Proceedings of the IASS 2024 Symposium
Redefining the Art of Structural Design
August 26-30, 2024, Zurich Switzerland
Philippe Block, Giulia Boller, Catherine DeWolf,
Jacqueline Pauli, Walter Kaufmann (eds.)

Structural Design and Optioneering of Form and Function Leveraging Generative Deep Learning

Seyedomid SAJEDI*, Alloy H. KEMP, Robert K. OTANI[b]

*Senior AI/ML Engineer, CORE Studio Thornton Tomasetti
120 Broadway, New York, NY, USA 10271
ssajedi@thorntotomasetti.com

[a] PE, AIA, Vice President, CORE Studio Thornton Tomasetti
[b] PE, Chief Technology Officer, CORE Studio Thornton Tomasetti

Abstract

This paper introduces a hybrid machine learning framework designed to accelerate optioneering in structural form and performance. We present a method that generates synthetic data through finite element analysis, leveraging structural performance metrics, such as Demand Capacity Ratios (DCR), in a computationally efficient way. A conditional variational autoencoder is customized to learn probability distributions in the multi-dimensional design space, encompassing structural performance, geometry, and external loads. We demonstrate the efficiency of this framework with a case study involving a dataset of 3.5 million steel arches. Addressing the challenges of evaluating generative AI models, we develop a statistical metric focused on the standard deviation of the differences between target and AI-generated design DCRs. This metric reveals a consistent design accuracy within a 6-8% range of the expected target DCR across 5,000 test cases. Our method synergizes physics and engineering principles to enhance model training for reliable structural design, even with limited computational resources. The generative nature of the model positions AI as a collaborative tool for engineers and architects, facilitating rather than dominating the design process. To demonstrate its practical application, we have developed a prototype web application that showcases this AI-assisted arch design workflow.

Keywords: Generative artificial intelligence, Finite-element analysis, Computer-aided optioneering, Steel arch structures, Accelerated-computing

1. Introduction

We are living in an era where computers are sparking signs of intelligence in a variety of narrow and general tasks [1]. Calling machines “intelligent” can be a subject of dispute, especially from the perspective of a structural engineer because intelligence is often demonstrated with engineering intuition, a characteristic that is partly developed through years of experience and exposure to a variety of structural design projects. Regardless, it is undeniable that modern machine learning algorithms have shown promising capabilities to explore (or better said, interpolation) in complex, high dimensional and even between multi-modal spaces (e.g., computer vision, natural language processing, audio, …) [2]. The family of algorithms used for such purposes are often categorized as generative Artificial Intelligence (AI) which opposed to the more conventional supervised learning, provide flexibility is the generated output.

The goal of this paper is to investigate the potential of generative AI in exploring the space of structural performance, geometric form, and external forces. We have studied the potential use case of this

Copyright © 2024 by Seyedomid Sajedi, Alloy H. Kemp, Robert K. Otani
Published by the International Association for Shell and Spatial Structures (IASS) with permission.

technology and how it can be integrated with the concepts of structural performance through a case study of steel arches subject to wind and gravity loads. The next section includes details of the physics-based simulations and synthetic data generation strategy. Subsequently, we will discuss the training and evaluation of the model. Considering the immense power of generative AI. We believe that AI will be one of the most impactful technologies in structural engineering. Therefore, we have dedicated a section to a high-level discussion on deployment challenges and how this technology could benefit a structural design workflow by show casing a web user interface.

2. Synthetic data

At the time of writing this paper, state-of-the-art deep learning models in computer vision and natural language processing have scaled up to billions of parameters, with larger models often demonstrating superior performance. However, the ability to train these models for generalization often owes itself to a large abundance of public internet data. In contrast, engineering design data, particularly in the context of machine learning, is not only limited in quantity but also frequently safeguarded due to valid intellectual property concerns. Structural engineering represents a distinct domain, differing significantly in the dimensionality and physics of data when compared to typical tasks for which generative AI models are trained. The volume of data necessary for obtaining reliable structural design performance from a machine learning model varies with the complexity of the design and the dimensionality of the output, presenting a question ripe for future research.

