



Optimisation of convolutional neural network architecture using genetic algorithm for the prediction of adhesively bonded joint strength

Edore G. Arhore¹ · Mehdi Yasaee¹ · Iman Dayyani¹

Received: 9 November 2021 / Revised: 8 June 2022 / Accepted: 3 August 2022
© The Author(s) 2022

Abstract

The classical method of optimising structures for strength is computationally expensive due to the requirement of performing complex non-linear finite element analysis (FEA). This study aims to optimise an artificial neural network (ANN) architecture to perform the task of predicting the strength of adhesively bonded joints in place of non-linear FEA. A manual multi-objective optimisation was performed to find a suitable ANN architecture design space. Then a genetic algorithm optimisation of the reduced design space was conducted to find an optimum ANN architecture. The generated optimum ANN architecture predicts efficiently the strength of adhesively bonded joints to a high degree of accuracy in comparison with the legacy method using FEA with a 93% savings in computational cost.

Keywords Adhesive joints · Convolutional neural network · Genetic algorithm · Composite adherend · Lightweight design

1 Introduction

Artificial neural networks (ANN) can be described as computational models comprised of interconnected layers of multiple processing units operating in parallel (Cichy and Kaiser 2019). They are comprised of an input and output layer and are referred to as deep when multiple layers are stacked between these layers. They can be trained to perform a certain task. During training, relationships between the processing units are established. The strength of these relationships is expressed numerically with values referred to as weights. These weights are, therefore, ‘learnt’ by the ANN algorithm during the training process. Deep ANN has an advantageous capability to learn feature representation from the availability of training data (Han et al. 2019).

There are different types of ANN ranging from perceptron to neural Turing machine as outlined by Tch (2017). These various types of ANN have advantages and disadvantages and have been implemented in various fields. Some of the

most used ANN includes Feedforward, Recurrent, and radial basis function among others.

Feedforward neural network, also known as multilayer perceptron (MLP), has been used to identify damages in planar truss structures (Truong et al. 2020). It was the first and most successful ANN algorithm which employed non-linearity in the identification of distinguishing features in the training data (Bengio et al. 2016). Other applications of feedforward neural networks include modelling of batteries (Dziechciaruk et al. 2020) and forecasting (Norouzi et al. 2019; Calvo-Pardo et al. 2020) among others.

Recurrent neural networks (RNN) are mostly used in tasks involving sequential data with varying input and output sizes (Bengio et al. 2016). It can be applied to varying sizes because of its parameter sharing property. This ANN type has been applied in the engineering field by Zhou et al. (2019) to study the impact load of non-linear structures. The result of this study showed a good performance/accuracy of the ANN. Other applications of RNN include speech recognition (Graves et al. 2013), forecasting and wind turbine power estimation (Medsker and Jain 1999) among others.

Another type of ANN is the radial basis function neural network. This method employs the use of the radial basis function as its activation function. Its application includes the prediction of the sound absorption coefficient of composite structures (Liang et al. 2020) and wind turbine power

Responsible Editor:: Shikui Chen

✉ Edore G. Arhore
edore.arhore@cranfield.ac.uk

¹ School of Aerospace, Transport and Manufacturing,
Cranfield MK43 0AL, UK

curve modelling (Karamichailidou et al. 2021). Other ANNs includes self-organising feature map more popular in speech recognition (Devi et al. 2020) applications and generative adversarial network demonstrated in structural topology optimisation (Li et al. 2020).

Convolutional Neural Network (CNN) can be described as the standard for image recognition (Traore et al. 2018). It is defined as a neural network that utilises convolution mathematical operation instead of matrix multiplication (Bengio et al. 2016). It has received wide attention in recent times due to extensive application in technical fields from physics, engineering, and automation to medicine, chemistry, and languages with multiple CNN architectures being proposed in the process (Sony et al. 2021). Applications of CNN range from image detection of potholes (Aparna et al. 2019), identification of pathogens (Traore et al. 2018), facial emotion recognition (Jain et al. 2019), handwriting recognition (Ptucha et al. 2019), and topology optimisation (Lin et al. 2018) among others. Despite its diversity, it has received little attention in the engineering field for structural optimisation tasks.

A study performed by Stoffel et al. (2020) compared CNN with feedforward and radial basis function neural network in the task of predicting structural deformation. The results showed that CNN offered the best performance in terms of accuracy. A study performed by Cao et al. (2018) resulted in a similar conclusion with the CNN extracting features better than the alternatives. However, some issues with CNN include the selection of key parameters. The accuracy and/or performance of CNN is dependent on these parameters which due to its size (depending on the depth of the CNN) could be difficult and expensive to optimise. These parameters are therefore selected manually or randomly with no guarantee of the accuracy of the CNN (Cao et al. 2018).

However, some studies have attempted to implement an optimisation process to improve the accuracy of the CNN. Results obtained from the studies performed by Lu (2021), Bakhshi et al. (2019), Yu and Zhang (2021), and Oh and Kim (2021) showed that genetic algorithms (GA) are capable of optimising CNN hyperparameters producing an optimum CNN with increased accuracy. Gonçalves et al. (2022) compared GA and particle swarm optimisation as an optimiser for CNN architecture with the result showing that the optimum CNN obtained from the GA optimisation process exhibited the highest accuracy. The study performed by Elskén et al. (2019) also showed GA as an effective tool for optimising the hyperparameters and neural network architecture.

A CNN architecture is comprised of an input and an output layer which has convolutional, activation function, pooling, and fully connected layers stacked in-between. Other layers such as dropout could also be included but these layers are task-orientated and therefore not compulsory. The

layers perform different tasks in the CNN architecture and depend on specific parameters. A summary description of some of the layers used in this study is described below.

Convolutional layers can ‘learn’ or identify features of an image through an extraction process (Rawat and Wang 2017). Most of the computational resources utilised by CNN are expensed on this layer (Sewak et al. 2018). The convolutional layer utilises sparse iteration, parameter sharing, and equivariant representation to improve the algorithm of the ANN. Matrix multiplication which requires every output to interact with every input is removed with sparse interaction. This is done by reducing the kernel size. These kernels allow for the detection of important features to be stored. As a result, the number of parameters to be stored is minimised reducing computational cost in terms of memory requirements. Parameter sharing results in only one parameter being used for every location. This means that, unlike other ANN algorithms in which members of the weight matrix are used once, just one set is learnt and used by the CNN. Finally, equivariant representation means that every change made to the input is translated to the output. These allow the capture, of features, such as edges in an input image. Some of the parameters utilised with this layer include the number of neurons to be connected to a region in the input (number of filters) and the size of the region on the input connected to the neuron (filter size) among others.

The activation function is used to decide which weights should be kept or penalised/removed. This layer can be used to improve the effectiveness of CNN for specific tasks when applied correctly (Gu et al. 2018). There are different types of activation functions which include rectified linear unit (ReLU), leaky ReLU, Maxout, and Probout among others. The ReLU is the most used activation function. It offers the best speed with deep CNN in comparison to the other types (Bengio et al. 2016) as well as induction of sparsity (Gu et al. 2018).

The pooling layer is usually included after an activation layer and is used to reduce the computational cost in CNN. This is achieved by decreasing the link to the convolutional layers (Gu et al. 2018). The pooling layer operates by replacing some positions on the output with a summary of the neighbouring positions (Bengio et al. 2016). There are many methods of pooling utilised with CNN including mixed pooling and stochastic pooling among others. The two most common methods are maximum and average pooling, which are classes of the mixed pooling method. These methods replace the positions with the maximum and average value (respectively) of the positions’ neighbourhood. The pooling layer also helps with making the representation equivariant thus allowing changes/translations in the input to be made in the output (Bengio et al. 2016).

