de Miguel, Jaime; Eugenia Villafane, Maria; Piskorec, Luka; Sancho-Caparrini, Fernando

Deep Form Finding Using Variational Autoencoders for deep form finding of structural typologies

Published in:
ECAADE SIGRADI 2019: ARCHITECTURE IN THE AGE OF THE 4TH INDUSTRIAL REVOLUTION, VOL 1

Published: 01/01/2019

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
de Miguel, J., Eugenia Villafane, M., Piskorec, L., & Sancho-Caparrini, F. (2019). Deep Form Finding Using Variational Autoencoders for deep form finding of structural typologies. In JP. Sousa, GC. Henriques, & JP. Xavier (Eds.), *ECAADE SIGRADI 2019: ARCHITECTURE IN THE AGE OF THE 4TH INDUSTRIAL REVOLUTION, VOL 1* (Vol. 1, pp. 71-80). (eCAADe proceedings). eCAADe.
http://papers.cumincad.org/data/works/att/ecaadesigradi2019_514.pdf

Deep Form Finding

Using Variational Autoencoders for deep form finding of structural typologies

Jaime de Miguel[1], Maria Eugenia Villafañe[2], Luka Piškorec[3], Fernando Sancho-Caparrini[4]
[1,4] University of Seville [2] The Bartlett UCL [3] Aalto University
[1,4] {jdemiguel\fsancho}@us.es [2] maria@sofiaproject.net [3] luka.piskorec@aalto.fi

In this paper, we are aiming to present a methodology for generation, manipulation and form finding of structural typologies using variational autoencoders, a machine learning model based on neural networks. We are giving a detailed description of the neural network architecture used as well as the data representation based on the concept of a 3D-canvas with voxelized wireframes. In this 3D-canvas, the input geometry of the building typologies is represented through their connectivity map and subsequently augmented to increase the size of the training set. Our variational autoencoder model then learns a continuous latent distribution of the input data from which we can sample to generate new geometry instances, essentially hybrids of the initial input geometries. Finally, we present the results of these computational experiments and lay out the conclusions as well as outlook for future research in this field.

Keywords: *artificial intelligence, deep neural networks, variational autoencoders, generative design, form finding, structural design*

INTRODUCTION

In recent years, we are witnessing a proliferation of machine learning tools and methods in both academic as well as professional fields. In this paper, we are aiming to present a methodology for generation, manipulation and form finding of structural typologies using variational autoencoders. Machine learning and especially neural networks are already widely implemented for solving problems of geometrical complexity and data manipulation inherent to pattern recognition of 2D images (Krizhevsky, Sutskever and Hinton, 2012) as well as various types of manifolds such as 3D geometry and graphs (Kelly et al, 2018) (Wang et al, 2018). Depending on the problem and the discipline, these manifolds can fall within a wide variety of geometrical domains. However, beyond the training involved in the initial tasks of recognizing patterns to streamline complex processes (Luo, Wang and Xu, 2018) (Yetiş et al, 2018) (Wu and Kilian, 2018), we argue that the most promising implementation techniques of neural networks in design rely on their ability to become generative.

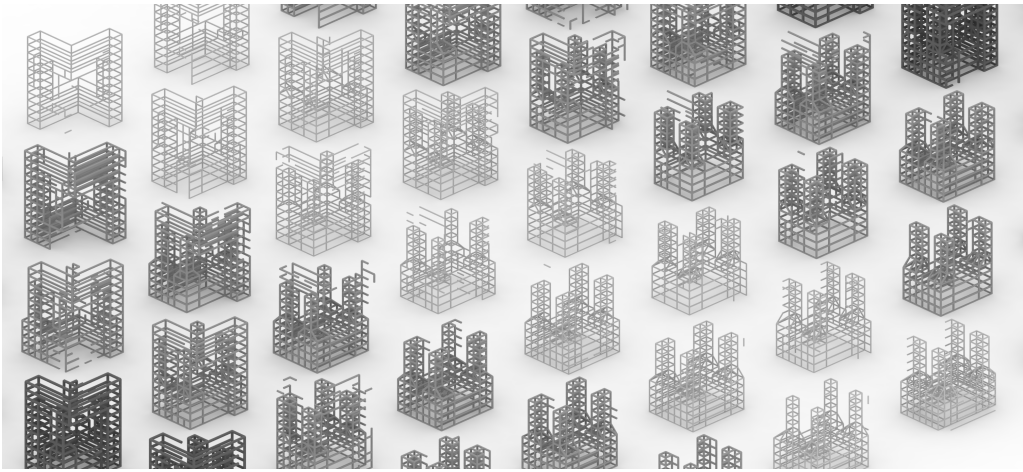


Figure 1
Output of VAE trained model obtained by taking a series of locations within the continuous latent distribution. These locations are then reconstructed following their resulting connectivity map.