Despite these challenges, which currently hinder the application of machine learning in structural design and optioneering, there is a silver lining. Engineers possess domain-specific expertise, such as knowledge of design codes and standards, alongside a proficiency in physical simulation methods like finite element analysis (FEA). Leveraging these strengths, we demonstrate that it is feasible to synthetically generate high-quality structural design data within a reasonable compute time, which can subsequently be employed to train a generative AI model. To this end, we have considered the problem of designing steel arches which is further explained in the following sections.

2.1. Dataset

The main variables required for synthetic data generation are provided in Table 1. The goal of this generative model is to provide engineers with a series of arch designs that meet user-defined input loads, span length, and a target Demand Capacity Ratio (DCR). All possible combinations of these variables are considered to build a training dataset. Tributary width is calculated by dividing half of the span length by the corresponding coefficient in the table. This value will be used to assign distributed gravity and wind loads to the arch elements. Wind load orientations are perpendicular to the arch curve in both windward and leeward directions.

Table 1: Range and increments of synthetic data generation variables.

Variable	Min	Max	Increments	Categories
Arch geometry type	-	-	-	Parabola, Catenary, Circular
Span length	6.1 m (20 ft)	30.5 m (100ft)	0.6 m (2 ft)	-
Shape factor	1.02	2.00	0.02	-
Tributary width coef.[(a)]	1	5	1	-
Live load	0.48 kN/m2 (10 psf)	1.44 kN/m2 (30 psf)	0.24 kN/m2 (5 psf)	-
Wind load	0.48 kN/m2 (10 psf)	1.44 kN/m2 (30 psf)	0.24 N/m2 (5 psf)	-

For each datapoint, a steel section is sampled from 283 standard American W shapes [3] where elements self-weights are automatically considered in FEA. Note that there is a distinction here between the proposed method and the common labeling that is used in supervised learning. In the current method, we sample a section and record its critical DCR as metric of structural performance. Our goal is to train the generative model in such a way to learn the distribution of under and overdesigned members as well as optimum designs. This strategy will provide users with target DCR as a control parameter which allows engineers to explore conservative designs, a feature that is often desired by the structural engineer to account for uncertainties. The other advantage compared to supervised learning is more efficient

compute times. Obtaining the most efficient section for a given geometry in supervised learning, requires a brute-force (or alternatively an iterative optimization method) to check of possible sections and selecting the most economical option. Afterward, the lightest member passing the target DCR will be kept, and the other 282 evaluation are discarded which not only creates a substantial computational cost, but also is wasteful as a supervised learning algorithm will not utilize the discarded evaluations during training. Later in the paper, we will show that the generative model is capable of learning from DCRs and generating designs close to the user-defined targets.

2.3. Stratified sampling

The sampling strategy used to select the member size for each arch is also important considering the imbalance in the strength of section in the W shape database (Figure 1). The cross-sectional area can be used as a good measure to estimate the structural strength. Most elements in the shape database have an area between 0-645 cm^2 (100 in^2). We have sorted the sections based on area and split the range into 6 approximately equal strata. Subsequently, we randomly sample one section from each stratum for each data point. Continuing this process for all arches will yield a total of ~4.5 million evaluated arch datapoints. To reduce the impact of outliers, we have filtered the data to keeps datapoints with DCR<1.2.