A fully Connected Layer is used to interpret the features extracted from the previous convolutional and pooling layers

(Rawat and Wang 2017). That is, the fully connected layer uses the output of the different layers to compute a meaning for the input data. It multiplies its input with a weight matrix and adds a bias value to determine the meaning of each extracted feature.

Poor performance and high computational cost have been described as the reasons behind the reduced application of CNN in the field of structural engineering (Lee et al. 2018). However, the recent application of CNN in engineering has yielded some good results outlining its potential. Alqahtani et al. (2021) proposed a CNN which produced a good level of accuracy when deployed to classify fatigue crack damage in structures. Zhang et al. (2021) deployed CNN for damage localisation for plate-like structures. The results showed that CNN was able to localise damage without the need for modelling which could result in the minimisation of computational cost.

In this study, an ANN specifically CNN would be designed and applied in the field of structural engineering. The aim is to optimise a CNN architecture capable of predicting the strength of adhesively bonded joints subjected to tensile load. The optimisation process of these joints could be very computationally expensive especially since the numerical simulation of the finite element analysis (FEA) requires non-linear material models, progressive damage, and complex contact interactions. This investigation, therefore, performs a novel study on the feasibility of deploying a CNN in place of a non-linear FEA to predict the properties of an adhesively bonded joint in an optimisation process. The goal of this would be to minimise the computational cost of predicting joint strength without adversely affecting the accuracy of the optimisation process.

This paper would start by discussing the methodology used in carrying out this study explaining it in detail. The methodology would explain the optimisation process performed on the architecture of the CNN including an initial manual search done to minimise the design space. Next, the evolutionary optimisation method (genetic algorithm) used to search the design space is explained. This means that the CNN architecture will be optimised first using a search space method to minimise the design space before employing a search strategy method to optimise it. The performance of the CNN would be tested by deploying it to an optimisation process of adhesively bonded joints. The results of the study will be discussed along with the performance and accuracy of the proposed optimised CNN in comparison to an FEA benchmark and other CNN architectures. Finally, this paper would be concluded by discussing the feasibility of deploying ANN in the optimisation process of structures as well as the possible computational gain.

2 Methodology

This study aims to generate a CNN architecture that can assist with the optimisation of an adhesively bonded joint. Therefore, for comparison, an optimisation process, dubbed 'legacy optimisation process', was performed first to set the benchmark. This was done with the implementation of a genetic algorithm (GA) and finite element method (FEM) to obtain the global optimum adhesively bonded joint design. The objective function of the legacy optimisation process was set to maximise the specific strength (SRF) of the joint. This strength was measured as the force required to completely debond the joint when subjected to an applied load divided by the mass of the joint. The GA optimisation process was chosen because the previous study (Arhore et al. 2021) carried out compared different optimisation methods highlighting the GA method as the better performing method.

Figure 1 shows the flowchart of the legacy optimisation process performed by Arhore et al. (2021). In the optimisation process, a MATLAB program was created to utilise GA to select design variables within a defined design space which dictates the geometry of the outer adherend (Fig. 2a). An ABAQUS input file was generated autonomously by the MATLAB program. Using the generated design variables, a spline function was used to define and autonomously generate the geometry of the outer adherend for the input file. Figure 2a shows an illustration of an autonomously generated joint model outlining different components as well as the loading and boundary condition. This input file is

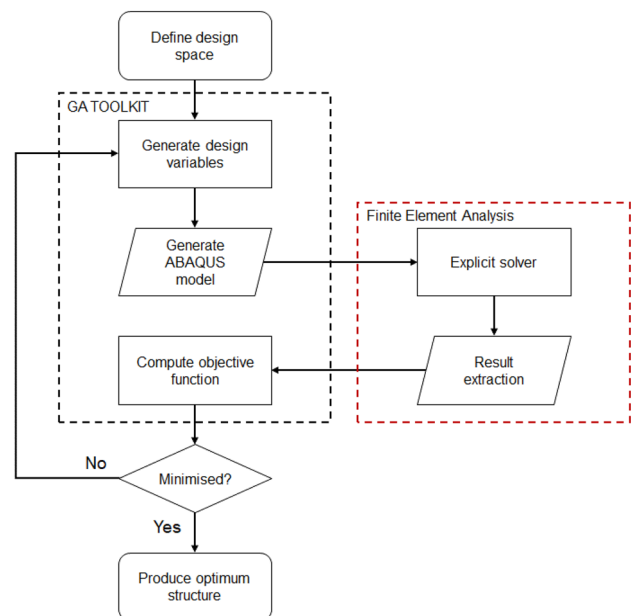


Fig. 1 Flowchart of the legacy optimisation process

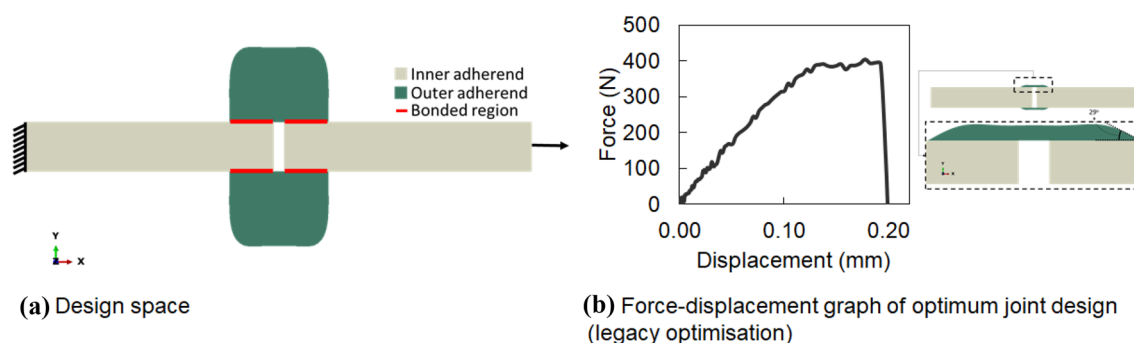


Fig. 2 Schematic illustration of adhesively bonded joint

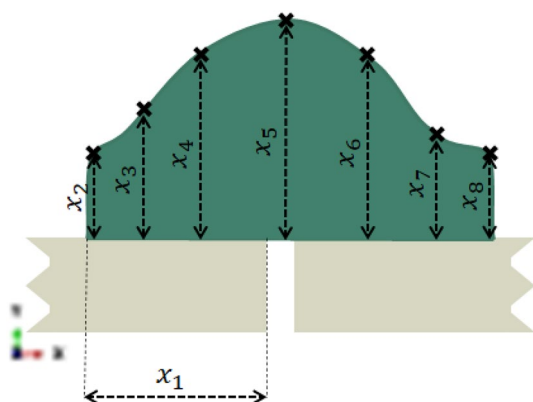


Fig. 3 Diagrammatic description of the design variables

then submitted to the ABAQUS explicit solver using the MATLAB system function and the results (force required for complete debonding and joint mass) of the analysis are extracted autonomously. The objective function (SRF) was computed for the design and the process was repeated until the optimum design was achieved. Details of the legacy optimisation process and the results can be read from work done by Arhore et al. (2021). Figure 2b shows the

force–displacement graph and the optimum joint geometry obtained from the legacy optimisation process.

Figure 3 shows a diagrammatic explanation of the design variables. The 2nd and 8th design variables defined the height of the left and right edges, respectively, as shown in the figure, while the other design variables were used to define y coordinate points within the overlap length defined by the 1st design variable. The overlap length was kept constant, that is, the bonded length between the inner adherend and outer adherend was constant on both sides of the joint. Therefore, the x coordinates of design variables 3–7 were selected by partitioning the total length of the outer adherend equally. A spline was used to connect the points defined by the design variables creating the outer adherend. The geometry of the inner adherend was kept constant.

