Contents lists available at ScienceDirect

Building and Environment

journal homepage: www.elsevier.com/locate/buildenv

A generative architectural and urban design method through artificial neural networks

Hao Zheng[a], Philip F. Yuan[b,*]

[a] Stuart Weitzman School of Design, University of Pennsylvania, Philadelphia, USA
[b] College of Architecture and Urban Planning, Tongji University, Shanghai, China

ARTICLE INFO

Keywords:
Architectural form finding
Artificial neural networks
Computational design
Machine learning

ABSTRACT

Machine learning, as a computational tool for finding mappings between the input and output data, has been widely used in engineering fields. Researchers have applied machine learning models to generate 2D drawings with pixels or 3D models with voxels, but the pixelization reduces the precision of the geometries. Therefore, in order to learn and generate 3D geometries as vectorized models with higher precision and faster computation speed, we develop a specific artificial neural network, learning and generating design features for the forms of buildings. A customized data structure with feature parameters is constructed, meeting the requirements of the neural network by rebuilding surfaces with controlling points and appending additional input neurons as quantified vectors to describe the properties of the design. The neural network is first trained with generated design data and then tested by adjusting the feature parameters. The prediction of the generated data shows the basic generative ability of the neural network. Furthermore, trained with design data collected from existing buildings, the neural network learns and infers the geometric design features of architectural design with different feature parameters, providing a data-driven method for designers to generate and analyze architectural forms.

1. Introduction

1.1. Background

In the design process, designers create their works using models or drawings based on their design requirements and limits [1]. Especially in the design of forms, such as the generative design for pavilions or high-rise buildings, its process is similar to the programming of algorithms; it inputs several controlling factors and outputs the generated geometric models or drawings [2,3].

There are two algorithmic methods to generate design solutions. One is rule-based, including the Metropolis algorithm [4], simulated annealing [5], and the genetic algorithm [6]. These algorithms regard the design process as an optimization problem, applying human-defined rules to iteratively approach the solution that meets the requirements.

In architectural design, researchers have been using these algorithms to refine their design work for decades. Simulated annealing has been applied to solve the facility layout problem in the interior design of a hospital [7] and generate the optimal floor plans [8], and to evaluate and optimize a lightweight structure with high performance and low cost [9]. Additionally, the genetic algorithm has been used to design massing options on the basis of a site as a pixel image [10] and to look for design solutions that optimize thermal and lighting performance in a building [11].

However, in order to “teach” the algorithms to find the solutions, users must clearly define the objective function $f(x)$ [12]. That means that, in the design field, designers must provide a clear evaluation function to state the quality of a design instead of giving just a binary judgment. It is difficult for designers to understand and express the exact objective function in a mathematical way.

Another method to generate design solutions is the data-driven process that involves artificial intelligence approaches. Artificial intelligence methods have been widely used in design [13] and non-design domains, especially the applications of machine learning techniques such as the neural networks. Machine learning as a decision-making tool differs from simulated annealing and the genetic algorithm in that it takes the input controlling factors and the output solutions as training data and then calculates the mappings between the inputs and outputs. To be specific, the process is presented as several neurons and a neural network describing the computational relations between the inputs and

* Corresponding author.
E-mail addresses: zhhao@design.upenn.edu (H. Zheng), philipyuan007@tongji.edu.cn (P.F. Yuan).

https://doi.org/10.1016/j.buildenv.2021.108178
Received 16 March 2021; Received in revised form 27 June 2021; Accepted 17 July 2021
Available online 27 July 2021

outputs [14]. When the input feature data and the output design data are given to the neural network, the program runs the training process with back propagation [15] to optimize the parameters in the neural network. Then, with the finalized network, the user can input a new set of feature data and obtain the output design data as feedback.

Using this method, the user needs to provide the neural network with only the design features and their corresponding design results, instead of the objective functions, to train a "data translator", helping transform the input data into the output design works (Fig. 1). Researchers have applied machine learning in solving architectural, urban, and environmental problems in previous research [16–20]. Successful applications of machine learning techniques in solving design problems include the following: learning design concepts from design examples [21], learning design shapes from point clouds [22], converting unstructured triangle meshes into ones with consistent topology [23], classifying design objects [24], and describing and categorizing the design process quantitatively [25].

1.2. Problem statement

In architectural design, researchers have recently applied different neural networks to learn and generate design works. Translating visual design data into 2D images is a convenient method to feed the design data into neural networks. Generative adversarial networks (GANs) [26] are used to generate design images. Previously, GANs have been used to generate satellite images of cities [27,28]; architectural floor plans and layouts [29,30]; street view images [31,32]; three view images of residential houses [33]; the geometry of curved surfaces as black-and-white images [34]; structural solutions with evaluation criteria [35]; and the assembly plans for bricks [36].

In addition to the 2D pixel-based tasks, 3D convolutional neural networks (3DCNNs) [37] and 3D generative adversarial networks (3DGANs) [38] represent another approach to learn and infer architectural data in a 3D voxel-based format. The uses of these techniques include classifying the 3D features of buildings [39], identifying difficult-to-manufacture features in 3D models [40], and generating modelings for industry and furniture [38].

However, training GANs requires large computational power, which cannot be easily applied to the problem-solving in design-related scenes [41,42]. According to the visualization of the parameters and kernels in the neural network [29], the convolution layers in image-based neural networks actually act to detect the boundary information from the original image and then combine features into deeper layers for computation. Thus, a neural network based on pixels or voxels first translates image data into vector-like data by creating continuous white or black pixels, then feeds the pixel data to full-connected layers for outputting a single value (image classifier) or to deconvolution layers for outputting another image (image generator). A large amount of computational power is used to update millions of kernels, which only acts to transform images into vectorized data. Thus, it is inefficient to train image-based neural networks if vectorized data already exists in CAD drawings or modelings in most of the architectural design cases.

Furthermore, because of the voxelization, precision is lost during the translation between the vector data and voxel data [43]. For the commonly used 3DGAN [38] model, the output layer only generates a 64*64*64 voxel model, which gives an aliased solution and is not smooth enough to represent an architectural form, especially the curved facade of a building under the design tendency of free-from surface.

In addition to the voxel-based method, there are some existing machine learning techniques based on vectorized data, for example point cloud [44,45] and graph topology [46,47]. However, the point cloud method also requires a large number of computation resources because the density of the point cloud directly influences the precision of the model. Additionally, the graph-based data structure is usually used when generating the topological relation of elements, so it is not suitable for generating design models with geometric information.

1.3. Objectives

Therefore, in order to train a neural network to learn and generate 3D architectural geometries as vectorized models rather than pixelized images or voxelized volumes, we introduce an artificial neural network with a data structure customized for generating 3D forms, which enables designers to generate architectural geometries with data-driven machine learning methods by training with design examples. In this research, our goal is to test whether this artificial neural network can learn and generate design cases as geometries with features from different designers and also whether it can become a tool architects can use to quickly generate architectural forms with a design style that is similar to the collected dataset. As a result, this method is intended to be used as a form-finding tool in the early design stage, helping architects improve their creativity and efficiency.

As compared with other methods on a technical level, first, the data structure should keep the precision of the original data while the training and generating processes should be accelerated. To achieve this, a new data structure should be developed, storing the modeling data as a collection of numerical vectors rather than as pixelized images. The

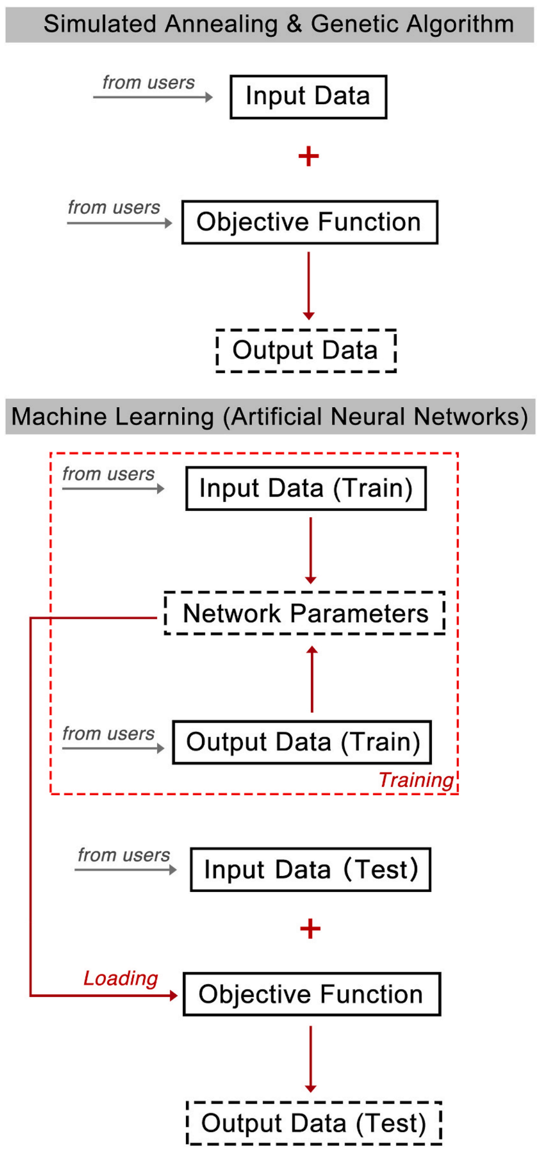

Fig. 1. Simulated annealing, genetic algorithm, and machine learning (artificial neural networks).

vectorization and the digital representation of the design data in this method will provide guidance and inspiration for further exploration of intelligent algorithms in architectural design. Also, this artificial neural network should be easy to train and implement, which does not require extensive computational power. Due to the lack of computational power in personal computers available to most architects, this method should provide a light-weight neural network structure and training process to allow fast training and generating in local computers.

Second, in order to provide more options for the generation of architectural forms - for example, the footprints and styles - this artificial neural network should include the input features to represent the different requirements and constraints in the design process. By defining the values in the input layer, designers should be able to generate the corresponding output geometry under the control of design features. Meanwhile, the modification of the features should be flexible, thus broadening the potential uses of this method by providing a general guideline for adjusting features quantitatively.

Third, by building a generative system for the case studies, this research also aims to show a general workflow for applying machine learning methods to predict design solutions. According to the input features and the output results, it should be possible to adapt this system to different generative tasks, thus helping designers to play with forms with data-driven methods.

Last, in order to verify the capability of the neural network, two datasets, the generated and the real, are proposed to be collected and tested using this neural network. The successful training and predicting of both datasets can then prove the above assumption.

2. Methodology

2.1. Data processing

In the process of generating 3D forms, usually a surface, the geometric data is recorded as a series of Non-Uniform Rational Basis Splines (NURBS) defining the sections in the surface in modeling software, for example Rhino [48]. A series of 3D points defines NURBS as controlling points (Fig. 2). Therefore, a surface is originally defined by a grid of points with coordinates of (x, y, z), while the points together define the UV NURBS and further become the surface.

For example, as Fig. 3 shows, when the surface of a tower design is deconstructed by an 8layer*10item grid of points, it can be presented as the collection of the coordinates of the controlling points from $(x00, y00, z00)$ to $(x79, y79, z79)$. Taking the process a step further, since the distance between each neighboring layer remains constant and all layers are parallel to the world XY plan, the coordinates can be simplified as a 2D vector of (x, y) and a 1D vector (H) as the height of the surface in Z

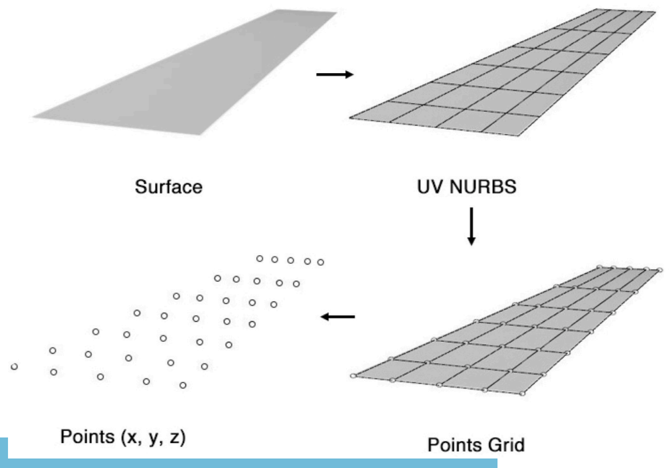

Fig. 2. Definition of the surface.