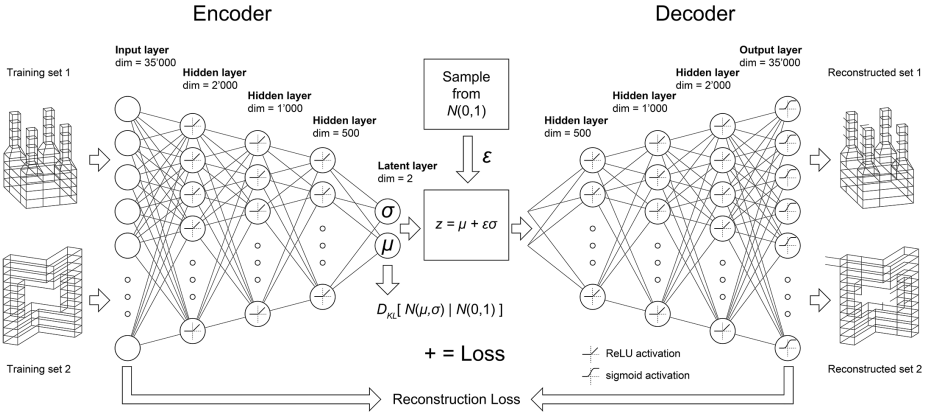
Some work within the design community that maps the potential of this approach has already been conducted (Cudzik and Radziszewski, 2018). Powerful techniques inherent to these models like sampling (White, 2016) and feature vector arithmetic (Radford, Metz and Chintala, 2016) show great promise in the context of design. Variational autoencoders, which are a recent development of neural networks, are shown to be able to generate many types of complex data. Although initially trained on sets of labeled images, proliferated use of these models in wider disciplines has driven the need for working with vector data and 3D geometry (Gregor et al, 2015) (Ha and Eck, 2015). To reduce the dimensionality of the input data, some neural network models transform the raw 3D geometry input into its voxelized representation (Wu et al, 2016).

Our approach involves taking the centerlines of the interconnected structural elements that compose a building to obtain a wireframe that is representative of its core geometry. Represented as a connectivity map, we used this geometry as an input data for a variational autoencoder, opening up a methodology within 3D machine learning that is applicable to the field of structural design. After training on an augmented data set, the network learned to identify various types of wireframes. In the last stage of the process, we made use of the generative capability of the network by sampling new points from the continuous latent distribution that the model has learnt. The autoencoder then outputs their corresponding connectivity maps that result in newly generated wireframes.

VARIATIONAL AUTOENCODER

Our variational autoencoder network is taken directly from the paper "Auto-encoding variational Bayes" by Diederik P. Kingma and Max Welling from 2014 (Kingma and Welling, 2014). We also consulted a very good summary of variational autoencoders and their inner workings written by Kevin Frans from 2016 [2] as well as "Tutorial on Variational Autoencoders" paper (Doersch, 2016). For implementation of the model in Keras we consulted the work by Francois Chollet from 2016 [1]. We adapted the number of neurons in intermediate layers to accommodate for our input layer dimension and added custom parts for data input and output. Code is implemented in Python using Keras as a high-level API for TensorFlow, a machine learning library developed by Google. Our

Figure 2
Diagram of VAE model architecture with 150 million trainable parameters.



variational autoencoder architecture therefore consists of an encoder and a decoder network with a latent layer in between that encodes two latent variables of the probability distribution.

The encoder network consists of the input layer with 35,100 neurons, 3 intermediate fully connected layers consisting of 2,000, 1,000 and 500 neurons respectively. This number of neurons in each intermediate layer is derived from a grid search approach. The decoder network is the inverse of the encoder network, ending with an output layer of the same dimension as the input layer. All neurons use rectified linear unit (ReLU) as an activation function, except in the output layer that uses sigmoid activation. The output of the network is fully reconstructed from the in-between latent layer that encodes the input data probability distribution. Results from the network are directly written into a connectivity map in plain text format, which is later read in Rhino to reconstruct the wireframe geometry.

