Sigma J Eng Nat Sci, Vol. 44, No. 3, pp. 2207–2218, June, 2026

Sigma Journal of Engineering and Natural Sciences
Web page info: https://sigma.yildiz.edu.tr
DOI: 10.14744/sigma.2026.2090

Research Article

Optimized convolutional neural networks for defect classification in fiber-reinforced composites

Pankaj BELDAR[1,*]

[1]*Department of Mechanical Engineering, K.K.Wagh Institute of Engineering Education and Research, Savitribai Phule Pune University, 665121, Maharashtra, India*

ARTICLE INFO

Article history
Received: 28 September 2024
Revised: 08 December 2024
Accepted: 08 January 2025

Keywords:
Composites; Defects; Machine Learning; Microstructure

ABSTRACT

The FRC materials have tremendous strength/weight ratio but are vulnerable to pull out, fracture, de-bond of fibre, fibre-matrix, and micro cracks thus, compromising tremendously with mechanical performance in different industries. It is important in detecting and classifying these defects so that the reliability and durability of these materials are guaranteed. This paper provides an automated recognition scheme based on convolutional neural networks to detect these five faults of a scanning electron microscopy image. Two hyper parameter tuners Keras Tuner and Particle Swarm Optimization were used to optimize the convolutional neural networks. The performance of Keras Tuner was 96.88 percent whereas Particle Swarm Optimization had a better result with an accuracy of 99.23 percent and a validation loss of 0.02 as compared to 0.1119 in Keras. Also, the model with Particle Swarm Optimization had greater precision (99.8 per cent), recall (99.75 per cent), and F1-score (99.77 per cent), and inference times of 25 milliseconds per batch were lower than the ones of Keras Tuner (35 milliseconds per batch). These findings demonstrate the usefulness of Particle Swarm Optimization in obtaining a greater classification accuracy. The innovation of the given research is that the developed technique integrates both modern optimization methods and convolutional neural networks to classify defects, and has surpassed the current approaches reaching an accuracy of nearly 100 percent with a much lower main complexity of computation. The results open the way to strong automated defect identification in fibre-reinforced composites, which would lead to the higher quality of materials and a guarantee of the materials performance in field.

Cite this article as: Beldar P. Optimized convolutional neural networks for defect classification in fiber-reinforced composites. Sigma J Eng Nat Sci 2026;44(3):2207–2218.

INTRODUCTION

Composites made of fibre are crucial in inspection of industries such as the aerospace and automobile industries due to their amazing strength-mass relationship. Nonetheless, flaws such as fibre-matrix de-bonding, fibre pull-outs, fibre fracture, fibre voids, and micro-cracks can

***Corresponding author.**
*E-mail address: prbeldar@kkwagh.edu.in

This paper was recommended for publication in revised form by Editor-in-Chief Ahmet Selim Dalkilic

have a notable negative effect on performance. The conventional ways of defect detection are manual, time consuming and inaccurate. CNN have recently become a popular approach in the detection of defects in composite materials because they are able to learn sets of complex patterns and features in images. Kim et al. [1] showed that terahertz electromagnetic waves in conjunction with CNN to identify micro-defects in the glass-reinforced polymer composites could be done non-destructively with good results as far as sensitivity and accuracy were concerned. On the same note, Wang et al. [2] derived a DA-CNN to detect internal de-bonding flaws in composites, demonstrating the high-resolution terahertz and its advantageous use in improving defect characteristics. The methodologies of defect classification have also grown beyond the more standard uses, with Guo et al. [3] demonstrating the flexibility of CNN to a welding defects classification problem, which continues to demonstrate the versatility of CNN in a wide range of tasks. In addition, alternative image processing aspects, including compression, and their effects on the accuracy of defect classification have been explored by Benbarrad et al. [4], where the pre-processing aspect of CNN performance has been identified. CNN together with other neural network structures have become a promising area of hybrid architecture integration to improve defect classification. Li et al. [5] suggested a hybrid model that included CNN and Transformers to detect the strip steel surface defects, which demonstrated better classification results than conventional CNN models. Transfer learning has also been successfully applied to defect classification problems; Kumaresan et al. [6] used CNN with transfer learning to identify welding defects and proved the use of the method to be effective in terms of accuracy improvement. Moreover, Dong et al. [7] researched a deep one-class multitask CNN to classify and detect defects with MM stressed on the adaptability of CNNs to a variety of defects. [8] paper shows a comparative analysis of the detection of distracted drivers with the help of Convolutional Neural Networks (CNN). It is aimed at testing CNN models to identify distracted driving behaviors by using video and image data. The authors contrast a range of CNN structures, indicate the performance indicators, and interpret the efficiency of these models in the correct recognition of distractions like mobile phone use (among others) and other inattention patterns during the driving process. Although such developments were achieved, it is important to note that there is still a significant gap in the comparative research in the hyper-parameter optimization methods to be used in CNN specifically on fibre-reinforced composite [6]. The majority of studies have focused on CNN architectures or transfer learning but have not adequately addressed the relationship between various optimization strategies, including Keras Tuner and Particle Swarm Optimization, and model performance and defect classification accuracy [4]. In order to address this gap, the current study would undertake a systematic assessment of these methods of hyper-parameter optimization, and as a result, would play a role in improving automated defect in fibre-reinforced composites. A study on the classification of microcracks in cement concrete by Wang et al. [9] with the X-ray CT imaging gave a basis of how CNN can be used in identifying the existence of microcracks. In the same light, Awaja et al. [10] surveyed the processes of crack formation and propagation in the polymer structures, insisting on the need to control the structural integrity in proactive mode by using automated techniques of structural detection. These studies have highlighted the need to have proper crack detection mechanisms that may be utilized in FRC. Simsek et al. [11] offered incomplete data modal with TLBO to enhance the accuracy of the damage detection. To detect damage in laminated composite beam, their approach solved the problem with limited data by using the finite element model (shear-deformable), which is optimized by harmonic search (HS). Kahya et al. [12] demonstrated the potential of damage estimation with TLBO in the optimization of damage detection in composite laminated beam. Their methodology revealed better accuracy of their detection, which is indicative of the usefulness of the sophisticated computational models and optimization algorithm.