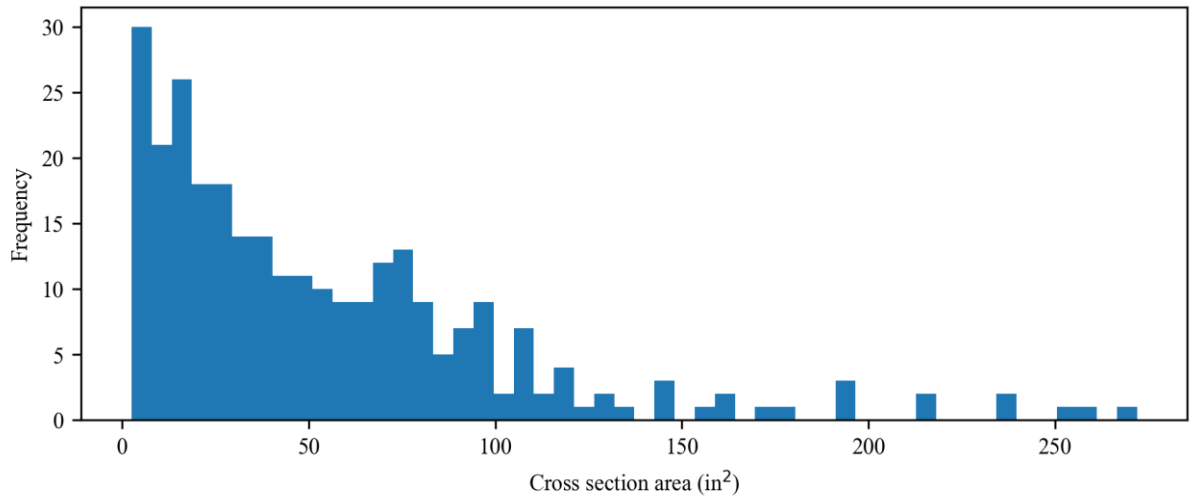

Figure 1. Histogram of cross-sectional area for the 283 W sections in the AISC shape database

2.4. Simulations

The arch geometry is approximated by 20 linear segments utilizing a Grasshopper script in Rhinoceros 3D [4]. The FEA is conducted by the Karamba3D plug-in in Grasshopper [5]. For simplicity we have assumed a 2D linear structural analysis. Simulations were conducted on 2 workstations at Thornton Tomasetti equipped with Intel Xeon Gold 6258 (28 cores), and AMD Ryzen Threadripper 5975WX (32 core) CPUs. It took approximately 10 days to complete the simulations where each workstation was running 6 analyses in parallel.

3. Machine learning

In this section we will discuss the process of training and inference of the generative model that leverages the dataset described earlier. Generative deep learning models often get their characteristic by sampling a tensor from an abstract latent space and decoding this latent representation into a meaningful output. We have adopted Conditional Variational Autoencoders (CVAE) [6] to generate arch designs considering their effectiveness and efficiency compared with more complex neural network architectures such as generative adversarial networks or diffusion models. The CVAE is comprised an encoder which is required during training and a decoder that is used both during training and inference (Figure 2).