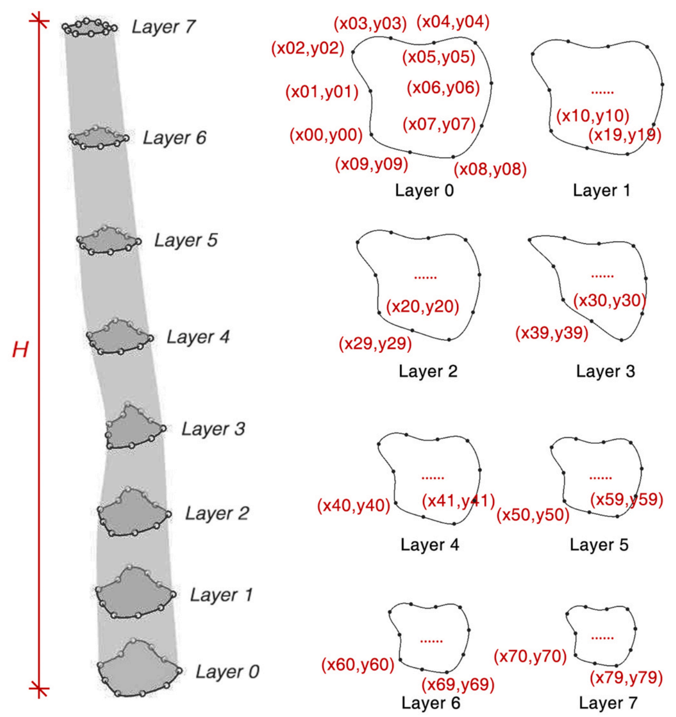

Fig. 3. Parameterization of a surface in the world coordinate system.

direction. That means that the surface can be expressed as a collection of 161 real numbers listed as $(x00, y00, x01, y01, \ldots, x79, y79, H)$, which is acceptable for a more efficient artificial neural network.

Next, it is important to note that the sigmoid activation functions in the neural networks are usually specially designed to learn and generate data between 0 and 1, such as solving the binary problem [49]. In order to obtain a better prediction of the form, all real numbers should be standardized into a range of 0–1.

Fig. 4 shows the process of data standardization. The original model is scaled down by scaling factors for X, Y, and Z axes, and fitted into a box with side lengths of 1 unit in the X direction, 1 unit in the Y direction, and 10 units in the Z direction. Equation (1) shows the definition of the scaling factors R_X, R_Y, and R_Z, in which k is a fixed coefficient, while X Ratio, Y Ratio, and Z Ratio are parameters with values of real numbers between 0 and 1. After the surface has been scaled down, its bonding box in the X, Y, and Z directions coincides with that specific box. Since all points are inside the bounding box, their coordinates (x, y) are also real numbers between 0 and 1.

$$\begin{aligned} R_X &= \frac{1}{k^* X_{Ratio}} \\ R_Y &= \frac{1}{k^* Y_{Ratio}} \\ R_Z &= \frac{1}{k^* Z_{Ratio}} \end{aligned} \tag{1}$$

However, when training with the data from different cases, points with the same index contributes to the training of the same neuron, which means they share the same weight parameter from which to learn and generate [50]. Thus, in order to align points in each case as much as possible, the points in each UV NURBS are rebuilt as Fig. 5 shows. After scaling up, the closest point on the UV NURBS to the coordinates origin $(0, 0)$ is found and assigned as the new start point (Point 0). Then the original UV NURBS is divided into 10 points again based on the new start point. As a result, all points with the same index in different cases have greater commonality and train the same neurons.

Lastly, feature parameters, as additional input neurons, describe design data with different design strategies or styles. With these

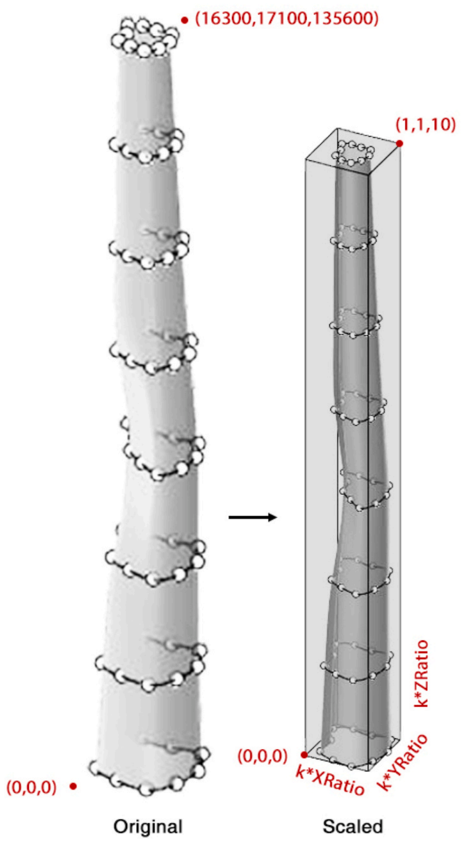

Fig. 4. Model scaling and data reparameterizing.

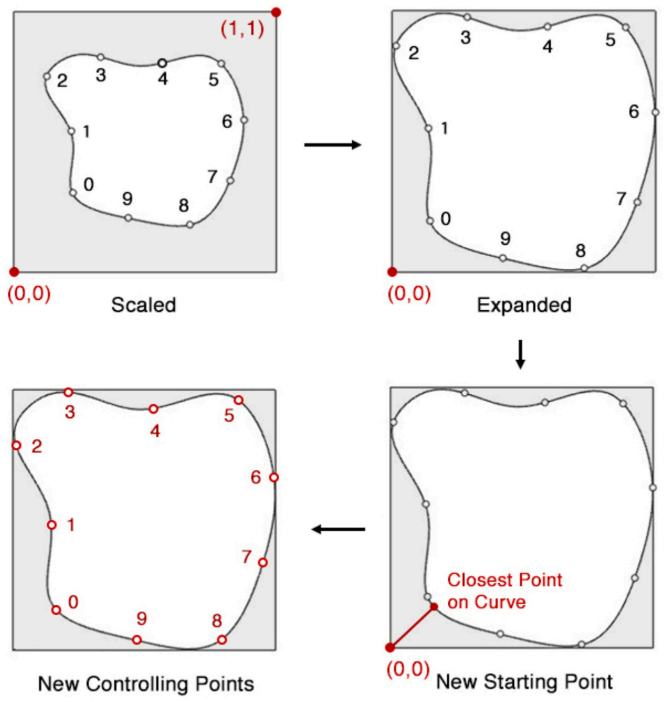

Fig. 5. Adjusting controlling points.

neurons, the neural network is able to learn and generate design data based on the given parameters, mathematically describing the related factors that influence the design outcomes [51].

To show an example of adding these feature parameters into the neural network structure and test the learning ability of the neural network, we define 9 different operations to generate the training dataset of tower-like forms (Fig. 6). For each style, a set of 9 numbers, either 0 or 1, is assigned as its feature parameter, showing the index of its style. For example, $(0,1,0,0,0,0,0,0,0)$ represents the 2nd style; $(0,0,0,0,0,0,0,0,1)$ denotes the 9th style. There is always one style assigned to each case in the training data, which means that there is only one count of number 1 in the 9 numbers. However, forms with combined styles can be generated by inputting feature parameters, such as $(0, 0.5, 0, 0, 1, 0.8, 0, 0, 0.3)$, in the generative design process. We discuss this over the following pages.

Therefore, with 9 sets of feature parameters in total, 9 styles of the tower forms are generated as the training data. As Fig. 6 shows, the forms contain two types of geometric operations and three options for each operation. The first operation defines the basic shape of the form: straight, gradual, or conical. Based on the overall shape, the second operation applies distortions in different positions in the form: no distortion, distortion in the lower position ($z = 2.5$), or distortion in the middle and upper positions ($z = 5$ and $z = 7.5$). The two operations together produce 9 types of distinguishable forms, in which the complicated shapes increase the difficulty or learning for the neural network. We anticipate being able to see the effects on the performance of the neural network and evaluate its abilities by analyzing the results.

Once 100 forms are randomly generated for each style, the training dataset contains 900 forms. For every single form, it is translated as 172 real numbers between 0 and 1, listed as $x00, y00, x01, y01, \ldots, x79, y79$, X Ratio, Y Ratio, Z Ratio, and 9 feature parameters.

2.2. Neural network training

The purpose of generative design is to generate a form based on several given requirements [52], such as the footprint of the building and the style of the form. To make a generative design machine for the tower design, the input data should include basic information to allow the neural network as well as a human designer to generate a unique form.

Fig. 7 shows the definition of the input and output data. The data of the controlling points in Layer 0 describes the geometry, which touches the ground as the starting section of the form. X Ratio (XR), Y Ratio (YR), and Z Ratio (ZR) represent the actual size of the form by showing the scaling ratios. And the 9 feature parameters, FP1 to FP9, provide the information on the styles, identifying the design strategies of the form. The output data contains the rest of the controlling points from Layer 1 to Layer 7. By building the surface according to the 1 inputted and the 7 outputted UV NURBS and then scaling the surface up by the three ratios, the form is constructed, providing the users with a 3D form for the tower design.

Fig. 8 shows the neural network structure. In addition to the input and output layers, each hidden layer between them acts as an operator to combine the parameters from the previous layer and forward the parameters to the next layer. Having more and larger hidden layers yields more accurate predictions, but they also require more time for training and predicting and increase the risk of overfitting [53]. After experiments, we found that the neural network with only one hidden layer of 200 neurons performed best on balance, considering both accuracy and efficiency [54]. Therefore, a neural network with a hidden layer of 200 neurons was chosen as the basic machine learning framework.

As mentioned before, the purpose of standardizing all the real numbers into the range of 0–1 is to allow the application of a sigmoid activation function to the neural network. Thus, the activation function, which calculates the parameter $\hat{y}$ in the current neuron from the parameter x in the previous neuron, can be expressed as the sigmoid formula (Eq. (2)).

$$\hat{y} = Sigmoid(w * x + b) = \frac{1}{1 + e^{-(w*x+b)}} \tag{2}$$

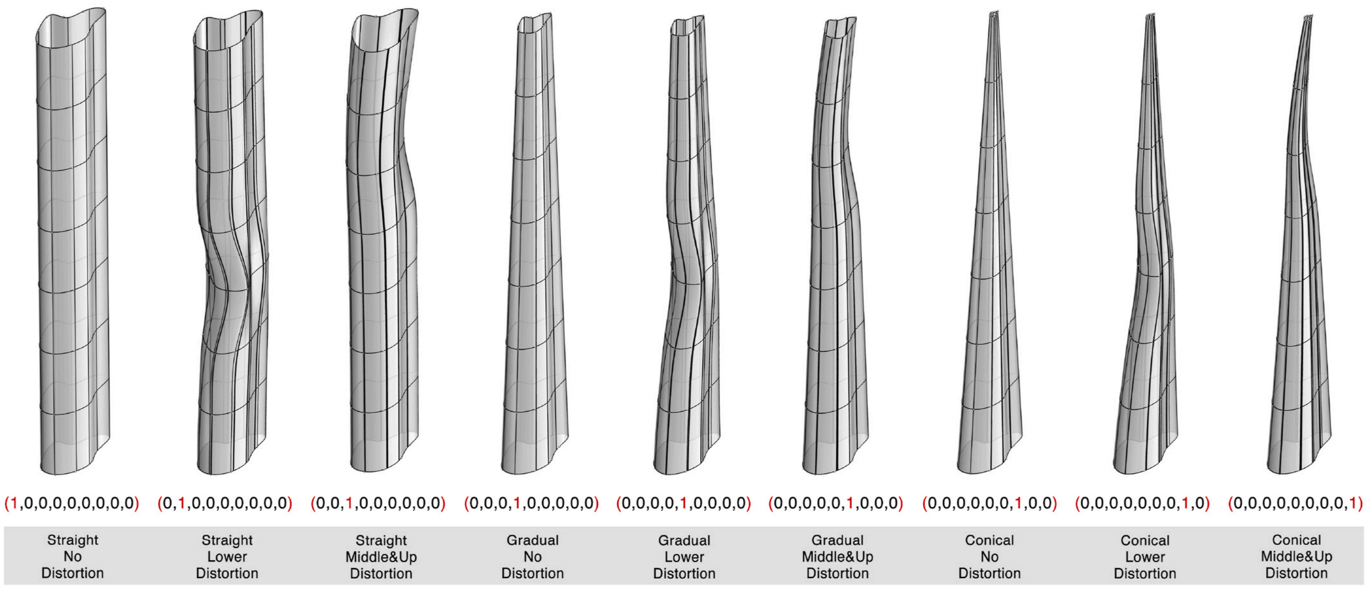

Fig. 6. Feature parameters and data generation.

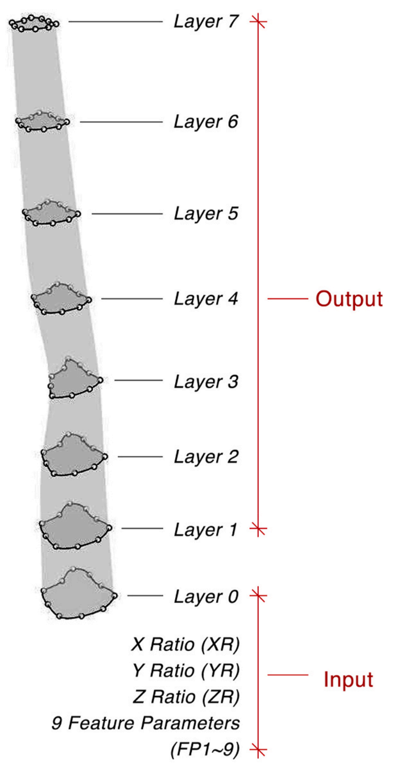

Fig. 7. Input and output data.

