Acta Mech
https://doi.org/10.1007/s00707-023-03520-7

ORIGINAL PAPER

Jad Maqdah · Milad Memarzadeh · Georgios Kampas · Christian Málaga-Chuquitaype

AI-aided exploration of lunar arch forms under in-plane seismic loading

Received: 8 November 2022 / Revised: 30 January 2023 / Accepted: 7 February 2023

Abstract Increasing computational power has led to the expansion of civil engineering research into using machine learning concepts for developing improved design strategies. These strategies are particularly useful for the design of extra-terrestrial habitats under uncertain environmental conditions. This paper focuses on building an unsupervised machine learning model (convolutional autoencoder) capable of detecting patterns in arch shapes and differentiating between their stress and displacement contours. Foremost, detailed discussions of the model's architecture and input data are presented. The variation of arch shapes and contours between cluster centroids in the latent space is determined, proving the capability of optimisation by moving towards clusters with optimal contours. Finally, a regression model is built to investigate the relationship between the input geometric variables and the latent space representation. We prove that the autoencoder and regression models produce arch shapes with logical structural contours given a set of input geometric variables. The results presented in this paper provide essential tools for the development of an automated design strategy capable of finding optimal arch shapes for extra-terrestrial habitats.

1 Introduction

1.1 Machine learning and structural design

Machine learning is a powerful tool that is capable of detecting and learning patterns in data, and, in turn, creating models for forecasting and aiding decision making under uncertain conditions [1,2]. In the last decade, machine learning techniques have been integrated into many various structural engineering applications that include monitoring structural performance, determining structural damage, and identifying recovery methods [2]. These machine learning applications in structural engineering and many more can be categorised into four groups: (a) predicting structural response and performance, (b) interpreting experimental data and formulating prediction models for component-level structural properties, (c) retrieving information using images and written text, and (d) recognising patterns in structural health monitoring data.

Burton, Huang, and Sun [2] investigated the studies conducted in each of these four categories in the past decade. It is evident from their paper that the machine learning models developed in prior studies focused on

J. Maqdah· C. Málaga-Chuquitaype (✉)
Department of Civil and Environmental Engineering, Imperial College London, South Kensington Campus, London SW1 2AZ, UK
E-mail: c.malaga@imperial.ac.uk

M. Memarzadeh
USRA, NASA Ames Research Center, Mountain View, USA

G. Kampas
RCube Private Company, Marousi, Greece

Published online: 24 February 2023

areas such as building performance, component-level design, seismic damage, damage control, and structural health. A limited number of studies have been conducted on the application of machine learning for the design and optimisation of whole structural systems, and some are outlined in the overview papers by Salehi & Burgueno [3] and Málaga-Chuquitaype [1]. However, there is no apparent research on the use of machine learning for the design of extraterrestrial habitats, making this a novel area of study. This paper will make use of the third of the options outlined above (c) to build an unsupervised machine learning model which extracts necessary features from image data sets of arch shapes and contour plots in order to find an alternative to traditional structural design for the purpose of designing extra-terrestrial structures. A design proposal of lunar outposts can be seen in Fig. 1.

A large benefit of shifting focus towards machine learning-based structural design is the large size of data that can be fed into the model [5]. The data set could be in the form of stress or displacement contour plots, giving rise to the potential of producing complex models that have an abundance of structural knowledge for better and more accurate optimisation results. This is particularly useful for extra-terrestrial design where environmental conditions are not well understood and, therefore, would require the consideration of a large set of design scenarios to compensate. In addition, machine learning models are able to easily identify patterns that would not be apparent to a human designer, spotting efficient designs that would not have been traditionally thought of from a structural viewpoint. This could be useful when coming up with creative structural shapes that could be easily 3D printed using extra-terrestrial soil.

1.2 Scope of investigation

In this paper, we use artificial intelligence concepts to build an unsupervised machine learning model that is able to detect features from images of black and white arch shapes and stress and displacement contours for extra-terrestrial habitats. The data sets of input arch images are created using algorithms in the form of Python scripts, which are run on the structural analysis software Abaqus. The machine learning model then trains on these data sets by reducing their dimensionality to produce a latent space and attempting to reconstruct the input images with minimal error. We then proceed to present the K-means clustering algorithm in order to group the latent space representation of each data set into distinct clusters with similar features, building a better understanding of this reduced space.

Finally, a regression model is built in order to determine and investigate the relationship between the input geometric variables and the latent space representation of the arch shapes and contours. Alongside the convolutional autoencoder, the regression model will allow the visualisation of the effect on the arch shape and contours when moving towards a certain direction in the input space.

2 Method and approach

2.1 Autoencoders and convolutional autoencoders

Autoencoders are unsupervised machine learning models that consist of two components: the encoder and the decoder [6]. As seen from Fig. 2, the encoder is a network that compresses the input data, $\bar{x}$, into an encoded

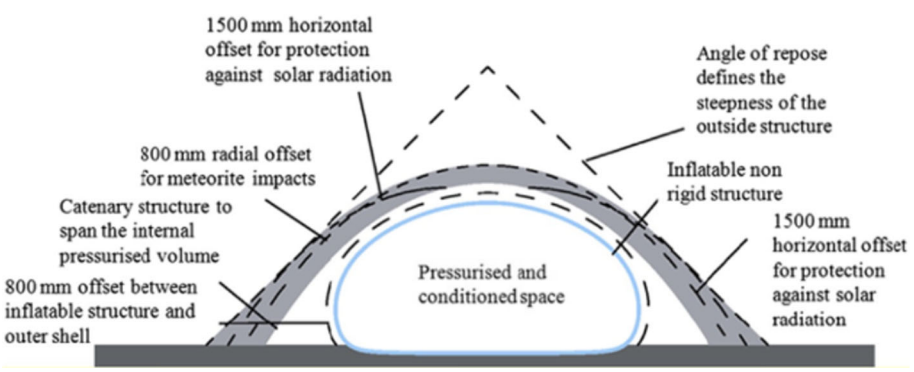

Fig. 1 Lunar habitat proposal using a regolith arch shield [4]

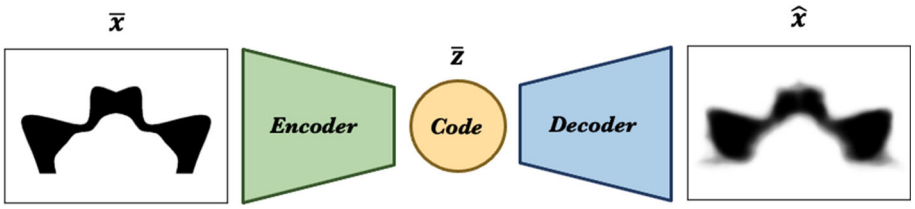

Fig. 2 Autoencoder architecture

feature vector, $\bar{z}$, whose dimensionality is smaller than that of the input's. The decoder then reconstructs the original input using the feature vector to form the output, $\hat{x}$. As visualised, the model forms a bottleneck between the encoder and decoder components, which serves an important purpose for the autoencoder: it ensures that the model does not simply copy the input and passes it along the network to the output, i.e. it forces the model to constrain the input data and use its own understanding to reconstruct it.

Convolutional autoencoders are a type of autoencoders which exploit the observation that an input signal can be decomposed as a sum of other signals [7]. A convolutional autoencoder takes an input, encodes it in a set of simple signals, and then attempts to reconstruct the input signal from them. An advantage of using convolutional autoencoders as opposed to conventional autoencoders is that they do not ignore the structure of the input 2D image [8]. Conventional autoencoders introduce a redundancy by only accepting 2D input images in the form of 1D vectors (i.e. 2D images must be unrolled) and have networks that are constrained to the number of inputs. Convolutional autoencoders, on the other hand, avoid the introduction of any redundancy in the parameters and prevent the features from becoming global, i.e. spanning the entire visual field. In addition to retaining a 2D structure, convolutional autoencoders increase a third dimension that corresponds to the number of filters that are necessary to represent the input image.

