Augmented Intelligence for Architectural Design

Building a vertical garden

27.04.2021 - Talks with Industry Partners

Luis Salamanca, Fernando Perez Cruz (Swiss Data Science Center)
Ania Apolinarska, Matthias Kohler (Gramazio Kohler Research, ETH Zurich)
Design paradigms: “classic” & parametric design

- **Design parameters (W)**: radii, constellations, grid points
- **Generate design**: outline curves
- **Simulate & analyse**: performance attributes (X)
  - sun occlusion
  - rain occlusion
  - surface area
Design paradigms: "classic" & parametric design
Design paradigms

“classic” & parametric design

ML-based design
Case Study: “Semiramis” the vertical garden

Task:
Design shapes (outlines)
of (5) stacked plant bowls
Case Study: “Semiramis” the vertical garden

Performance attributes $X$:
- Sun occlusion
- Rain occlusion
- Total planting area

$m^2$
Case Study: “Semiramis” the vertical garden

Design parameters $W$:

“Constellation” of support points → Outline curve using a signed distance function
Radii

Constants:

Heights
Grid points
115 possible constellations

→ $115^5 = 20'113'571'875$
combinations of constellations
for 5 plant bowls
Endless different outline shapes
Design workflow
Autoencoders - intuition and applications

A deep neural network to learn efficient latent representations of some input data in an unsupervised way

Objective: lower dimensional latent representation $Z$ that still allows a precise reconstruction of the input $W$

- Compressed representation that makes used of the inherent structure of the data
Autoencoders - for generative design

![Diagram of autoencoder](image)

- **Encoder**
  - Design parameters → Encoding → Latent space

- **Hidden layer**

- **Decoder**
  - Latent space → Training phase

- **W** (Input)
- **Z** (Latent space)
- **W'** (Output)
Autoencoders - for generative design

**Generation phase**

- The latent space learns the “fundamentals of the geometry”

- Sampling from $Z$ should lead to the generation of new design parameters, i.e. feasible geometries
Conditional Autoencoders for architectural generative design

### Diagram

- **Encoder**
  - Design parameters → Simulation
  - Simulation → Performance attributes

- **Decoder**
  - Performance attributes → Training phase

- **Hidden layer**
  - $W$ → $X$ → $W'
Conditional Autoencoders for architectural generative design

Targeted generation

- Given some requested $X$, provide a set of solutions that satisfy it
- The designer can then focus on the aesthetic process, or improving the obtained geometries
A fundamental problem with $X$: the non-bijectivity between the performance attributes and the design parameters, i.e., totally different geometries can lead to the same values of attributes.
Conditional Autoencoders - model for vertical gardens

- During training we can learn the mapping in the encoder from $W$ to $X$
- The latent dimensions $Z$ allow capturing the degrees of freedom
Conditional Autoencoders - model for vertical gardens

To **train** a neural network we need to define some objective function to minimize

- The reconstruction error between $W$ and $W'$ (MSE for radii and NLL for constellations)
- Error between given $X$ and output of the encoder $X'$
- Regularizer on the mean and variance of $Z$ to impede it overtakes the generation

$$L(W, X) = \lambda_x L_X + \lambda_w L_W + \alpha L_Z + \beta L_{Z_\sigma}$$
Conditional Autoencoders - pipeline for training and generation

**Training**: generate with Grasshopper thousands of random instances, i.e. tuples of \( W \) and \( X \)

1. Train the autoencoder using all generated samples

\[
L(W, X) = \lambda_X L_X + L_W + \alpha L_Z + \beta L_{Z\sigma}
\]
Conditional Autoencoders - pipeline for training and generation

**Training:** generate with Grasshopper thousands of random instances, i.e. tuples of $W$ and $X$

1. Train the autoencoder using all generated samples
2. Only forward pass to learn the distribution of $X$ and $Z$ as multivariate Gaussian

\[ p(X, Z) \sim \mathcal{N}(0, \Sigma_{xz}) \]
Conditional Autoencoders - pipeline for training and generation

**Training:** generate with Grasshopper thousands of random instances, i.e. tuples of $W$ and $X$

1. Train the autoencoder using all generated samples
2. Only forward pass to learn the distribution of $X$ and $Z$ as multivariate Gaussian

**Prediction:** given some requested $X_d$

1. Use the estimated distribution to obtain samples of $X_r$ and $Z$

$$X_r, Z \sim \mathcal{N}(\mu, \Sigma_{X_r, Z|X_d})$$
Conditional Autoencoders - pipeline for training and generation

Training: generate with Grasshopper thousands of random instances, i.e. tuples of $W$ and $X$

1. Train the autoencoder using all generated samples
2. Only forward pass to learn the distribution of $X$ and $Z$ as multivariate Gaussian