For the output layer, in order to return the feedback on how much deviation there is between the predicted output $\widehat{y}$ and the ground truth output y, a mean squared error (MSE) function is chosen as the loss function to evaluate the neural network performance and allows the back propagation to update the neural network weight w and bias b parameters (Eq. (3)).

$$Loss(y,\widehat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \quad (3)$$

Therefore, with the neural network described above, the training dataset of 900 forms was assigned to train and update the neural network parameters. It took an i7-6700HQ CPU-only laptop 0.8 h to run roughly 30,000 epochs, and ultimately, it reached an error value of 0.04 for each case (Fig. 9), which means that, on average, there is a shift of just 0.024 units for each point.

After training, to test the performance of the neural network, the testing dataset was generated and inputted to the neural network. In the training process, the neural network only updates itself by learning the training dataset without loading the testing dataset. Thus, through a comparison of the expected forms and the predicted forms of the testing dataset, the performance of the neural network can be evaluated.

Fig. 10 shows the forms generated from the input data with the same controlling points in Layer 1 and different feature parameters. In the cases of 1), 4), and 7), when no distortion applies to the forms, the loss value is insignificant; the predicted forms and the expected forms are analogous. However, when distortions exist, as in 2), 3), 5), 6), 8), and 9), the loss value is larger. Especially in the cases of 3), 6), and 9), in which distortions happen in two positions, the loss value is 3–10 times larger than that in the training dataset, and there are obvious differences between the two corresponding forms in geometric renderings, although the overall tendency looks synchronous. However, for the basic straight, gradual, or conical shape, the performance also varies. In the cases of 1), 2), and 3), when the basic form is straight, the loss value is smaller than that in the cases of 4) to 9). This is because, mathematically, the straight form means there are no significant changes between the higher layers and the first layer. As a result, after the distortions are applied, the changes are smaller, and the design features are simpler for the neural network to learn.

Generally speaking, the overall loss is in anticipation, and the predicted forms are very similar to the ground truth forms. The neural network was successfully built and trained, and it learned and practiced the different design features in the generated dataset.

2.3. Result evaluation

Using the trained neural network as a transforming tool, one can explore the generative design process, generating forms by inputting different feature parameters, such as combining two feature parameters to represent the mixture of the two styles.

First, to test the influence of the feature parameters on the forms, a series of forms are generated by inputting a single feature parameter with different scales, such as $(0.4, 0, 0, 0, 0, 0, 0, 0, 0)$ or

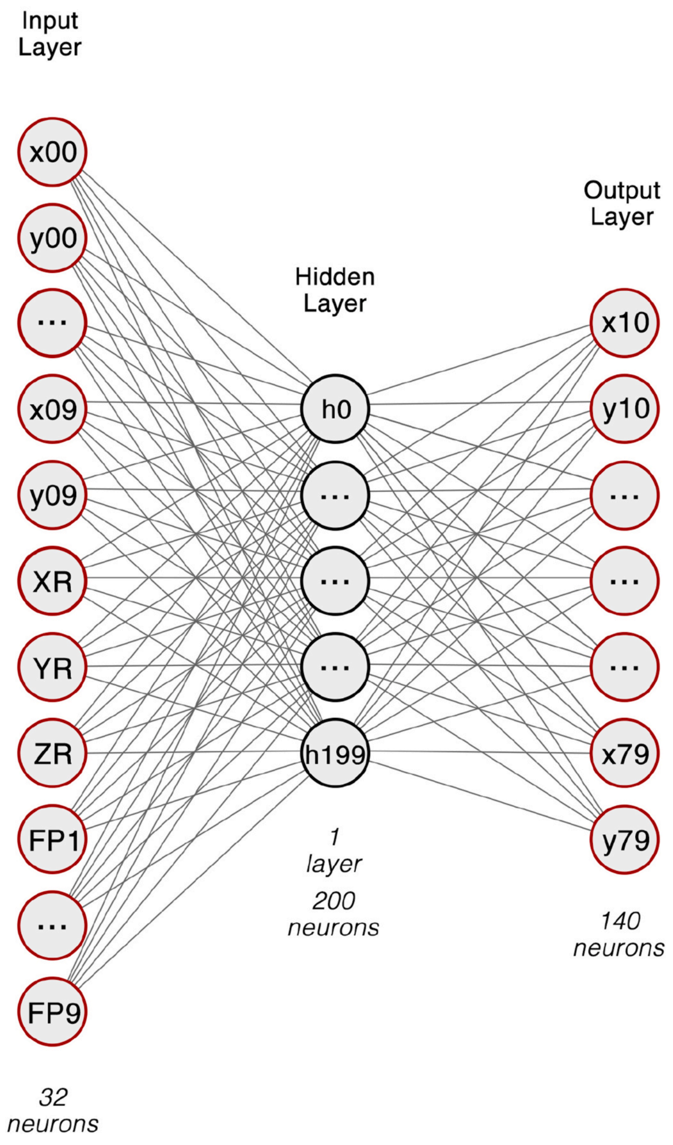

Fig. 8. Neural network structure.

$(0, 0, 0, 0, 0, 0, 0, 0, 0.8)$, as Fig. 11 shows.

In the case of 1-1), when all feature parameters are 0, an initial form is generated, showing a basic form. It is more complicated than a simple form because it allows one of the 9 geometric operations to deform it and neutralize the curved parts while maintaining the ability to accept the other 8 operations equally with different feature parameters. Then in the cases from 1-2) to 1-6), 5 forms with the first feature parameter gradually changed from 0.2 to 1 are generated. The curved parts are gradually neutralized and disappear, and finally, the form becomes straight, as expected. Additionally, in the cases from 2-2) to 2-6), in which conical forms with distortions from the middle up are generated, the same phenomenon can be observed for this geometric operation. As the feature parameter increases, the curvature of the form gradually changes, resulting in smooth parts.

Therefore, inputting a single feature parameter in different scales with real numbers from 0 to 1 allows the form to be generated based on different degrees of the corresponding operation. Designers can adjust the scale of the design strategies by inputting different numbers and generating a series of forms. The generation of the desired forms demonstrates that the neural network is suitable in data-driven generative tasks.

Next, we test form generation with two feature parameters at the same time. Fig. 12 shows the selected output forms with two feature parameters, where the feature parameters are both 1. However, the results are not as convincing as expected. This is exemplified by 4), whose form is difficult to understand by looking into the feature parameters and their corresponding operations. Moreover, in the cases of 2), 3), and 5), in the middle of the forms, it is difficult to explain why the UV NURBS curves in the front and back run in different directions. Only the form in the case of 1) looks reasonable, being gradual (straight + conical) with middle-and-up distortion (no distortion + middle-and-up distortion).

Further, a special form that inputs all feature parameters as $(1, 1, 1, 1, 1, 1, 1, 1, 1)$ is generated. But the neural network outputs all numbers as 0, which means the generation process fails, without any results. Therefore, we can assume based on the above observation that, when a feature parameter is inputted, the corresponding operation acts on the form. But in the training dataset, all operations squeeze and shrink the form to some degree. Thus, if too many operations act on the form too much, that is, the sum of all 9 feature parameters exceeds a certain number, the form cannot exist with legal operations.

Accordingly, in the next test, whose results are shown in Fig. 13, the sum of the two inputted feature parameters is always 1. The results show a clear path of change in the forms from the case of 1) to the case of 6).

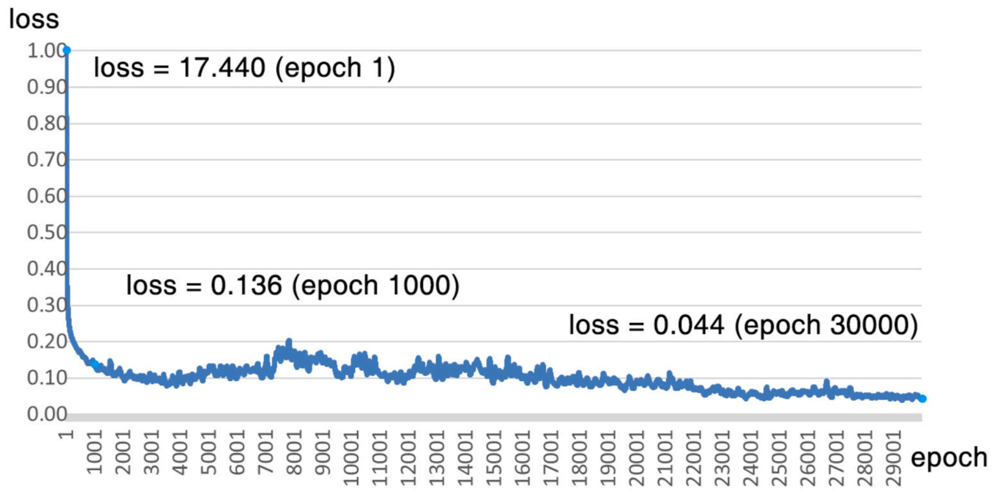

Fig. 9. Training epoch and loss.

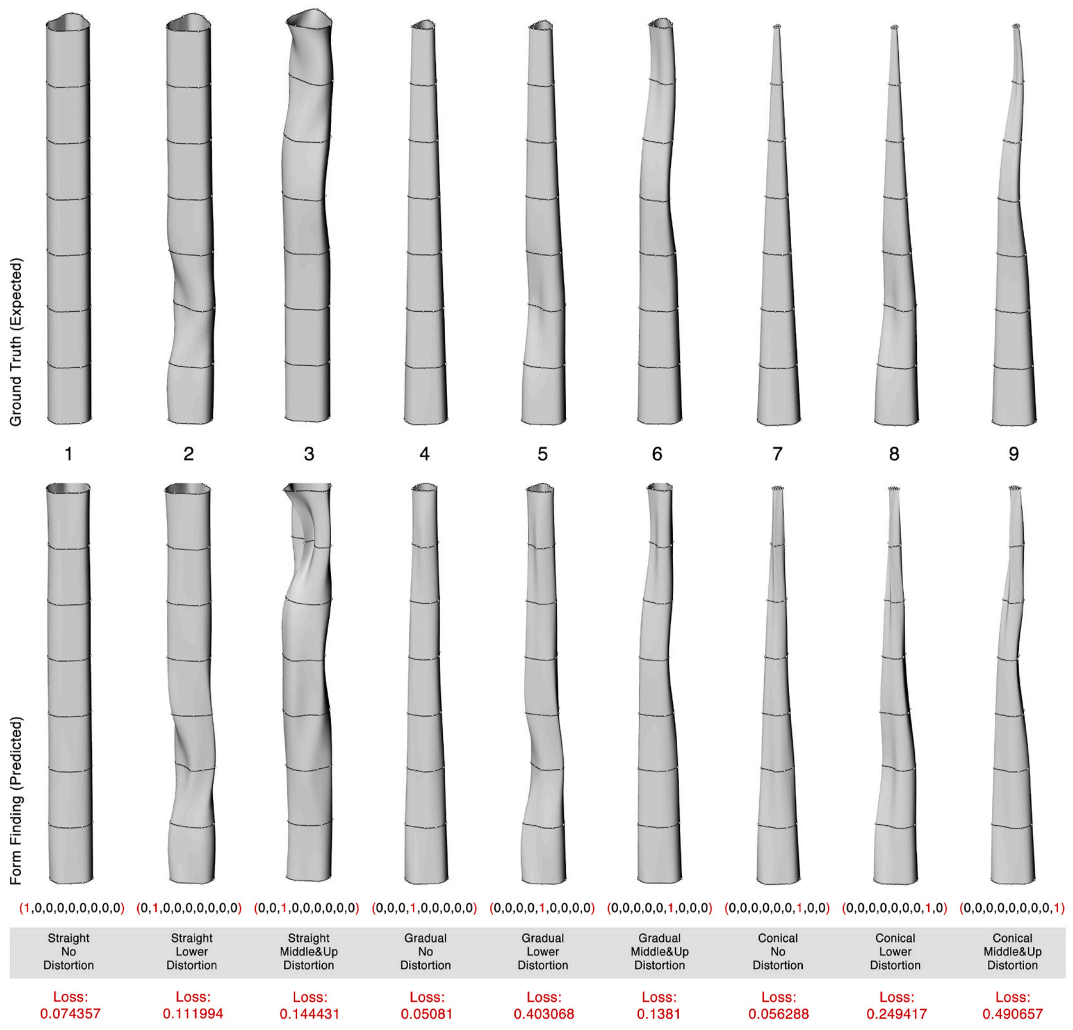

Fig. 10. Expected forms and predicted forms from test dataset.

Unlike in the previous forms, feature parameters with different proportions lead to the generation of a series of forms, showing the intermediate forms between two styles.