2.2 Building the convolutional autoencoder with Keras and TensorFlow

The Python library TensorFlow with the toolkit Keras is used to build, model, and train the convolutional autoencoder. Keras is able to construct and train complex machine learning architectures easily and, for this reason, is used [6].

The code for the encoder starts with an input layer that acts as an entry point to the network. Following this, a block of code is repeated several times (seven in this case) consecutively. Each block of code is made up four functions or layers in the following order:

1. 2D Convolutional layer: detects and extracts features from the input data.
2. Batch normalisation: standardises the values of its input to accelerate the training process.
3. Activation: applies an activation function to its input (ReLU activation is used for this model).
4. 2D Maxpooling: reduces the dimensionality of its input and its computational complexity.

After the encoder reduces the dimensionality of the input images as seen in Fig. 3, the decoder attempts to reconstruct the input images using only the latent space vector by reversing the encoder's operations [9]. The decoder's input is thus the encoder's output.

As seen in Fig. 4, the decoder starts with creating an input layer of size equal to the latent space dimension. Following this, a block of code that starts with batch normalisation and ends with a transposed convolution is repeated several times (again, seven in this case), consecutively. The purpose of these consecutive code blocks is to reverse the operations performed by the encoder, in hopes to reconstruct the input image from the latent space. Each block of code is made up four functions:

1. Batch normalisation.
2. Activation.
3. 2D Upsampling: increases the dimensionality of its input by duplicating its rows and columns.
4. 2D Inverse Convolution: takes an element from its input matrix and distributes it into a specific-sized matrix in the output.

The process of convolutional layers comprises of the matrix multiplication between the input image and multiple learning matrices or kernels (filters) to produce feature maps [10]. This produces as many feature

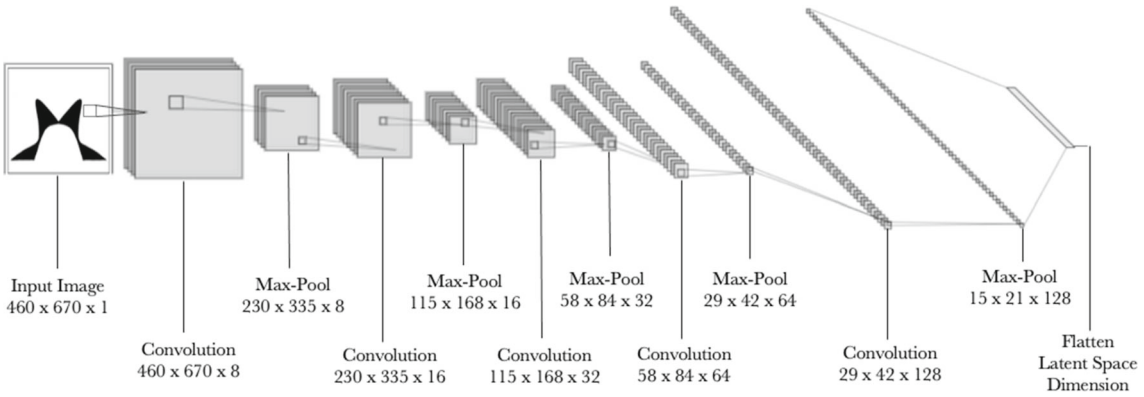

Fig. 3 Encoder architecture of the built model (only five layers showed for clarity)

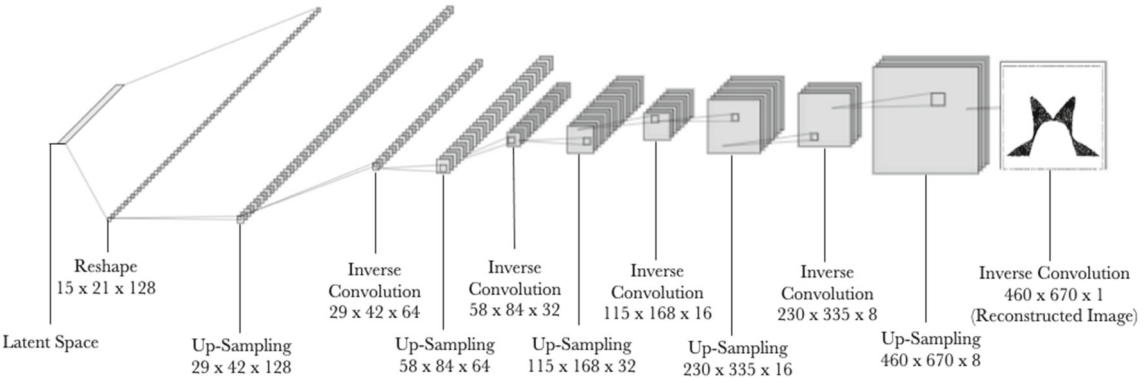

Fig. 4 Decoder architecture of the built model (only five layers showed for clarity)

maps as there are filters (i.e. adding to the third dimension of the input image). Each of these feature maps corresponds to the feature that has been detected by a particular filter.

Since the types of images that are inputted into the built convolutional autoencoder are images of structures with complex shapes and detailed contours that represent stress and displacement variations, a deep network is required. Deeper models have higher capacity to learn complex patterns present in the input data and have a higher redundancy, which is suitable for the design optimisation. The number of layers is also affected by the size of input data, as the learning of the latent features is done gradually through convolutional operations. Increasing the number of convolutional layers, as well as the number of filters in each layer, would increase the number of features the model will detect, since more feature maps would be generated, and would allow the model to learn more complex features.

In order to find a suitable number of convolutional and max pooling layers to use, an input image was fed into the autoencoder and the quality of the reconstructed image (i.e. a measure of how well the encoder was able to pick up the complex and detailed features from the input image) was observed. To avoid the model overfitting the data and becoming computationally expensive, a relatively simple model (a model with three convolutional and maxpooling layers) was used as a starting point and additional layers were added as needed. It was found that using seven convolutional and max pooling layers (in conjunction with a suitable latent space dimension) resulted in reconstructed images that highly resembled the input image.

When choosing an activation function for the model, three of the most popular activation functions were considered (see Fig. 5): rectified linear units (ReLU), sigmoid, and hyperbolic Tangent. The ReLU activation function was used in this model due to its simple mathematical interpolation ($y = \max(0, x)$) and its efficiency in training deep machine learning models.

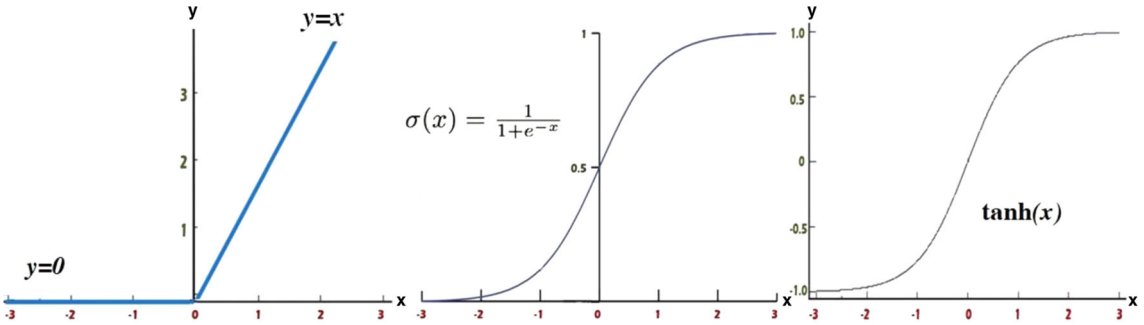

Fig. 5 Most popular activation functions used in activation layers; (left) rectified linear units, (centre) sigmoid, and (right) hyperbolic tangent [10]

