A Novel Self-Updating Design Method for Complex 3D Structures Using Combined Convolutional Neuron and Deep Convolutional Generative Adversarial Networks

Zewen Gu, Xiaonan Hou,* Mohamed Saafi, and Jianqiao Ye

Mechanical design is one of the essential disciplines in engineering applications, while inspirations of design ideas highly depend on the ability and prior knowledge of engineers or designers. With the rapid development of machine learning (ML) techniques, artificial intelligence (AI)-based design methods are promising tools for the design of advanced engineering systems. So far, there have been some studies of 2D patterns and structural designs based on ML techniques. However, a particular challenge remains in allowing complex 3D mechanical designs using ML techniques. Herein, a novel and experience-free method to equip ML models with 3D design capabilities by combining a convolutional neuron network (CNN) with a deep convolutional generative adversarial network (DCCGAN) is developed. The model directly receives 2D image-based training data that define the complex 3D structures of a specific machine part. After the training process, an infinite number of new 3D designs can be generated by the proposed model, with their geometric and mechanical properties being accurately predicted at the same time. Moreover, the generated new designs can be fed back to expand the original input datasets for further ML model training and updating.

1. Introduction

Engineers have been the creators of mechanical designs for over centuries. Two essences of mechanical designs are creativity and novelty, which are deemed human “privilege.” Standard automatic design tools have been developed to aid design engineers to improve design quality but seldom bring creativity and novelty into a design.[1] Hence, engineering designs are highly experience-related, while the designer’s ability and imagination can also restrain the variety of designs. This is particularly true in designing mechanical parts of complex structures, for instance, complex 3D helical structures. Many novel design concepts were only proposed by researchers after being inspired by objects in nature, for example, a DNA-inspired helical structure was found showing a high designability.[2] It confirmed that human’s ability and efficiency of exploiting the feature space of a design problem are still limited. To conduct engineering structural design more effectively, researchers have been working on formulating the relationship between design form (i.e., parametric representation of geometry) and design behaviors (i.e., intended purpose of designs) by specific functions.[3] However, this has introduced significant difficulties to designers when a design problem is extended from 2D to 3D, from a simple to a complex geometry, and from linear to nonlinear. As a result, some of the 3D and nonlinear features would have to be neglected to reduce the mathematical complexity of the design procedure, and the efficiency of the process is also limited.[4] In short, these traditional design methods demand substantial prior knowledge from engineers and naturally restrain the number of possible new designs.

With the rapid advancement of artificial intelligence (AI), data-driven methods, such as generative adversarial network (GAN), recently have gained great success in creatively generating fake faces, speeches, and videos.[5] More recently, GAN has begun to draw the attention of engineers in conducting generative designs. For instance, it was used to generate virtual microstructural graphs for various types of steels as it was claimed that the visual images were more helpful than parameterized descriptors in predicting macroscale material properties from microstructures.[6] Apart from its application in the design of material microstructures, GAN has gained more considerable popularity among researchers who work on macroengineering structures and mechanical parts’ design. For instance, GAN was used to generate 2D patterns of complex architectural materials, which aided in finding 2D patterns with extreme properties.[7] Conditional GAN (CGAN) was proposed by labeling the training data to generate cross-section shapes of airfoil toward particular classes of labeled samples.[8] GAN models were also used to create complex 2D bicycle structural designs based on a dataset of...
existing bicycle designs.\[1\] Moreover, GAN was used to generate anisotropic metamaterial unit cell design by training a small set of meta-atom structures.\[9\] More comprehensive studies on integrating topology optimization and GANs were conducted using a 2D automotive wheel design problem as a case study.\[19\] These design applications have delivered good outcomes in various design practices. However, the training data and the learning objectives of these studies were simple 2D patterns and intended to be reduced to simpler 2D shapes. Very few studies have focused on conducting complex 3D structural designs by applying GAN. To the authors’ best knowledge, only one study presented a GAN model for generating 3D aircraft models by training 4045 3D existing aircraft models.\[3\] The training dataset of that study was composed of 3D point clouds of the 3D model surfaces, by which a dataset of very high quality is required to achieve a high training rate. As such, it is essential to obtain precise 3D points of each model using, for instance, 3D scanning or numerical computer-aided design techniques. However, the addition of these data acquisition processes violates one of the most important objectives of the automated AI designs, which aims to conduct automated designs from simpler and more direct input resources, such as images or sketches for reduced human interventions. It is also a very import step to reduce any human interventions and professional knowledge for achieving automatic machine learning (ML) methods.\[11\]