Therefore, inputting two feature parameters in different proportions with a sum of 1 generates a form in a combined style, presenting both features. Designers can adjust the proportion of the two design strategies by inputting different numbers, and they can generate a series of forms, even if the same feature parameters are not included in the training dataset. It also shows the ability of the neural network to infer nonexistent design strategies by learning from the existing data.

2.4. Methods comparison

With the initial experiment above, we fine-tuned the hyperparameters for the training process while comparing the performance of each combination of hyperparameters, in order to find the neural network settings with highest accuracy.

First, we compared the efficiency of ANN and GAN. As we state in the objectives, the training cost of our vector-based ANN model should be lower than pixel-based GAN models. Table 1 shows the comparison of the two models. The initial hyperparameters for our ANN model contains: 2 layers (1 hidden layer); 200 neurons in the hidden layer; and Adam optimizer with the learning rate of 0.001. We applied pix2pixHD [55] as the GAN model for comparison. Using an i7-6700HQ CPU, the training of our ANN model cost 0.8 h for 30000 epoch. To train the GAN model, we transformed the curvatures into 8 groups of black-and-white images, thus it required the training of 7 models for generating the surface information. With the acceleration of a Tesla-P100 GPU, the training still cost around 60 h in total. This comparison supports our assumption that our ANN model is light-weighted and fast for training.

Second, besides ANN, there are other vector-based machine learning models being used in solving design-related problems. Therefore, we compared the accuracy of different vector-based machine learning models with our ANN model. Table 2 shows the 5-fold cross-validation accuracy of each model. Compared with other models such as the Random Forest Regressor and the Linear Regressor, our ANN model presents a highest accuracy of 98.05%.

Furthermore, we adjusted the number of layers to compare the accuracy of each ANN model derived from our initial model (Table 3). It shows a highest accuracy of the initial model with 2 layers (1 hidden layer). Thus, the 2-layer ANN model performs best in our generative task.

Third, the hyperparameters during the training process also influence the accuracy. Table 4 shows the comparison of the training results with different optimizers. According to the 5-fold cross-validation, we selected Adam optimizer since it showed a highest accuracy with a fixed number of training epoch.

With Adam optimizer, we further changed the learning rate and compared the performance of each model (Table 5). When the learning rate was set to 0.001, it resulted in a highest accuracy. Therefore, we set our ANN model with 2 layers and Adam optimizer with the learning rate of 0.001.

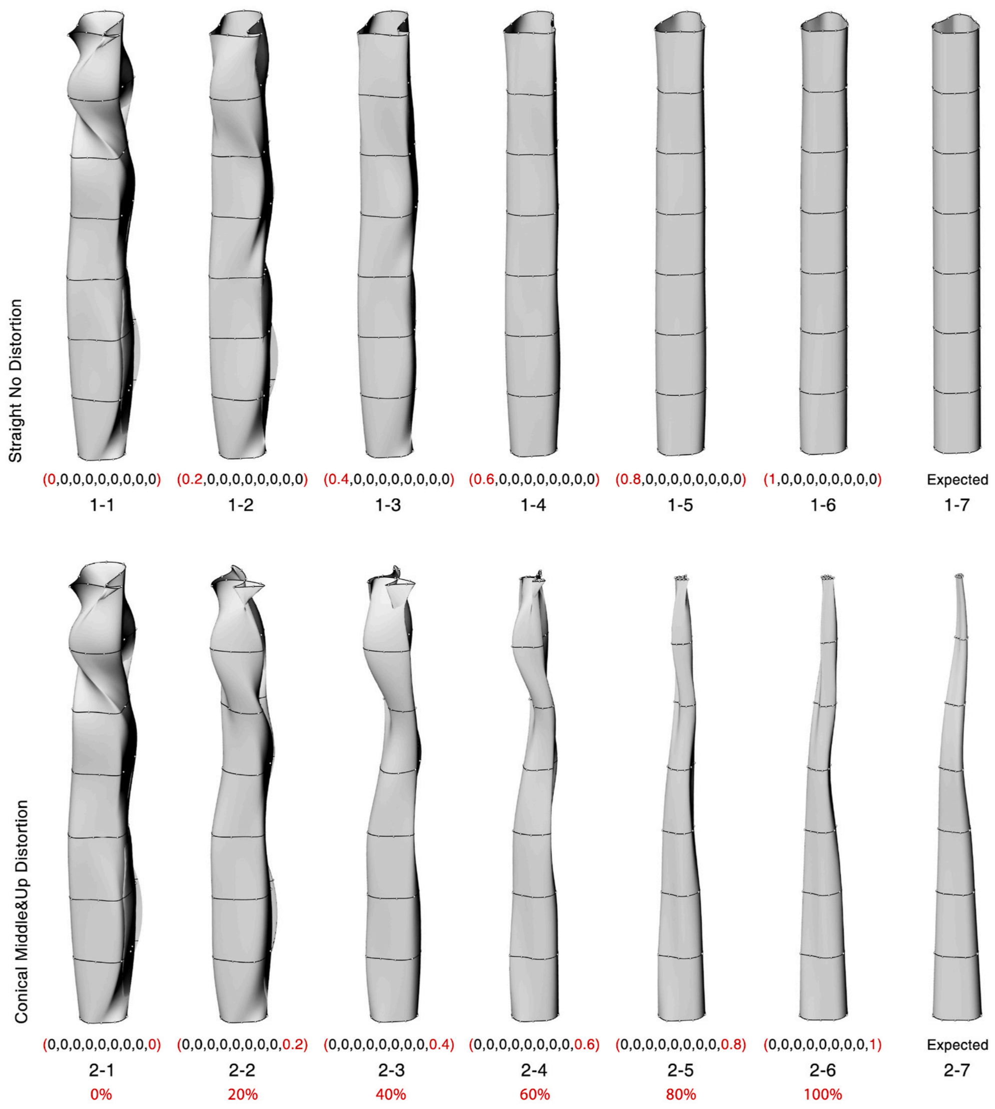

Fig. 11. Generating with gradually changed feature parameters.

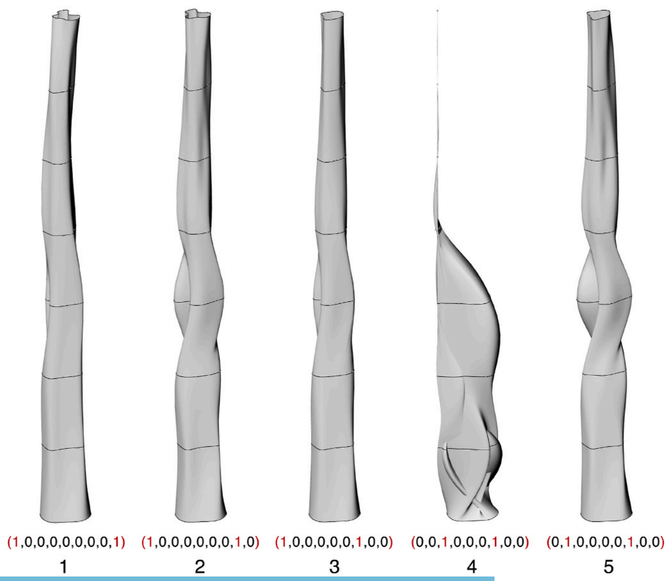

Fig. 12. Generating forms with two binary feature parameters.

3. Implementations

Based on the generated results above, it has been shown that the artificial neural network has the ability to learn the features of the 3D forms in the training process. As a new method for finding forms, various geometries can be created by inputting certain site conditions, while combining with adjustable design strategies. Moreover, in order to add more practical application meanings to the neural network, it is necessary to use real-world data to test and develop the feasibility of the neural network. To that end, in addition to the training of the generated data, data on existing buildings in the real world were collected as the training data.

However, when the influencing factors on the forms of the existing buildings are considered, there are several choices for defining the feature parameters, such as the completion year, architectural firm, location, material, or function. But since the amount of data is limited, and an excess of feature parameters causes the neural network to fail and output confusing results [56] only one type of feature parameter is set to be included in each training. In addition, in order to improve the neural network's future prediction ability on a wider scale, the completion year and the architecture firm for individual styles are defined as the feature parameter in two separate training sessions below, representing the general building context. Both of the generating processes can be

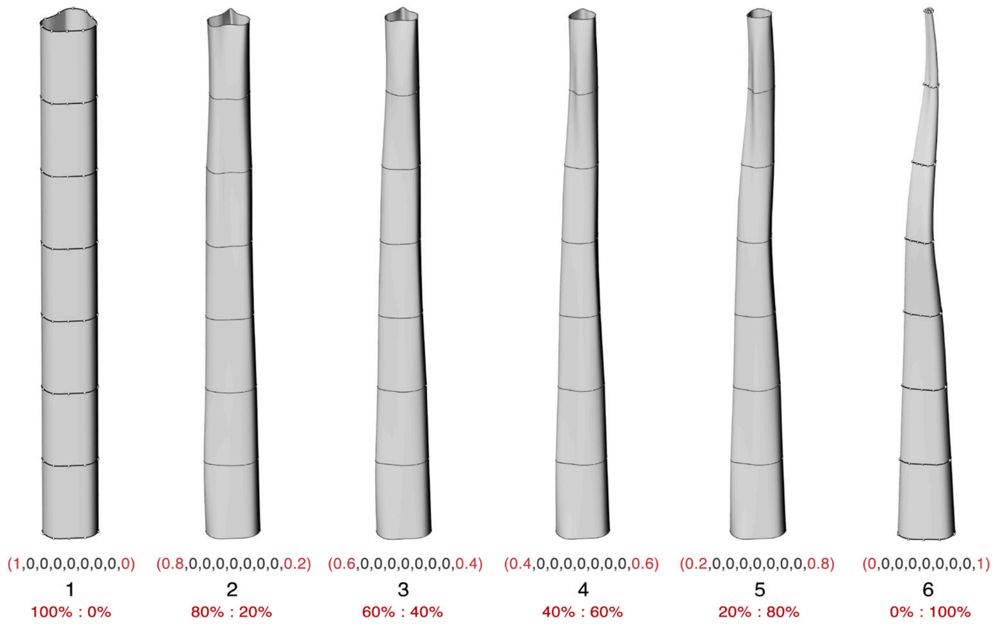

Fig. 13. Generating forms with two real feature parameters.

Table 1
Comparison of the training cost in ANN and GAN.

	Resolution	Device	Training Epoch	Training Time
ANN	numeric	i7-67000HQ (CPU)	30000	**0.8 h**
GAN	512*512 (pixel)	Tesla-P100 (GPU)	100*7	8.5*7 h

Table 2
Comparison of the accuracy of different machine learning methods.

	5-Fold Cross-validation Accuracy (%)
ANN	**98.05**
Decision Tree Regressor	95.57
Bagging Regressor	96.62
Random Forest Regressor	96.58
Extra Trees Regressor	96.96
Linear Regressor	97.85

Table 3
Comparison of the accuracy of ANNs with different numbers of layers.

	5-Fold Cross-validation Accuracy (%)
1-layer ANN	97.45
2-layer ANN	**98.05**
3-layer ANN	97.39
4-layer ANN	95.43

Table 4
Comparison of the accuracy of ANNs with different optimizers in training.

	5-Fold Cross-validation Accuracy (%)
Adadelta	74.97
Adagrad	88.26
Adam	**98.05**
Ftrl	95.86
Gradient Descent	88.05
RMSProp	97.34

Table 5
Comparison of the accuracy of ANNs with different learning rate for Adam optimizer in training.

	5-Fold Cross-validation Accuracy (%)
0.0001	96.98
0.001	**98.05**
0.01	81.15
0.1	56.14

summarized as three main steps: 1) data processing, 2) neural network training, and 3) design generation.

3.1. Training with completion years

In this training session, the feature parameter is defined as the completion years of the existing buildings. A sample of 90 high-rises completed from 1990 to 2030 worldwide was gathered and used as the training dataset for predicting the general development trends of global tower designs.

First, at the data preparation stage, it is necessary to edit the models to match the neural network's data structure. As Fig. 14 shows, the original model collected from the internet is usually too complicated, with a lot of details that have little influence on the overall form. Therefore, a simplified model needs to be built by extracting the main section curves and lofting them together. In detail, the model simplification process starts with closing all geometries in the original model, resulting in a new model that only contains closed meshes. Then, eight boundary sections are generated by contouring the new model from bottom to top as curves. Last, the section curves are lofted together as the final simplified model.