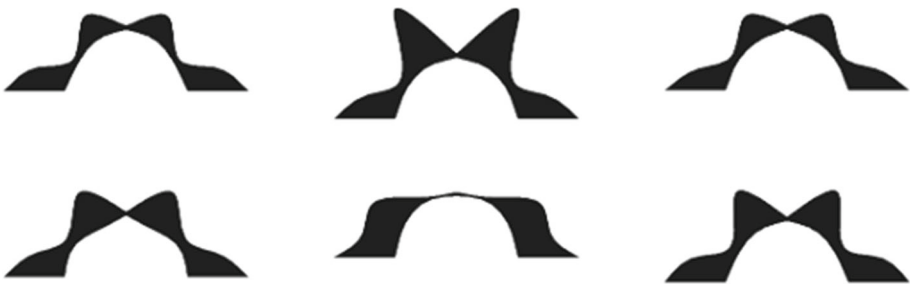

Fig. 6 Data Set 1 examples

Fig. 7 Data Set 2 examples

2.3 Input data sets

The arch shapes investigated using thrust line analysis under lunar environments in Málaga-Chuquitaype et al. [11] paper act as motivation for the arch shapes that the machine learning model trains on (details on this are covered in Sect. 3.2). In order to fully understand and make use of the convolutional autoencoder, three arch image data sets are set up:

1. Data Set 1 (3000 arch shapes): simple and repetitive arch shapes in order to identify what type of features the model is able to pick up (see Fig. 6).
2. Data Set 2 (5000 arch shapes): wide variety of arch shapes to investigate the extent to which the model is able to pick up features and cluster arch shapes (see Fig. 7).
3. Data Set 3 (4000 contour plots): this data set is similar to that of data set 2 but is constrained to reduce the occurrences of extreme arch shapes, i.e. shapes that have very little structural integrity. A total of 1000 arch shapes were structurally analysed to form 4000 contour plots (1000 each for maximum in-plane stress, minimum in-plane stress, horizontal displacement, and vertical displacement) (see Fig. 8).

When creating the different sets of inputs for the arches, it is important to use methods that produce arches that cover the full range of the input geometric variable space. This is to maximise the range of arch shapes

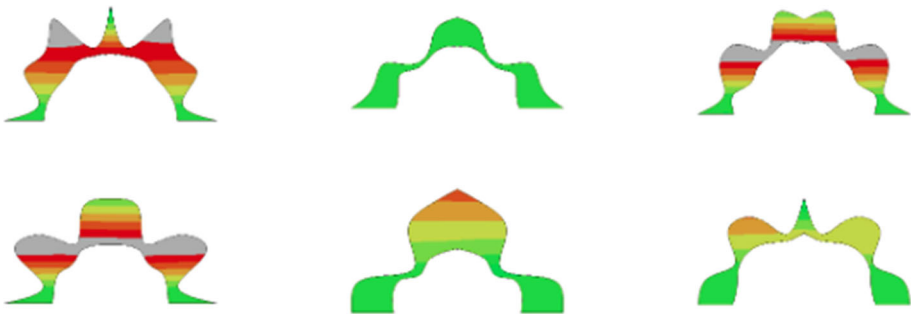

Fig. 8 Data Set 3 examples

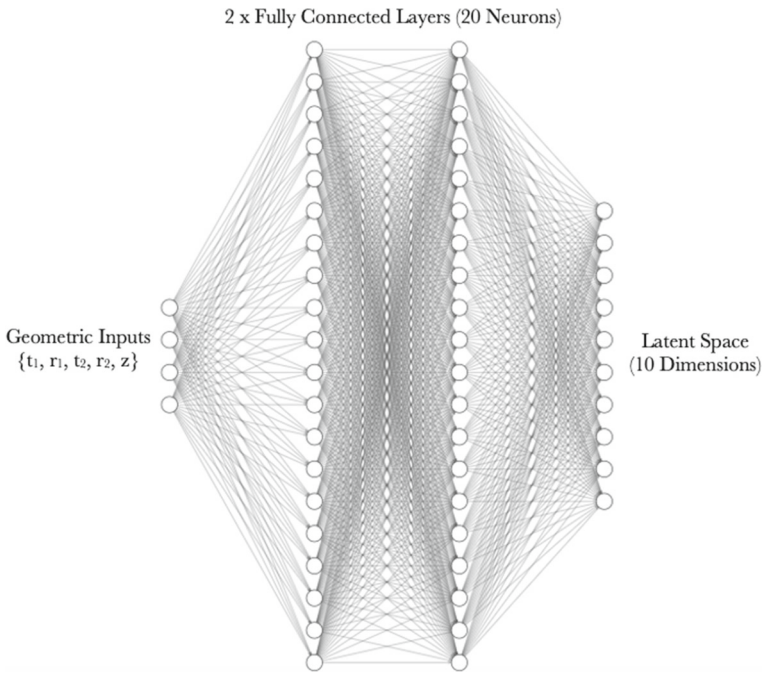

Fig. 9 Regression model architecture

that are fed into the autoencoder model and to prevent skewing the data in the direction of specific arch shapes. Latin hypercube sampling was used as it is useful for complex models with many variable inputs. It is a random sampling method that increases computational efficiency by reducing the need for very large input data sets, which is achieved by ensuring full coverage of the input space [12].

2.4 Regression model

In order to build a better understanding on how the input geometric variables that define the arch shapes affect the latent space representation, which, in turn, affects the output arch image, a regression neural network is built using Keras. In the context of this paper, the regression model is able to determine the relationship between the input geometric variables (independent variables) and the latent space representation (dependent variables) to obtain the effects on the latent space vectors when the input variables are moved in a specific direction, e.g. increasing a thickness at a certain section in the arch. Figure 9 displays the architecture of the built regression model.

Since the relationship that the regression model was attempting to find is simple, i.e. a relationship between two small sets of numbers, a wide (network with more nodes in each layer) and shallow (networks with less layers) neural network was used as a starting point. According to Chollet [13], simple networks can learn simpler patterns more efficiently, but they are also more prone to overfitting. In order to avoid overfitting, a

Table 1 Comparison of different lunar seismic sources (m_b = short-period body wave) [14]

Natural seismic source	Rarity	Cause	Magnitude
Deep Moonquakes	Most common	Tidal forces	Maximum of 3 m_b
Shallow Moonquakes	Rarest	Unknown	Maximum of 4.8 m_b
Thermal Moonquakes	Unlikely	Extreme temperature variations on the lunar surface	Very small amplitude
Meteorite Impacts	Abundant (but less than deep moonquakes)	Thin atmosphere	Amplitude and frequency vary greatly

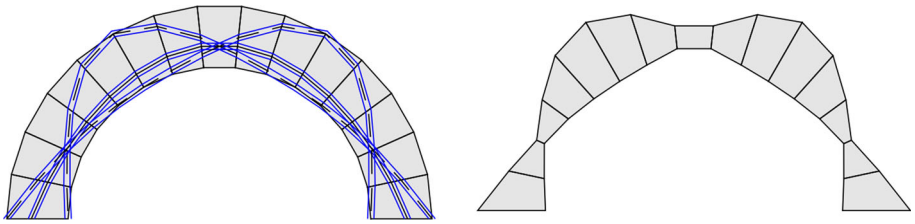

Fig. 10 Progression of an input semi-circular arch to the optimal form-found arch [11]

Table 2 Regolith material properties [19]

Material property	Units	Value
Density	kg/m^3	2300
Young's Modulus	MPa	287.3
Compressive strength	MPa	4.2
Tensile strength	MPa	0.42
Poisson's ratio	–	0.2

very simple model (a model with only one fully connected layer and 10 neurons) was used as a starting point and additional layers and neurons have been added as required. In order to measure whether the model has successfully detected a pattern between the input geometric variables and the latent space representation of the arches, the output from the regression model (i.e. the predicted latent space representation) is fed into the decoder part of the convolutional autoencoder in order to visualise the arch that the predicted latent space vector represents. The model's architecture was not augmented with additional layers or neurons when the decoder generated arches with shapes that exhibit conformity with the input geometric variables (i.e. the predicted latent space representation was accurate enough for the decoder to reproduce the input geometric variables in its output).

