8667, Nov. 2022, doi: 10.3390/en15228667.

[22] D. Neupane, Y. Kim, J. Seok, and J. Hong, ''CNN-based fault detection for smart manufacturing,'' *Appl. Sci.*, vol. 11, no. 24, p. 11732, Dec. 2021, doi: 10.3390/app112411732.

[23] F. Farahnakian, L. Koivunen, T. Mäkilä, and J. Heikkonen, ''Towards autonomous industrial warehouse inspection,'' in *Proc. 26th Int. Conf. Autom. Comput. (ICAC)*, Portsmouth, U.K., Sep. 2021, pp. 1–6, doi: 10.23919/icac50006.2021.9594180.

[24] M. Hussain, T. Chen, and R. Hill, ''Moving toward smart manufacturing with an autonomous pallet racking inspection system based on MobileNetV2,'' *J. Manuf. Mater. Process.*, vol. 6, no. 4, p. 75, Jul. 2022, doi: 10.3390/jmmp6040075.

[25] Thepihut.com. *Official Raspberry Pi Products | The Pi Hut*. Accessed: Jan. 5, 2023. [Online]. Available: https://thepihut.com/collections/raspberry-pi/products/raspberry-pi-4

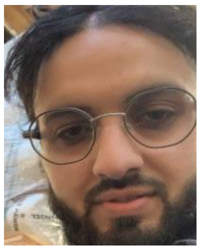
[26] M. Hussain, H. Al-Aqrabi, M. Munawar, R. Hill, and T. Alsboui, ''Domain feature mapping with YOLOv7 for automated edge-based pallet racking inspections,'' *Sensors*, vol. 22, no. 18, p. 6927, Sep. 2022, doi: 10.3390/s22186927.

[27] R. Lal, B. Bolla, and S. Ethiraj, ''Efficient neural net approaches in metal casting defect detection,'' 2022, *arXiv:2208.04150*, doi: 10.48550/arXiv.2208.04150.

[28] Y. He, K. Song, Q. Meng, and Y. Yan, ''An end-to-end steel surface defect detection approach via fusing multiple hierarchical features,'' *IEEE Trans. Instrum. Meas.*, vol. 69, no. 4, pp. 1493–1504, Apr. 2020, doi: 10.1109/TIM.2019.2915404.

[29] X. Feng, X. Gao, and L. Luo, ''X-SDD: A new benchmark for hot rolled steel strip surface defects detection,'' *Symmetry*, vol. 13, no. 4, p. 706, Apr. 2021, doi: 10.3390/sym13040706.

[30] D. Yang, Y. Cui, Z. Yu, and H. Yuan, ''Deep learning based steel pipe weld defect detection,'' *Appl. Artif. Intell.*, pp. 1–13, Sep. 2021, doi: 10.1080/08839514.2021.1975391.



MUHAMMAD HUSSAIN was born in Dewsbury, West Yorkshire, U.K., in 1995. He received the B.Eng. degree in electrical and electronic engineering and the M.S. degree in the Internet of Things from the University of Huddersfield, Charlottesville, in 2019, where he is currently pursuing the Ph.D. degree in artificial intelligence for defect identification. His research interest includes the detection of various faults in particular micro-cracks forming on the surface of photovoltaic (PV) cells because of mechanical and thermal stress. He has a particular interest in the field of machine vision, focusing on the development of light-weight architectures that can be optimized for deployment on edge devices and ultimately on the production floor. He is also researching into design-level architectural interpretability, with a focus on explainable AI for sensitive fields, such as medicine and healthcare.

RICHARD HILL is currently the Head of the Department of Computer Science and the Director of the Centre for Industrial Analytics, University of Huddersfield, U.K. He has published more than 200 peer-reviewed articles. His research interest includes digital manufacturing. He has been a recipient of several best paper awards, having been recognized by the IEEE for outstanding research leadership in the areas of big data, predictive analytics, the Internet of Things, cyber-physical systems security, and Industry 4.0.