Another essential consideration in an AI-based design process is how the 3D geometric features could be achieved from 2D images for new 3D AI-generated designs. Unlike a 2D pattern or structure, whose dimensions can be extracted using a coordinate system,\[9\] 3D geometric features of a complex structure embedded in a 2D image cannot be extracted easily without the aid of feature extraction tools.\[12\] As one of the most representative deep learning algorithms, the convolutional neural network (CNN) has been widely used in image classifications\[13\] and property prediction.\[14\] It was reported that the good performance of 3D features’ extraction from a typical image was observed using the method in the study by Zhao et al.\[15\] CNN has also shown its robust ability in predicting both local and global properties of a particular structure from 2D images, for example, in predicting the effective elastic stiffness of a specific 3D microstructure of composites (formed by a series of 2D images).\[16\] In predicting the effective thermal conductivity of composites from 2D cross-sections images,\[17\] and in predicting the main features of the stress–strain relationships of composite microstructures.\[18\] These successes of CNN applications in predicting properties of structures from 2D images rely on its robust ability to automatically discover desired features from a vast set of potential features when no explicit feature design is needed.\[13,19\]

In these studies, the input images to the CNN models contain only 2D microstructures, most of which are simple arrangements of blocks. Nevertheless, the CNN technique still demonstrates significant potentials in extracting desired features from a complex 3D structure illustrated in a 2D image.

In this article, a systematic and innovative design method is developed by harnessing combined ML algorithms (deep convolutional GAN (DCGAN) and CNN) to generate AI-based designs of complex 3D structures. The 3D geometric features of the structures are trained directly using 2D images of existing designs instead of 3D structural point data. To demonstrate the new design method, a complex helical structure (nonlinear valve springs) is selected as a case study to test the functional performance of the developed ML model. The schematic working flowchart of the developed design method is shown in Figure 1. First, 130 valve spring samples and their corresponding geometric properties are generated using Latin Hypercube sampling techniques. Next, these samples are represented by 3D geometric models developed using the 3D computer-aided design software Solidworks. Each of the 3D geometric models is transferred by saving its dimetric projection into an individual 2D image, where its 3D geometric information is embedded. Next, the 130 2D images are divided into training and testing groups for the proposed CNN and DCGAN models, respectively. Armed with the trained knowledge, the DCGAN model can generate 2D images of new spring design samples, and the trained CNN model can accurately predict the 3D geometric properties from these generated 2D images. In addition, the mechanical properties (spring stiffness and first-order natural frequency) of the new AI-based spring samples can be predicted using the already developed genetic programming (GP) model in our previous study.\[20\] More importantly, the generated new 2D spring images with the predicted properties can feed backward to generate new 3D computer aided design (CAD) models that can be used to expand and self-update the input dataset to be trained for further use.

2. Results and Discussion

2.1. Predictions of Geometric Parameters by the Multiregression CNN Model

To train and test the proposed CNN model, 80% (104 images of the valve springs) of the whole dataset (130 images) together with
their eight geometric parameters P1–P8 (spring pitches and diameters of coils 02–05 of a helical spring) are used as the training set, while the rest 20% (25 images) of the dataset is used as the testing set to evaluate the accuracy of the CNN model prediction. Before the training process, all inputs and the eight regression targets P1–P8 are normalized between 0 and 1 to improve the efficiency of training data-based models.[20,21] As the CNN model individually seeks a set of viable weights to fit the neuron networks for simultaneously predicting the eight outputs, it can be treated as a multiregression CNN model. The size of the training group is 104 and the batch size of the CNN model is set to 30, thus allowing the model to study a part of the training data at a time, and the epoch is 500, which updates the weights of the neuron networks by 500 cycles. In this study, the Google online service “Colab” with “Graphics Processing Unit” hardware accelerator and ‘High-Ram’ runtime shape are used to train and test the CNN model. In average, it costs 83.32 s to train and only around 0.25 s for each prediction. The training and validation loss over the 500 training epochs are shown in Figure S2, Supporting Information, both of which converge after 300 epochs.