The success of CNN in categorizing composite defects has been looked into more and more. As an illustration, Santos et al. [13], investigated the contribution of microcracks in collagen networks, and this means that microstructural analysis is relevant in material performance. Still, regarding the interpretation of machine learning as a process that detects microcracks in the dentin, which is intimately connected to FRC, Haupt et al. [14] have proven the applicability of this methodology to non-destructive dentin using machine learning. This depicts that CNN based models do have the ability to efficiently process imaging data that is complex and then identify a microstructural defect in the image data. Multiple investigations have concentrated on the hyper-parameter optimization to improve the accuracy of machine learning models to detect defects. Keras Tuner has become popular in the parameter s optimization of CNN such as filters, kernel sizes and dense units. As an example, Wang et al. [15] also applied this method to optimize CNN in their defect detection models. Nevertheless, effective as Keras Tuner is, it has some drawbacks in searching the global optimum, since the technique is usually limited to a set domain of potential search and it can omit important hyper-parameters. PSO has become a serious alternative in global optimization. PSO, unlike in the conventional methods, reacts dynamically to swarm behavior by dynamically changing the search space positions of the particles. The application of PSO was used to optimise the fracture of fibres in composite by Lv et al. [16] and Ren et al. [17]. Their work illustrates how PSO may be effectively used to search over large hyper-parameter space, increase model accuracy and detect defects more effectively. The performance of CNN optimized with PSO in terms of defects detection has also been found to be higher than that of Keras tuner. Other research variants, including

the ones conducted by Mehdikhani et al. [18] and Xue et al. [9], have shown that higher accuracies may be obtained with the PSO-optimized models due to a good capability of exploring global optima in the hyper-parameter space. Detecting the fibre pull-outs and other advanced defects in FRC are also improved by the use of PSO. The latest efforts in fibre pull-out detection in composite drilling have only increased awareness to the role of machine learning in the classification of defects. Abdul Kudus et al. [19] established an image processing technique to determine fibre pull-outs, whereas Kahl et al. [20] studied hybrid effects in glass/cellulose-reinforced polypropylene pull-out testing. Through these studies, it is highlighted that CNN and PSO can be used to enhance detection accuracy of the defects in fibre-reinforced composites.

Fibre-Matrix De-Bonding

Edge Detection: Image processing methods, e.g. edge detection algorithms (e.g., Sobel, Canny), can be used to emphasise the edges between the fibre and matrix. In fibre-matrix de-bonding a separation would occur between the fibre and the surrounding matrix. The visibility of these separations is increased by edge detection creating an easier way of the CNN to detect de-bonding. Contrast Enhancement: This feature can be used to adjust the contrast of the SEM images to bring out the more prominent fibre-matrix interface as this will allow the model to determine places where the bond has failed. Figures 1 up to 5 show different SEM images demonstrating the main phenomenon in fibre-reinforced composites: fibre-matrix de-bonding (Fig. 1), fibre pull-out (Fig. 2), fibre fracture (Fig. 3), voids in the composite (Fig. 4), and microcracks (Fig. 5).

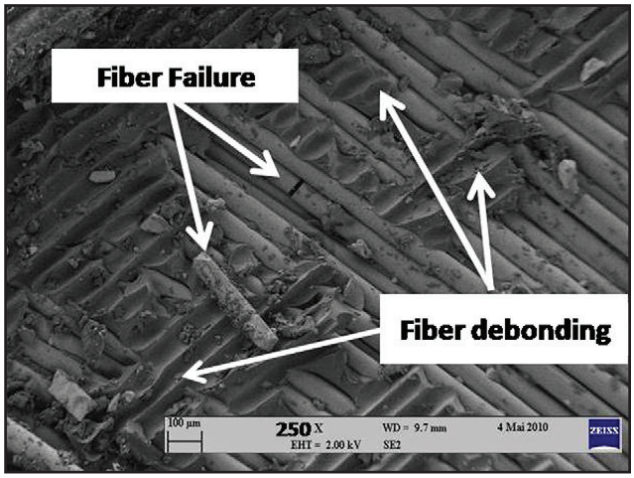


Figure 1. Fibre-matrix de-bonding.

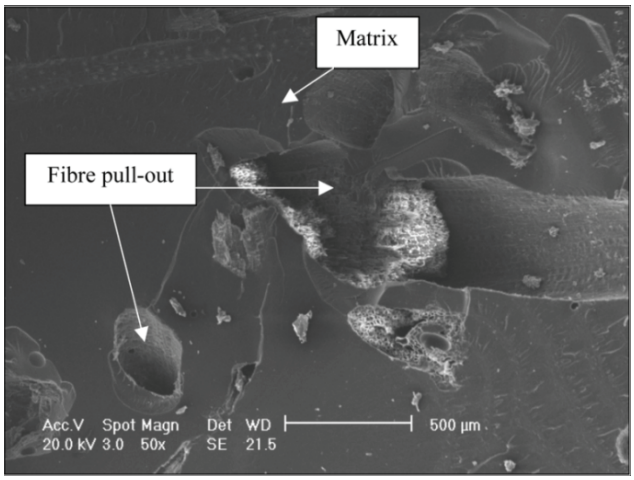
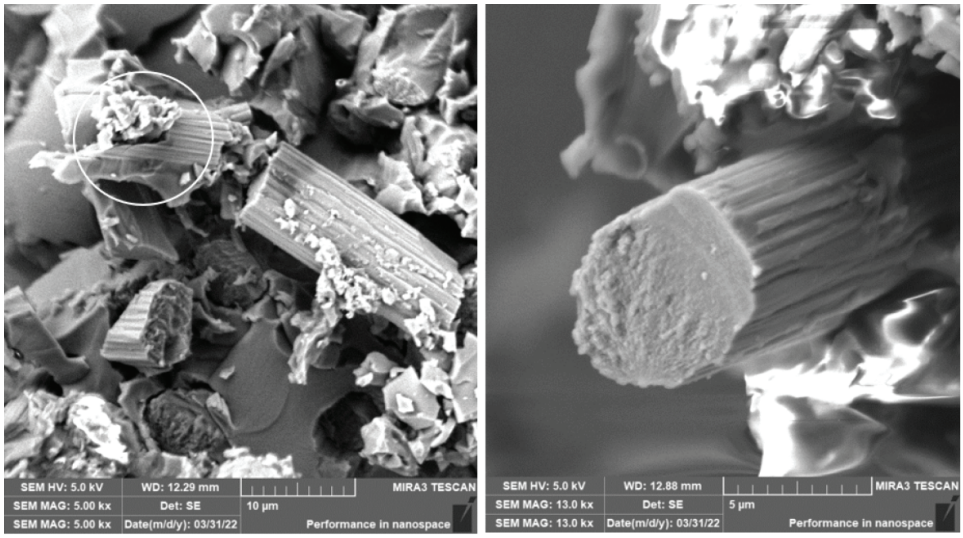


Figure 2. Fibre Pull-Out.