After obtaining the proper models, the completion years of the buildings are converted into machine-learnable data by specifying a real number series from 0 to 1 as the year from 1990 (or earlier) to 2030 (or later) for standardization. As there are only 5 towers in the dataset built before 1990, it is more efficient to set them all as the minimum year of 1990 to avoid the waste of parameter range. In addition, the maximum year is set as 2030 to make the neural network predict the form of towers in the future by inputting a feature parameter larger than the values in the training dataset. Therefore, the input data contains 24 neurons: 20

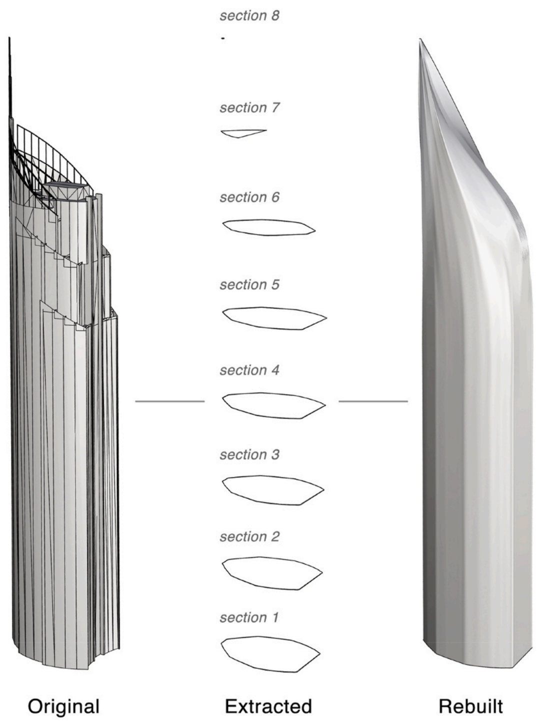

Fig. 14. Model simplification.

real numbers for the controlling points in the first layer, 3 real numbers for the scaling ratios, and 1 real number for the completion year.

Nevertheless, another problem is the lack of training data. Compared with the 900 generated training forms, 90 collected forms might not be enough to train the same neural network. But since a form has no orientation, rotated forms with the same model are also acceptable as separate data. This fact allows for 270 more forms to be created by rotating the original forms by 90, 180, and 270°. Then, the controlling point data extracted from the 360 total buildings are prepared for the training process. After training, the loss value reached the same level compared to the previously generated neural network, indicating the new neural network should have the same ability to generate forms with different feature parameter values.

With the trained neural network, the generative design process can be achieved using a similar method for the previous neural network. The difference is that, besides the 20 parameters for the first layer representing the footprint, the feature parameter contains only the completion year in the new neural network; however, the parameter can be any continuous real number between 0 and 1. Therefore, a set of gradually changed feature parameter values can be assigned to the inputted footprint to generate a series of forms, representing the predicted tower designs in different completion years with the same footprint.

To be specific, in Fig. 15, the case of 2005) is the original simplified model, Q1 Tower in Queensland completed in 2005. With the same footprint but different feature parameters, 41 predicted forms from 1990) to 2030) are listed. The gradual changes in forms are clear from left to right, showing the different simplified design strategies in different years. Although the design should not be similar from year to year, the results show gradual rather than sudden change because the training data is still not sufficient to cover all conditions. As a result, the neural network outputs a mathematically average form between two existing cases, causing the inadequate prediction. However, the remarkable trait of this generative design method is the ability to produce new and variable forms according to the limited training data, giving designers more choices to select from. Especially when upcoming years are used as feature parameters, the predicted forms represent a possible design trend of the future, showing the ability of the neural network to learn and deduce design rules.

However, the design trend sometimes varies according to different boundary conditions. For example, Figs. 16 and 17 show the forms predicted from the footprints of two buildings, Q1 Tower and The Torch. The change tendencies of the forms display different results. In the case of Q1 Tower, Form 1990) is more distorted than Form 2030), while in the case of The Torch, Form 2030) is more distorted than Form 1990). This phenomenon indicates that footprint also performs a major role in predicting, even if it varies a little in the two cases.

3.2. Training with design styles

In this experiment (also presented at the DigitalFUTURES Shanghai 2019 Workshop Group 7-2), the feature parameter is redefined as the design styles of different architects. To build the training dataset, 184 models of completed buildings are collected, representing 7 groups of designers (Le Corbusier, MAD, Gensler, Zaha Hadid Architects, Tadao Ando, BIG, and SOM). However, the objectives are expanded to common architectural works, thus the combined proportions of different architects can generate forms with mixed design styles (Fig. 18).

Next, the models gathered online should be simplified and then unified into closed surface. In contrast to the tower collections, the architectural works in general increase the complexity in forms and

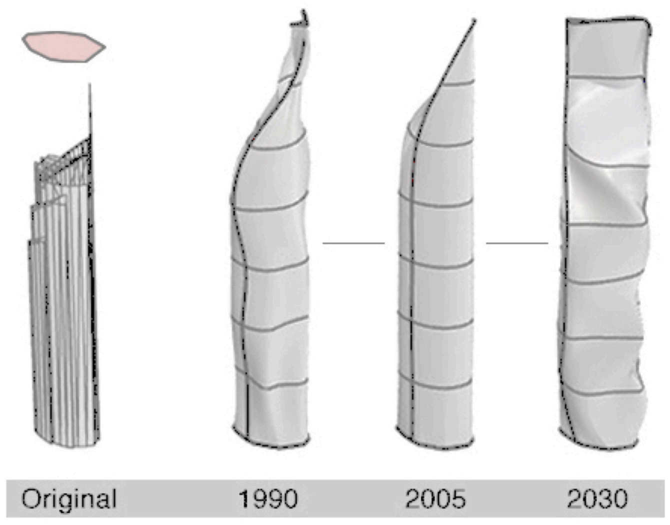

Fig. 16. Predicted forms with the footprint of Q1 Tower.

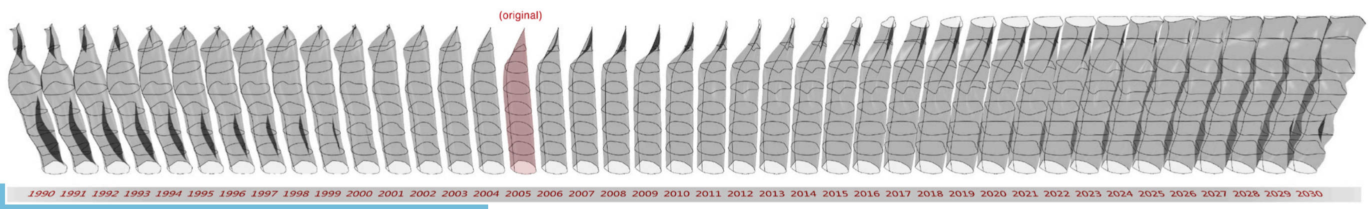

Fig. 15. Predicted forms with the same footprint but different feature parameters (completion year).

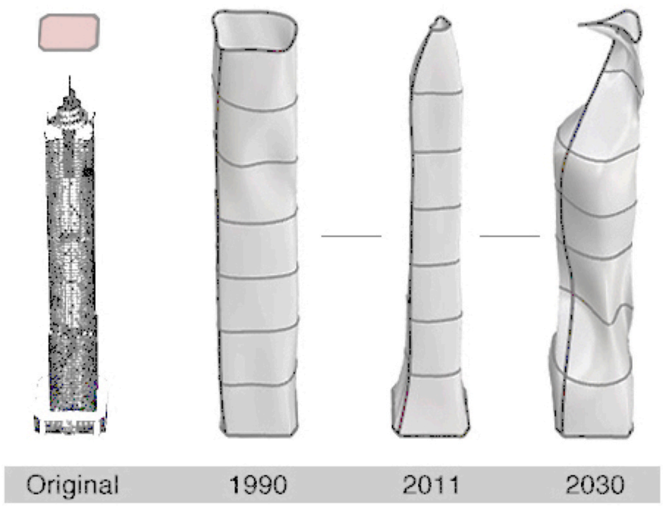

Fig. 17. Predicted forms with the footprint of The Torch.

design methods while decreasing the values of building levels. Specifically, when extracting the 2D vectors form 3D models, the previously stated rule of an 8layer*10item grid of points may not be enough to summarize the complicated forms. Thus, the controlling points extracted from each building are increased to 10 layers with 20 points per layer, indicating that the 3D mesh can be translated into a collection of 400 real numbers, which develops more inclined mapping between the original forms and the training data (Fig. 19).

In addition, since the difference of the heights of the collected buildings is smaller than that of the towers in the generated case, the height set in the bounding box for the data reparameterization (10 units in the *Z* direction) is not suitable for scaling the values of this model collection. As a result, the unit value in the *Z* direction is decreased to 2, in order to obtain a more even numeric distribution of Z Ratio. Thus, after the controlling points are aligned and rebuilt, the training data structure is then changed into $x0, y0, \ldots, x199, y199$, X Ratio, Y Ratio, Z Ratio, and 7 feature parameters - that is, a series of 410 real numbers for each building.

As a result, the input data contains 50 neurons - 40 real numbers for the controlling points in Layer 0, 3 real numbers for the scaling ratios, and 7 feature parameters indicating 7 architecture companies. The number of neurons in the hidden layer is also enlarged to 500 due to the increased complexity of the data structure. The final outputs involve 360 neurons from Layer 1 to Layer 9 with 180 controlling points in total (Fig. 20). Furthermore, due to the complicated forms of the distinguished styles, the rotation method is also used for strengthening the training data; 1288 more forms are created by rotating the original models by 45, 90, 135, 180, 225, 270 and 315°. Finally, 1472 stylized forms in total are prepared for the next stage of the neural network training.

Before training, we implemented a k-means clustering method [57] with a *K* value of 7 to test whether the seven groups of forms are similar or can be easily distinguished by seven clusters. Specifically, we implemented the script in Python with Numpy for numeric processing. For the data structure, we maintained the 410-dimensional vector for

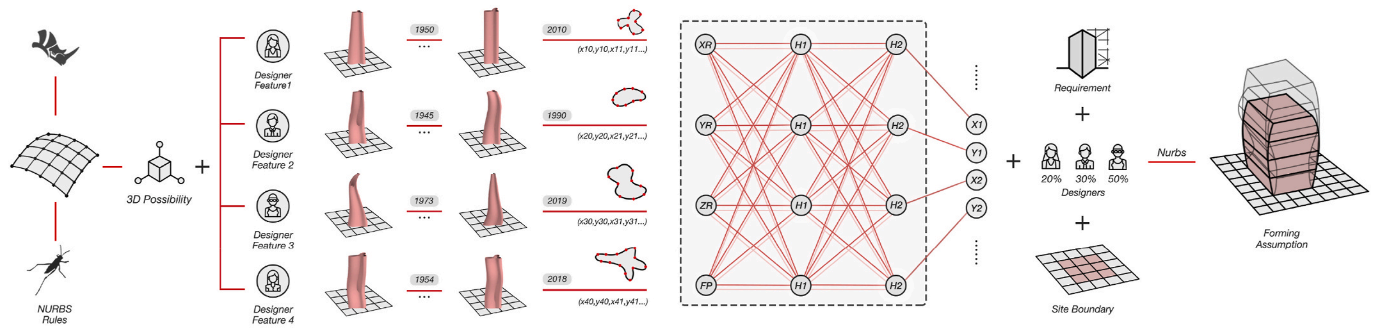

Fig. 18. Workflow of the training and generative design with different architects' design styles.

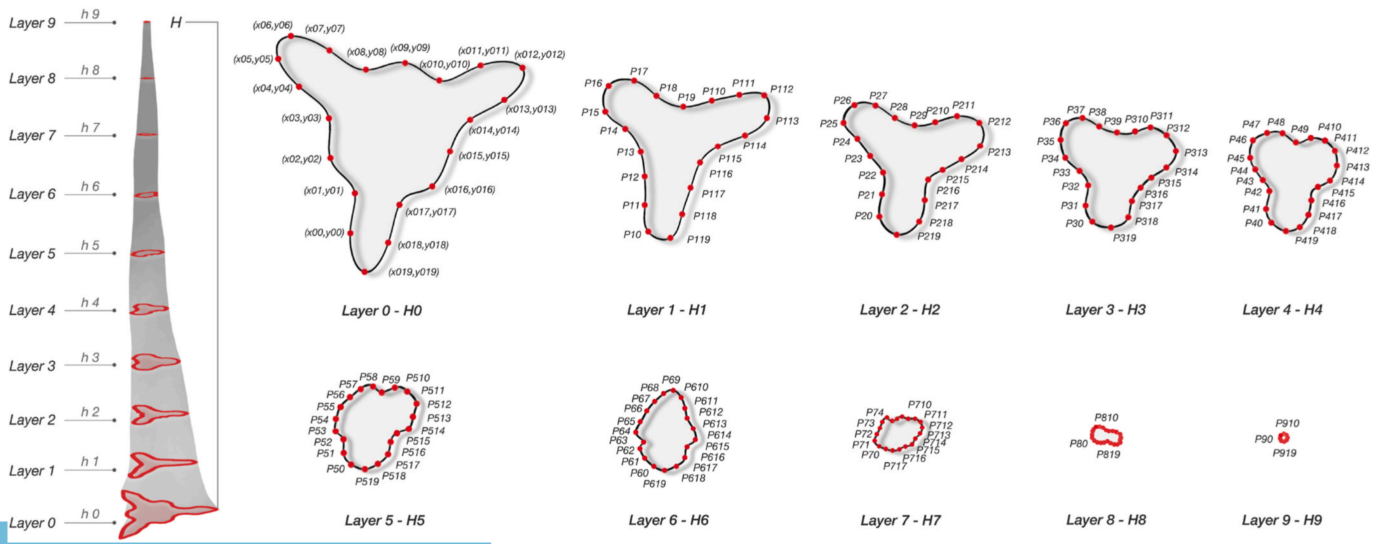

Fig. 19. Controlling points extraction process with 10 layers and 20 points for each layer.