Next, the testing set is input into the trained CNN model, and the prediction results are compared with the ground truth results of the testing set to assess the prediction accuracy of the trained model. In this study, the scatter plots of the comparisons between the ground truth values of the eight regression targets and the eight sets of predicted results by the CNN model are used to illustrate the goodness of fit of these data. In addition, $R^2$ values of each set of ground truth values and predicted results are used to quantify and evaluate the prediction accuracy of the CNN model. Figure 2a–d presents the comparisons between the ground truth values and the predicted results of both the training and the testing sets for the first four geometric parameters P1–P4 (pitches of coils 02–05), and the comparisons for the last four geometric parameters P5–P8 (diameters of coils 02–05) are displayed in Figure 2e–f, where the $R^2$ values are calculated for every training and testing set in each plot. For the eight sets of training data, it can be seen that all the eight $R^2$ values are greater than 0.944 (Figure 2e) and the maximum is 0.978 in Figure 2b. These values demonstrate that the CNN model has gained high learning rates using the developed CNN framework and the defined hyperparameters, which map the latent relationships between the input parameters. For the eight sets of testing data, six of the eight $R^2$ values are over 0.90, although the $R^2$ values in Figure 2c,d are slightly lower. However, it is noteworthy that the testing data in Figure 2c,d...
are well correlated along the diagonal with only small fluctuations along the line. It illustrates that these fluctuations slightly decrease the $R^2$, but the average accuracy of the predictions is still high. From the earlier observations, it can be concluded that the CNN model is well trained and can simultaneously and accurately predict the eight geometric parameters within the predefined design domain from the images of the valve springs.

### 2.2. Development of New 3D AI-Based Designs by DCGAN

The DCGAN model can be trained by feeding the whole dataset (130 images of valve springs) and by applying the developed frameworks and hyperparameters in the previous section. However, to simultaneously execute convolutional neuron networks and transpose convolutional neuron networks in a single model, the DCGAN model requires significant computational resources for training. Therefore, to reduce the computational cost, 130 sets of training data are split into batches of the 30 images. In addition, 4000 epochs are required to update the parameters of both the neuron networks for achieving better learning rates. When executing in “Colab,” it costs around 6578.43 s to train the DCGAN model.

Figure 3 presents the outcomes at various epochs when training the DCGAN model with the 130 sets of image data. In the initial 100 epochs, the DCGAN model can change the pixels on the input image of random noise and distinguish the main object from the background. At 400 epochs, most of the background area is changed to white color, and the positions of every coil are displayed. At 1200 and 2400 epochs, the model gradually improves the quality of the profiles of all the coils. As a result, the shapes and positions of each coil are clearly shown at 3200 epochs. Finally, at 4000 epochs, the edges of the spring become smoother, and more importantly, the lighting effect is significantly improved to enhance the 3D representation of the valve spring. It can be seen that the first and the last coil of the valve spring are already well depicted from 1200 epochs and virtually with the same clarity during the rest of evolution. This is because the shapes and positions of these two coils barely change in the predefined design domain, allowing the DCGAN model to learn them easily and quickly. In this case, the DCGAN model is trained satisfactorily after 4000 epochs and can generate new and clearly defined spring designs.