The final regression model starts with an input layer of size equal to the number of input geometric variables for Data Set 1 (five variables $t1$ $t2$, $r1$, $r2$, and z as shown in Fig. 11 in Sect. 3.4), continues with two fully connected neural layers with 20 neurons each, and ends with an output layer of size equal to the latent space dimension. For similar reasons to the convolutional autoencoder, the ReLU activation function is used for this regression neural network. The regression model is then trained on the input geometric variables ($t1$, $t2$, $r1$, $r2$, and z) that have created the data set and the latent space representation of the corresponding arches.

3 Structural analysis

3.1 Lunar ground motion

Kalapodis et al. [14] discussed the seismic aspects of the Moon by referring to the data gathered by the seismometers installed during the Apollo 11, 12, 14, 15, and 16 missions. Table 1 summarises the corresponding findings.

From Table 1, it is evident that shallow moonquakes will govern the seismic design of the arches due to their large maximum body wave magnitude (4.8 m_b). Given the very small observation period of these motions (only 8 years), it is anticipated that larger events should occur during the life cycle of a lunar structure;

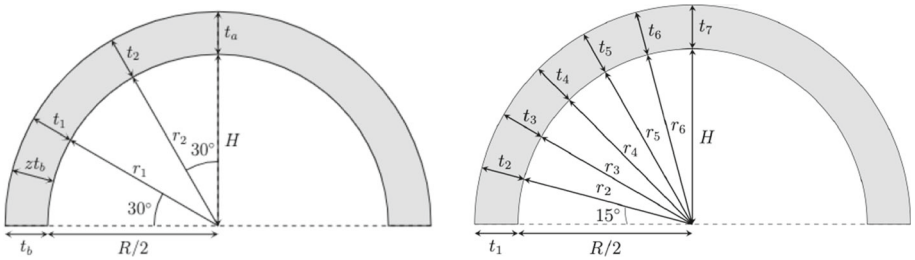

Fig. 11 (Left) Data Set 1 geometric variables (Right) Data Set 2 and 3 geometric variables

hence, a design level acceleration of 0.2 g (where g is the acceleration of Earth's gravity) is considered. This level of acceleration is consistent with previous assumptions (i.e. Málaga-Chuquitaype et al. [11], Kalapodis et al. [15]) and is in accordance with the observations of Oberst and Nakamura [16] who compared the magnitude–recurrence relationships for shallow moonquakes with interpolate earthquake in Central-Eastern USA. Therefore, static forces equivalent to a peak acceleration of 0.2 g were used in the structural analysis of Data Set 3 as seen in Sect. 3.3 for direct comparison purposes with Málaga-Chuquitaype et al. [11] case study.

3.2 Motivation of arch shapes

The definition of the range of arch shapes used as input for this study was done with two principles in mind: (i) first, to be able to include wide variations of geometry that would enable the exploration of shapes and structural responses beyond what would be intuitively obvious for a human engineer so that the benefits of AI could be materialised; and (ii) the need to include arch shapes that had been previously identified as optimal. To this end, McLean et al. [17] and Málaga-Chuquitaype et al. [11] have developed a form finding algorithm that obtains an optimal arch shape given any standard geometry of minimum thickness for a given loading. In brief, this algorithm finds the limit and admissible thrust lines of the arch shape, offsets them by a required finite strength offset, and uses these offset lines to form a new arch shape envelope. The algorithm then attempts to find the most optimal shape that has the least area compared to the original input arch. Figure 10 shows the progression of an input semi-circular arch to the optimal form-found arch using limit thrust lines. It is evident that the final (optimal) shape is associated with a significant reduction in structural material, forming a 'pinch point' at the crown of the arch and two symmetrical ones to the sides of the arch.

As seen in Sect. 2.3, the algorithm used for creating the input data is able to produce shapes that resemble the general shape of an optimal form-found arch from the limit thrust-line analysis based procedure devised by McLean et al. [17]. By doing so, we prevent the machine learning model from working with extremely large data sets of shapes that equate to low structural capacities and make the model training process much more efficient.

3.3 Contour plot requirements

To produce the stress and displacement contours using the commercial finite element analysis software Abaqus [18] for Data Set 3, material properties, loading conditions, and boundary conditions need to be defined.

3.3.1 Material properties

Goulas et al. [19] investigated the mechanical behaviour of additively manufactured lunar regolith simulant components, giving a deeper insight into how extraterrestrial 3D printed structures might respond in environmental conditions. Table 2 gives a summary of the findings of this material.

3.3.2 Loading conditions

Abaqus has a gravity loading condition that applies a uniformly distributed acceleration in a fixed direction. Abaqus then obtains the actual loading applied on the structure using the acceleration magnitude and the material density. For this scope of study, the following was applied:

– Component 1: lateral component representing the seismic load applied on the arch originating from moonquakes, meteorite impacts, etc. A value of 0.2 g (1.962 m/s^2) is chosen in consistency with Málaga-Chuquitaype et al. [11] case study.
– Component 2: vertical component representing the self-weight load on the arch. This component will therefore be equal to the Lunar gravitational field, which equals to -0.166 g (-1.628 m/s^2).

3.3.3 Boundary conditions

When defining the arches' boundary conditions with the ground, it was decided to use fixed conditions (horizontal and vertical displacements and rotations are set to zero). This prevents any damages occurring on the inner inflatable structure due to a movement of a base.

3.4 Abaqus algorithms

To create all the input training images (shapes and contours), the structural analysis software Abaqus is used. In order to create thousands of input images, three Python-Abaqus scripts have been created:

1. Script that creates black and white images (used for Data Sets 1 and 2): this python script loops through an array of input geometric variables (as defined in Fig. 11) that define the shapes of the arches and creates a model for each set of inputs (i.e. for each row in the input array). For each model, the input geometric variables are used to create lines and splines to draw the shape of the arch. The model and its background are then coloured black and white and saved to a folder as a '.png' file.
2. Script that creates input files (used for Data Set 3): similar to the script that creates Data Set 1, this script iterates through the array of input geometric variables (as defined in Fig. 11 (Right)) and for each row, creates the model and shape of the arch, applies the loading and applies the boundary conditions. The script then saves the models as Abaqus input files that can be used to run the analyses in batch.
3. Script that runs the input files (used for Data Set 3): this script iterates through the input files, runs the analysis jobs, plots contour diagrams of the stresses and displacements, and saves these contours as '.png' files.

3.5 Structural analysis

As mentioned above, the arches comprising Data Set 3 were subjected to detailed numerical analyses using Abaqus. To this end, the arches were discretised into reduced integration 4-noded bilinear plane stress quadrilateral elements. A preliminary mesh sensitivity analysis indicated that a maximum element dimension of 0.2 m was enough to ensure stability in the response of our models. A geometric nonlinear static analysis was carried out with a one second time period.