(**a**) (**b**)

Figure 3. Fibre Fracture.

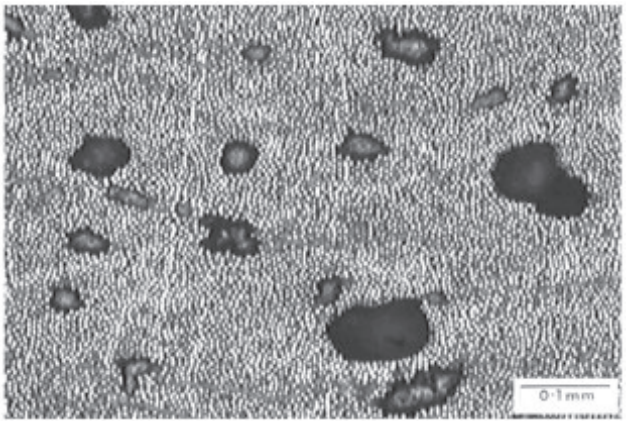


Figure 4. Voids in composite.

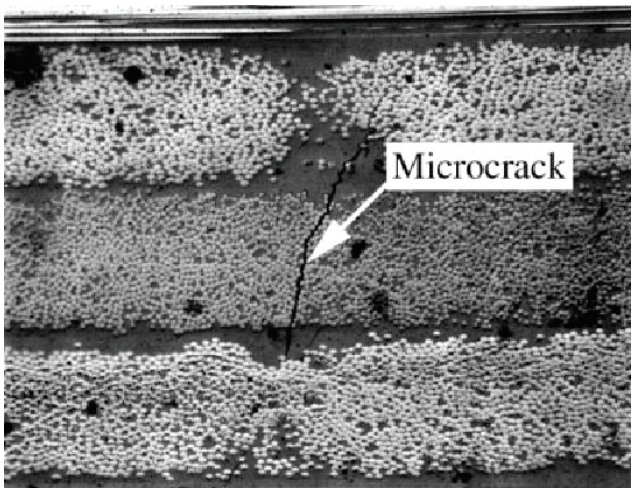


Figure 5. SEM of Micro cracks.

Fibre Pull-Out

Contour Analysis: In fibre pull-out a fibre will partially pull out leaving a hole where it was originally deposited. The visibility of these cavities can be improved using image processing in which the fibre and empty spaces surrounding the fibre are distinguished through the application of thresholding techniques [21]. Morphological Operations: Morphological techniques like dilation and erosion may be used to highlight the edges of the fibres and crevices created in the process of the pull-out and consequently readily identify this particular defect [20].

Fibre Fracture

Edge and Texture Analysis: Fibre ends that are broken are characterized by a sharp edge. The broken ends of the fibres can be marked by image processing functions like edge detecting along with texture analysis. How the fractured fibres become irregular and coarse in texture is enhanced and they become distinguishable in SEM images [16]. Segmentation: Image processing can be used to target the fibres and isolate them against the surrounding matrix to enable the CNN to respond to the fibres and whether they are intact or fractured [22].

Voids

Blob Detection: Voids are literally air pockets that are represented by dark and empty areas of SEM images. These round or irregularly shaped voids can be distinguished by using the blob detection algorithms. The technique separates the darker parts which are air pockets and distinguishes them against the solid matrix [18]. Thresholding: Image processing can be used to detect the voids (darker areas) of the image by using thresholding methods which increase the accuracy of detecting the voids (learned 23).

Micro Cracks

Crack Detection Algorithms: Image processing method like crack detection algorithm can be used to focus on fine and linear features in the images. Such algorithms are used to boost the visibility of the micro cracks that are usually present along the fibre-matrix interface [9]. Noise Reduction and Filtering: Micro cracks may be minute and hard to detect particularly when the SEM picture is full of noise. It can be effectively filtered using methods such as Gaussian filters that can remove noise without reducing finer details, so that CNN models can be able to detect micro cracks more accurately [13].

Major image processing techniques that are applied during the Process: Threshold (to distinguish defects and the matrix), Edge Detection (to bring out sharp pixel intensity variances), Morphological Operations (in order to improve shapes and structures), Noise Reduction Filters (to remove artefacts which hide the defects), Increasing or minimizing the Contrast and Brightness (Better see millennial defects) Through these image processing techniques, SEM images of the fibre-reinforced composites are improved and the process of detecting defects becomes more accurate [7].

MATERIALS AND METHODS

The proposed research works on hyper parameter optimization on CNN and classification of defects in fibre reinforced composites. A library of 3670 SEM images was created – 734 super images of 5 categories of faults. The images, which are obtained in different research laboratories of Pune, India, represent a variety of materials and the fibre orientations to provide solid training. To create the dataset from imageSet function-directory we split the dataset into training and validation in the ratio of 80:20. The image scale was reduced to 150x150 pixels and pixels were scaled to the range of 001. The identified Methodology of this research is indicated as shown in Figure 6. All SEM images of the dataset are resized to 150x150 pixels, meaning that all images will form 150x150x3 dimension matrices, where x3 is RGB colors, i.e. “color channels. The pixel values are divided by 255 to bring the values of the pixels

within the range 0 to 1 before encoding of images to the CNN. In CNN, inner layers are convolutional in nature meaning that an array of filters (or kernels) are applied to the input images. These filters are taken in a sliding format with the input image- dot products with the filter and regions of the image are calculated. It is the method that identifies geometric aspects such as edges, textures and patterns. The input image is represented by the values I(x,y) with xxx and y being the pixel co-ordinates and the filter (or kernel) being known as K(i, j), iii and j being the size of the kernel. Equation 1 below gives the output of the convolution operation:

$$O(x,y)=\sum_j \sum_j I(x+i, y+j) \cdot K(i,j) \quad (1)$$

This operation is performed for every position of the kernel on the image, generating an output feature map that highlights important features of the image. The CNN used in this study includes two convolutional layers, each with tunable filter sizes optimized by Keras Tuner. The number of filters and their sizes are determined through hyper-parameter tuning.After the convolution operation, a nonlinear activation function is applied, commonly the Rectified Linear Unit (ReLU). The ReLU function is defined as equation 2:

$$f(x) = \max(0,x) \quad (2)$$

This introduces non-linearity into the model, allowing the CNN to learn complex patterns.