![Figure 3. Generation process of new designs. Training process and training results of the developed DCGAN model at multiple training epochs.](image-url)

### 2.3. Predictions of Geometric and Mechanical Properties and 3D Model Representation of New AI-Based Designs

Using the trained DCGAN model, in theory, an infinite number of new AI-designed valve springs can be developed. The 2D images of the generated new designs are subsequently input into the developed CNN model to predict the eight geometric parameters of these new springs. Then, based on the predicted geometric parameters, the 3D geometric representation is processed using the parameterized geometric model developed in Section 2.1 to produce 3D geometric models of the AI-based designs. Figure 4 illustrates three representative designs developed by the AI algorithm. The 2D images shown in Figure 4a,c,e are design sample 01, 02, and 03 generated by the trained DCGAN model, respectively. The 3D models shown in Figure 4b,d,f are the 3D CAD models defined by the geometric parameters predicted by the developed CNN model. The shadows of these geometric models indicate that they are 3D CAD models other than 2D images. It can be seen that the design sample 01, shown in Figure 4a, possesses relatively larger and similar-coil diameters at both lower and upper ends of the helical springs. The design sample 02 in Figure 4c has an even distribution of the coil diameter, while design sample 03 in Figure 4e has a smaller coil 02, which is only 14.403 in diameter.

The earlier design results show that the developed DCGAN model can generate new AI-based designs with detailed 3D geometric information, including the dimension and the relative positions of every coil, as predicted accurately by the CNN model. Hence, it shows that the DCGAN model has a reasonably good learning ability to generate new designs similar to the practical designs, and the CNN model can predict the geometric properties of complicated 3D helical structures with high accuracy.

A GP model, which was previously developed by the authors,[20] is used to predict the mechanical properties of the newly designed springs. The GP model can accurately predict spring stiffness and first-order natural frequency of complex helical springs based on their geometry. The GP model was trained using finite-element simulation results from 300 valve springs and validated by experimental results. The spring forces and natural frequency formulas generated by the GP model in the study by Gu et al.[20] can be directly used along with the ML models developed in this study without executing the GP model. The geometric properties of the AI-based design samples predicted by the developed CNN model were imported into these formulas. Assuming that the material of these springs is spring steel Oteva 90 (with Young’s modulus $E$ of 206 GPa and shear modulus $G$ of 79.6 GPa), the stiffness and the first-order natural frequency of the three new springs are predicted by the formulae from the GP model, as shown in the tables in Figure 4. When compared with the design 03, the observed mechanical properties of design 02
and the design 03 are closer to each other as they present similar geometric features. The earlier examples clearly show that the mechanical and the geometric properties of the DCGAN designed valve springs can be accurately predicted and represented by the 3D CAD models using the developed CNN model and the previously developed GP model.

3. Conclusion

This article presents a novel approach for conducting AI-based mechanical design of complex 3D structures with minimum human interventions. 2D images of complex 3D geometries are the original dataset and the direct inputs of the developed model. A DCGAN model and a CNN model are trained using these 2D images to generate new AI-based design samples and accurately predict their geometric properties. The generated new samples, together with the predicted geometric properties, are then represented by 3D CAD models using the developed CNN model and the previously developed GP model.
generated output designs of the current cycle to form a self-updated input dataset for the next cycle of training and designing. The earlier general procedure is followed in this article to design nonlinear helical structures as a case study. In addition, by combining the GP model developed previously by the authors, the mechanical properties of these new spring designs can be predicted simultaneously. The main findings of this study are as follows. 1) Using the advanced ML techniques (DCGAN, CNN, and GP), a small volume of 2D projection images of 3D mechanical parts can be used as input to produce an infinite number of computer-generated complex 3D mechanical designs, whose geometric and mechanical properties are subsequently predicted by a GP model; 2) Human interventions in designing complex mechanical parts can be significantly reduced by applying the proposed method, which is one of the primary aims of introducing AI into mechanical designs; 3) The present AI-based design method has great potential to be further developed to design complex mechanical parts with desired geometric and mechanical properties. In addition, the method will also be expanded to the design of unparameterizable 3D mechanical parts, which is a limitation of the present work.