2.1 Convolutional neural network data creation

To create the data required to train (train dataset), validate (dev dataset), and test (test dataset) the CNN, a new MATLAB program was created to randomly produce this data. Figure 4a shows an illustration of the tasks performed by the MATLAB program which generates random joint design models and outputs an image representation of the joints (Fig. 4b). To create

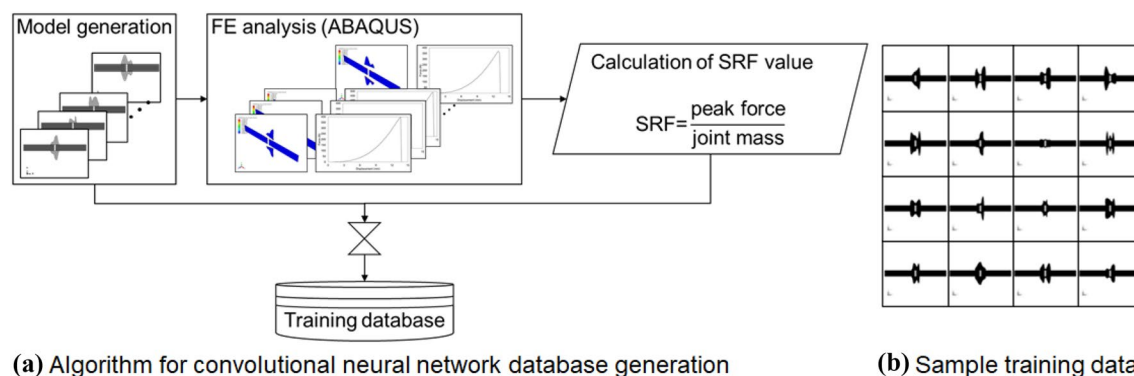


Fig. 4 Process flow for convolutional neural network database generation

the joint models, design variables (Fig. 3) which fit within the design space were randomly generated. For each joint model, a Finite element model (FEM) was autonomously generated and analysed using ABAQUS explicit solver. To accomplish this, a MATLAB program was created which included a script to generate an ABAQUS input file for a joint model based on the randomly generated design variables. The design variables were used to model the outer adherend as shown in Fig. 3. The input file was submitted to the ABAQUS explicit solver using the MATLAB system function, therefore, performing a FEA. The FEA produced an ABAQUS ODB (result) file which was accessed using the system function in MATLAB to extract the required data. The force required for the complete debonding of the joint and the mass of the joint were extracted from the results. This force was divided by the mass of the joint to produce the SRF value which was stored as the label of the image representation in the training database. This process was performed autonomously by the created MATLAB program requiring no user input. Each FEA required on average 1.5 min using 1 CPU. The overall training data creation process was performed in parallel using 16 CPUs and required on average 11 h for completion.

A total of 9000 unique joint designs were generated as the total CNN training data. The term unique is used to indicate that no joint design was repeated in the training data. This was done to prevent the possibility of overfitting. The train, dev, and test dataset were randomly selected from the training data by splitting them using the ratio 8:1:1, respectively. The data size of 9000 was selected because it was noticed for peak accuracy of the CNN to occur at approximately 7000 unique designs. Increasing the training data further offered little to no improvement to the CNN's accuracy. Figure 5 shows a graph which highlights the effect of the amount of data on the accuracy of the CNN. The accuracy was calculated as what percentage of the test designs were predicted by the CNN with an error which was less than 40% using Eq. 1. The architectures of the CNNs (CNN-1 and CNN-2) are shown in Figs. 10 and 11.

The CNN was trained to predict this SRF value for joint designs using its image representation. The training database included the images randomly generated as shown in Fig. 4b and the calculated SRF values as their labels which were regularised between the values of 1 and 10 to prevent overfitting during the training process. The known optimum joint design from the legacy optimisation process was excluded from the CNN training data to prevent bias during the deployment of the trained CNN.

2.2 Convolutional neural network architecture optimisation (manual search)

A manual search optimisation process was performed on the architecture of the CNN using multiple architectures with

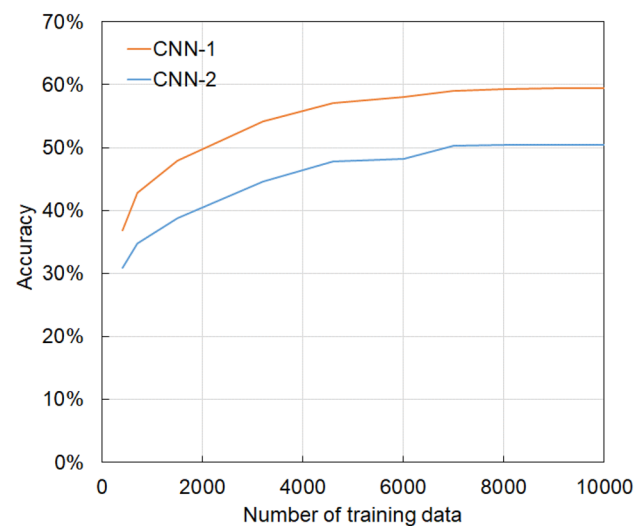


Fig. 5 Effect of number of training data on CNN accuracy

depths ranging from 26 to 123 layers. The increment of layers was performed in series to allow for the inclusion of modified parameters from the previous architectures to the future ones. The purpose of this task is to reduce the vast architecture design space to a smaller/more manageable set of variables and constraints. It served as a search method for the method of optimising CNN architecture as outlined by Elsken et al. (2019). The ideal architectures obtained from this process were used as guides to limit the architecture design space and significantly reduce the computational cost for the optimisation of the CNN architecture.

Figure 6a shows the process used in training, deploying, and retraining the CNN architectures. The generated train dataset was used to train the selected CNN architecture, while the dev dataset was used to validate the CNN during the training process. This allowed for an estimation of the CNN accuracy during the training process with data distinctively different from the train dataset. The test dataset was used to determine the accuracy of the CNN once training was completed, this allowed for the analysis of the CNN performance outside the train and dev dataset.

After training, the trained CNN was then deployed to perform a structural optimisation process identical to the legacy optimisation process (Fig. 1) with the FEA section replaced with CNN prediction as shown in Fig. 6b. For this process, different joint models were generated within the predefined design space using GA as performed by the legacy optimisation process (Arhore et al. 2021). However, unlike in the case of the legacy optimisation process, an image representation of the generated joints such as the illustrations shown in Fig. 4b was autonomously produced and sent to the trained CNN for the prediction of its SRF value which the optimiser was set to maximise. The structural optimisation

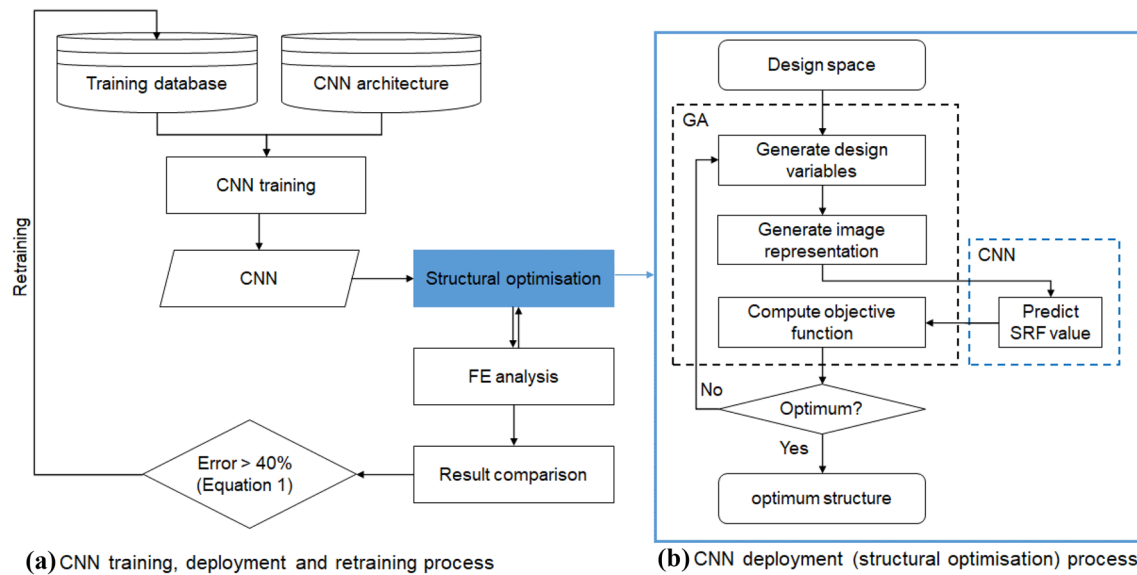


Fig. 6 Convolutional neural network manual search optimisation process. * **b** Illustrates a flowchart of the optimisation process performed on adhesively bonded joints using the trained CNN from (a)