Max-pooling layers down sample the feature maps by taking the maximum value from a patch of the image. This reduces the spatial dimensions of the data while retaining the most important features. If we apply a 2×2 max-pooling operation, the output for each 2x2 region is given by equation 3:

$$P(x,y) = \max\{I(x+i,y+j)\},\text{for } i,j\in\{0,1\} \quad (3)$$

This pooling operation helps reduce computational complexity and the risk of overfitting.

After the convolutional and pooling operations, the 2D feature maps are flattened into a 1D vector, denoted as equation 4:

$$F = [f1, f2,\ldots,fn] \quad (4)$$

This vector is then fed into a fully connected layer for classification.

The fully connected layer takes the flattened feature vector and computes weighted sums to classify the image. If Wk represents the weights of the kth neuron and bk is its bias term, the output zk of the fully connected layer is given by equation 5:

$$zk = \sum_{i=1}^{n} Wk[i] \cdot F[i] + bk \quad (5)$$

This linear transformation is applied for each neuron in the fully connected layer, and the results are then passed through a softmax activation function to get the final class probabilities. The softmax function is used to convert the raw output of the fully connected layer into class probabilities. For a class k in a total of C classes, the probability Pk is calculated as equation 6:

$$Pk = \frac{\exp(zj)}{\sum_{j=1}^{C} \exp(zk)} \quad (6)$$

This ensures that the output values are between 0 and 1, with the sum of all probabilities equal to 1.

The loss function used for training is sparse categorical crossentropy. For a given true class label y and predicted probability Py, the loss L is computed as equation 7:

$$L=-\log (Py) \quad (7)$$

This loss is minimized during training to improve the model's accuracy in classifying defects. The model uses the Adam optimizer, which updates the weights based on the gradients of the loss function with respect to the weights. Adam combines the advantages of both RMSProp and stochastic gradient descent with momentum. The weight update rule is given by equation 8:

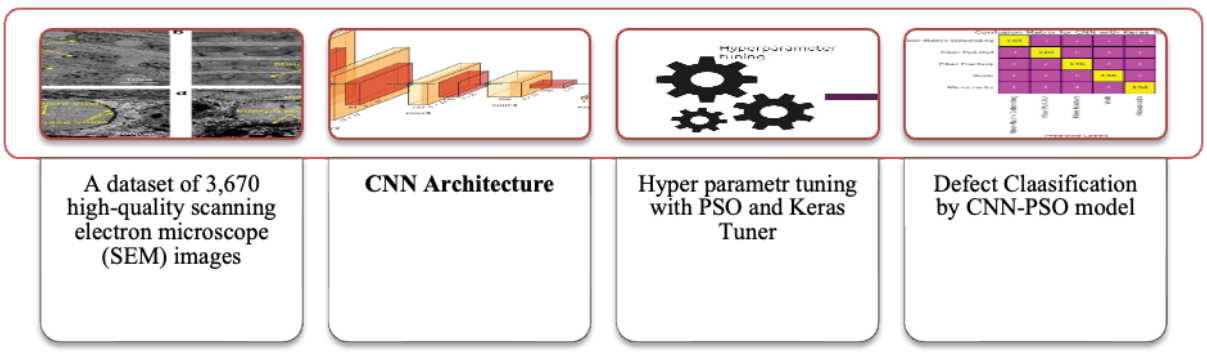


Figure 6. Methodology for this research.

$$\theta t+1=\theta t-\eta\cdot\frac{m_t}{\sqrt{v_t+\epsilon}} \quad (8)$$

Where:
θt is the weight at time step t,
η is the learning rate,
mt is the first moment (mean of gradients),
vvt is the second moment (variance of gradients),
ϵ is a small constant to avoid division by zero.

A CNN can be used to highlight spatial information, such as edges, textures and patterns, by applying different filters (kernels) to the input image through a convolution. Output at a particular position consists of dot product of input image values and the values of the kernel at the respective coordinates (equation 1). A Rectified Linear Unit (ReLU) activation is applied to this convolution and as a continuation of this, the model is non-linear with further propagation of positive values, equation 2 below: In order to down-sample the spatial size of the feature map but preserve significant features, max-pooling is used. This operation will return the maximum value to a part of the image (equation 3) - this is useful to reduce the computational load and overfitting. Subsequently, the 2D feature maps are projected onto a 1D vector (equation 4), and passed on to the fully connected layer. The fully connected layer which is the weighted combination of the inputs combined with each neuron (equation 5). The result comes out once we then use a softmax class activation function (equation 6) to get the class probabilities, which should total to one. During training, the difference between the actual probability and the predicted probability (equation 7) is implemented using a sparse categorical crossentropy loss function. The Adam optimizer modifies weights of the model by a rule that helps to blend the benefits of both the RMSProp and stochastic gradient descent and momentum (equation 8) that enables the model to enhance its performance as time goes by. To optimize the hyper-parameters of the CNN model, i.e., the two convolutional layers with the sizes of filters as parameters, a number of max-pool layers, a flattening layer, and a fully connected layer with softmax as the activation function for the classification layer were created using Cnn With Keras Tuner. Adam was used to compile the model, allowing the option of a tunable hyper-parameter that is the learning rate, and sparse categorical crossentropy was used as the loss function. Keras Tuner has performed a Random Search which was done on five trials to locate the optimal hyper-parameter s. Once ten epochs were completed, the performance of the model was plotted and it was stored as cnn_image classifier best.h5 to use in future runs of the automated classification system. CNN With PSO The Particle Swarm Optimization (PSO) is a method of optimization based on the natural behavior of swarms (e.g. birds or fish). Each particle of the swarm is a possible solution (or a set of hyper-parameters, such as the number of filters, kernel size, pooling size and so on) of a CNN. A particle has a position (set of hyper-parameter s that the particle has currently), and a velocity (that determines how it moves in the search space). The objective is to identify hyper-parameter s in the most optimum way that will reduce the validation loss or maximize the validation accuracy of the CNN. For each PSO time step the fitness of each particle is evaluated by learning the CNN using the hyper-parameter s being the particle being considered. The fitness of the fitness will be computed based on the performance of CNN in a validation data. Each particle also has a memory of its best position (called personal best or p) and each particle learns the global best (g), the best solution found by any particle in the entire swarm. The status and the velocity of every particle are revised basing on three elements namely the former velocity of the particle (inertia), the appeal of the particle to a personal best and that of the particle to a global best. Alternate solutions for the swarm test are added in to avoid these random factors. The three factors are used to query its velocity, and then uses the new velocity to offset its position. As time goes on, the swarm particles get closer to the best solution optimizing hyper-parameter s. The output is the solution to a problem; that is, a result at which a problem ceases, as when a maximum number of iterations is reached, or the accuracy is satisfactory. In this study, PSO is used to learn the best hyper-parameter s of a CNN for defect identification of fibre-reinforced composites. The CNN is optimized by the optimum hyper-parameter s discovered by PSO to enhance an efficient performance in the accurate classification of defects fibre-matrix de-bonding, fibre pull-out, fibre fracture, voids and microcracks. Through PSO, the CNN has an enhanced classification accuracy than the conventional ways of optimization.