Special care was taken to ensure that the stress and displacement contour scales remained consistent between all analyses in order for them to be comparable and suitable for input into our convolutional autoencoder network. To this end, the stress contours produced range from the assumed tensile (0.42 MPa) and compressive (-4.2 MPa) strengths of the material, as described previously in Sect. 3.3. Likewise, the displacement contours vary from $+1$ to -1% of the largest arch thickness considered (20 m), i.e. from -0.2 to 0.2 m. The contour bar legend that corresponds to the contour plots in Data Set 3 can be seen in Fig. 12. Any stresses or displacements above or below that contour range are represented by a solid grey or black contours, respectively.

3.5.1 Maximum in-plane stress

Since the maximum in-plane principal stress represents the maximum stress component for a specific stress state, it is expected that most results will be closer to the tensile stress limit (red). Furthermore, the tensile strength of regolith has been slightly surpassed for many arch shapes due to the small grey contours, indicating that the maximum in-plane stress could govern the structural design of these arches. Figure 13 displays arches from two different clusters with the maximum in-plane stress plotted over them.

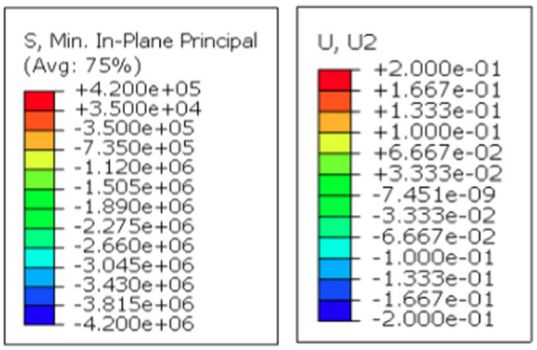

Fig. 12 (Left) Stress contour bar for Data Set 3 in Pa (Right) Displacement contour bar for Data Set 3 in *m*

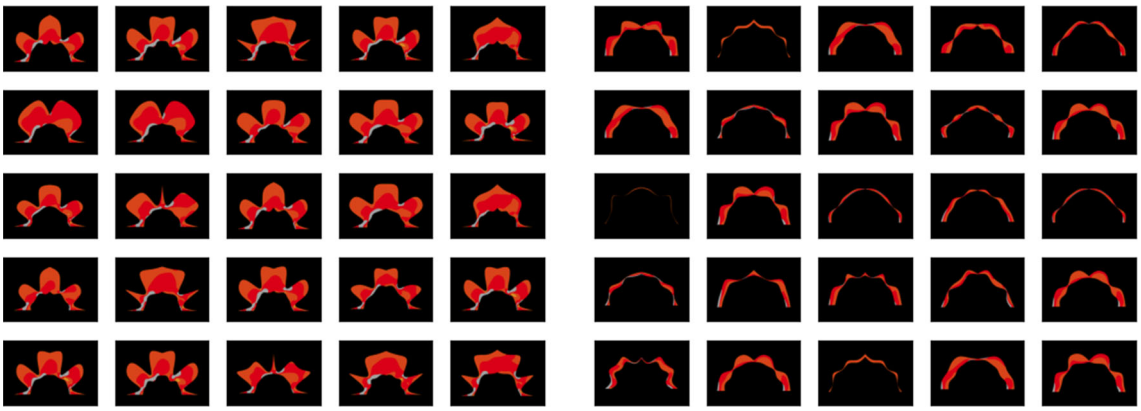

Fig. 13 Maximum in-plane stress arches from two random clusters

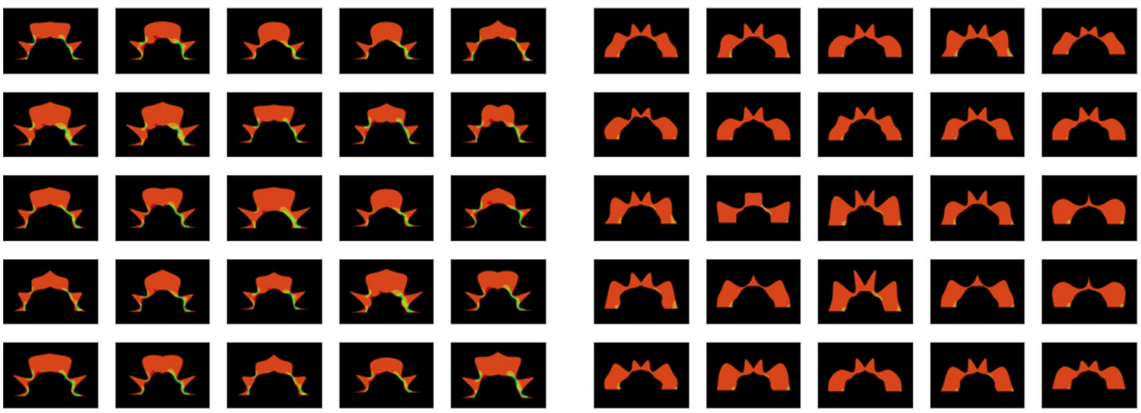

Fig. 14 Minimum in-plane stress arches from two random clusters

3.5.2 Minimum in-plane stress

The minimum in-plane stresses represent the minimum stress components for any stress state, and thus, it is expected that the results are closer to the compressive strength compared to the ones in the maximum in-plane stresses. On average, there are more stress variations compared to the results of the maximum in-plane stresses and fewer failures, and thus, compressive stresses do not appear to be a dominant factor on the structural design of the arches. Figure 14 displays arches from two different clusters with the minimum in-plane stress plotted over them.