Figure 3
Our loss function is summing two separate losses: the reconstruction loss and KL divergence.

$$l_i\ (\theta, \phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}\Big[\log p_\phi\ (x_i \mid z)\Big] + \mathbb{KL}(q_\theta(z \mid x_i)\ ||\ p(z))$$

Reconstruction loss is measuring how well do the outputs of the network match the inputs and is calculated as a mean squared error between the two. KL divergence measures the difference between the distribution of the latent variables and a unit Gaussian. Added together, these two losses force the network to generate latent vectors that roughly follow a unit Gaussian distribution (Doersch, 2016). As the latent layer contains only two variables, we can sample the distribution and plot its results on a 2D grid. For network training, we used the GPU cluster at the University of Seville. It took approximately 20-30 minutes to train a network for 25 epochs.

DATA REPRESENTATION

In order to train a variational autoencoder it was crucial to prepare our 3D geometry in a way that can be parsed through the network. This means choosing a most compact way to represent the geometry, avoiding redundancies whilst retaining full information. We call our proposed model 3D-canvas as it consists of a rectangular 3D volume discretized in cube-shaped cells within which the input geometry is contained. Each cell of the 3D-canvas contains labeled connectivity vectors that can be activated or deactivated depending on the input geometry. These connectivity vectors represent wireframe segments in different orientations that are used to

approximate the input geometry in 3D space. To keep our data representation compact and without overlaps, we discarded parallel connectivity vectors and chose only 13 for each cell according to Figure 4. For understanding, we included a corresponding Figure 3 that shows the same principle working in 2D space.

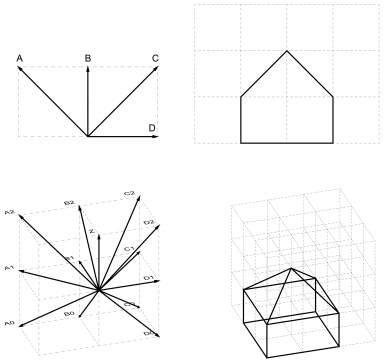


Figure 4
Connectivity diagram for 2D.

Figure 5
Connectivity diagram for 3D.

At the beginning of the routine, a 3D-canvas is generated "around" the input geometry, so that the full extent of the input can be encoded within it. Wireframe line segments of the input geometry are then snapped to the grid defined by the cube-shaped cells of the 3D-canvas and corresponding connectivity vectors are mapped according to each one of the orientations of the snapped line segment. This process is iterated throughout each cell of the 3D-canvas (which acts as the container of the input geometry) until the full extent of the input geometry is described in this way. This information is then stored in a connectivity map with its corresponding grid coordinate (3D-canvas) and values for the 13 connectivity vectors are marked with a letter and a number indicating its orientation in each cell. To make it easier for a variational autoencoder to train, instead of using a binary value for the presence of a connectivity vector (0 or 1) we used a continuous value representing a percentage with a domain [0, 1]. Our chosen activation threshold within this domain was arbitrarily set to 0.6 and kept for all training samples, which means that values above this threshold indicate the presence of a wireframe segment in that cell, while values below indicate it's absence. Although not implemented in the current model, this continuous value could be used to encode the thickness of the structural wireframe element, where smaller values would correspond to thinner elements and larger values to thicker elements. This principle could be used to encode other structural or material properties as well.

Connectivity map

Parsing the input geometry into a connectivity map was implemented in Rhino using a custom written Python script. The connectivity map itself is stored as a plain text file and read directly by the Python script used to train the variational autoencoder. In order to be used as an input, the connectivity map needs to be "flattened" and the value for each connectivity vector for each cell of the 3D-canvas mapped onto a single input neuron. In our training examples, we used a 3D-canvas with dimensions 14x14x11. The 3D canvas is then composed of 14 cells along the x-axis, 14 cells along the y-axis, and 11 cells along the z-axis. This resolution is sufficient to distinguish structural typologies like arches, wall and surface elements, volumes with cavities, openings etc. and allowed us to design and implement instances of neural networks which are trained for recognition and handling of data pertaining to the field of architectural geometry. With each cell having 13 connectivity vector values assigned to them, the number of input neurons for the variational autoencoder is therefore calculated as 15x15x12x13 = 35,100. Here we calculate with 15 instead of 14 as a dimension, as we need to consider one additional node to define the snapped lines that might be inscribed in the top, rear and lateral faces of the 3D-canvas. As the number of training parameters in a neural network grows with the number of input neurons, this size of the 3D-canvas was at the limit of what we could work with in terms of available computational power. Future improvements to the