4. Machine Learning Frameworks

Data Generation and Preprocessing: A manufactured beehive valve spring (Figure 5a) was used in this study as a benchmark design. The nonlinear 3D geometric model of the spring was generated accordingly using the commercial software SolidWorks (Figure 5b). It can be seen that the spring pitch and the coil diameter varied at different helical positions, which brought difficulties to present the whole structure by simple analytical models.[22] The detailed modeling process and validation of the model can be found in the authors’ previous studies.[20,23] In this study, the spring pitch and the diameter of coils 02–05 were parameterized and denoted as P1–P8 for constructing the parameterized geometric model (Figure 5c). The overall helical structure can be altered by modifying one or more parameters from P1–P8, and the ranges of these parameters construct the accessible design space of the valve spring.

To select the best samples that represent the properties of the whole design domain, constrained Latin hypercube sampling (cLHS) and variant of Latin hypercube sampling (LHS) were used. LHS can generate more representative samples than other sampling techniques, such as Monte Carlo Sampling (MCS).[24] The working principle of LHS is to divide the ranges of variables into n intervals and then select samples from the stratified design domain.[25] Thus, the number of design variables determines the number of stratified layers. However, as LHS exploits the entire design domain evenly, it cannot seek samples with specific relationships between design variables. Hence, cLHS is used in this study, which can define constraints when exploiting samples from the design domain.[20]

The cLHS method was achieved using the “LHS DOE generator” in the commercial software MATLAB. The eight geometric parameters, P1–P8, were defined as the design variables. The respective ranges of the variable are listed in Table 1, together with a constraint, that is, the relationship between the variables. The constraint aimed to limit the height of the helical spring to a specific range from 45.0 to 46.0 mm. In practice, this type of constraint is usually an essential consideration as most engineering parts are designed to fit into specific systems. In this case, the specified height range of the valve spring was used to satisfy the dimensional requirements of a commercial sports car engine.

Using the cLHS method, 130 sets of geometric parameters of the valve spring (Table S1, Supporting Information) were generated inside the design domain. The 130 geometric parameters were then imported into the 3D-parametrized model to generate 130 distinctive 3D valve spring samples. Finally, each sample was exported as a 2D image, where all geometric models were displayed in dimetric projection. The original dimension of the sample image was 1693 pixels in width and 890 pixels in height, where most of the area was occupied by pure background. Therefore, the original image of each spring sample was cropped to an identical size of

Table 1. Ranges and constraints of the design variables (mm).

|        | P1  | P2  | P3  | P4  | P5  | P6  | P7  | P8  |
|--------|-----|-----|-----|-----|-----|-----|-----|-----|
| Lower Bound | 6   | 6   | 6   | 6   | 15  | 15  | 15  | 15  |
| Upper Bound  | 9.3 | 10  | 10  | 10  | 25  | 25  | 25  | 25  |
| Constraint  | .31 < P1 + P2 + P3 + P4 < .32 |

Figure 5. Manufactured and numerical models of nonlinear springs. a) Manufactured beehive valve spring. b) 3D geometric model and c) parameterized geometric model of the helical structure of the beehive spring.
400 pixels in width and 700 pixels in height to remove the useless background. Eventually, 130 sets of the original images were modified to 130 sets of cropped images, which improved the quality of the datasets and the training efficiency of the ML models in the following section.

The two essential processes in AI-based designs were to train the AI algorithm to recognize the specific features of every identical design sample and create new samples from the design domain. This study used CNN and GAN frameworks to process the 130 sets of image-based data samples. Furthermore, they were trained to recognize the geometric properties of various valve springs and create new AI-featured new designs, respectively.

CNN: CNN has already been widely used in tackling data-driven problems of image classification, recognition, and segmentation. As a deep neural network class, CNN can process multiple dimensional datasets, making it a popular candidate for handling images and video input data. The main features of a CNN framework include introducing convolutional layers to detect local conjunctions of image pixels and pooling layers to reduce dimensions of input data by summarizing features in the certain regions. CNN was also used in image regression as discussed in the previous section. However, it is notable that all these studies are single-output-based CNN models. In this section, a multiple-output-based CNN model was developed by applying the Keras package based on the Tensorflow backend in Python to process the image inputs of the 130 valve springs. The ultimate outputs of the framework were the eight predicted geometric parameters P1–P8.