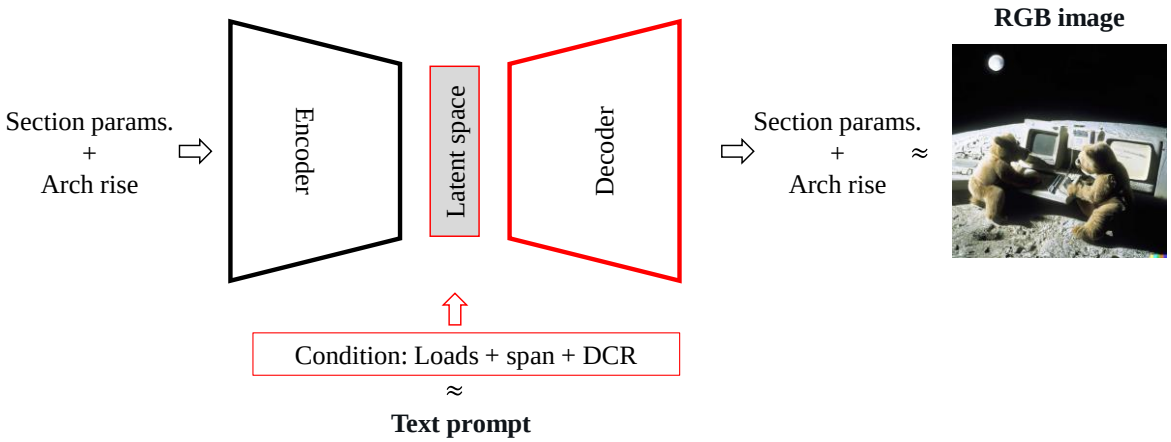

Figure 2. The CVAE architecture and its conceptual similarity to text-to-image models are explored for conditional generation. The image and text prompt, as taken from the model in [7] , assist readers in understanding the analogy of how generative models operate. It should be noted, however, that the CVAE architecture described in this paper differs from most text-to-image AI pipelines. Components outlined with a red border are utilized during the inference phase.

3.1. Training

A separate model is trained for each arch type: catenary, parabola and circular. We observed that separate models perform better than combining different arches in a larger model. The design output is characterized by vector **X** that contains normalized values of arch rise and section properties: area (A), web thickness (t_w), flange thickness (t_f), section depth (d), strong axis elastic modulus (S) and moment of inertia (I). Elements of **X** are independently normalized with their mean and standard deviation in the training set.

The encoder neural network will receive batches of **X** as an input and try to compress them into a latent representation **z** and later reconstruct the same **X** through the decoder. The loss function (L_{vae}) is the summation of Reconstruction (R) and Kullback–Leibler (KL) divergence loss functions. The second term will penalize the model for learning **z** vectors that are not following a standard normal distribution. The KL loss is essential as it allows to later sample new value from the **z** space and generate designs with the decoder model. In this study, we have noticed that introducing an additional weight factor $\alpha = 0.005$ to the loss function can improve the performance of the model at testing.

$$L_{vae} = -\alpha KL(\mathbf{X}, \mathbf{z}) + R(\mathbf{X}, \mathbf{z}) \quad (1)$$

It is also essential to condition the generated arch designs on the user input loads, span length and target DCR. These values represent another vector which we call **C**. The authors of [6] have shown that by simply concatenating **z** and **C**, during training, the decoder network will generate designs (**X**) based on the user input constraints (**C**) at inference time by sampling normally distributed random **z** vectors. This relationship is akin to how text prompts guide the generation in text-to-image models, steering the resultant designs (**X**) based on specified parameters (**C**).

For both the encoder and decoder, we consider two dense layers with 1024 hidden nodes, and a latent dimension of 16. We utilized Adam optimizer with a learning rate of 1.0e-3 and a learning rate scheduler with a patience of 5 epochs at plateau and reduction coefficient of 0.2. As an example, the training logs are provided in for the circular arch model. A validation set is used to monitor the stability of training process by holding out 10% of the original dataset. This validation set is not used to measure the structural performance of the model. A testing strategy is studied in the evaluation section.