process is concluded when the optimum joint design has been obtained. The use of the CNN in place of the FEA is aimed at minimising the overall computational cost by removing the computationally expensive FEA segment from the legacy optimisation process. This also provided a process for estimating the performance of the CNN outside of the created data thus verifying its viability.

For each CNN depth, modifications were made to the hyperparameters to better improve the architecture based on its performance. These modifications included longer training (higher number of epochs) in the case of underfitting among others. Other hyperparameters that were modified include the mini-batch size, the inclusion of dropout and its regularisation technique, learning rate, and the solver (stochastic gradient with momentum or Adam optimiser). For individual convolutional and pooling layers, the number of filters, filter size, and stride were also modified to improve the performance of the CNN.

The availability/distribution of training data also affects the performance of the CNN, therefore, increasing the number and/or using a better distribution of training data could improve the performance of the CNN. Therefore, it was necessary to consider the effect of the training data on the accuracy of the trained CNN. As such, during the deployment of the trained CNN, FEA was performed in parallel for the generated joint models. The predicted SRF values from the CNN were thus compared with the actual FEM values. The joint designs in which the predicted SRF values exhibited an error difference greater than 40% in comparison

with the actual FEM values were added to the training data. The CNN was then retrained and redeployed to perform a new structural optimisation process. Equation 1 describes how the error was calculated, FEM represents the SRF value obtained from the FEA while CNN represents the CNN predicted SRF value. However, only joint designs which were not already included in the training data were added to ensure they remained unique. The optimum joint design obtained from the legacy optimisation method was also permanently excluded from the training data.

$$\frac{|FEM - CNN|}{FEM} \times 100 \quad (1)$$

2.3 Genetic algorithm optimisation of convolutional neural network architecture

With the determined number of layers from the manual search, an optimisation process of the CNN architecture was performed using the GA toolkit available from MATLAB. The objective function was set to maximise the accuracy of the CNN. The accuracy was defined as how well the CNN predicted the SRF values of the test dataset and was calculated using root mean square error (RMSE) in Eq. 2, comparing the CNN values with the SRF values obtained from the parallel performed FEA. The design variables generated from the GA were set to modify the number of layers and CNN architecture (location of each layer). It was also set to modify the parameters of each layer.

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (FEM - CNN)^2}{N}} \quad (2)$$

An illustration of architecture modification by the GA is shown in Fig. 7. The image input, fully connected and output layers were kept constant during the optimisation process. The optimisation process was therefore set to determine the number and position of the convolutional, pooling, and dropout layers in the architecture. The design variable was also used to set the parameters of these layers as shown in the figure. For the convolutional layer, the parameters represented in the brackets are the filter size, stride, and the number of filters, respectively. The filter size and stride were the parameters modified in the pooling layers while the modified parameter of the dropout layer was the probability to drop outputs from the previous layer. A constraint was added to place a batch normalisation and a ReLU (activation function) layer between convolutional and pooling layers. The objective function was set to penalise architectures that required a training time greater than 1 h using 16 CPUs.

3 Results and discussion

3.1 CNN architecture optimisation (manual search)

The results for the CNN architecture optimisation using manual search are given in Fig. 8. In all the runs, the target value (dashed line) is the optimum SRF value obtained from the legacy optimisation process and is used as a benchmark to see how well the trained CNN architecture deployed in the structural optimisation (see Fig. 6) performs. Each of the charts on the left compares the SRF values of the optimum

joint design that the CNN deployed structural optimisation produced with the corresponding FEM value acquired from a FEA performed on ABAQUS. Once the structural optimisation run is complete, the CNN is retrained with additional data and a new structural optimisation run is conducted. The plots on the right side of Fig. 8 show the predicted SRF value by the CNN for one optimisation run, ordered in ascending order. The equivalent SRF value obtained from the FEA is also plotted to show a visual indication of the CNN accuracy in predicting the joint strength.

The accuracy of the trained CNN when deployed was calculated using RMSE as illustrated in Eq. 2, where FEM and CNN represent the actual SRF value obtained from the FEA and the predicted SRF value by the CNN, respectively, while N indicates the number of joint models analysed. RMSE shows the distance between the actual and predicted value, therefore the smaller this value is, the higher the accuracy. The RMSE value for 10 different deployments of the trained 26 layers CNN architecture (Fig. 12) is also shown in Fig. 8a. As stated earlier, the CNN was retrained after every deployment. This was done to consider the additional training data from the poorly predicted joint design from previous deployments in subsequent ones. This allowed for the analysis of the effect the training data had on the accuracy of the CNN architecture.

The results showed that increasing the training dataset could also adversely affect the accuracy of the CNN. This negative effect was mostly noticed when the training data were unevenly distributed due to the additional data from previous deployments. This phenomenon was usually corrected during the next deployment with the additional data more likely to even out the data distribution. However, despite the increasing training dataset, it was noticed that this architecture failed at capturing certain features of the