The CNN framework developed in this study for identifying multiple geometric properties (P1–P8) of the valve springs is shown in Figure 6. The first part of the framework is to input the training images into the CNN. Among the generated 130 spring images, 80% (104 images) were used as the training set, and the rest 20% (25 images) were used as the testing set. The original data structure of each image in both sets was 700 × 400 × 3, representing 700 pixels in length, 400 pixels in width, and 3 color channels. However, as the ultimate objective is to predict the geometric properties, the color information is not essential. A high volume of input elements will significantly increase the computational time of the CNN model. Therefore, the dimension of each input image was reduced to 180 × 100 × 1 in the first layer of the CNN framework, as it was confirmed later that the reduced black and white image provides sufficient geometric information for the model. Finally, the 104 sets of dimensionally reduced training images were input to the developed convolution layers.

Hyperparameters of the model should be defined to control the learning process of the CNN model in this study. Table 2 presents the hyperparameters defined in the developed CNN models and the outputs of each layer. The schematic structure of the outputs of these layers is also presented in Figure 6 to show the data structure of each layer. As shown, 8 × 8 kernels were used in convolution layers 01, 02, 03, and the numbers of kernels for these layers were 32, 64, and 128, respectively. Mathematically, given an image I and a filter f, the convolutional computation can be represented by

\[
\text{conv}(I)_{xy} = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} I_{i-1,j-1,k} 
\]

The main features of a CNN framework include introducing convolutional layers to detect local conjunctions of image pixels and pooling layers to reduce dimensions of input data by summarizing features in the certain regions. CNN was also used in image regression as discussed in the previous section. However, it is notable that all these studies are single-output-based CNN models. In this section, a multiple-output-based CNN model was developed by applying the Keras package based on the Tensorflow backend in Python to process the image inputs of the 130 valve springs. The ultimate outputs of the framework were the eight predicted geometric parameters P1–P8.

The CNN framework developed in this study for identifying multiple geometric properties (P1–P8) of the valve springs is shown in Figure 6. The first part of the framework is to input the training images into the CNN. Among the generated 130 spring images, 80% (104 images) were used as the training set, and the rest 20% (25 images) were used as the testing set. The original data structure of each image in both sets was 700 × 400 × 3, representing 700 pixels in length, 400 pixels in width, and 3 color channels. However, as the ultimate objective is to predict the geometric properties, the color information is not essential. A high volume of input elements will significantly increase the computational time of the CNN model. Therefore, the dimension of each input image was reduced to 180 × 100 × 1 in the first layer of the CNN framework, as it was confirmed later that the reduced black and white image provides sufficient geometric information for the model. Finally, the 104 sets of dimensionally reduced training images were input to the developed convolution layers.

Hyperparameters of the model should be defined to control the learning process of the CNN model in this study. Table 2 presents the hyperparameters defined in the developed CNN models and the outputs of each layer. The schematic structure of the outputs of these layers is also presented in Figure 6 to show the data structure of each layer. As shown, 8 × 8 kernels were used in convolution layers 01, 02, 03, and the numbers of kernels for these layers were 32, 64, and 128, respectively. Mathematically, given an image I and a filter f, the convolutional computation can be represented by

\[
\text{conv}(I)_{xy} = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} I_{i-1,j-1,k} 
\]

where \(m\), \(n\), and \(k\) are the sizes of height, width, and the number of color channels of the input images. As a result of passing convolution layers, the input data structure became 3D, and its depth was equal to the number of kernels of the layer. In addition, two pooling layers with a 2 × 2 pooling kernel were added between the convolution layers. These pooling layers summarized the features of the input data by reducing its spatial size to reduce the computational time. The max-pooling results were obtained by calculating the \(l_p\) norm of the input data using Equation (2),

\[
P_l(I)_{xy} = \left[ \sum_{i=1}^{m} \sum_{j=1}^{n} (I_{i,j})^p \right]^{1/p} \quad p \to \infty
\]

where \(s_b\) is the size of the pooling kernel. Hence, the width and the length of input images were halved in every pooling layer when applying the 2 × 2 pooling.
pooling kernel. The fully connected layers received the input data from the convolution layer 03 and flattened the $45 \times 25 \times 128$ 3D data to 144,000 1D data in the flattened layer. The following two dense layers were to conduct regression, leading to eight outputs of the final layer. The activation functions of the convolution and dense layers, the pooling layers, and the output layer were ReLU, max pooling, and linear with multiple regression outputs, respectively.