Pseudo Code for PSO with CNN

1. Initialization
# Initialize parameters
N = 1000 # Number of particles in the swarm (population_size)
M = 10 # Optimization iterations desired
num_classes = 5 # Number of defect classes
data = load_dataset() # Load the dataset

Initialise swarm (particles, velocities and best positions)
swarm = [] # List to store all particles in the swarm
global_best_value = -inf # define the global (lowest possible) value
global_best_position = -1 # Set up global best position

Initialize each particle of the swarm.
for i in list of lowest to highest numbers under the population size:
x = get_attr("particle").position.x256 (for example) = initialize_particle(… hyper-parameters s…)particle = initialize_particle(… hyper-parameters s…, …) #

Randomly generate a particle with hyper-parameters s (see get_attr("particle"))
particle.best_value = -inf # Initialize particle's best value
particle.best_position = None # begin a blank particle's best position
flock.append(XMParticle) # Add particle to the flock

Construct CNN model(With Hyper-parameters)
Defines a CNN model using the hyper-parameters.
def build_cnn_model(hyper_parameters):
model = Sequential() # Create a Sequential model

#Add Convolutional layers including hyper-parameters: filters, kernel_size, activation
for layer_params in hyper_parameters["conv_layers"]:
model.add(Conv2D(layer_params['filters'], kernel_size=layer_params['kernel_size'], activation=layer_params ['activation']))

Add MaxPooling/AveragePooling layers (etc.).
Do the following for pool_params in every iteration of hyper_parameters["pooling_layers"]:
model.add(MaxPooling2D(pool_size=pool_params['pool_size']))

# Flatten the output
model.add(Flatten())

A dense layer activation function and a dropout layer are added.A dense layer, activation function and a dropout layer are added.
for dense_params in hyper_parameters["dense_layers"]:
model.add(Dense(dense_params['units'], activation=dense_params['activation']))
if dense_params.get('dropout', 0) > 0:
model.add(Dropout(dense_params ['dropout']))

Compile using optimizer and loss function.
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

return model

3. Objective Function
Function to return the fitness (accuracy) for each particle
def objective_function(particle):
Pruned EVP hyper-parameters: hyper_parameters = decode_particle(particle) # Extract hyper-parameters from the particle

Create CNN model using the hyper-parameters. Design CNN model using the hyper-parameters.
model = build_cnn_model(hyper_parameters)

Stage 1: Train the model using the training data.
history = model.fit(dataset.train_images, dataset. train_labels, validation_split=0.2, epochs=E, batch_size=B).

Using the validation samples, compare the performance of the model.Compare the model's performance with the validation samples.
The model's accuracy on a validation dataset.accuracy = evaluate_model(model, dataset.validation_images, dataset.validation_labels)

return accuracy - return the accuracy as fitness value

4. Main PSO Loop
The main loop of Particle Swarm Optimization (PSO) is shown below.The main particle swarm optimization (PSO) loop is:
for i in iterable range(1, num_iterations + 1):
Evidence that evaluates fitness of each particle. Evidence to assess fitness of individual particles.
In a swarm of particles (clumps):
Fitness = objective_function(particle), Evaluate fitness (accuracy)

To update the personal best record for each particle.To update the particle personals best record.
if fitness_value > particle.best_value:
particle.fitness_value = fitness_value # Update particle's personal best
If we're iterating our array, then we need to save each particle's best position.If we are traversing our array, we need to know each particle's best position.

Remove the old solution and pick up the new solution, if there is a better one.
max_particle = max(swarm, key=lambda p: p.best_value) # choose the particle having the best fitness value
if max_particle.best_value > global_best_value:
global_best_value = max_particle.best_value
For each particle, find its best position:

Iterate the velocities and positions of each particle.Repeat calculation of velocities and positions of each particle.
for each particle in swarm:
alpha = 0.95 # Velocity update parameter of inertia
H = get_normal_height_above_ground(particle) # What is the height the particle is above the ground?

5. Final Model Training
After a few iterations, the global_best_position will contain the best hyper-parameters.

best_hyper_parameters = global_best_position (Extraction of the optimal hyper-parameters)

Learn the final model using an optimized set of the hyper-parameters.Fit the final model with the most appropriate hyper-parameters.

final_model = build_cnn_model(best_hyper_parameters)

Let's fit the model to the training data using the following line up to E epochs, with a batch size of B examples: final_model.fit(dataset.train_images, dataset.train_labels, epochs = F, batch size = B)

When the experiment is finished, test the final model with the test set

test_accuracy = evaluate_model(final_model, dataset.test_images, dataset.test_labels)

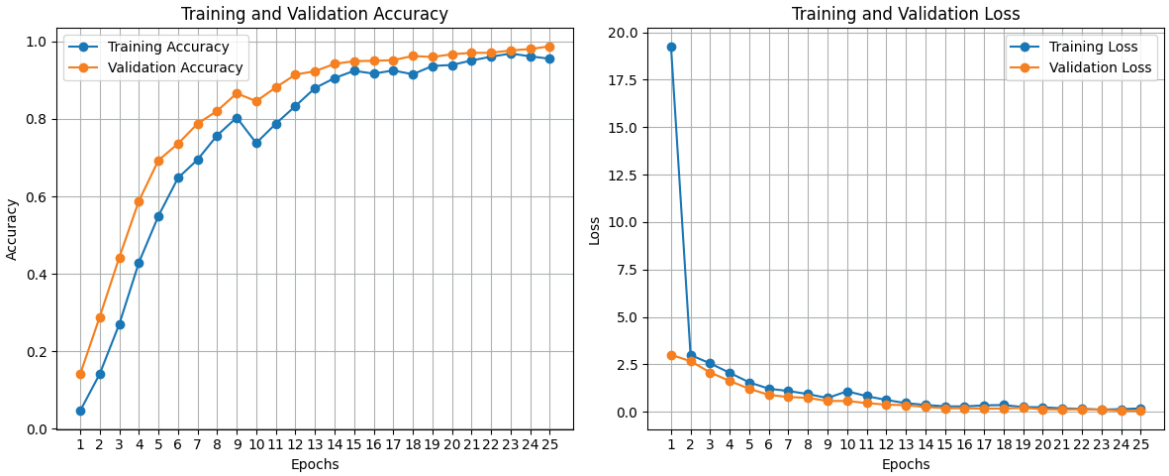
print("Test Accuracy:", test_accuracy)