Fig. 7 Comparison of design variables in two different CNN architectures

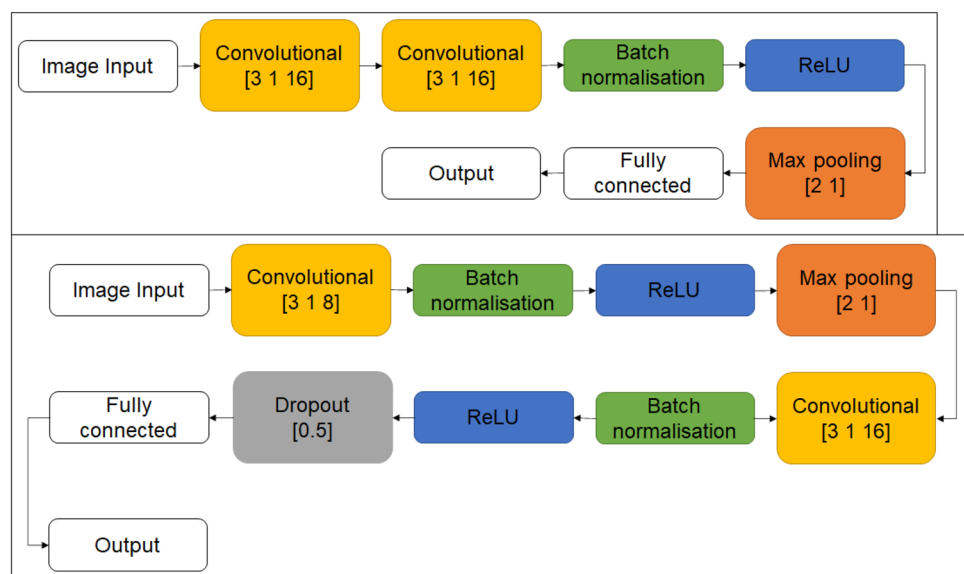
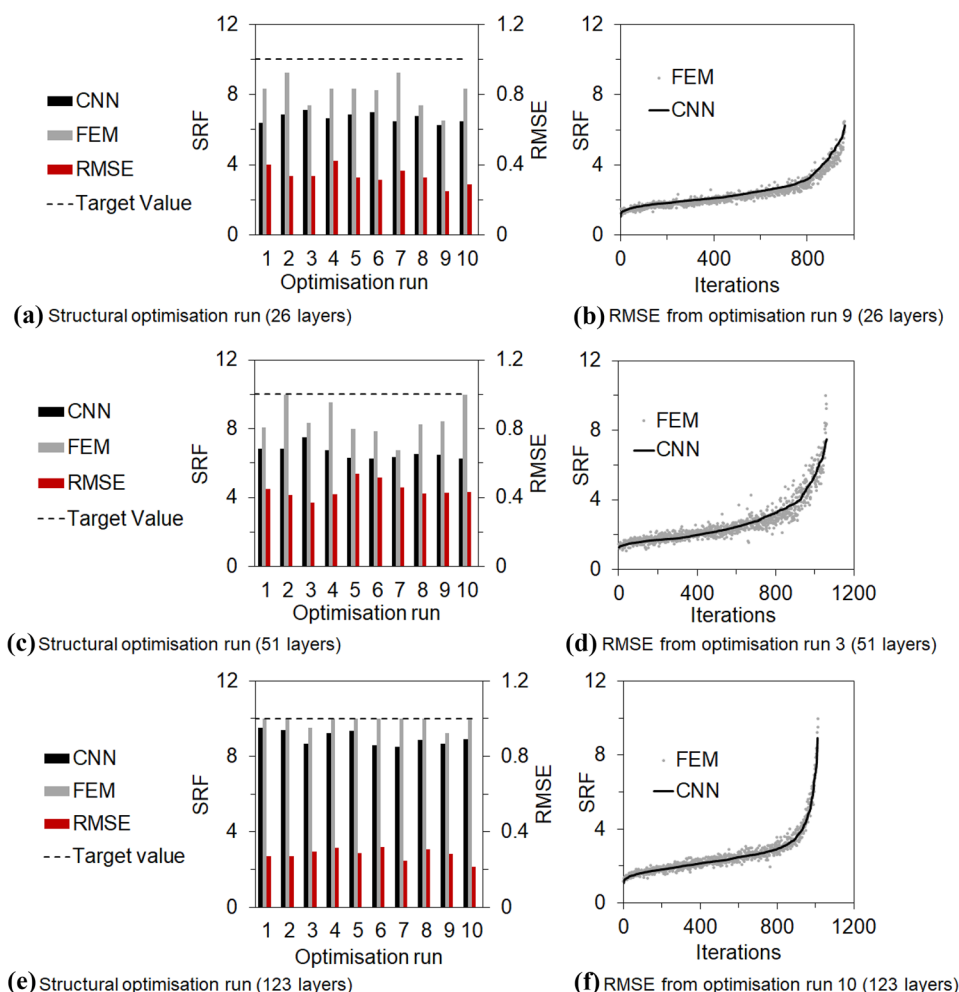


Fig. 8 Performance of convolutional neural network architectures. * **a** Shows the objective value for the optimum designs obtained from the structural optimisation process performed using the trained CNN and compares them to the objective function obtained from the legacy optimisation process



joint designs such as the overlap length and edge angles. These features are important in determining the strength of adhesively bonded joints. Therefore, this architecture produced a high error during training with the dev dataset, during testing with the test dataset and when deployed. This result suggests that a bigger/deeper CNN was required, the high bias in the performance of the CNN could also be addressed by the modification of the CNN architecture.

Figure 8b shows the data plot from the 9th structural optimisation run which produced the best RMSE value (Eq. 2) for an architecture depth of 21 layers. The figure contains the FEM SRF value of every joint design generated during the deployment along with the corresponding predicted CNN SRF values. The data are sorted by increasing CNN SRF values for better readability. Sorting the data this way allows for the CNN SRF values to appear as a trend line showing how well it fits the FEM values. As stated earlier, RMSE was used to determine the accuracy of the CNN by calculating how close the FEM values are to their corresponding CNN values, therefore, the smaller the RMSE value, the higher the accuracy.

To improve the accuracy of the CNN, a deeper architecture was implemented to reduce the bias. CNN architectures with 51 layers deployed are shown in Fig. 8c. It is worth noting that the training data were reset to the original dataset. The results showed an improvement in the CNN with runs 2 and 10, achieving the correct optimum joint design. However, it is still evident, the CNN has a poor prediction of the joint SRF compared with FEA. A significant difference with the 26 layers architectures was its capability at detecting joint designs with high SRF properties. This explains its higher probability of generating the global optimum joint design when deployed.

However, the 51 layers of CNN architecture exhibited high errors in its predictions when deployed. It also exhibited high errors in predicting the dev and test dataset, although, this error was on average approximately 1.5 times lower than the error obtained when deployed. For this architecture, the additional training data did not improve the accuracy of the CNN. The addition was noticed instead to result in overfitting with the CNN accuracy only increasing for joint models with certain distinct features. Figure 8d

illustrates the CNN accuracy on optimisation run 3 which produced the best RMSE value. The architecture of the best performing 51 layers CNN is shown in Fig. 13.

To account for the high variance/overfitting, the architecture of the CNN was modified further with an increased depth to reduce bias. Increasing the number of layers to 123 resulted in improved accuracy of the CNN, as is shown in Fig. 8e. The results showed that the 123 layers produced a lower average RMSE when compared to the other architectures, with an example shown in Fig. 8f. The increasing training dataset was noticed to offer some improvement to the overall accuracy of the network. This architecture reduced the high bias/underfitting problem experienced with the smaller architectures. Overall, the architectures which exhibited a high number of filters in the convolutional layers produced the best accuracy. This feature was noticed to improve the accuracy of the CNN by minimising overfitting. Importantly, this CNN architecture did not consistently produce the global optimum joint design, from the legacy optimisation process, as its optimum design for every deployment (Fig. 8e). On the occasions, it did not produce the global optimum design, it produced a variant of the design exhibiting the same geometric profile but a different length for the bonded region. Figure 14 shows the architecture of the best performing 123 layers of CNN.

3.2 CNN architecture optimisation (genetic algorithm)

A GA optimisation process was performed to investigate whether an optimum CNN architecture for the task of predicting SRF values of adhesively bonded joint designs can be produced. For this optimisation process, the objective function was set to maximise the accuracy of the CNN (minimise RMSE) in predicting the test dataset. The design variables were set to define the CNN architecture by modifying

the layer position and parameters. To minimise the computational cost, the lower and upper bound were set using the results obtained from the manual optimisation process. The minimum number of layers was set to 50 while the maximum number of layers was set to 150. This is because, for architectures with more than 150 layers, the adverse effect of the vanishing gradient problem was detected during the manual search.

The optimum CNN architecture generated by the GA optimisation process (Fig. 15) is composed of 71 layers. Similarly, with the results obtained from the manual optimisation process, the GA optimiser was noticed to converge towards an increasing number of filters in the convolutional layers. Due to a maximum training time constraint set for the GA optimisation process, the optimiser was noticed to reduce the number of layers while increasing the filter size. The minimum number of filters in the optimum architecture is 64. This ultimately generated a CNN architecture that was smaller than the 123 layers described in Sect. 3.1 but required a higher computational cost in terms of training time. This architecture produces RMSE values which were less than 0.2 every time it was deployed. Figure 9b shows the accuracy of the optimum architecture from the tenth (most accurate) optimisation run.