**DCGAN:** As a class of unsupervised ML frameworks, GAN was recently proposed by Goodfellow et al.\(^{(5)}\) The framework was inspired by the zero-sum game of game theory, which constructs two neural networks to contest each other. Figure S1, Supporting Information, shows the schematic framework and working principles of GAN. The two core functional modules in GAN are the two neural networks called generator and discriminator. The generator is used first to generate massive samples from randomly generated noise data. Then, both the generated and actual samples are input into the trained discriminator to distinguish the generated samples from the actual samples. The classification results propagate backward to iteratively tune the parameters of both the generator and the discriminator. The ultimate goal of a GAN is to train the generator to generate new samples that cannot be distinguished from actual samples by...
the well-trained discriminator. To achieve this goal, loss functions derived from the formula of binary cross-entropy loss are defined to evaluate the performances of both the generator and the discriminator.\(^\text{[3]}\) The objective of the discriminator is to classify the generated data and the actual data correctly by maximizing the following loss function.

\[
\text{Loss}^{(D)} = \max \{-\log(D(x)) + \log(1 - D(G(z)))\} \tag{3}
\]

where \(D\) and \(G\) represent the discriminator and the generator, and \(x\) and \(z\) are the inputs from the dataset and the random noise, respectively. In contrast, the generator was competing against the discriminator, which needed to minimize the loss function as

\[
\text{Loss}^{(G)} = \min \{-\log(D(x)) + \log(1 - D(G(z)))\} \tag{4}
\]

Equation (3) and (4) were combined to form the loss function of GAN and introduce the expectation to consider multiple data points. Finally, the objective function of GAN valid for the entire dataset can be given as

\[
\min_G \max_D V(D, G) = \max_D \left(\min_G (-\log(D(x)) + \log(1 - D(G(z))))\right) \tag{5}
\]

By exploiting the best parameters to satisfy the earlier objective function, a GAN model can be trained to generate various new samples, which contain similar properties as the real datasets.

In this study, a DCGAN model was developed to generate new valve springs in the given design domain. As a type of GAN, DCGAN uses transpose convolutional networks and deep convolutional networks to construct the generator and the discriminator. Figure 7 shows the schematic framework of the developed DCGAN model, which used the Keras package based on the Tensorflow backend in Python. The hyperparameters of both the generator and the discriminator are presented in Table 3. A random noise map with a shape of \((1, 100)\) was generated first and input into the generator. Then, the transpose convolution layer, which is simply a converse process of the convolution layer, extended the dimension of the random noise to be \(90 \times 50 \times 256\). The dimensions of the input data became \(90 \times 50 \times 128\) and \(180 \times 100 \times 64\) in the transpose convolution layer 02 and layer 03, respectively. The final outputs of the generator were 2D valve spring images of 360 pixels in length and 200 pixels in width. The generated spring images were mixed with the spring images from the actual dataset, divided into \(360 \times 200\) pixels, and input into the discriminator. These composite image data were transformed to 3D datasets with a shape of \(180 \times 100 \times 128\) in the convolution layer 01 of the discriminator and then reshaped to \(90 \times 50 \times 256\) in the convolution layer 02. In the following flattened layer, the 3D datasets were reconstructed to form a series of \(1,152,000\) data points. Leaky ReLU was used in all the discriminator convolution layers and the generator’s transpose convolution layers as the activation functions. In the last layer of the discriminator, the Sigmoid activation function was used to classify the images from the real datasets and the ones generated by the generator.

**Supporting Information**

Supporting Information is available from the Wiley Online Library or from the author.