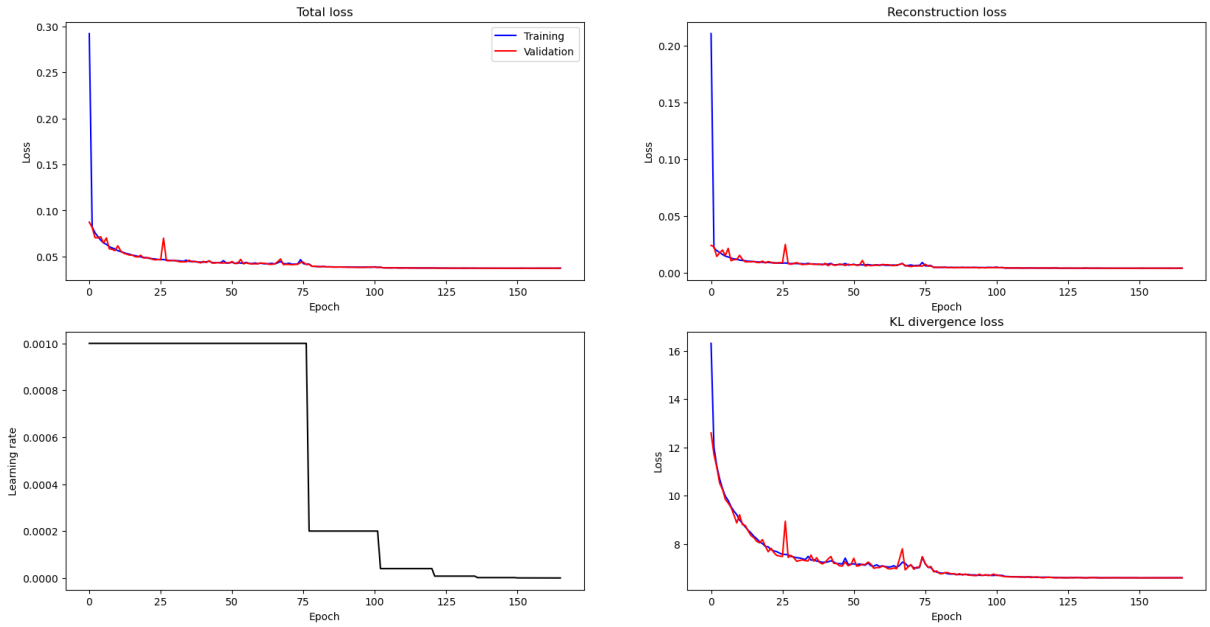

Figure 3. Training and validation losses of the CVAE model during training for circular arches.

In our study, we conducted experiments to train the CVAE models on two distinct GPUs: a modestly powered NVIDIA A2000, equipped with 4 GB of VRAM, and a more advanced NVIDIA A6000, with 48 GB of memory. We maintained identical hyperparameters for each training session. On the A2000, a lower-tier GPU, it took approximately 16 seconds to complete a single epoch, cumulating to less than an hour overall. Recall that this duration is significantly shorter than that required for synthetic data generation. Conversely, training on the high-end A6000 GPU yielded a quicker average time of 4 seconds per epoch.

This comparison aims to underscore a critical insight: engineers tasked with developing such models can significantly reduce the model size, consequently lowering the demand for computational resources, while still attaining satisfactory outcomes. It's important to note, however, that tackling more intricate design challenges, characterized by higher dimensionalities in **X**, **C**, and **z** vectors, might necessitate larger models and, by extension, greater computational power.

3.2. Inference

X contains properties of W sections which are continuous values instead of a section index. When a decoder generates new arch designs, it will be a combination of arch rises and this section parameters which do not exactly match the existing values in the shape database. Inspired by the concept of text embeddings in language models [8], we utilized cosine similar between generated properties and sections in the shape database and return to the user the closest match.

3.3. Evaluation

Measuring the performance of generative models is more challenging compared to the traditional machine learning models that are trained on labeled data. This is mainly because for a given input, there might be several feasible correct answers. As we will see in the next section, it is quite possible to have multiple arches with different combinations of rise and steel section that yield similar DCR and therefore, a one-to-one comparison is not appropriate.

To address this issue, we generate a test sample of 5000 **C** tensors that include combinations of different spans, loads and target DCRs within the acceptable bounds of the training set. Then 5000 designs are generated conditioned on **C** and the critical DCR values are measured using the physics-based approach with FEA that was used in the synthetic data generation. The quality of a model can be measuring by studying the difference between AI-generated design DCRs and the user-defined target DCR. We refer to this metric as ΔDCR. The standard deviation of ΔDCR is a good metric to quantify the structural

performance of designs. This metric is provided in Table 2 for the complete test set and two typical range of DCR that are often used by engineers. The reliability of model in the design space can also be investigated using the visualization techniques demonstrated in Figure 4.