Table 1 compares the accuracy of the optimum CNN (in bold) with conventional CNNs and the other layers described in Sect. 3.1. The rows in italics indicate CNNs and FEA created in this study. The results show a higher accuracy of the optimum CNN in comparison to other architectures. The ResNet-18 (MathWorks 2018) CNN architecture which was composed of a similar number of layers also produced a high level of accuracy. However, the ResNet-18 CNN architecture requires 83% less computational cost than the optimum CNN despite their similarities. This computational gain can be attributed to the architecture of ResNet-18 which uses a residual learning framework (He et al. 2016). The

Fig. 9 Performance of optimum convolutional neural network architecture. * **a** Shows the objective value for the optimum designs obtained from the structural optimisation process performed using the trained CNN and compares them to the objective function obtained from the legacy optimisation process

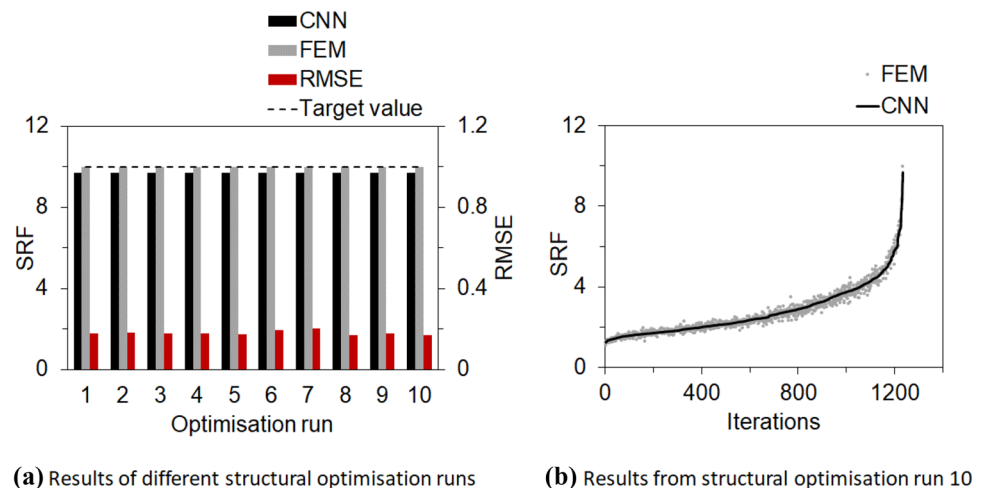


Table 1 Comparison of convolutional neural networks accuracy and computational cost

Neural network	Number of layers	Training time (hours)	RMSE (test data)	RMSE (optimisation data)	Runtime (hours)
Optimum	71	0.88	0.1428	0.1808	0.91
26 layers	26	0.03	0.3006	0.4227	0.69
51 layers	51	0.42	0.3014	0.4450	0.72
123 layers	123	0.80	0.1800	0.2807	1.10
AlexNet (MathWorks 2017)	24	0.05	0.5875	1.0212	1.32
VGG-16 (MathWorks 2017)	40	0.64	0.9295	1.4234	1.41
VGG-19 (MathWorks 2017)	46	0.75	0.9331	1.4283	1.62
ResNet-18 (MathWorks 2018)	70	0.15	0.1430	0.2108	1.03
ResNet-50 (MathWorks 2017)	176	0.35	0.2522	0.3806	1.22
ResNet-101 (MathWorks 2017)	346	0.63	0.4121	0.4977	2.64
GoogLeNet (MathWorks 2017)	143	0.17	0.8279	1.3454	1.13
ABAQUS FEM	–	–	–	–	15

possibility of generating architectures with this feature was not included in the optimisation process. Therefore, it could be assumed that the accuracy of the optimum CNN could be further improved while minimising the computational cost. This could be accomplished by modifying the design variables of the optimisation process to generate more complex architectures. As stated earlier, architecture with more than 150 layers exhibited higher RMSE values as shown in the table, indicating an upper limit of layers above which no gains in accuracy are made. These higher errors were noticed to result from a vanishing gradients problem attributed to the depth of the CNN resulting in a training process generating poor weights.

Table 1 also compares the run time for different architectures when deployed to perform the optimisation of the adhesively bonded joint with the legacy method. The optimum CNN consistently performed the optimisation process in under an hour resulting in a 93% saving in computational cost in comparison with the legacy method (ABAQUS FEM). On average, the replacement of the FEM analysis with the CNN prediction resulted in a 92% drop in the computational cost. As stated earlier, training data did not include the optimum joint design. However, the optimum architecture was able to consistently identify and produce it as the optimum design. This showcases the possibility of identifying the properties of novel structures with the use of CNN without performing FEA.

4 Conclusion

This study aimed at improving the optimisation process of structures by the minimisation of computational cost. To achieve this, a novel optimisation process was proposed that employed an artificial neural network (specifically

convolutional neural networks) to replace the computationally expensive finite element analysis (FEA). Several different convolutional neural network (CNN) architectures were analysed and discussed including CNN architecture that was optimised using a genetic algorithm (GA) to perform the required task of predicting the strength of adhesively bonded joints. The GA optimised CNN architecture is composed of 71 layers. The optimisation process also showed that for this task, a minimum of 64 filters was required in every convolutional layer for improved accuracy. Although this feature increased the computational cost in terms of the training cost, the results showed that it is feasible to perform structural optimisation by deploying ANN to replace the computational expensive FEA in the process and achieve up to 92% faster optimisation as a result.

In comparison with conventional CNN architectures, the optimum CNN produced the highest overall accuracy for the specified task. However, it required the highest computational cost in terms of training time. Training aside, when deployed, the different architectures required similar computational time to achieve optimum structure design. The results obtained from the study show the importance of CNN architecture to perform a specific task for improved accuracy. The study also shows that other architectures could be tested and used as a guide to minimise the search space for the optimum architecture. This would allow for a better implementation of parameters for each layer to guarantee improved accuracy of the CNN.

The use of a genetic algorithm (GA) as the optimisation process proved to be very effective. This process allowed for flexible modifications of the CNN's architecture and thus a more thorough analysis of the search space. It allowed for more strategic placement of individual layers as well as the modification of the parameters affecting these layers. These modifications were performed in parallel along

with modifications to the hyperparameters of the CNN. This allowed for the discovery of some interesting CNN architectures that proved to be significantly more accurate than manually designed schemes.

For future studies, the GA optimisation process of the CNN architecture could be modified to account for residual learning frameworks. Although this modification would result in an increased complexity to the GA optimisation process, it could produce an optimum CNN architecture with a reduced training time. However, as stated earlier, when deployed, the CNNs performed similarly in terms of computational cost. The CNN could also be deployed to perform the optimisation process on more complex structures.

Appendix 1: Convolutional neural network architectures

See Figs. 10, 11, 12, 13, 14, 15.