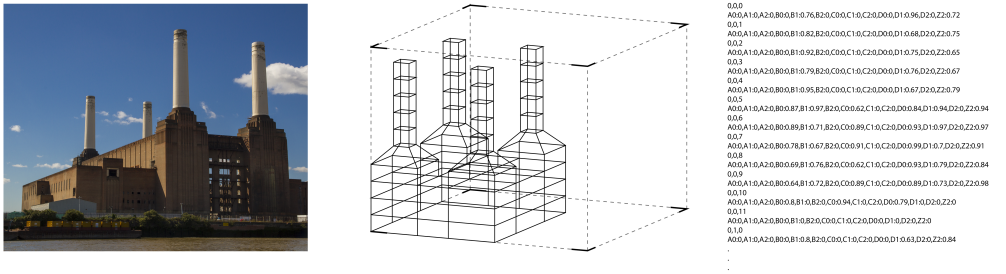


Figure 6
Battersea Power Station, its representative geometry as a snapped set of lines within the 3D-canvas and a part of its corresponding connectivity map.

training model should include a different approach to parsing the 3D input geometry, possibly using a convolutional neural net architecture. This would enable using larger 3D-canvases and correspondingly parsing models in higher resolution while at the same time significantly reducing the number of trainable network parameters.

Data augmentation

Training neural networks of any kind requires large amount of input data. Additionally, this data should ideally be as continuous as possible in terms of raw input values. For images of 3D objects, this continuity implies gradual changes in translation, rotation, scale, color and brightness values of the input data. Artificially added noise on the level of pixels can as well help the network to learn the underlying meaningful features in the data. A technique called data augmentation is often used in order to enlarge the number of training samples by artificially adding variation to the input data (Goodfellow, Bengio and Courville, 2016). In our model, the number of initial training samples was only two, corresponding to two input models between which the autoencoding was to be done. To augment our training data, we used simple random translation of the input models inside the 3D-canvas to increase the number of training samples to 3,000. At the beginning, we also experimented with rotation, but this was not successful, as rotations in a rectangular grid could not be implemented as gradual transformations but only as 90-degree "jumps". These discretely rotated samples would be considered as separate training sets by the neural network and slow down the training. We found similar limitations for scaling the input models. Based on these findings, we used only discrete randomized translation to augment the training data. Future improvements of the model could include introduction of noise in the form of randomly activating or deactivating connectivity vectors based on some small probability value. This would increase the robustness of the variational autoencoder training process and make the network better learn features that are invariant to the noise. Finally, we exported the complete training set with augmentation into the connectivity map from which it can be read into the training model.

CASE STUDIES AND RESULTS

As mentioned previously, for training we ran tests where we used input models augmented to 3,000 samples each, totaling 6,000 samples for the two-input models tests and 9,000 samples for the three-input models tests, at each epoch. Variational autoencoder network then learned the underlying distribution of the input data, mapping each initial data set (the learnt input geometries) onto a continuous latent space. Because this latent space is continuous and two-dimensional (there are only two latent variables), we can sample this space and plot the results on a grid, thus obtaining the geometries that lay in between the distribution of the original in-

puts. Due to their "bottleneck" architecture, autoencoders are forced to learn compact representation of the input data (Goodfellow, Bengio and Courville, 2016). This process can most efficiently be achieved by learning the underlying translationally invariant high-level features in the data and map it into a higher-dimensional latent space. Assumption is that all geometries that share certain higher-level features (like a spatial gap under the bridge or an arch or protruding thin volumes of towers from a castle for example) are mapped as close points on a higher-dimensional manifold in the latent space (Sohn, Lee and Yan, 2015).

Variational autoencoders enable us to sample this higher-dimensional manifold in a continuous manner and to obtain variations or hybrids of initial input geometries, success of which we can directly evaluate visually by examining the geometries sampled from the distribution. Thus, we obtained interpolated geometries resulting from the sampling of the positions within the learnt distribution of the original input models. By being able to identify positions representative of specific geometric instances within the latent space, the possibility for thinking about pseudo-semantic operations with vectorized geometric entities can be considered as further implementation of this technique. Following our 3D-canvas and connectivity vector approach, this might be challenging due to the high n-dimensional vector that is representative of the geometry of each sample within the dataset. We hope that this paper can help exploring further methods that allow for this type of vector-based geometric operations.