Table 2. Standard deviation of DCR difference between actual and AI-generated designs

Data subset	Standard deviation of ΔDCR
Complete test set	9.09%
Target DCR≤1.0	6.11%
0.9 ≤Target DCR≤1.0	7.25%

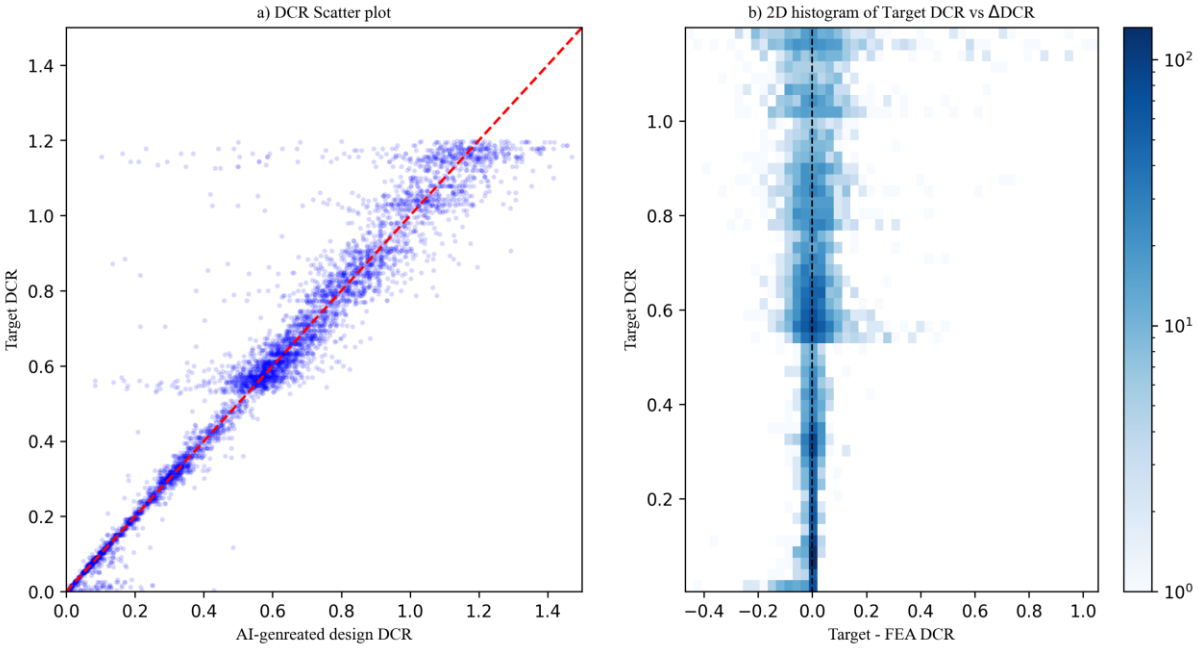

Figure 4. Performance visualization of generative model's designs with respect to different target DCRs.

4. Deployment and production

Once the training is complete, the decoder network of CVAE can be used for inference and generating new arch designs conditioned on the user inputs. A user can define a target DCR and the number of generated designs with different random seeds. As an optional step, the designs can be evaluated with FEA to measure the actual DCR of AI-generated designs. In our user testing sessions, we have noticed that structural engineers often appreciate having this additional step as it enables them with an additional physics and code-based safety check. It is also possible that some generated designs slightly exceed the target DCR. The post processing step provides the option of filtering such designs and sorting the generated designs from lowest to highest weight.