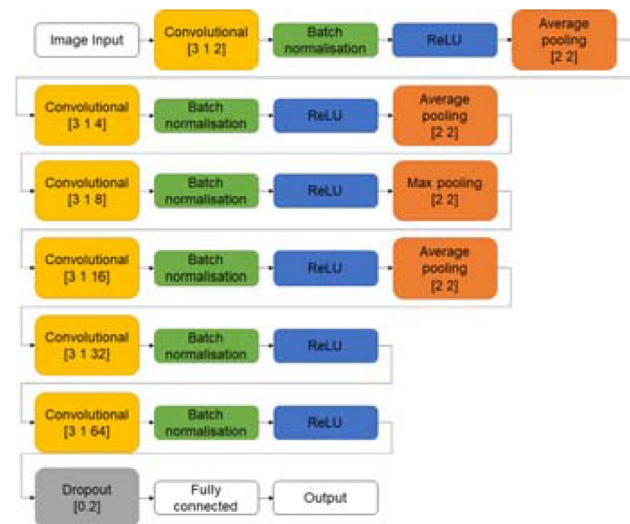


Fig. 10 CNN-1 architecture

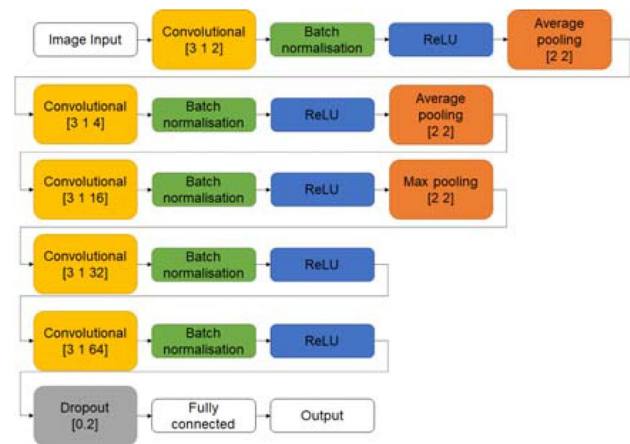


Fig. 11 CNN-2 architecture

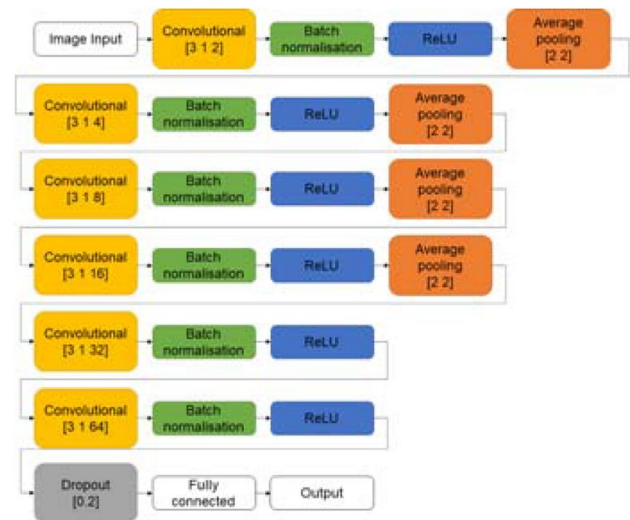


Fig. 12 Best performing 26 layers architecture

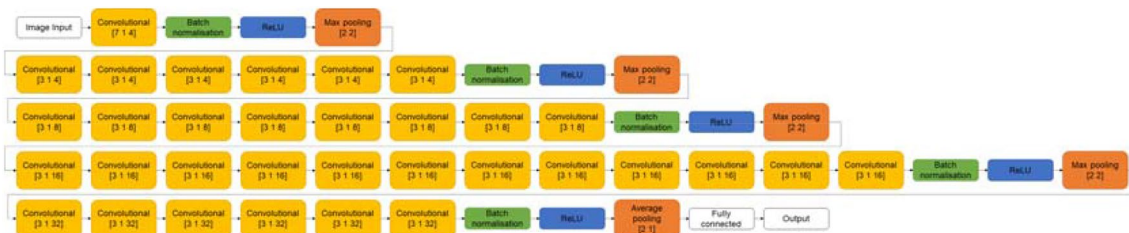


Fig. 13 Best performing 51 layers architecture



Fig. 14 Best performing 123 layers of architecture



Fig. 15 Optimum CNN architecture

Appendix 2: Main code sample

```
% Main Code
clc
clear
fclose all;

% Create a Folder to Store the Generated Images
mkdir(Folder-Name);

% Define a Global Variable to Count the Number of Iterations
setGlobalnIteration(1);

% The genetic algorithm
objFcn = % @Objective function
nvars = % Number of variables

lb = % Lower Bound
ub = % Upper Bound

LB = zeros(1,nvars);
UB = zeros(1,nvars);
LB(1,:) = lb;
UB(1,:) = ub;

% Enforce only integers for design variables
IntCon = 1:nvars;

% Perform the GA Optimisation: Sample Code Line
[x, fval] = ga(objFcn, nvars, [], [], [], [], LB, UB, [], IntCon);
```

Appendix 3: Objective function sample code

```
function [objective] = objectiveFunction(Design_Variable)

%% Get the number of Iteration
nIteration = getGlobalnIteration;

%% Create a Python Script to Generate the Joint ABAQUS Model and
generate Image
% Sample: JointModel = fopen('JointModel.py', 'w'); % To open a script
WriteScriptUsing(Design_Variable);

%% Predict the Joint Model Properties Using the CNN (CNN_Predictor)
% Resize if required: Image = imresize(Image, [64 64]);
load ('CNNFile.mat', 'CNN_Predictor');
objective = predict(CNN_Predictor, Image);

%% Compare results with FEM Simulation
% Run Finite Element Analysis on ABAQUS if Required
% Sample: system('abaqus job=JointJob.inp interactive cpus=14');
objectiveABAQUS = % extractResultsFileABAQUS_ODB_File;

%% Store Data and Values
StoreIterationData(x, objective, objectiveABAQUS);

%% Increase the Iteration Count
setGlobalnIteration(nIteration+1);

%% Return Inverse to the Optimiser if Maximising
objective = 1/objective;
```

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Replication of results Information has been provided throughout the paper on replicating the results which include the material properties and convolutional neural network architecture used. Sample codes are provided in the Appendices (2, 3) to illustrate the deployment phase (Fig. 6b). Access to the fully trained CNN as well as sample testing data has been provided here.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Alqahtani H, Bharadwaj S, Ray A (2021) Classification of fatigue crack damage in polycrystalline alloy structures using convolutional neural networks. *Eng Fail Anal* 119:104908. <https://doi.org/10.1016/j.engfailanal.2020.104908>
- Aparna, Bhatia Y, Rai R, Gupta V, Aggarwal N, Akula A (2019) Convolutional neural networks based potholes detection using thermal imaging. *J King Saud Univ Comput Inf Sci*. <https://doi.org/10.1016/j.jksuci.2019.02.004>
- Arhore EG, Yasae M, Dayyani I (2021) Comparison of GA and topology optimization of adherend for adhesively bonded metal composite joints. *Int J Solids Struct* 226–227:111078. <https://doi.org/10.1016/j.ijsolstr.2021.111078>
- Bakhshi A, Noman N, Chen Z, Zamani M, Chalup S (2019) Fast automatic optimisation of CNN architectures for image classification using genetic algorithm. In: 2019 IEEE congress on evolutionary computation. IEEE, pp 1283–1290. <https://doi.org/10.1109/CEC.2019.8790197>
- Bengio Y, Goodfellow I, Courville A (2016) Deep learning. MIT Press, Cambridge
- Calvo-Pardo H, Mancini T, Olmo J (2020) Granger causality detection in high-dimensional systems using feedforward neural networks. *Int J Forecast*. <https://doi.org/10.1016/j.ijforecast.2020.10.004>