VAE experiment no. 03. Wireframes of 2-distinct input buildings: Battersea Power Station (in purple) and CCTV (in yellow). Top view of chart representing the learnt distribution in latent space, together with reconstructed wireframes from a grid of sample points taken from this distribution. With only 25 epochs, the output proved smooth transitions in-between the reconstructed samples resembling the original input. KL = -0.5

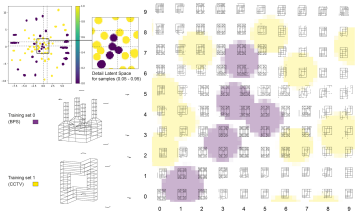


Figure 7
Experiment no. 03

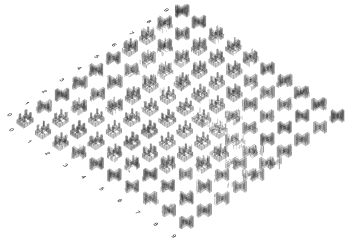


Figure 8
Experiment no. 03

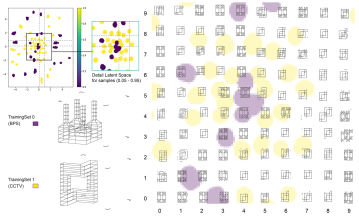


Figure 9
Experiment no. 05

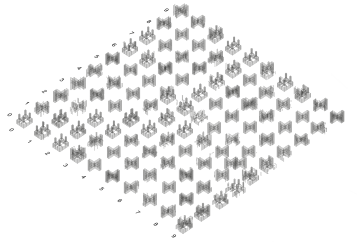


Figure 10
Experiment no. 05

VAE experiment no. 05. We have now increased the training of the model, sampling points from the latent space after 75 epochs. The model seems to be over-trained (after only 50 epochs more) and to over-recognize the outputs in latent space, offering no smooth transition in-between the grid of reconstructed wireframes. KL = -0.5

Figure 11
Experiment no. 06

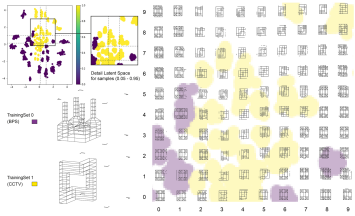


Figure 12
Experiment no. 06

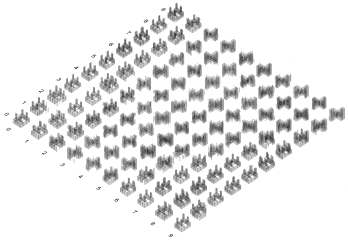


Figure 13
Experiment no. 13

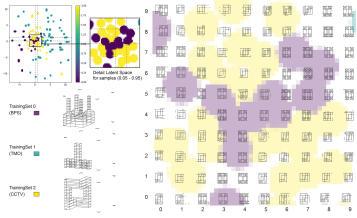
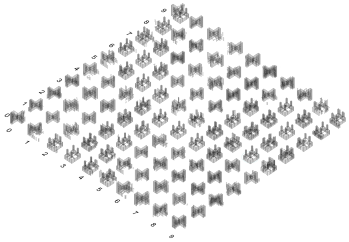


Figure 14
Experiment no. 13



VAE experiment no. 06. We sample points after 25 epochs, and change KL to -10.5. The resulting distribution is mapped into clearly defined regions. Similar to Experiment no. 05, the model seems to over-recognize the outputs in latent space, offering no smooth transition in-between the grid of reconstructed wireframes.

VAE experiment no. 13. Wireframes of 3-distinct input buildings: Battersea Power Station (in purple), Tate Modern (in cyan) and CCTV (in yellow). Top view of chart of latent distribution, together with corresponding reconstructed wireframes. With only 25 epochs, the output proved smooth transitions in-between the reconstructed samples. KL = -0.5

VAE experiment no. 15. We have increased the training of the model, sampling points from the latent space after 75 epochs. Again, the model seems to be over-trained (after only 50 epochs more) and to over-recognize the outputs in latent space, offering no smooth transition in-between the grid of reconstructed wireframes. KL = -0.5

In the series of experiments presented here, we are working with models that handle around 150 million parameters, whilst only training with 6,000 and 9,000 samples respectively as the dataset for the corresponding tests. From our experience with training the models, the number of samples in the dataset should be around 10% of the number of parameters in the model, which makes 15 million samples for the models that we used an optimal number. Therefore, further work on these models would require a greatly increased dataset in order to have a better balance of training and validation.