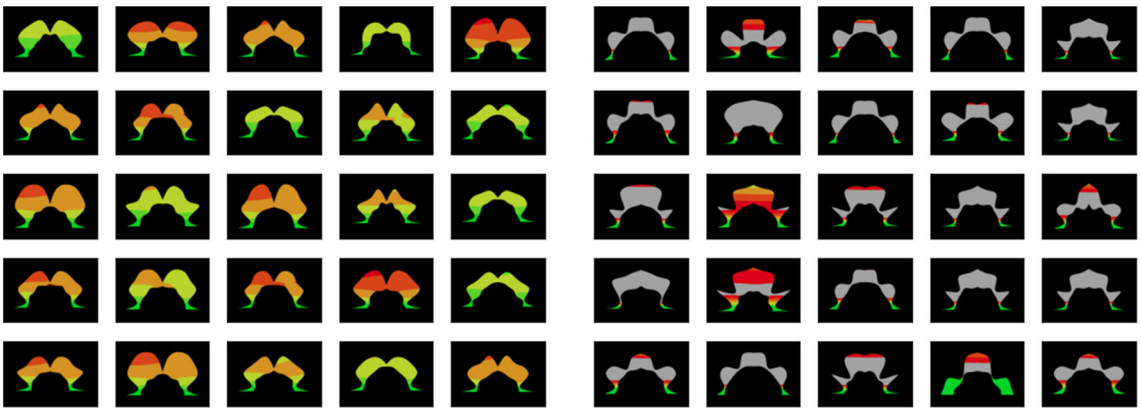

Fig. 15 Horizontal displacement of arches from two random clusters

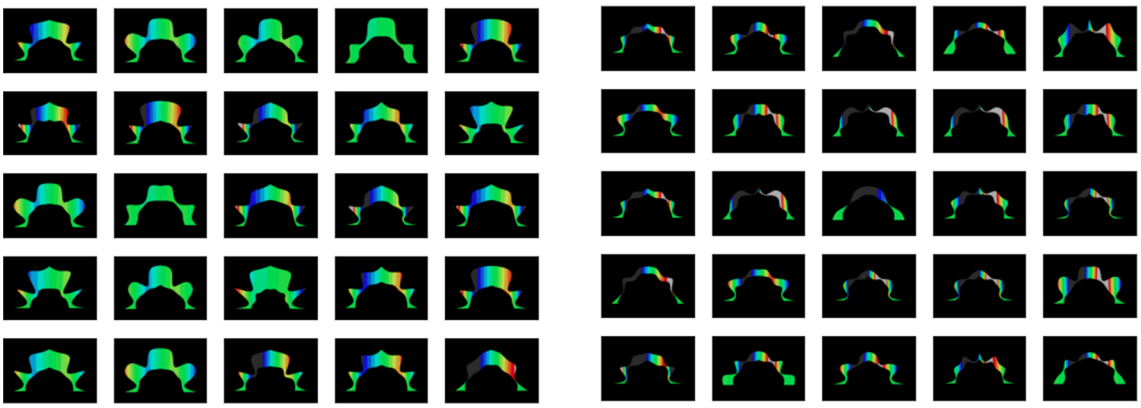

Fig. 16 Vertical displacement of arches from two random clusters

3.5.3 Horizontal displacement

The horizontal displacement, unlike the maximum and minimum in-plane principal stresses, poses more stringent constraints on the structural design of the arches. This is represented by the large proportion of arches with solid grey contours, i.e. surpassing the displacement limit, and the large dependency of contours on the arch shapes. That said, some clusters contain arch shapes with suitable structural configurations which prevent any horizontal displacement, represented by green contours. Figure 15 displays arches from two different clusters with the horizontal displacement plotted over them.

3.5.4 Vertical displacement

To a smaller extent than the horizontal displacement, the variation of vertical displacement is influenced by the arch shape, with some shapes exceeding the upper and lower displacement limits. Figure 16 shows the arches from two different clusters with the vertical displacements plotted over them.

4 Results

4.1 K-means clustering results

After the convolutional autoencoder has been trained on the three data sets, it is important to ensure that the model was able to pick up and differentiate the different features of the arches. In order to do so, the unsupervised machine learning algorithm K-means clustering is implemented on the latent space representation of the data.

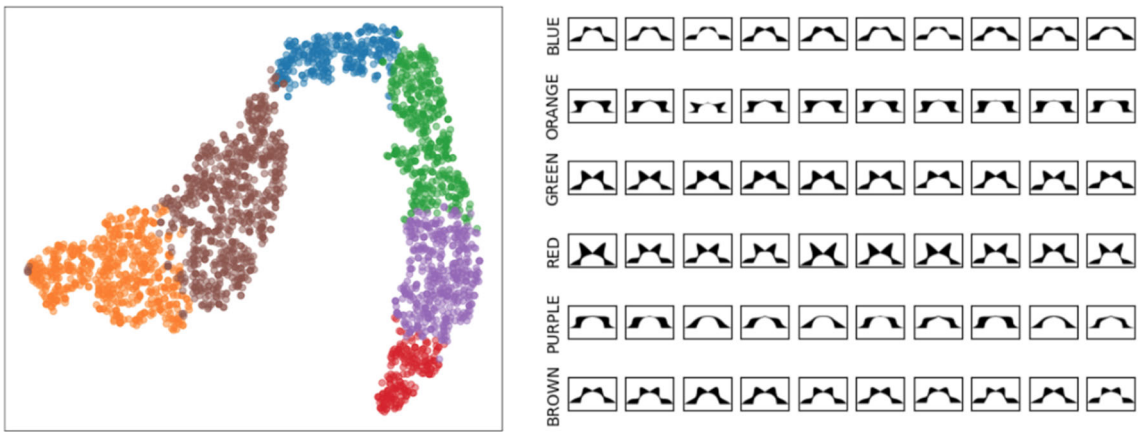

Fig. 17 (Left) K-means clusters of latent space representation of arches in Data Set 1 and (Right) corresponding arches from each cluster

By doing so, the arches will be grouped according to the features that the encoder was able to extract from the data set.

After using the Elbow method (a mathematical method in which the optimal number of clusters is sought for a specific data set) to find the correct number of clusters (k), the data are then reduced to two components using the t-distributed stochastic neighbour embedding (t-SNE) method, which is a statistical tool that allows better visualisation of high-dimensional data by reducing it into fewer components.

Figure 17 (left) shows the reduced data with six clusters ($k = 6$) for Data Set 1. In order to ensure the model was able to categorise the images correctly, ten random arch images from each of the six clusters are visualised in Fig. 17 (right). Each row represents a different cluster, and it is clear that the model was able to group very similar looking arch shapes with each other. Because all arches in a specific row contain very similar looking shapes and no anomalies were present, the value chosen for the number of clusters, $k = 6$, is ideal. The model was also able to distinguish between similar arch shapes with different thicknesses, for example, rows 1, 3, 4, and 6 all have similar looking arches but have different bulging thicknesses near the apex.

4.2 Latent space variation

In order to be able to use the model for optimisation, it is crucial to understand how the arch shapes and contours are varying from one point in the latent space to another. For this purpose, the latent space representation of a certain data set is clustered using the K-means clustering algorithm and the cluster centroids are found. The clustered data are then reduced to two components and plotted alongside their corresponding cluster centroids. Straight lines are then fitted between the centroids in order to define candidate points along these lines. These two-dimensional candidate points are then transformed back to their latent space representation form, which are, in turn, reconstructed using the decoder to visualise how the arches vary from one centroid to another. It should be noted that a different unsupervised technique called the principal component analysis (PCA) is used for the reduction and inverse-reduction of this data as opposed to t-SNE. The PCA, as opposed to t-SNE, is able to perform inverse-reductions, enabling the transformation of the reduced data back to a latent space representation form.

Figure 18 shows the 2D representations of the clusters and the variation of arches along the lines connecting the cluster centroids for different data sets and number of clusters (k values). Each row in Fig. 18b, d, and f represents the variation along a linear line between two centroids in their corresponding 2D visualisations (a), (c), and (e), respectively. The starting point of the lines, and thus the first row of arch images, is the red cluster for all three data sets, and the order continues along the lines to the end point at the brown cluster for Data Set 1 and the blue cluster for Data Sets 2 and 3.

It is evident that when moving along a line or a row of images, one or more arch features are changing to form a new arch. For example, along the first row in Fig. 18d (i.e. from the red to the orange cluster in (c)), the thicknesses of the two bulging zones are progressively increasing, as well as the side thicknesses in the last row in (d). As for the contour data set, (e) and (f), the shape along with the contours of the arches progressively