- Cao W, Wang X, Ming Z, Gao J (2018) A review on neural networks with random weights. *Neurocomputing* 275:278–287. <https://doi.org/10.1016/j.neucom.2017.08.040>
- Cichy RM, Kaiser D (2019) Deep neural networks as scientific models. *Trends Cogn Sci* 23:305–317. <https://doi.org/10.1016/j.tics.2019.01.009>
- Devi KJ, Singh NH, Thongam K (2020) Automatic speaker recognition from speech signals using self organizing feature map and hybrid neural network. *Microprocess Microsyst* 79:103264. <https://doi.org/10.1016/j.micpro.2020.103264>
- Dziechciaruk G, Michalczyk M, Ufnalski B, Grzesiak LM (2020) Dynamic model of a lithium-ion cell using an artificial feed-forward neural network with dynamical signal preprocessing. *J Energy Storage* 31:101503. <https://doi.org/10.1016/j.est.2020.101503>
- Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: a survey. *J Mach Learn Res*. <https://doi.org/10.48550/arXiv.1808.05377>
- Gonçalves CB, Souza JR, Fernandes H (2022) CNN architecture optimization using bio-inspired algorithms for breast cancer detection in infrared images. *Comput Biol Med* 142:105205. <https://doi.org/10.1016/j.compbiomed.2021.105205>
- Graves A, Mohamed A, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech, & signal processing. IEEE, pp 6645–6649. <https://doi.org/10.1109/ICASSP.2013.6638947>
- Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liub T, Wang X, Wang G, Cai J, Chen T (2018) Recent advances in convolutional neural networks. *Pattern Recognit* 77:354–377. <https://doi.org/10.1016/j.patcog.2017.10.013>
- Han H, Li Y, Zhu X (2019) Convolutional neural network learning for generic data classification. *Inf Sci (Ny)* 477:448–465. <https://doi.org/10.1016/j.ins.2018.10.053>
- He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition, vol 2016-December. IEEE, pp 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- Jain DK, Shamsolmoali P, Sehdev P (2019) Extended deep neural network for facial emotion recognition. *Pattern Recognit Lett* 120:69–74. <https://doi.org/10.1016/j.patrec.2019.01.008>
- Karamichailidou D, Kaloutsas V, Alexandridis A (2021) Wind turbine power curve modeling using radial basis function neural networks and Tabu search. *Renew Energy* 163:2137–2152. <https://doi.org/10.1016/j.renene.2020.10.020>
- Lee S, Ha J, Zokhirova M, Moon H, Lee J (2018) Background information of deep learning for structural engineering. *Arch Comput Methods Eng* 25:121–129. <https://doi.org/10.1007/s11831-017-9237-0>
- Li M, Jia G, Cheng Z, Shi Z (2020) Generative adversarial network guided topology optimization of periodic structures via subset simulation. *Compos Struct*. <https://doi.org/10.1016/j.compstruct.2020.113254>
- Liang L, Guo W, Zhang Y, Zhang W, Li L, Xing X (2020) Radial basis function neural network for prediction of medium-frequency sound absorption coefficient of composite structure open-cell aluminum foam. *Appl Acoust* 170:107505. <https://doi.org/10.1016/j.apacoust.2020.107505>
- Lin Q, Hong J, Liu Z, Li B, Wang J (2018) Investigation into the topology optimization for conductive heat transfer based on deep learning approach. *Int Commun Heat Mass Transf* 97:103–109. <https://doi.org/10.1016/j.icheatmasstransfer.2018.07.001>
- Lu T-C (2021) CNN convolutional layer optimisation based on quantum evolutionary algorithm. *Connect Sci* 33:482–494. <https://doi.org/10.1080/09540091.2020.1841111>
- MathWorks (2018) ResNet-18 convolutional neural network—MATLAB resnet18—MathWorks United Kingdom. <https://uk.mathworks.com/help/deeplearning/ref/resnet18.html>. Accessed 3 Mar 2021
- MathWorks (2017) ResNet-50 convolutional neural network—MATLAB resnet50—MathWorks United Kingdom. <https://uk.mathworks.com/help/deeplearning/ref/resnet50.html>. Accessed 3 Mar 2021
- MathWorks (2017) VGG-19 convolutional neural network—MATLAB vgg19—MathWorks United Kingdom. <https://uk.mathworks.com/help/deeplearning/ref/vgg19.html>. Accessed 3 Mar 2021
- MathWorks (2017) VGG-16 convolutional neural network—MATLAB vgg16—MathWorks United Kingdom. <https://uk.mathworks.com/help/deeplearning/ref/vgg16.html>. Accessed 3 Mar 2021
- MathWorks (2017) AlexNet convolutional neural network—MATLAB AlexNet—MathWorks United Kingdom. https://uk.mathworks.com/help/deeplearning/ref/alexnet.html?s_tid=srchtitle. Accessed 3 Mar 2021
- MathWorks (2017) GoogLeNet convolutional neural network—MATLAB GoogLeNet—MathWorks United Kingdom. <https://uk.mathworks.com/help/deeplearning/ref/googlenet.html>. Accessed 3 Mar 2021
- MathWorks (2017) ResNet-101 convolutional neural network—MATLAB resnet101—MathWorks United Kingdom. <https://uk.mathworks.com/help/deeplearning/ref/resnet101.html>. Accessed 3 Mar 2021
- Medsker L, Jain LC (1999) Recurrent neural networks design and applications, 1st edn. CRC Press, Boca Raton. <https://doi.org/10.1201/9781003040620>
- Norouzi R, Kosari A, Sabour MH (2019) Real time estimation of impaired aircraft flight envelope using feedforward neural networks. *Aerosp Sci Technol* 90:434–451. <https://doi.org/10.1016/j.ast.2019.04.048>
- Oh BK, Kim J (2021) Optimal architecture of a convolutional neural network to estimate structural responses for safety evaluation of the structures. *Measurement* 177:109313. <https://doi.org/10.1016/j.measurement.2021.109313>
- Ptucha R, Petroski Such F, Pillai S, Brockler F, Singh V, Hutkowski P (2019) Intelligent character recognition using fully convolutional neural networks. *Pattern Recognit* 88:604–613. <https://doi.org/10.1016/j.patcog.2018.12.017>
- Rawat W, Wang Z (2017) Deep convolutional neural networks for image classification: a comprehensive review. *Neural Comput* 29:2352–2449. https://doi.org/10.1162/neco_a_00990
- Sewak M, Karim R, Pujari P (2018) Practical convolutional neural networks: implement advanced deep learning models using Python. Packt Publishing, Birmingham, Mumbai
- Sony S, Dunphy K, Sadhu A, Capretz M (2021) A systematic review of convolutional neural network-based structural condition assessment techniques. *Eng Struct* 226:111347. <https://doi.org/10.1016/j.engstruct.2020.111347>
- Stoffel M, Gulakala R, Bamer F, Markert B (2020) Artificial neural networks in structural dynamics: a new modular radial basis function approach vs. convolutional and feedforward topologies. *Comput Methods Appl Mech Eng* 364:112989. <https://doi.org/10.1016/j.cma.2020.112989>
- Tch A (2017) The mostly complete chart of Neural Networks, explained. Toward Data Science. <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>
- Traore BB, Kamsu-Foguem B, Tangara F (2018) Deep convolution neural network for image recognition. *Ecol Inform* 48:257–268
- Truong TT, Dinh-Cong D, Lee J, Nguyen-Thoi T (2020) An effective deep feedforward neural networks (DFNN) method for damage identification of truss structures using noisy incomplete modal data. *J Build Eng* 30:101244. <https://doi.org/10.1016/j.jobbe.2020.101244>

- Yu Y, Zhang M (2021) Control chart recognition based on the parallel model of CNN and LSTM with GA optimization. *Expert Syst Appl* 185:115689. <https://doi.org/10.1016/j.eswa.2021.115689>
- Zhang S, Li CM, Ye W (2021) Damage localization in plate-like structures using time-varying feature and one-dimensional convolutional neural network. *Mech Syst Signal Process* 147:107107. <https://doi.org/10.1016/j.ymssp.2020.107107>
- Zhou J, Dong L, Guan W, Yan J (2019) Impact load identification of nonlinear structures using deep Recurrent Neural Network. *Mech Syst Signal Process* 133:106292. <https://doi.org/10.1016/j.ymssp.2019.106292>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



DSSPACE

<https://dspace.org/>

Optimisation of convolutional neural network architecture using genetic algorithm for the prediction of adhesively bonded joint strength

Arhore, Edore G.; Yasaee, Mehdi; Dayyani, Iman

2022-09-02

Springer

<https://doi.org/10.1007/s00158-022-03359-x>

Arhore EG, Yasaee M, Dayyani I. (2022) Optimisation of convolutional neural network architecture using genetic algorithm for the prediction of adhesively bonded joint strength. Structural and Multidisciplinary Optimization, Issue 65, September 2022, Article number 256