CONCLUSION AND OUTLOOK

Due to the growing international academic interest in generative machine learning methods and its wide commercial applications, it is clear that the workflow presented in this paper is just the beginning of a burgeoning field. Although developed 5 years ago, the use and usefulness of variational autoencoders in the context of architectural design remains mostly unexplored. We hope that this paper will show the potential of these methods to a wider design audience, specifically in application to working with 3D geometries. Unfortunately, this is as well one of the technical bottlenecks in extending the application to larger geometries, namely the fact that the amount of input parameters scales proportionally with the bounding volume of the input geometry. This makes working

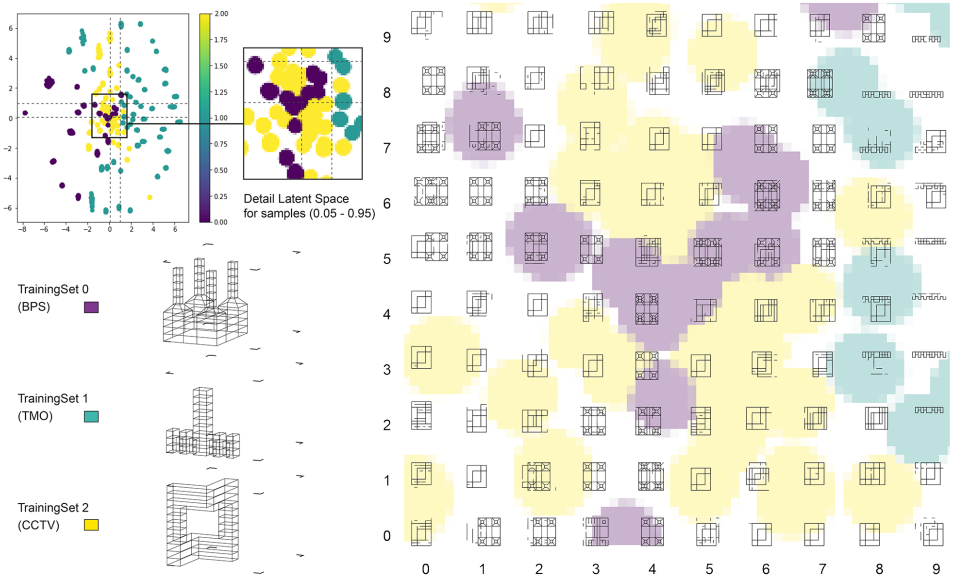


Figure 15
VAE experiment no. 15: Battersea Power Station, Tate Modern and CCTV.

with large bounding boxes unpractical due to long network training times, even if the training is conducted on a GPU cluster. Existing techniques like introducing convolutional layers into the variational autoencoder could be used to solve this problem. To improve the learning of the model proposed in this paper, it would be useful to use a data set not derived from few augmented instances, but rather from a larger repository of 3D wireframe geometries of structural typologies. The authors of this paper are currently not aware of the existence of any such repository.

In addition, to improve the quality of generated geometries, switching to a completely different neural network model could be called for, as generative adversarial networks (GANs) proved to be even more powerful in terms of learning higher-level features of the data and using this to reconstruct plausible instances of it. In that sense, GANs could be used instead of variational autoencoders as design generators. Future research should focus on pros and cons of using either of these models, especially in terms of current research being conducted on merging the two models to parse 3D geometries (Wu et al, 2016). Finally, as already mentioned earlier in the paper, continuous values in variables representing connectivity vectors could be used to encode other structural or material properties of the wireframe members like thickness, material strength, etc. This would contribute to pushing this research towards a more material driven design with holistic structural understanding, rather than just focusing on the underlying geometry.

The proposed workflow challenges designers to acquire a critical perspective of the impact and potential of AI in our society and design practices. We believe that generative neural network models have a large potential to redefine how architects and designers work with architectural precedents, namely to use them directly as data for design generation.

Figure 16
VAE experiment no. 15: Reconstructed wireframes from a grid of sample points taken from this distribution.