RESULTS AND DISCUSSION

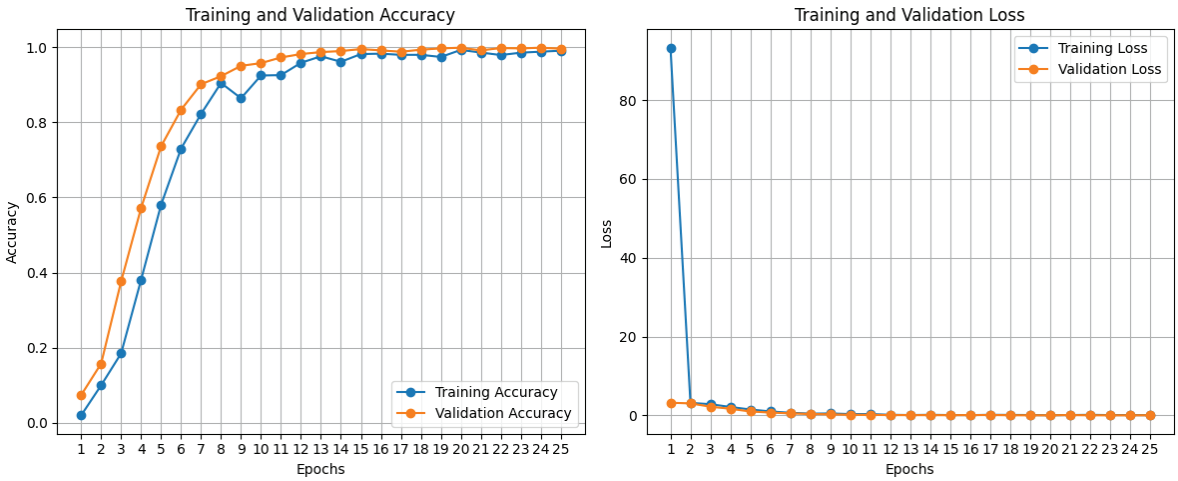
Epoch wise training-validation, accuracy and loss with Keras tuner and PSO are shown in Figure 7A and 7B respectively.

The results obtained when comparing CNN trained on Keras Tuner and Particle Swarm Optimization (PSO) as demonstrated in table 1 yield significant differences in terms of performance and optimization approaches. The CNN optimized using PSO shows better results in all measurements, as it has a better training and validation accuracy (99.23% and 99.82% respectively) and an F1-score (99.77%). Its validation loss (0.02) is much smaller than that of the Keras Tuner-optimized (0.1119) model, which means that the model generalizes better and is less overfitted.

Performance comparison of CNN with Keras tuner and CNN with PSO is shown in figure 8.



a) Keras



b) PSO

Figure 7. Epoch wise training-validation, accuracy and loss.

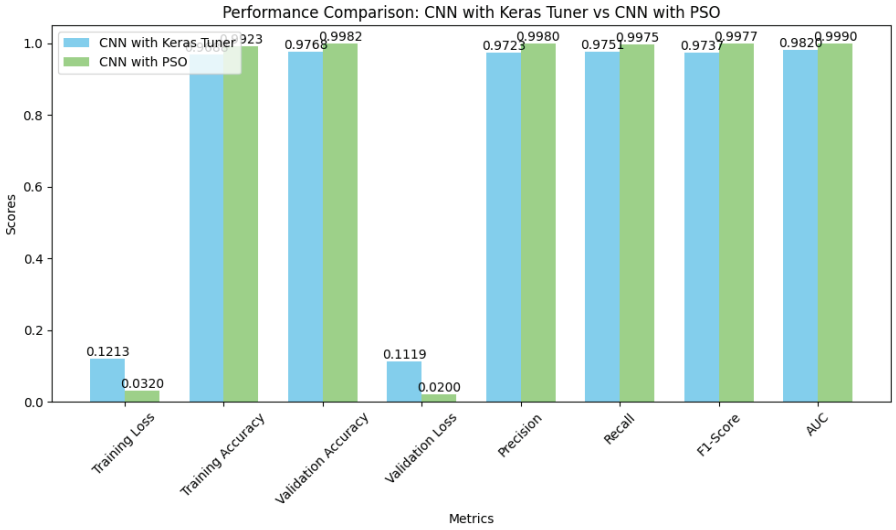


Figure 8. Performance comparison of CNN.

Table 1. The comparison of CNN models Evaluation metrics

Model	Optimization & Hyper-parameters	Training Performance	Validation Performance	Classification Metrics	Inference Time
CNN with Keras Tuner	Keras Tuner (Random/Bayesian Search); Fixed search space; Tuned parameters: $conv_1$ filters = 128, kernel = 5; $conv_2$ filters = 192, kernel = 3; dense units = 256; learning rate = 0.000458	Loss = 0.1213; Accuracy = 0.9688	Loss = 0.1119; Accuracy = 0.9768	Precision = 0.9723; Recall = 0.9751; F1 = 0.9737; AUC = 0.982	35 ms
CNN with PSO	Particle Swarm Optimization; Dynamic particle-based search; Optimized parameters: filters = 117, kernel = 8, pool size = 2, layers = 4	Loss = 0.032; Accuracy = 0.9923	Loss = 0.020; Accuracy = 0.9982	Precision = 0.9980; Recall = 0.9975; F1 = 0.9977; AUC = 0.999	25 ms

Although Keras Tuner operates with fixed search spaces and even resorts to such techniques as random or Bayesian search, PSO dynamically reacts particles in the search space effectively samples a larger space and settles on solutions overall optimal on a global scale. Such dynamic nature of PSO ensures enhanced precision and recollection, reducing false positive and false negative. Keras Tuner is based on a simple method, so it finds the most optimal configuration after a certain number of experiments, which is computationally inexpensive yet can fail to detect global optima. Conversely, PSO generates an iterative process which depends on particle fitness which, though computationally more weighty, leads to high performance. Moreover, PSO-optimized CNN needs fewer inference time (25 ms/batch) than the Keras Tuner model (35 ms), which is important to real-time applications. All in all, PSO is a more robust strategy, with a much more appropriate performance and accuracy, whereas Keras Tuner offers a computationally efficient method with more or less satisfactory performance, and therefore it can be assumed that this tool is preferred in those cases when performance is the key factor.