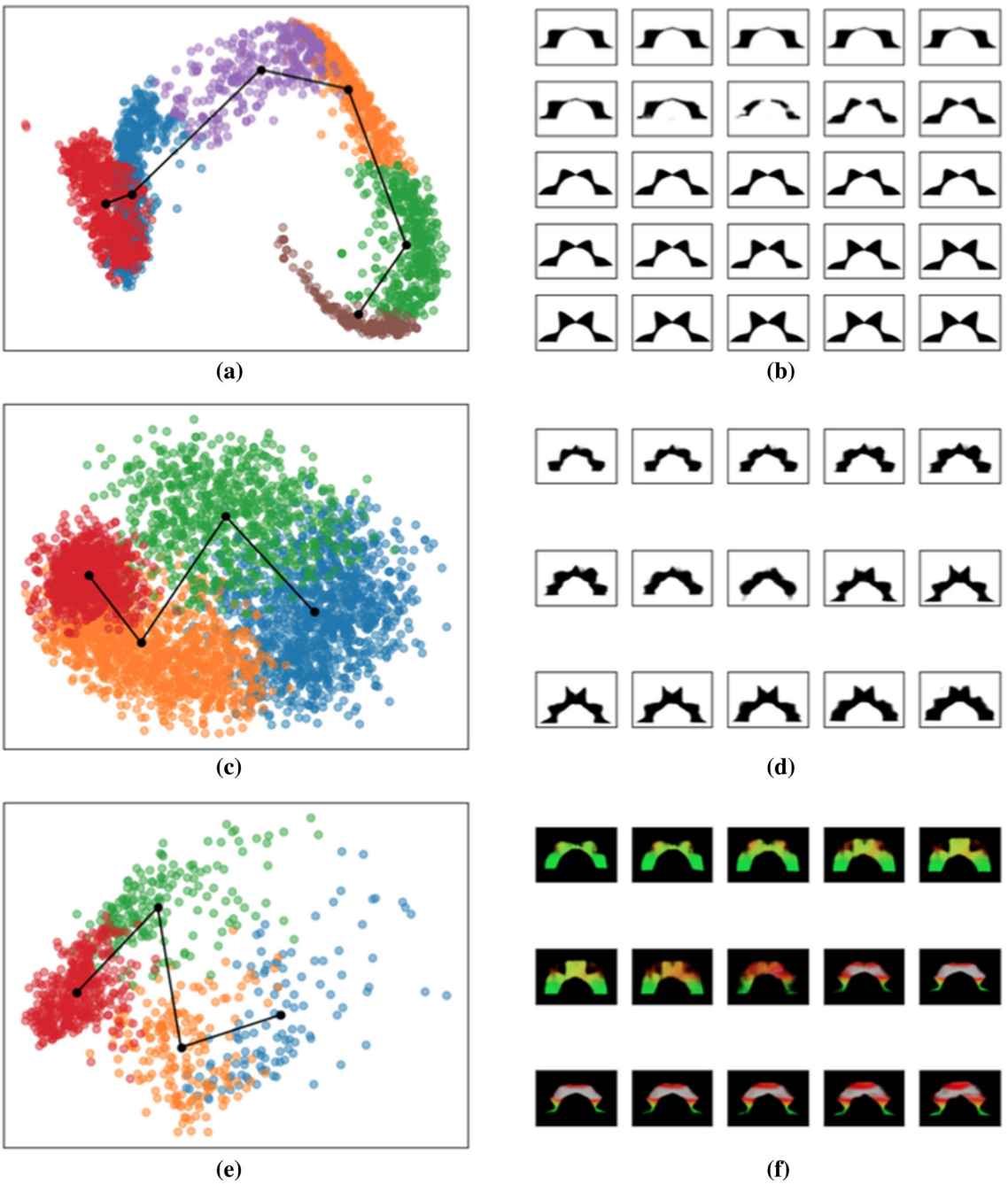

Fig. 18 Design-space exploration of arches using the latent space representation. **a** Data Set 1 with $k = 6$, **b** variation of Data Set 1 arches along the lines connecting the centroids, **c** Data Set 2 with $k = 4$, **d** variation of Data Set 2 arches along the lines connecting the centroids, **e** Data Set 3 with $k = 4$, and **f** variation of Data Set 3 arches along the lines connecting the centroids

changes along the lines that connect the cluster centroids, which can be clearly seen in rows one and two in (f) and the two lines connecting the red, green, and orange clusters in (e), in that order.

Optimisation is a function of the latent space representation and, therefore, understanding where the optimal arches, i.e. ones that have green contours, are situated in this latent space is important. Referring to Fig. 18e, f, when moving from the red cluster to the green cluster, to the orange cluster, and ending at the blue cluster, the contours of arches are turning from green (zero horizontal displacement) to red (limit horizontal displacement)

Table 3 Input geometric variables with t_1 varying for Data Set 1

	t_1 [m]	1_1 [m]	t_2 [m]	r_2 [m]	z[−]
Arch 1	1	15	1	15	0.3
Arch 2	5	15	1	15	0.3
Arch 3	10	15	1	15	0.3
Arch 4	15	15	1	15	0.3
Arch 5	20	15	1	15	0.3

to grey (failure). This indicates that optimisation in this case should include an objective function that moves towards the red cluster that contains the green contoured arches.

4.3 Regression results

After the regression model has been trained and has determined a relationship between the input geometric variables and the latent space variables, the effect of moving towards one direction in the input space on the latent space can be determined. This is done by creating input vectors and increasing one input variable from its minimal value up to its maximum value and feeding these vectors into the regression model. The regression model then outputs the equivalent latent space vectors for these input vectors, which can be fed into the decoder in order to visualise the effect on the arch images. For example, t_1 in Data Set 1's input variables is varied as seen in Table 3 (seen as the grey cells) to produce five input vectors (i.e. five arches) with increasing thicknesses at 30^o from the horizontal. The definitions of the geometric variables can be seen in Fig. 11 (Left).

In order to better visualise the effect of the input geometric variables on the latent representation, the output latent space vectors from the regression model are reduced using t-SNE into two components and plotted on top of the reduced latent space representation of the whole data set. A line is then drawn between the points from the regression model in order to visualise the movement in the latent space as a result of movement in the input space. As seen from this Fig. 19, it is expected that the greater the impact of the input variable on the overall shape of the arch, the greater the distance traversed in the latent space (rows 1 and 3), and vice versa (rows 2, 4, and 5). The start and end points are represented as the green and red dots, respectively.

5 Conclusion

The main contribution of this study is the development of an unsupervised machine learning model (convolutional autoencoder, CAE) to detect trends and features from large data sets of arch shapes and contours for lunar habitats under in-plane seismic loading. The CAE model presented in this paper has demonstrated the potential of AI-based structural optimisation by being able to accurately learn and classify the arches into groups according to specific similar features. In the case of Data Set 3 (stress and displacement contour plots), each category that was found by the model corresponds to a specific structural trend; this suggests that optimisation algorithms could be utilised to create structurally optimised lunar habitats, where material efficiency is a critical factor. The utilisation of machine learning in the structural design of lunar habitats holds great significance as it has the potential to yield innovative shapes that may not have been conceived of by human designers. These unique shapes have the potential to be both material efficient and structurally stable against the harsh environmental conditions present on the lunar surface.

Three Abaqus algorithms in the form of Python scripts have been created and used to produce the training data: the first algorithm created the black and white arch images, the second created input files containing arch shapes and applied the material properties, boundary conditions, and loading conditions, and the final algorithm took the output from the second algorithm and ran the analyses to output stress and displacement contours.

K-means clustering was applied to the learned features by the model to identify the significant patterns that the model finds in the data. An experimental study was conducted to examine the variations of arches along lines connecting cluster centroids in the latent feature space. The results of the study indicated that for Data Set 3, each cluster contained arches with distinct contours. Based on these findings, it can be inferred that it is possible to achieve favourable contours by moving towards cluster centroids that contain arches with desirable contours, thus facilitating optimisation.