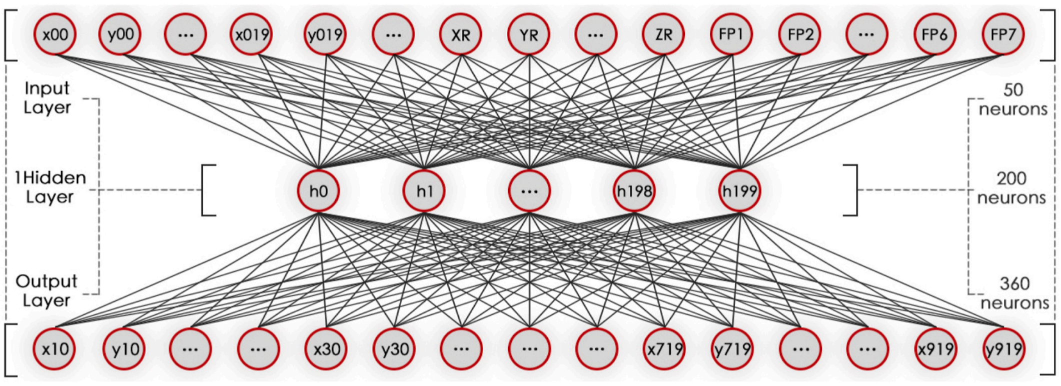

Fig. 20. Neural network structure with FC layers.

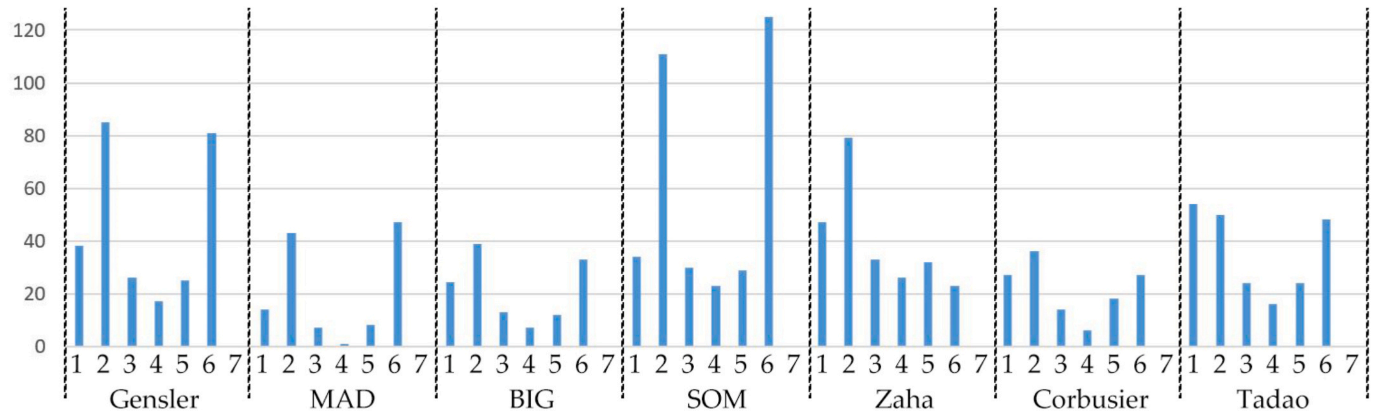

Fig. 21. Distribution of the clusters for each group of forms.

each form while standardizing the values (Eq. (4)). After clustering, Fig. 21 shows the distribution of the members in each cluster. According to the distribution, we used purity measure [58] to evaluate the accuracy. The result shows an overall accuracy of 26.06%, which is higher than the random-guess accuracy of $\frac{1}{7}$ (14.29%).

$$x_{standardized} = \frac{x - mean(x)}{standarddeviation(x)} \tag{4}$$

Additionally, according to Fig. 21, each group of forms is gathered into different clusters, with a tendency to be certain styles. For example, in the group of Gensler, MAD, BIG, and SOM, a large amount of forms are grouped into clusters 2 and 6. However, most forms in the group of Zaha are collected into clusters 1 and 2, whereas most forms in the group of Le Corbusier and Tadao are gathered into clusters 1, 2, and 6. We might understand this phenomenon as the classification of design styles, in which cluster 2 represents the general architectural design style for all groups and cluster 6 represents the commercial design style for companies such as Gensler and SOM, whereas cluster 1 represents the emerging design style with more complex forms and more for individual architects, for example Zaha, Le Corbusier, and Tadao. Therefore, the features of each group might be represented in a higher dimension that combines different design styles.

Generally speaking, however, from the k-means clustering experiment above, we can reach a primary conclusion that the features in the dataset can be learned with machine learning techniques. Although the clustering does not indicate a high accuracy to distinguish the features, it is still available to use networks to further learn the features.

After training, the loss value reached a relatively stable level compared with the neural network trained by the generated data, thus showing the ability of the neural network model to generate forms with various mixtures of design styles. According to the 7 styles and their corresponding parameters in the features, different combinations of the influencing parameters are tested in the next stage.

In the generative design process, Fig. 22 shows a path of predicted results in the cases from 1) to 7) with the same footprint and height conditions and with two gradually changed proportions of the Le Corbusier style and the MAD style with a feature parameter sum of 1. The form designed by Le Corbusier is usually centroid and linear [59], whereas MAD usually prefers a non-linear form with complex surfaces [60].

As our result shows, a higher proportion of the Le Corbusier style led to simpler and more regular generated forms. However, increasing the proportion of MAD caused the geometries to become more distorted and finally result in a curved form. These traits visually reflect the differences in the design styles of these two architects. By training with self-collected datasets, designers can apply this method to combine two or more design styles into one, to create inspirational forms for design exploration.

Similarly, with the same footprint and height, more stylized combined forms are generated and compared during this process. Fig. 23 also presents the changeable feature parameters between Gensler and Tadao Ando and between BIG and Zaha Hadid Architects, leading a series of hybrid forms with intermediary mappings of contrastive styles. Generally speaking, the generated results point out a tendency in Tadao Ando's style to use more regular geometries, whereas others present a preference for uncommon curvatures. This observation matches the design styles by those architects [61–64]. A more detailed cross-comparison can be achieved similarly by testing other combinations of the feature parameters, thus providing designers with an analytical tool for design styles.

To further verify the above conclusions, we designed a questionnaire

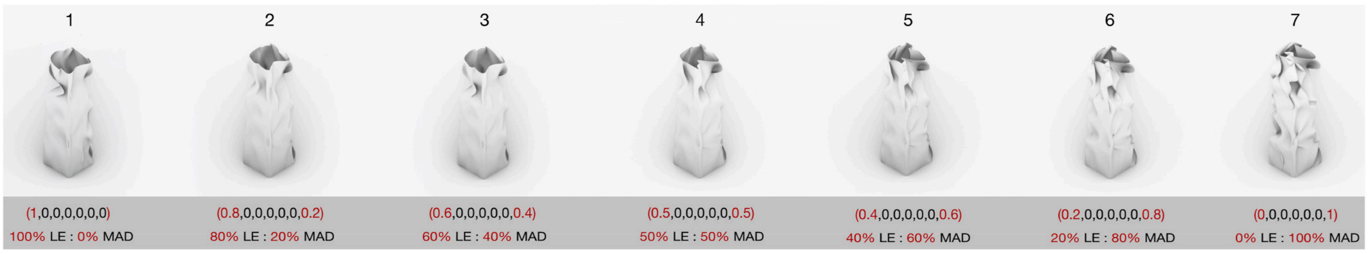

Fig. 22. Predicted forms with changing feature parameters of Le Corbusier and MAD on the same footprint.

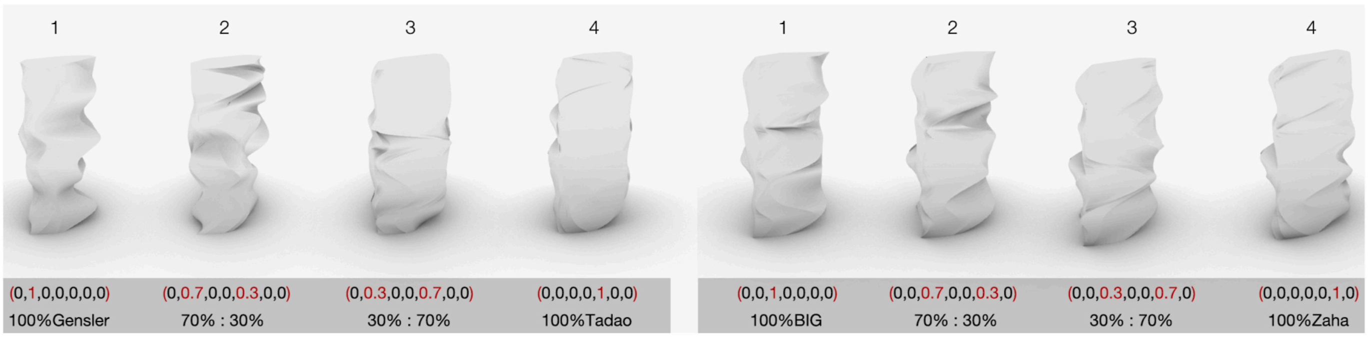

Fig. 23. Predicted forms with different changing feature parameters for other architects on the same footprint.

with three binary evaluations for the generated forms in Figs. 22 and 23, and we asked trained architects to distinguish the design styles. Each question asks a participant to match the forms with the architects (Fig. 24). The participant were required to have a bachelor's or master's degree in architecture or a related field, and they were required to have no prior knowledge of this research. A total of 40 architects participated in this questionnaire.

According to the results, 100% of the participants successfully distinguished the Le Corbusier form from the MAD form and 90% of the participants distinguished the Gensler form from the Tadao form, but only 55% of the participants successfully distinguished the BIG form from the Zaha form. When questioned about the reason, most of the participants mentioned that they rely on the complexity of the forms to identify the styles. For example, the right form in the first question is more complex than the left form, thus it is regarded as the MAD form instead of the Le Corbusier form. The second question is similar in that the more complex form is marked as the Gensler form rather than the Tadao form. However, in the third question, the two forms are similar; even though the Zaha form is usually more complex than the BIG form, the participants could not distinguish these two similar forms. This result reveals that the design styles of BIG and Zaha have some common points from the perspective of data science and machine learning.

Moreover, when considering the geometry parameters, the generated results are not always the same for different heights with fixed feature parameters. In Fig. 25, the increase of height decreases the surface complexity and the degree of distortion of the output geometries. This phenomenon indicates that the height setting also has an influence on predicting the forms.

Lastly, for further testing the applicability of the neural network on an urban scale larger than a singular architectural unit, experiments are performed on different city patterns. Based on the generating effects of the two real-world training neural networks, 3 urban functional blocks and an ideal city panorama are both created for inspiring the city designers at an early stage (Fig. 26 and Fig. 27).

In summary, in the training of real-world data, this paper has shown the feasibility of applying neural networks in 3D generative design. Different settings and inputs of feature parameters can be used to train variable neural networks and generate forms according to the requirements of designers. Table 6 shows the computation cost of the three cases discussed in this paper, which supports that our ANN model is light-weighted and requires small computation cost.

4. Conclusion and discussion

The artificial neural network is a novel tool for 3D generative design, especially when only the input and output design data is given, rather

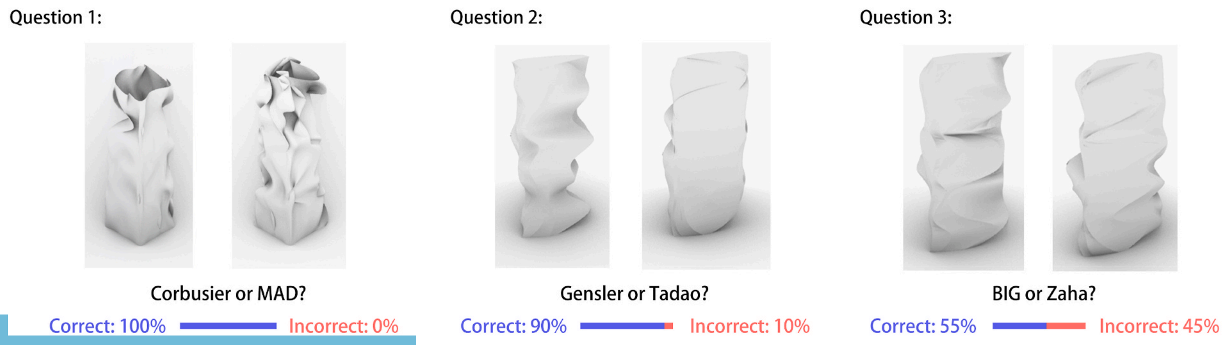

Fig. 24. The questionnaire contained three questions asking the participants to select the architects for the forms.