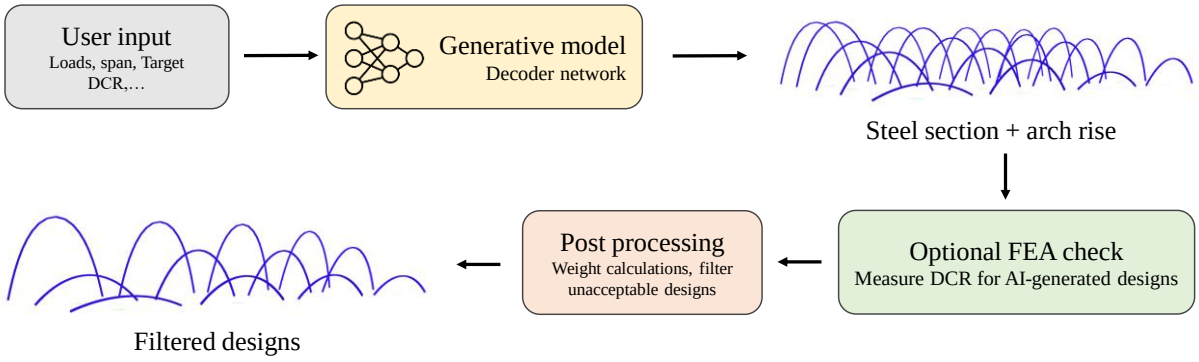

Figure 5. Inference steps, from user inputs to post processed arch designs.

The decoder network is deployed to a cloud CPU instance using Cortex (Thornton Tomasetti's MLOps infrastructure). The ML inference time varies depending on network communication overhead and the requested number of seeds. For 20 arch designs, the inference time varies between 75-150 ms which we consider acceptable for quick optioneering.

Figure 6 demonstrates the final web application that is powered by the generative arch designer. Such applications can serve as tools that can provide real time feedback for architects. Furthermore, structural engineers can also evaluate and fine-tune a larger number of designs by getting a head start using the generated options.

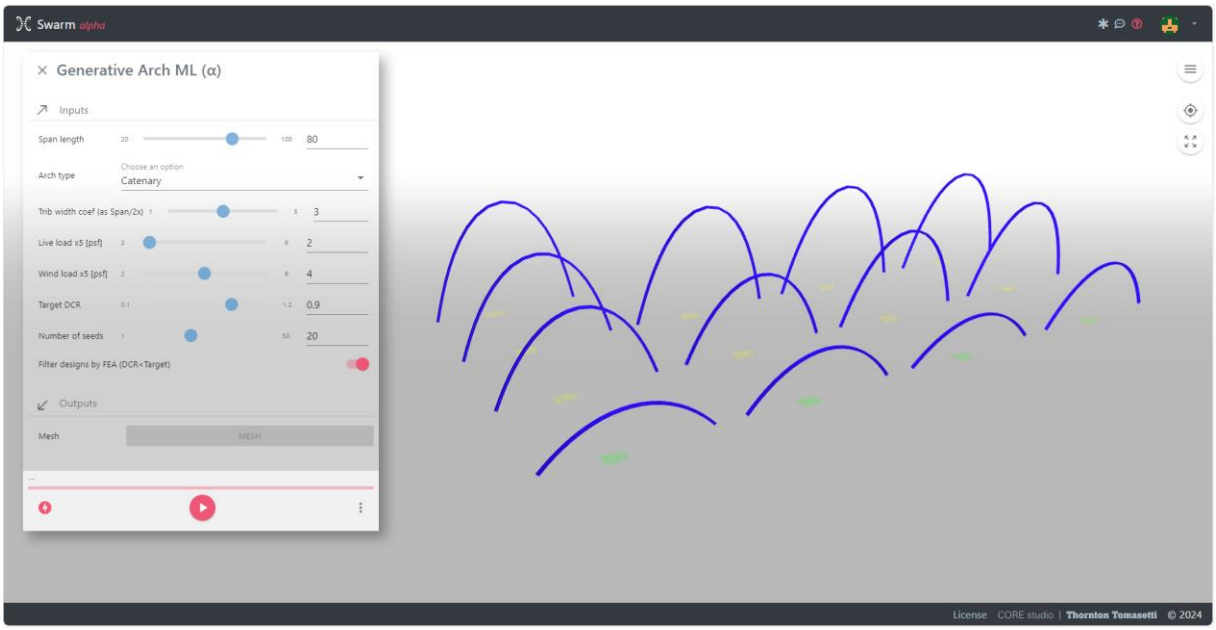

Figure 6. User interface of the arch design web application. The app will filter the designs that exceed the target DCR and sorts designs from lowest to highest tonnage.