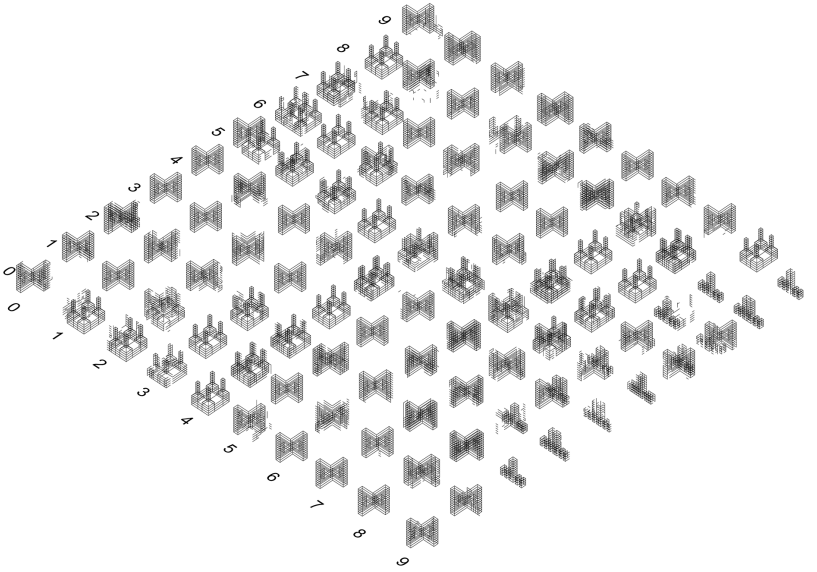


Figure 17
Output of VAE trained model obtained by taking a series of locations within the continuous latent distribution - CCTV on the left and Battersea Power Station on the right.

Work presented in this paper shows the potential of such techniques and the future of AI in the context of architectural geometry generation and the potential that bespoke AI-based tools for architects can bring to the field of structural deep form finding.

REFERENCES

Cudzik, J and Radziszewski, K 2018 'Artificial Intelligence Aided Architectural Design', *Proceedings of the 36th eCAADe Conference - Volume 1*

Doersch, C 2016, 'Tutorial on Variational Autoencoders', *arXiv preprint*, arXiv:1606.05908

Goodfellow, I, Bengio, Y and Courville, A 2016, *Deep Learning*, MIT Press

Gregor, K, Danihelka, I, Graves, A, Rezende, D and Wierstra, D 2015 'Draw: A recurrent neural network for image generation', *ICCV 2015*

Ha, D and Eck, D 2017, 'A neural representation of sketch drawings', *arXiv preprint*, arXiv:1704.03477

Kelly, T, Guerrero, P, Steed, A, Wonka, P and Mitra, NJ 2018, 'FrankenGAN: Guided Detail Synthesis for Building Mass-Models Using Style-Synchonized GANs', *arXiv preprint*, arXiv:1806.07179

Kingma, DP and Welling, M 2014 'Auto-encoding variational Bayes', *ICLR 2014*
Krizhevsky, A, Sutskever, I and Hinton, G 2012 'Imagenet classification with deep convolutional neural networks', *NIPS 2012*
Luo, D, Wang, J and Xu, W 2018 'Applied Automatic Machine Learning Process for Material Computation', *Proceedings of the 36th eCAADe Conference - Volume 1*
Radford, A, Metz, L and Chintala, S 2016, 'Unsupervised representation learning with deep convolutional generative adversarial networks', *arXiv preprint*, arXiv:1511.06434
Sohn, K, Lee, H and Yan, X 2015 'Learning structured output representation using deep conditional generative models', *NIPS 2015*
Wang, TY, Cylan, D, Popovic, J and Mitra, NJ 2018, 'Learning a shared shape space for multimodal garment design', *arXiv preprint*, arXiv:1806.11335
White, T 2016, 'Sampling generative networks', *arXiv preprint*, arXiv:1609.04468
Wu, K and Kilian, A 2018 'Designing Natural Wood Log Structures with Stochastic Assembly and Deep Learning', *Robotic Fabrication in Architecture, Art and Design 2018*
Wu, J, Zhang, C, Xue, T, Freeman, WT and Tenenbaum, JB 2016, 'Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling', *arXiv preprint*, arXiv:1610.07584
Yetiş, G, Yetkin, O, Moon, K and Kılıç, Ö 2018 'A Novel Approach for Classification of Structural Elements in a 3D Model by Supervised Learning', *Proceedings of the 36th eCAADe Conference - Volume 1*
[1] https://blog.keras.io/building-autoencoders-in-keras.html
[2] http://kvfrans.com/variational-autoencoders-explained/