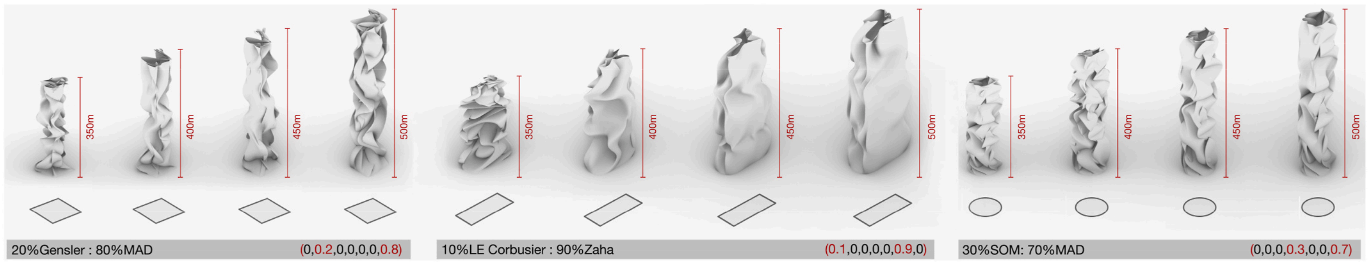

Fig. 25. Predicted forms with gradually changed height and a fixed feature parameter.

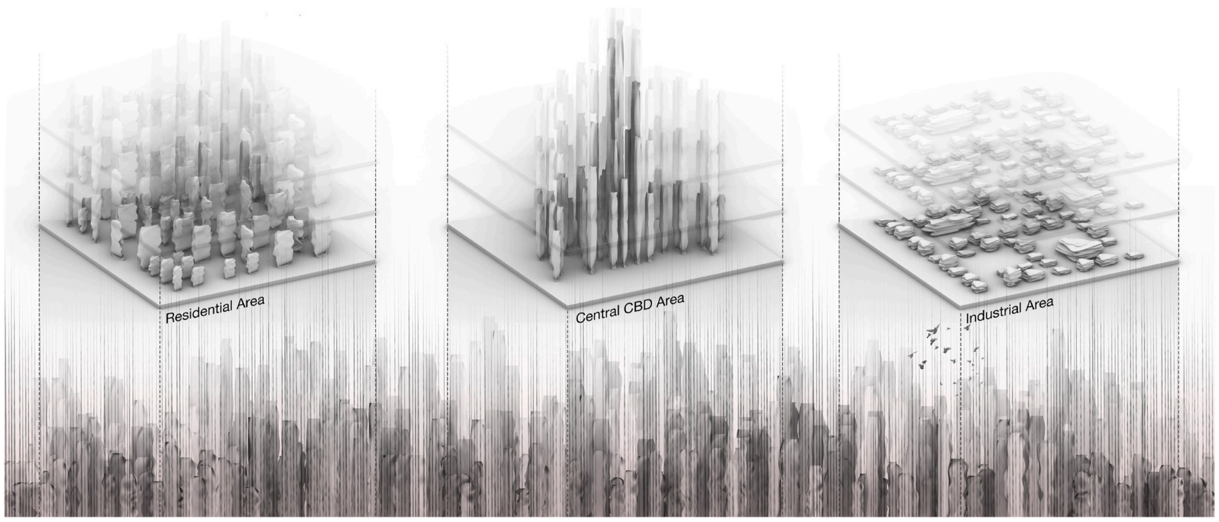

Fig. 26. Generation of functional urban blocks.

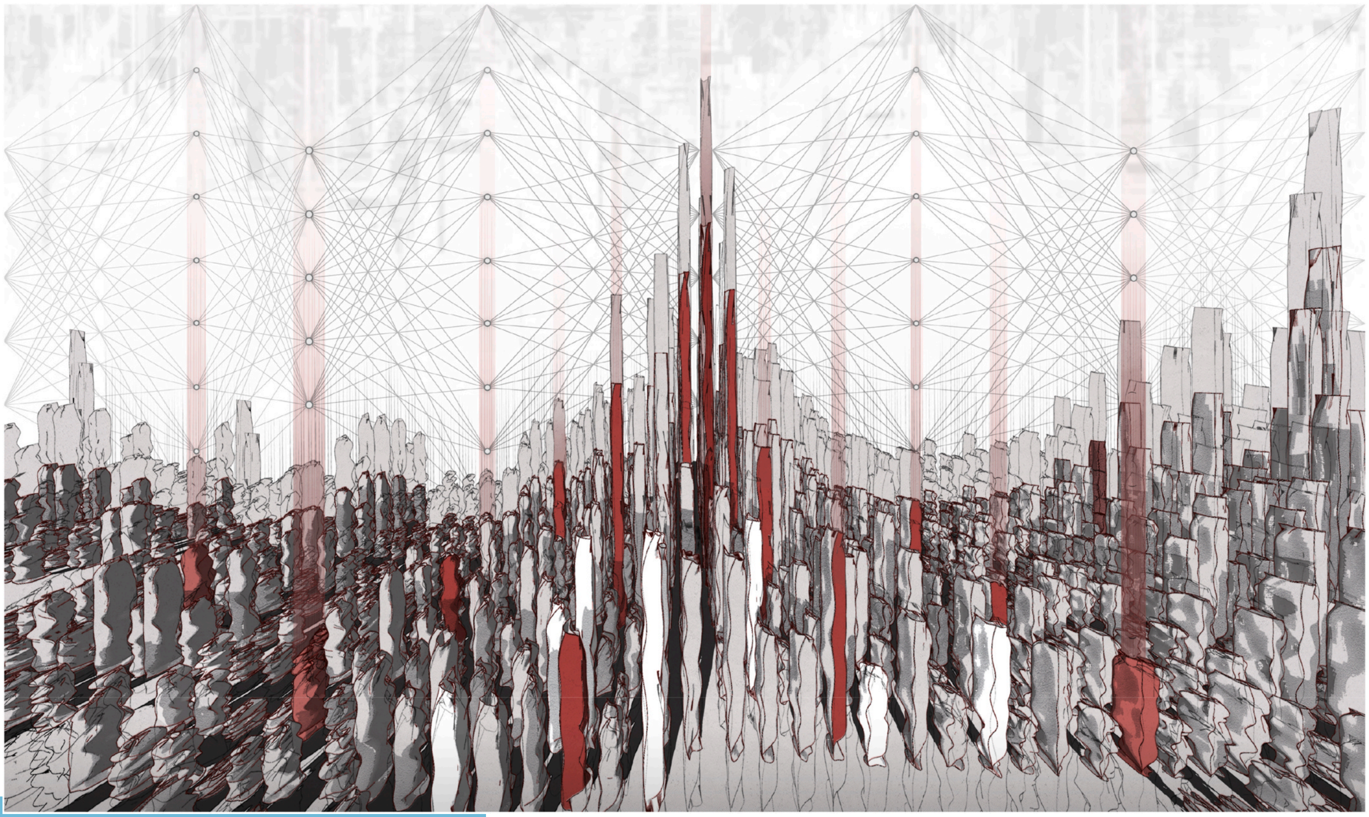

Fig. 27. Predicted panorama of a city.

Table 6
Training device and computation cost of each case in this paper.

	Device	Number of Samples	Training Epoch	Training Time
Generated Data	i7-67000HQ (CPU)	900	30000	0.8 h
Completion-Year Data	i7-67000HQ (CPU)	360	30000	0.3 h
Design-Style Data	i7-67000HQ (CPU)	1472	30000	3.2 h

than the clear design strategies. Customized data and neural network structures help to translate the design forms into computational data and map design features to several controllable parameters. By inputting different feature parameters, either a single parameter with different scales or combined parameters with different proportions, this generative design machine can generate 3D forms according to the features given by designers.

When the neural network is trained with the data collected from existing architecture, it can be used to learn and infer design data with different features, helping architectural researchers digitally redefine design strategies hidden among massive and variable design data. Then, designers can easily apply the trained neural network to the generation of forms and quickly generate designs with different features.

However, training a highly accurate neural network to generate design solutions requires a large amount of data, which shows the limitation of this data-driven generative design method, especially given the lack of architectural datasets [30]. Compared with other machine learning tasks, such as predicting medical [65] or facial data [66], the data collecting process for design tasks is harder and more time-consuming, and it requires experts with professional knowledge in design domains [67]. Therefore, there is still much work to be done before AI masters architectural design, from the primary design stages to the deeper and more detailed tasks. Therefore, this research raises the possibility of transforming the features in the design process into machine-learnable formats and applying a neural network to assist designers in the early design stages.

In the future, the application of 3D generative design via machine learning methods mostly includes design generation in the early design steps. This helps designers reduce the burden of creating variable forms, such as the urban design, which generates variable forms quickly in batches [68–70]. Besides, to improve the training of the neural network, it is important to develop a more efficient and applicable data collection process for architectural data, for example an automatic system for collecting building information modeling data [71]. Such a resource would increase the accuracy of the model and create a more powerful "machine designer" to assist human designers. Therefore, considering more design features and enlarging the dataset to increase the prediction accuracy are the main tasks in our future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

All real-world training 3D models are collected from Google 3D Warehouse or built by the authors. The selection of the towers refers to the data from the Skyscraper Center. Tensorflow in Python is used as the programming framework for the neural network.

This article is supported by the National Natural Science Foundation of China (Grant No. U1913603), and Shanghai Science and Technology Committee (Grant No. 18DZ1205604). Special thanks to all students (Chao Weng, Donglai Yang, Feng Shi, Hainan Yan, Hanyong Xu, Jiageng Chen, Jiewei Li, Li Yang, Qi Yang, Shilong Zhu, Shiqi Liang, Xiaobai Ji, Xinxing Chen, Xinyu Xia, Yibo Zhong, Yifan Huang, Yue Ren, Zhiming Niu, Zilin Zhou) and the teaching assistant (Youyuan Luo) in the DigitalFUTURES Shanghai 2019 workshop group 7-2!

References

[1] M. Carpo, The Alphabet and the Algorithm, Mit Press, 2011. ISBN 9780262515801.
[2] M. Carpo, The Digital Turn in Architecture 1992-2012, John Wiley & Sons, 2013. ISBN 9781118425916.
[3] M. Carpo, The Second Digital Turn: Design beyond Intelligence, MIT press, 2017. ISBN 9780262534024.
[4] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (6) (1953) 1087–1092, https://doi.org/10.1063/1.1699114.
[5] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680, https://doi.org/10.1142/9789812799371_0035.
[6] J.H. Holland, et al., Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT press, 1992, https://doi.org/10.7551/MITPRESS/1090.001.0001.
[7] I.-C. Yeh, Architectural layout optimization using annealed neural network, Autom. ConStruct. 15 (4) (2006) 531–539, https://doi.org/10.1016/J.AUTCON.2005.07.002.
[8] H. Zheng, Y. Ren, Architectural layout design through simulated annealing algorithm, in: Proceedings of the 25th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA), 2020, pp. 275–284. http://papers.cumincad.org/cgi-bin/works/paper/caadria2020_024.
[9] F.O. Sonmez, C. Tan, Structural optimization using simulated annealing, Simulat. Annealing (2008) 281–306, https://doi.org/10.5772/5567.
[10] G. Barczik, R. Kruse, Shifting design work from production to evaluation - an evolutive design tool, in: *Proceedings of the 34th eCAADe Conference*, 2016, pp. 109–115. http://papers.cumincad.org/cgi-bin/works/paper/ecaade2016_150.
[11] L.G. Caldas, L.K. Norford, A design optimization tool based on a genetic algorithm, Autom. ConStruct. 11 (2) (2002) 173–184, https://doi.org/10.1016/S0926-5805(00)00096-0.
[12] D. Whitley, A genetic algorithm tutorial, Stat. Comput. 4 (2) (1994) 65–85, https://doi.org/10.1007/BF00175354.
[13] T. Hong, Z. Wang, X. Luo, W. Zhang, State-of-the-art on research and applications of machine learning in the building life cycle, Energy Build. 212 (2020) 109831, https://doi.org/10.1016/j.enbuild.2020.109831.
[14] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133, https://doi.org/10.1007/BF02459570.
[15] P. Werbos, Beyond Regression:" New Tools for Prediction and Analysis in the Behavioral Sciences, Ph. D. dissertation, Harvard University, 1974, https://ci.nii.ac.jp/naid/10004070196/.
[16] H. Park, D.Y. Park, Comparative analysis on predictability of natural ventilation rate based on machine learning algorithms, Build. Environ. (2021) 107744, https://doi.org/10.1016/j.buildenv.2021.107744.
[17] R. Zhang, P.A. Mirzaei, Virtual dynamic coupling of computational fluid dynamics-building energy simulation-artificial intelligence: case study of urban neighbourhood effect on buildings' energy demand, Build. Environ. (2021) 107728, https://doi.org/10.1016/j.buildenv.2021.107728.
[18] Y. Liu, Z. Pang, M. Karlsson, S. Gong, Anomaly detection based on machine learning in iot-based vertical plant wall for indoor climate control, Build. Environ. 183 (2020) 107212, https://doi.org/10.1016/j.buildenv.2020.107212.
[19] J. Kim, Y. Zhou, S. Schiavon, P. Raftery, G. Brager, Personal comfort models: predicting individuals' thermal preference using occupant heating and cooling behavior and machine learning, Build. Environ. 129 (2018) 96–106, https://doi.org/10.1016/j.buildenv.2017.12.011.
[20] S.-J. Cao, C. Ren, Ventilation control strategy using low dimensional linear ventilation models and artificial neural network, Build. Environ. 144 (2018) 316–333, https://doi.org/10.1016/j.buildenv.2018.08.032.
[21] M.L. Maher, H. Li, Learning design concepts using machine learning techniques, AI EDAM (Artif. Intell. Eng. Des. Anal. Manuf.) 8 (2) (1994) 95–111, https://doi.org/10.1017/S0890060400000706.
[22] P. Achlioptas, O. Diamanti, I. Mitliagkas, L. Guibas, Learning Representations and Generative Models for 3d Point Clouds, arXiv preprint arXiv:1707.02392, 2017.
[23] N. Umetani, Exploring generative 3d shapes using autoencoder networks, in: SIGGRAPH Asia 2017 Technical Briefs, 2017, pp. 1–4, https://doi.org/10.1145/3145749.3145758.
[24] M. Rucco, F. Giannini, K. Lupinetti, M. Monti, A methodology for part classification with supervised machine learning, AI EDAM (Artif. Intell. Eng. Des. Anal. Manuf.) 33 (1) (2019) 100–113, https://doi.org/10.1017/S0890060418000197.
[25] W. Huang, X. Su, M. Wu, L. Yang, Category, process, and recommendation of design in an interactive evolutionary computation interior design experiment: a data-driven study, Artif. Intell. Eng. Des. Anal. Manuf. (2020) 1–15, https://doi.org/10.1017/S0890060420000050.
[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680. https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[27] H. Zheng, Drawing with bots: human-computer collaborative drawing experiments, in: Proceedings of 23rd International Conference on Computer-Aided Architectural Design Research in Asia, 2018, pp. 1–6. https://www.researchgate.net/publication/325117957_Drawing_with_Bots_Human-computer_Collaborative_Drawing_Experiments.