5. Conclusion

This paper has explored the potential of generative AI for structural design and form finding. Through a case study of steel arches subject to wind and gravity loads, we have proposed solutions to address challenges in certain key areas listed as follows:

- **Synthetic data generation:** The structure of **X** and **C** tensors enables the generative model to learn from all observations including over and under designed arches. The proposed strategy allows for a more efficient use of compute power and dramatically reduces the simulation time. The steel arch dataset generation in this paper is approximately ×283 faster than a standard brute force optimization needed to generate labeled data.
- **Evaluation:** We have proposed a physics-based approach to measure the standard deviation of difference between AI-generated and target DCRs as a quantifiable performance metric that can be used to assess and optimize the structural performance of such models.
- **AI as a copilot:** This paper also presents a prototype web application that utilizes a deployed version of the generative model. It is shown that the AI generated designs can be automatically checked with an FEA engine, providing further insight for the engineers and responsible application of AI in structural design.
- **Efficient Computing:** This paper illustrates that scaling the generative model and tailoring synthetic data to precise design problems can significantly curtail computational expenses across synthetic data generation, training, and inference. We benchmark solution times to highlight these efficiencies. Furthermore, our findings reveal that the proposed framework can be effectively trained on commonly accessible workstations, eliminating the need for industrial-

grade computer clusters. This approach democratizes access to advanced modeling capabilities, making them feasible for a broader user base.

We believe that generative AI has a great potential to be an impactful technology in the structural engineering industry. The combination of efficiency, precision, and flexibility offered by such models, promises a new era in engineering design, where creativity is augmented by accelerated AI computing. However, building effective design tools is only possible by leveraging domain expertise during various stages of application development including data preparation, training, inference and even building a user interface for the responsible use of AI in structural engineering workflows. As we stand at the forefront of integrating generative AI into structural design, there is an imperative need for continued research, particularly in the realms of scalability and integration with physics-based methods. As demonstrated in this paper, leveraging the domain expertise presents new opportunities to address the challenges in training more capable models in the future.

References

[1] M. R. Morris, J. Sohl-dickstein, N. Fiedel, T. Warkentin, A. Dafoe, A. Faust, C. Farabet and S. Legg, "Levels of AGI: Operationalizing Progress on the Path to AGI," *Google DeepMind*, 2023 [Online]. Available: arXiv:2311.02462. [Accessed: 24/3/2024].

[2] J. Schulman, B. Zoph, C. Kim, J. Hilton, J. Menick and J. Weng, "ImageBind: holistic AI learning across six modalities," *Meta AI,* 2023 [Online]. Available: arXiv:2305.05665. [Accessed: 24/3/2024].

[3] American Institute of Steel Construction, Specification for Structural Steel Buildings (ANSI/AISC 360-22), 2022.

[4] R. McNeel, "Rhinoceros 3D (Version 6.0.) [Computer Software]," Robert McNeel & Associates.

[5] C. Preisinger, "Linking Structure and Parametric Geometry," *Architectural Design*, vol. 83, no. 2, pp. 110-113, 2013.

[6] K. Sohn, H. Lee and X. Yan, "Learning structured output representation using deep conditional generative models.," in *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

[7] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," 2022 [Online]. Available: arXiv:2204.06125. [Accessed: 24/3/2024].

[8] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019 [Online]. Available: arXiv:1810.04805. [Accessed: 24/3/2024].