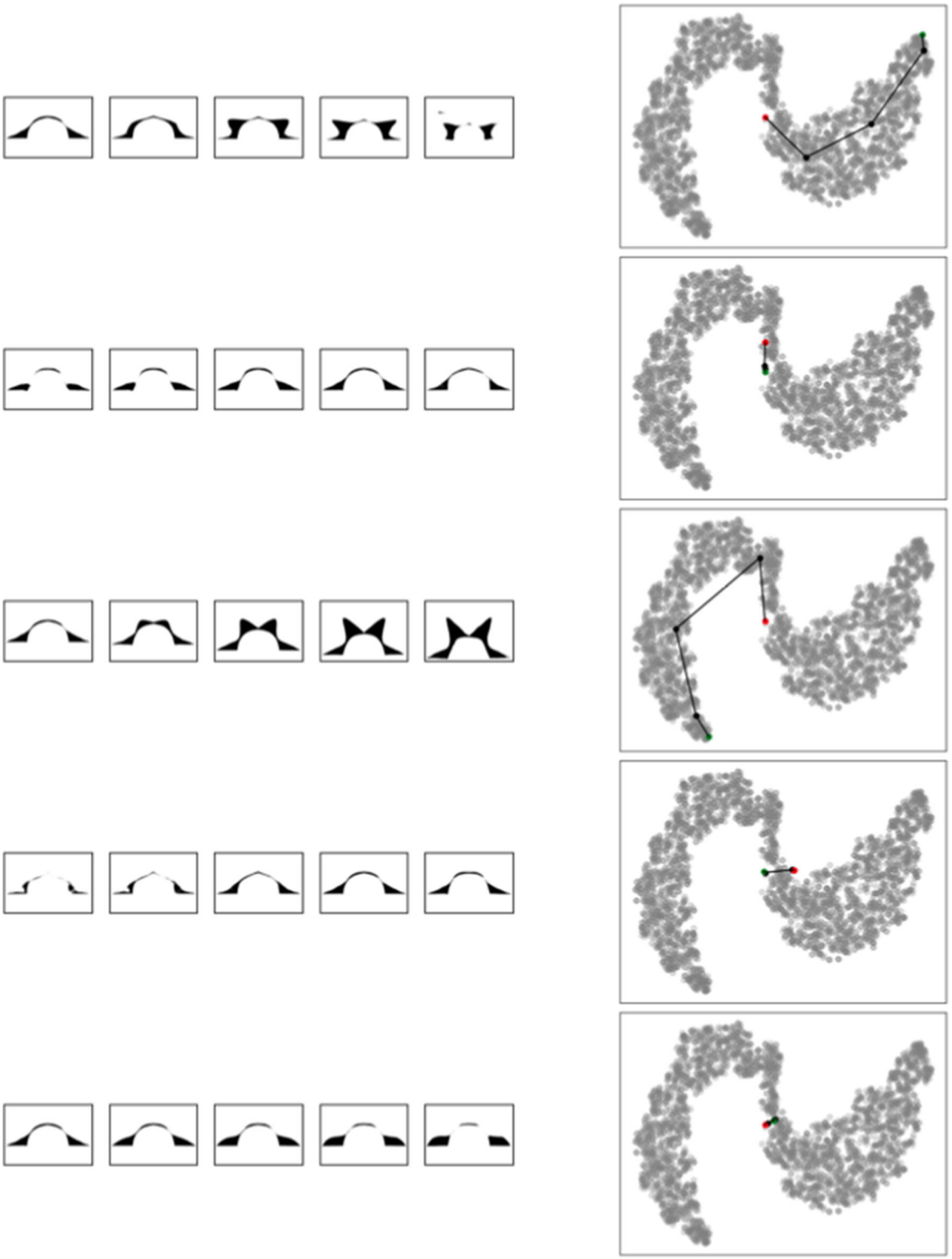

Fig. 19 Change in latent space due to a change in a geometric variable for Data Set 1. (Left) Change of arch shape due to a change in a geometric variable, and (right) corresponding movement in latent space due to the change in the geometric variable

Finally, a regression model was developed to investigate the correlation between input geometric variables and the latent space representation. Through this model, the variation of the latent space was determined, when an input variable is adjusted in a specific direction. As hypothesised, the greatest distance travelled in the latent space corresponded to the geometric variables, the more significant was the impact on the shape of the arch, such as thickness variables. It is noteworthy that when a geometric variable was modified in Data Set 3 using the regression model, the decoder was able to produce logical structural contours that were consistent with the altered arch shape, demonstrating that the model had the ability to structurally analyse the arch shape through its own knowledge. As a result, the decoder and regression models together constitute an effective tool for generating arch contours, given a set of geometric variables.

While the model presented in this paper has demonstrated the potential for AI-based structural optimisation, there are still some limitations that need to be addressed in future work. For example, the model was only trained on a limited data set, and so further research is needed to test its performance on a larger and more diverse data set. This includes a data set with a larger variety of loading conditions and material properties.

Acknowledgements The first and last authors of this paper would like to thank Dr. Amiya Basu for his support of the use of AI technologies in Civil Engineering. The third author would like to thank EPSRC for the grant EP/S0363931.

References

1. Málaga-Chuquitaype, C.: Machine learning in structural design: an opinionated review. Front. Built Environ. (2022). https://doi.org/10.3389/fbuil.2022.815717
2. Burton, H., Huang, H., Sun, H.: Machine learning applications for building structural design and performance assessment. J. Build. Eng. (2020). https://doi.org/10.1016/j.jobe.2020.101816
3. Salehi, H., Burgueño, R.: Emerging artificial intelligence methods in structural engineering. Eng. Struct. **171**, 170–189 (2018). https://doi.org/10.1016/j.engstruct.2018.05.084
4. Cesaretti, G., Colla, V., De Kestelier, X., Enrico, D.: Building components for an outpost on the Lunar soil by means of novel 3D printing technology. Acta Astronaut. (2014). https://doi.org/10.1016/j.actaastro.2013.07.034
5. Flair, D.: Advantages and disadvantages of machine learning language. https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/. Accessed 10 Mar 2021
6. Bonaccorso, G.: Mastering Machine Learning Algorithms. Packt Publishing, Birmingham (2020)
7. Galeone, P.: Convolutional autoencoders. https://pgaleone.eu/neural-networks/2016/11/24/convolutional-autoencoders/. Accessed 15 Apr 2021
8. Ciresan, D., Masci, J., Meier, U., Schmidhuber, J.: Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA) (2011)
9. Cholyshkina, O., Dolgikh, S., Karpenko, D., Prystavka, P.: Automated object recognition system based on convolutional autoencoder. In:10th International Conference on Advanced Computer Information Technologies (ACIT) (2020). https://doi.org/10.1109/ACIT49673.2020.9208945
10. Sabrol, H., Singh, N.: Convolutional neural networks—an extensive arena of deep learning. A comprehensive study. Arch. Comput. Methods Eng. (2021). https://doi.org/10.1007/s11831-021-09551-4
11. Mílaga-Chuquitaype, C., McLean, T., Kalapodis, N., Kolonas, C., Kampas, G.: Optimal arch forms under in-plane seismic loading in different gravitational environments. Earthq. Eng. Struct. Dyn (2022). https://doi.org/10.1002/eqe.3626
12. Deutsch, C., Deutsch, J.: Latin hypercube sampling with multidimensional uniformity. J. Stat. Plan. Inference (2012). https://doi.org/10.1016/j.jspi.2011.09.016
13. Chollet, F.: Deep Learning with Python. Manning (2017)
14. Kalapodis, N., Kampas, G., Ktenidou, O.: A review towards the design of extraterrestrial structures: From regolith to human outposts. Acta Astronaut. **175**, 540–569 (2020). https://doi.org/10.1016/j.actaastro.2020.05.038
15. Kalapodis, N., Zalachoris, G., Ktenidou, O.-J., Kampas, G.: On the seismic behaviour of monolithic lunar arches subjected to moonquakes. Earthq. Eng. Struct. Dyn. (2023)
16. Oberst, J., Nakamura, Y.: A seismic risk for the lunar base. In: The Second Conference on Lunar Bases and Space Activities of the 21st Century, vol. 1 (No. CONTRIB-769). NASA. Johnson Space Center (1992)
17. McLean, T., Málaga-Chuquitaype, C., Kalapodis, N., Kampas, G.: OpenArch: an open-source package for determining the minimum-thickness of arches under seismic loads. SoftwareX (2021). https://doi.org/10.1016/j.softx.2021.100731
18. SIMULIA.: Abaqus scripting reference guide. Abaqus 6.14 http://130.149.89.49:2080/v6.14/books/ker/default.htm (2021). Accessed 21 May 2021
19. Goulas, A., Binner, J.G., Engstrøm, D.S., Harris, R.A., Friel, R.J.: Mechanical behaviour of additively manufactured lunar regolith simulant components. Proc. Inst. Mech. Eng. (2019). https://doi.org/10.1177/1464420718777932