**Acknowledgements**

This work was supported by the EPSRC (grant number EP/R51357X/1).

**Conflict of Interest**

The authors declare no conflict of interest.

**Data Availability Statement**

The data that support the findings of this study are available in the supplementary material of this article.

**Keywords**

convolutional neuron networks, deep convolutional generative adversarial networks, engineering designs, helical structures, machine learning

---

\[1\] A. H. Nobari, M. F. Rashad, F. Ahmed, arXiv preprint arXiv:2103.06242 2021.

\[2\] B. Zheng, Y. Liu, J. Liu, S. Yin, J. Xu, Int. J. Mech. Sci. 2019, 161, 105025.

\[3\] D. Shu, J. Cunningham, G. Stump, S. W. Miller, M. A. Yukish, T. W. Simpson, C. S. Tucker, J. Mech. Des. 2020, 142, 071701.

\[4\] F. Buonamici, M. Carfagni, R. Furfari, L. Governi, A. Lapini, Y. Volpe, J. Comput. Des. Eng. 2018, 5, 145.
[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Adv. Neural Inf. Process. Syst. 2014, 27, 2672.
[6] J. W. Lee, N. H. Goo, W. B. Park, M. Pyo, K. S. Sohn, Eng. Rep. 2021, 3, e12274.
[7] Y. Mao, Q. He, X. Zhao, Sci. Adv. 2020, 6 eaaz4169.
[8] G. Achour, W. J. Sung, Q. J. Pinon-Fischer, D. N. Mavris, in AIAA Scitech 2020 Forum, ARC (Aerospace Research Central), Orlando, FL 2020, p. 2261.
[9] J. A. Hodge, K. V. Mishra, A. I. Zaghloul, in 2019 IEEE Inter. Symp. on Phased Array System & Technology (PAST) IEEE, Waltham, MA, USA 2019, pp. 1–6.
[10] S. Oh, Y. Jung, S. Kim, I. Lee, N. Kang, J. Mech. Des. 2019, 141, 111405.
[11] Y. Liu, J. M. Wu, M. Avdeev, S. Q. Shi, Adv. Theor. Simul. 2020, 3, 1900215.
[12] S. Jang, S. Yoo, N. Kang, arXiv preprint arXiv:2008.07119 2020.
[13] J. Schmidhuber, Neural Networks 2015, 61, 85.
[14] Y. Liu, B. Guo, X. Zou, Y. Li, S. Shi, Energy Storage Mater. 2020, 31, 434.
[15] X. Zhao, H. Wei, H. Wang, T. Zhu, K. Zhang, Sol. Energy 2019, 181, 510.
[16] A. Cecen, H. Dai, Y. C. Yabansu, S. R. Kalidindi, L. Song, Acta Mater. 2018, 146, 76.
[17] Q. Rong, H. Wei, X. Huang, H. Bao, Compos. Sci. Technol. 2019, 184, 107861.
[18] C. Yang, Y. Kim, S. Ryu, G. X. Gu, Mater. Des. 2020, 189, 108509.
[19] Y. LeCun, Y. Bengio, G. Hinton, Nature 2015, 521, 436.
[20] Z. Gu, X. Hou, J. Ye, in Proc. of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, SAGE Journals, London 2021, 09544062211010210.
[21] Z. Gu, Y. Liu, D. J. Hughes, J. Ye, X. Hou, Composites, Part B 2021, 217, 108894.
[22] I. Kacar, V. Yildirim, Mech. Based Des. Struct. Mach. 2016, 44, 340.
[23] Z. Gu, X. Hou, E. Keating, J. Ye, Int. J. Mech. Sci. 2020, 173, 105476.
[24] A. Olsson, G. Sandberg, O. Dahlblom, Struct. Saf. 2003, 25 47.
[25] B. Minasny, A. B. McBratney, Comput. Geosci. 2006, 32, 1378.
[26] J. B. Estrach, A. Szlam, Y. LeCun, in Inter. Conf. on Machine Learning PMLR, 2014, p. 307–315.