Prediction: given some requested $X_d$

1. Use the estimated distribution to obtain samples of $X_r$ and $Z$
2. Pass them through the decoder to generate design instances

$$X_r, Z \sim \mathcal{N}(\mu, \sum_{X_r,Z|X_d})$$
Conditional Autoencoders - pipeline for training and generation

**Training:** generate with Grasshopper thousands of random instances, i.e. tuples of $W$ and $X$

1. Train the autoencoder using all generated samples
2. Only forward pass to learn the distribution of $X$ and $Z$ as multivariate Gaussian

**Prediction:** given some requested $X_d$

1. Use the estimated distribution to obtain samples of $X_r$ and $Z$
2. Pass them through the decoder to generate design instances
3. Feed the generated geometries again through the encoder

Assess errors

This predicted value of performance attributes may differ from the requested ones $X_d$
Conditional Autoencoders - error assessment

Given the geometries generated by the decoder, and the values requested by the designer $X_d$, now we can calculate the following errors:

**Prediction error**
$$|\tilde{X} - X_d|$$

“How useful the AE is to select geometries”

**Request error**
$$|\tilde{X}^{DA} - X_d|$$

“How good the surrogate model is”

**Model error**
$$|\tilde{X}^{DA} - \tilde{X}|$$

“How good the AE model is generating new geometries”
**Quantitative evaluation**

**Prediction error**

$$|\tilde{X} - X_d|$$

“How useful the AE is to select geometries”

- This error can be obtained for thousands of geometries in few seconds.
- Used to select the best geometries that are provided to the designer.
Qualitative evaluation

Proposed geometries for total area of 220 square meters, and 50% of rain occlusion
Qualitative evaluation

- Some crazy requests... which allowed us drawing the following conclusions

Other geometries are quite common, and easily satisfiable

There are geometries that are simply unfeasible

The model can actually extrapolate, and suggest reasonable solutions even when it has seen only a few training samples, or none at all

The model is not a simple look-up table... is it creative then?
Qualitative evaluation

10 random renders (chosen using the prediction error) for the previously requested cases - plan view
Qualitative evaluation

10 random renders (chosen using the prediction error) for the previously requested cases - plan view
What’s next?

• Add binary constraints (“flags”) to performance attributes

• **Generalize**: test on different design tasks

• Interactive sliders that adjust according to confidence intervals

• Active learning for improving the generation of off-distribution samples

• Understanding the latent space $\mathbf{Z}$
Take home messages

• Explore the design space fast & easy

• Discover the unexpected

• Augment designer’s intuition
Thank you!

Ania Apolinarska  Matthias Kohler  Luis Salamanca  Fernando Perez
Backup slides
Conditional Autoencoders - implementation details

- Performance attributes $X$: rain and sun occlusion, and outline and surface for each platform
- Fully connected, batch normalization and RELU layers
- Hyperparameters tuned through grid search
- Input dimensionality: 630 (115 constellations + 11 supports, for 5 platforms)
- FC layers: dimensionality of 800
- Dimensionality of hidden layer: 30, 12 for $X$ and 18 for $Z$
- Adam optimiser and LR of $10^{-4}$
- We can also provide a total surface and it is split among the platforms randomly (Dirichlet to split the surface)
- 470,000 samples randomly generated in GH
Further exploration of the design space

• Such a plot can be readily obtained using the AE predictions.

• It can provide information to the designer about eligible areas of exploration.
Further exploration of the design space
Stop cheating model!

(a) large & exposed

(b) requested small size

small & occluded
• **Auto encoder:** in order to enable learning an encoder that does the mapping from $\mathbf{X}$ to $\mathbf{W}$

$$L = MSE(X, X') + \lambda \cdot EXP(W) + \beta \cdot MSE(X^*, X')$$

Now, we also force the decoder to mimic the behaviour of Grasshopper.
General scheme of FAILURE

• Losses:
  • **X**: MSE between the $X^*$ and the current $X$ in the network
  • **W**: MSE between $W'$ and $W$, as we treat $W$ categorical as integers with values between -1 and 1, using a TanH in the last layer of the decoder
Not really working well: we need extra losses to force a correct configuration and number of pillars, and it is not working adequately.
• **Training phase:** learn the parameters using the training data.

• **Predict phase:** once achieved, we fix a $X$ to predict, freeze the parameters of the network, and use the gradients to only learn $W$. Some sort of adversarial training
FAIL 3

• It worked... but required many random valid initialisation of $W_{\text{cat}}$, and it is not that interactive
  • Early stopping to quickly discard some faulty configurations
  • Parallelize the prediction: loss per sample and gradient per sample.
The encoder finds the mapping to the lower dimensional space.

And the decoder undoes the transformation to reconstruct the input.

**Autoencoders - intuition and applications**

**Dimensionality reduction**
Autoencoders - intuition and applications

The AE learns a generalizable representation of the data discarding the non-structured information, i.e. the noise.
Design pipeline

Switch input (radii + constellations)

- random
- predicted by AE
- ...