[28] R. Tian, Suggestive site planning with conditional gan and urban gis data, in: Proceedings of the 2nd International Conference on Computational Design and Robotic Fabrication (CDRF), Springer, 2020, pp. 103–113, https://doi.org/10.1007/978-981-33-4400-6_10.

[29] W. Huang, H. Zheng, Architectural drawings recognition and generation through machine learning, in: *Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture*, 2018, pp. 156–165. Mexico City, Mexico, http://papers.cumincad.org/data/works/att/acadia18_156.pdf.

[30] D. Newton, Deep generative learning for the generation and analysis of architectural plans with small datasets, in: Proceedings of ECAADE SIGRADI 2019, 2019, pp. 21–28, https://doi.org/10.5151/PROCEEDINGS-ECAADESIGRADI2019_135.

[31] H. Kinugawa, A. Takizawa, "Deep learning model for predicting preference of space by estimating the depth information of space using omnidirectional images, in: Architecture in the Age of the 4th Industrial Revolution-Proceedings of the 37th eCAADe and 23rd SIGraDi Conference, 2, 2019, pp. 61–68, https://doi.org/10.5151/PROCEEDINGS-ECAADESIGRADI2019_339.

[32] A. Noyman, K. Larson, A deep image of the city: generative urban-design visualization, in: The 11th annual Symposium on Simulation for Architecture and Urban Design (SimAUD), 2020, pp. 161–168, in: http://simaud.org/2020/proceedings/35.pdf.

[33] K. Steinfeld, K. Park, A. Menges, S. Walker, "Fresh eyes – a framework for the application of machine learning to generative architectural design, and a report of activities at smartgeometry 2018, in: Proceedings of the 18th International Conference on CAAD Futures, 2019, pp. 32–46. http://papers.cumincad.org/data/works/att/cf2019_003.pdf.

[34] G. Rossi, P. Nicholas, Re/learning the wheel: methods to utilize neural networks as design tools for doubly curved metal surfaces, in: ACADIA 2018: Recalibration. On Imprecision and Infidelity, 2019, pp. 146–155. http://papers.cumincad.org/cgi-bin/works/paper/acadia18_146.

[35] M. Turlock, K. Steinfeld, Necessary tension, in: Design Modelling Symposium, Springer, Berlin, 2019, pp. 250–262, https://doi.org/10.1007/978-3-030-29829-6_20.

[36] B.A. Zandavali, M.J. García, Automated brick pattern generator for robotic assembly using machine learning and images, in: Proceedings of ECAADE SIGRADI 2019, 2019, pp. 217–226, https://doi.org/10.5151/PROCEEDINGS-ECAADESIGRADI2019_605.

[37] S. Ji, W. Xu, M. Yang, K. Yu, 3d convolutional neural networks for human action recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2012) 221–231, https://doi.org/10.1109/TPAMI.2012.59.

[38] J. Wu, C. Zhang, T. Xue, B. Freeman, J. Tenenbaum, Learning a probabilistic latent space of object shapes via 3d generative adversarial modeling, in: Advances in Neural Information Processing Systems, 2016, pp. 82–90, in: https://proceedings.neurips.cc/paper/2016/file/44f683a84163b3523afe57c2e008bc8c-Paper.pdf.

[39] D. Newton, Multi-objective qualitative optimization (moqo) in architectural design, in: Proceedings of the 36th eCAADe, 2018, pp. 187–196. http://papers.cumincad.org/cgi-bin/works/paper/ecaade2018_323.

[40] S. Ghadai, A. Balu, S. Sarkar, A. Krishnamurthy, Learning localized features in 3d cad models for manufacturability analysis of drilled holes, Comput. Aided Geomet. Des. 62 (2018) 263–275, https://doi.org/10.1016/J.CAGD.2018.03.024.

[41] H. Zheng, K. An, J. Wei, Y. Ren, Apartment floor plans generation via generative adversarial networks, in: *Proceedings of the 25th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA)*, 2020, pp. 601–610. http://papers.cumincad.org/cgi-bin/works/paper/caadria2020_015.

[42] J. Shen, C. Liu, Y. Ren, H. Zheng, Machine learning assisted urban filling, in: Proceedings of the 25th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA), 2020, pp. 681–690. http://papers.cumincad.org/cgi-bin/works/paper/caadria2020_054.

[43] K.T. Islam, R.G. Raj, A. Al-Murad, Performance of svm, cnn, and ann with bow, hog, and image pixels in face recognition, in: *2017 2nd International Conference on Electrical&Electronic Engineering (ICEEE)*,, IEEE, 2017, pp. 1–4, https://doi.org/10.1109/CEEE.2017.8412925.

[44] P. Achlioptas, O. Diamanti, I. Mitliagkas, L. Guibas, Learning representations and generative models for 3d point clouds, in: International conference on Machine Learning, PMLR, 2018, pp. 40–49. https://arxiv.org/abs/1707.02392.

[45] A. Bidgoli, P. Veloso, Deep cloud. The Application of a Data-driven, Generative Model in Design, arXiv preprint arXiv:1904.01083, 2019, https://arxiv.org/abs/1904.01083.

[46] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Network. 20 (1) (2008) 61–80, https://doi.org/10.1109/TNN.2008.2005605.

[47] W. Jabi, A. Alymani, Graph machine learning using 3d topological models, in: *Proceedings of the 11th Annual Symposium on Simulation for Architecture and Urban Design* (SimAUD), 2020, pp. 427–434. Vienna, Austria, http://orca.cf.ac.uk/131855/.

[48] R. McNeel, " Rhinoceros, NURBS modeling for Windows, Comput. Software (2020). http://www.rhino3d.com/.

[49] A. Wanto, A.P. Windarto, D. Hartama, I. Parlina, Use of binary sigmoid function and linear identity in artificial neural networks for forecasting population density, Int. J. Inform. Syst. Technol. 1 (1) (2017) 43–54, https://doi.org/10.30645/IJISTECH.V1I1.6.

[50] M.H. Hassoun, et al., Fundamentals of Artificial Neural Networks, MIT press, 1995. ISBN 9780262514675.

[51] A. Lorensuhewa, S. Geva, B. Pham, Inferencing design styles using bayesian networks, Ruhuna J. Sci. 1 (1) (2012), https://doi.org/10.4038/RJS.V1I0.71.

[52] D. Piker, Kangaroo: form finding with computational physics, Architect. Des 83 (2) (2013) 136–137, https://doi.org/10.1002/AD.1569.

[53] S. Karsoliya, Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture, Int. J. Eng. Trends Technol. 3 (6) (2012) 714–717. http://www.ijettjournal.org/volume-3/issue-6/IJETT-V3I6P206.pdf.

[54] N. Hovakimyan, F. Nardi, A. Calise, N. Kim, Adaptive output feedback control of uncertain nonlinear systems using singlehidden-layer neural networks, IEEE Trans. Neural Network. 13 (6) (2002) 1420–1431, https://doi.org/10.1109/TNN.2002.804289.

[55] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1125–1134. https://arxiv.org/abs/1611.07004v3.

[56] B. Azhagusundari, A.S. Thanamani, Feature selection based on information gain, Int. J. Innovative Technol. Explor. Eng. 2 (2) (2013) 18–21. https://www.oalib.com/paper/2173542.

[57] S. Lloyd, Least squares quantization in pcm, IEEE Trans. Inf. Theor. 28 (2) (1982) 129–137, https://doi.org/10.1109/TIT.1982.1056489.

[58] S.C. Sripada, M.S. Rao, Comparison of purity and entropy of k-means clustering and fuzzy c means clustering, Indian J. Computer Sci. Eng. 2 (3) (2011) 343–346. ISSN 0976-5166.

[59] G. Baker, Le Corbusier-An Analysis of Form, Taylor & Francis, 2017. ISBN 9781351226240.

[60] R. Garber, Sinuous workflows: mad architects, the harbin opera house, Architect. Des 87 (3) (2017) 128–135, https://doi.org/10.1002/ad.2183.

[61] K. Frampton, T. And¯o, S. Wrede, *Tadao Ando*. Museum of Modern Art, 1991. ISBN 0870701983.

[62] A. Iannacci, Gensler Architecture: Form + Strategy, Edizioni Pr, 1999. ISBN 9780966223033.

[63] B.I. Group, BIG Recent Project, TASCHEN (Evergreen), 2020. ISBN 9784871406789.

[64] K.B. Hiesinger, Z. Hadid, P. Schumacher, Zaha Hadid: Form in Motion, Philadelphia Museum of Art, Philadelphia, PA, 2011, 2011. ISBN 9780300179828.

[65] C. Subbulakshmi, S. Deepa, Medical dataset classification: a machine learning paradigm integrating particle swarm optimization with extreme learning machine classifier, Sci. World J. 2015 (2015), https://doi.org/10.1155/2015/418060.

[66] Y. Guo, L. Zhang, Y. Hu, X. He, J. Gao, Ms-celeb-1m: a dataset and benchmark for large-scale face recognition, in: European Conference on Computer Vision, Springer, 2016, pp. 87–102, https://doi.org/10.1007/978-3-319-46487-9_6.

[67] W. Wu, X.-M. Fu, R. Tang, Y. Wang, Y.-H. Qi, L. Liu, Data driven interior plan generation for residential buildings, ACM Trans. Graph. 38 (6) (2019) 1–12, https://doi.org/10.1145/3355089.3356556.

[68] Z. Shi, J.A. Fonseca, A. Schlueter, A review of simulation based urban form generation and optimization for energy-driven urban design, Build. Environ. 121 (2017) 119–129, https://doi.org/10.1016/J.BUILDENV.2017.05.006.

[69] K. Karimi, A configurational approach to analytical urban design:' space syntax'methodology, Urban Des. Int. 17 (4) (2012) 297–318, https://doi.org/10.1057/UDI.2012.19.

[70] J. Beirão, G. Mendes, J. Duarte, R. Stouffs, Implementing a generative urban design model, in: eCAADe 2010 Conference: Future Cities, 2010, p. 265. http://papers.cumincad.org/cgi-bin/works/Show?ecaade2010_084.

[71] J. Döllner, B. Hagedorn, Integrating urban gis, cad, and bim data by service based virtual 3d city models, R. et al.(Ed.), in: Urban and Regional Data Management-Annual, 2007, pp. 157–160, https://www.researchgate.net/publication/313517089_Integrating_urban_GIS_CAD_and_BIM_data_by_service-based_virtual_3D_city-models.