The confusion matrix of CNN and PSO is shown in Figure 9 and when using the Keras tuner the confusion matrix is shown in Figure 10 [23-25]. Recently, studies have focused on CNN's ability to detect defects – which is often enhanced using approaches such as Particle Swarm Optimization. Two attempts have been made to identify faults in 3D-printed components, a project that successfully used a standard CNN [26] while another won over by implementing Fast R-CNN detected rail cracks as an example [25]. Significantly, tuning influents the results of the deep learning settings; the work of Sezer and Altan [27] emphasizes the importance of tuning in improving precision for complex data. The one thing that's struck me is the effect that can be seen when the model is refined when patterns are difficult to follow. So far, results indicate that

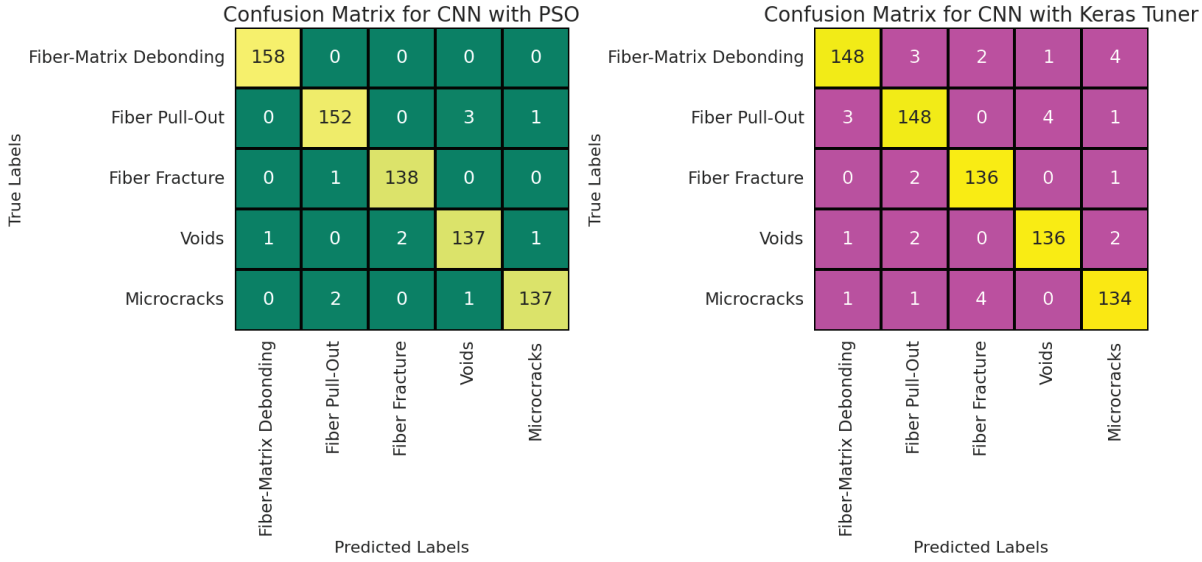


Figure 9. Confusion matrix for CNN with PSO and CNN with Keras tuner.

the tuning can be done beforehand, such as hyper-parameters, and actually improve the accuracy of convolutional networks in finding flaws from fiber-based materials. In factories, it is the setup used to test that lets you know if and how validation scores like numbers matter. Take particle swarm optimization: It should be able to say 9923%, aligned with the truth, which is okay, but only if what it is saying matches what happens in reality such as catching faults as a part is being made. A mere correct use would be one important criterion, but also the rate of use and regular production. But limits there are... One big factor? The sharpness and clarity of electron microscope images at which used; its performance heavily relies on these images. Another reason results in differences in images from SEM devices is due to factors such as resolution, background disturbances, sample treatment and so forth, which affect model performance in industrial settings. We should also remember the lesson on the different clarity of images that can impact their results, and create assumptions based on the environment where these kinds of systems might fail. Next up is data volume or size during test—small amounts do not allow for insight into the applications that might exist. Lightly can do a decent job in a narrow range of conditions doesn't mean that he can do the same level of work in every type of condition, under all classes of defect, in all manufacturing conditions. Regional considerations also should be kept in mind - how larger and more diverse sets of data might improve the reliability and enhance the reach for current boundaries.

CONCLUSION

In this paper, the authors compare the results of two hyperparameter optimization methods, Keras Tuner and Particle Swarm Optimization with automated fibre-reinforced composite defect classification by Convolutional Neural Networks. The findings prove that the Particle Swarm Optimization-optimized Convolutional Neural Network is superior to the one that was optimized with the help of the Keras Tuner in a variety of quantitative indicators. The model using Particle Swarm Optimization obtained a training accuracy of 99.23 percent and a validation of 99.82 percent, whereas Keras Tuner had 96.88 percent and 97.68 percent, respectively. Also, the loss on validation was also much lower with Particle Swarm Optimization of 0.02, as opposed to 0.1119 with Keras Tuner, which implies better generalization ability. Particle Swarm Optimization also exhibited a much higher precision (99.8 percent), recall (99.75 percent), and F1-score (99.77 percent), indicating that it is very effective with reducing false positives and false negatives. The Particle Swarm Optimisation-optimized model also had a higher area under the curve of 0.999 than the area under the curve of 0.982 of Keras Tuner indicating that it has better overall classification performance. To consider the computational efficiency, the Particle Swarm Optimization model had a lower inference time (25 milliseconds per batch) relative to Keras Tuner (35 milliseconds), which makes it appropriate to real-time defect classification systems. The optimization framework of Particle Swarm Optimization offered superior global search over hyper parameters since it is a dynamic particle-based optimization framework, and thus it was able to find a superior model convergence. By comparison, Keras Tuner at least was limited by a fixed search space, avoiding the global optima in certain instances, but was not as computationally heavy. The weakness of the selected methods are the reliance on massive data sets to train CNN models and the computational cost and requirements of hyper

parameter optimization algorithms. This paper numerically confirms the benefits of Particle Swarm Optimization in hyper parameter optimization to the Convolutional Neural Networks in classifying defects with regards to having superior accuracy, strength, and computational power. Future research will concentrate on applying Particle Swarm Optimization together with domain specific hybrid optimization to further enhance the model performance as well as extend of its use in other defect classification fields.

AUTHORSHIP CONTRIBUTIONS

Authors equally contributed to this work.

DATA AVAILABILITY STATEMENT

The authors confirm that the data that supports the findings of this study are available within the article. Raw data that support the finding of this study are available from the corresponding author, upon reasonable request.

CONFLICT OF INTEREST

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

ETHICS

There are no ethical issues with the publication of this manuscript

REFERENCES

[1] Kim HS, Park DW, Kim SI, Oh GH, Kim HS. Non-destructive detection of thin micro-defects in glass reinforced polymer composites using a terahertz electro-magnetic wave based on a convolution neural network. Compos Part B Eng 2023;257:110694. [CrossRef]

[2] Wang X, et al. DA-CNN-based similar terahertz signal identification for intelligent characterization of internal debonding defects of composites under high-resolution mode. Compos Struct 2023;322:117412. [CrossRef]

[3] Guo B, Wang Y, Li X, Zhou Y, Li J, Rao L. Welding Defect Classification Based on Lightweight CNN. Int J Pattern Recognit Artif Intell 2023;37:2350026. [CrossRef]

[4] Benbarrad T, Eloutouate L, Arioua M, Elouaai F, Laanaoui MD. Impact of image compression on the performance of steel surface defect classification with a CNN. J Sens Actuator Netw 2021;10:73. [CrossRef]

[5] Li S, Wu C, Xiong N. Hybrid Architecture Based on CNN and Transformer for Strip Steel Surface Defect Classification. Electronics 2022;11:1200. [CrossRef]

[6] Kumaresan S, Aultrin KSJ, Kumar SS, Anand MD. Transfer Learning with CNN for Classification of Weld Defect. IEEE Access 2021;9:3093487-3093495. [CrossRef]

[7] Dong X, Taylor CJ, Cootes TF. Defect Classification and Detection Using a Multitask Deep One-Class CNN. IEEE Trans Autom Sci Eng 2022;19:3109353-3109362. [CrossRef]

[8] Jagadale SH, Attar DPR. Distracted Driver Detection: A Comparative Study Using CNN. In: Proceedings of the International Conference on Industrial Engineering. IEEE; 2023. p. 1-8. [CrossRef]

[9] Wang R, Wu H, Zhao M, Liu Y, Chen C. The Classification and Mechanism of Microcrack Homogenization Research in Cement Concrete Based on X-ray CT. Buildings 2022;12:1011. [CrossRef]

[10] Awaja F, Zhang S, Tripathi M, Nikiforov A, Pugno N. Cracks, microcracks and fracture in polymer structures: Formation, detection, autonomic repair. Progress in Materials Science 2016;83:1-15. [CrossRef]

[11] Şimşek S, Kahya V, Adıyaman G, Toğan V. Damage detection in anisotropic-laminated composite beams based on incomplete modal data and teaching-learning-based optimization. Struct Multidiscip Optim 2022;65:3421-3435. [CrossRef]

[12] Kahya V, Şimşek S, Toğan V. Vibration-based damage detection in anisotropic laminated composite beams by a shear-deformable finite element and harmony search optimization. Struct Multidiscip Optim 2022;65:3270-3285. [CrossRef]

[13] Santos S, Neu CP, Grady JJ, Pierce DM. Genipin does not reduce the initiation or propagation of microcracks in collagen networks of cartilage. Osteoarthr Cartil Open 2022;4:100233. [CrossRef]

[14] Haupt F, Riggers I, Konietschke F, Rödig T. Effectiveness of different fiber post removal techniques and their influence on dentinal microcrack formation. Clin Oral Investig 2022;26:4338-4345. [CrossRef]

[15] Wang J, Zhang J, Feng P. Effects of tool vibration on fiber fracture in rotary ultrasonic machining of C/SiC ceramic matrix composites. Compos Part B Eng 2017;129:1-15. [CrossRef]

[16] Lv D, et al. Effects of fiber orientations on fracture mechanisms in rotary ultrasonic drilling of carbon fiber reinforced plastic laminates. J Manuf Process 2023;105:1-20. [CrossRef]

[17] Ren F, Yu Y, Cao M, Li Y, Xin C, He He Y. Effect of pneumatic spreading on impregnation and fiber fracture of continuous fiber-reinforced thermoplastic composites. J Reinf Plast Compos 2017;36:718085. [CrossRef]

[18] Mehdikhani M, Gorbatikh L, Verpoest I, Lomov SV. Voids in fiber-reinforced polymer composites: A review on their formation, characteristics, and effects on mechanical performance. Journal of Composite Materials 2019;53:8772152. [CrossRef]

[19] Ashrafi SA, Miller PW, Wandro KM, Kim D. Characterization and effects of fiber pull-outs in hole quality of carbon fiber reinforced plastics composite. Materials 2016;9:100828. [CrossRef]

[20] Kahl C, Bagnucki J, Zarges JC. Demonstration of Hybrid Effect in Single Fiber Pull-Out Tests for Glass/Cellulose-Reinforced Polypropylene with Different Fiber-Matrix Adhesions. Polymers 2022;14:2517. [CrossRef]

[21] Abrishambaf A, Barros JAO, Cunha VMCF, Frazão C. Time dependent behaviour of fibre pull-out in self-compacting concrete. Cem Concr Compos 2017;77:1-15. [CrossRef]

[22] Hashem OM, Wahba SME, Nasr-Eldin TI. Urban voids: Identifying and optimizing urban voids potential as a revitalization source in enhancing developing countries' city income. J Eng Appl Sci 2022;69:1-15. [CrossRef]

[23] Doh J, Rodrigues S, Saka-Helmhout A, Makhija M. International business responses to institutional voids. Journal of International Business Studies 2017;48:1-20. [CrossRef]

[24] Zhu YC, et al. A generic deep learning framework to classify thyroid and breast lesions in ultrasound images. Ultrasonics 2021;110:106300. [CrossRef]

[25] Choi JY, Han JM. Deep Learning (Fast R-CNN)-Based Evaluation of Rail Surface Defects. Applied Sciences 2024;14:1874. [CrossRef]

[26] Wen H, Huang C, Guo S. The application of convolutional neural networks (CNN) to recognize defects in 3D-printed parts. Materials 2021;14:2575. [CrossRef]

[27] Sezer A, Altan A. Optimization of Deep Learning Model Parameters in Classification of Solder Paste Defects. In: HORA 2021 - 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings. IEEE; 2021. p. 1-6. [CrossRef]