Proceedings of the IASS Annual Symposium 2025
"The living past as a source of innovation"
27 October– 31 October 2025, México City, México>
<Juan Gerardo Oliva, Juan Ignacio del Cueto, Elisa Drago (eds.)

Deploying cGANs for Stress-Informed Adaptation and Material Redistribution in Hybrid Shell Structures

Gabrielle BROOKING*, Luis BORUNDA[a]

*The Virginia Polytechnic Institute and State University
201 Cowgill Hall 1325 Perry St Blacksburg, VA 24061
gabby.brooking@gmail.com

Abstract

This paper explores how Conditional Generative Adversarial Networks (cGANs) can be used as a tool for structural adaptation in hybrid systems such as gridshells and plate structures. The cGAN is trained to identify stress trajectories to enhance structural performance. Height-to-stress mappings act as the basis for testing how cGANs can assist in geometric adjustments decision making by providing quickly produced generalized patterns of a design system's structural identity. By embedding structural analysis and quantitative constraints into a machine learning dataset, the method provides iterative feedback on stress distribution rather than treating computational analysis as a terminal output. There are two deployment strategies: 1. Applying various magnitudes on a plate structure to evaluate the model's ability to distinguish between different levels of force based on limited representational differences in the cGANs training, 2. Redistributing material in response to generatively mapped stress on a shell structure.

Keywords: Structural Morphogenesis, Machine Learning, Hybrid Shells, Stress Maps, cGANs

1. Introduction

Structural engineers and architects have long utilized methods like form-finding and topology optimization to design efficient shell structures. However, traditional techniques often lack the ability to incorporate real-time stress feedback during the design process. This research addresses that gap by introducing a Conditional Generative Adversarial Network (cGAN)-based framework for stress-informed design adaptation. A cGAN can learn the complex relationship between applied loads, resulting stress fields, and optimal material layouts, enabling it to suggest where to reinforce or lighten a structure. Unlike conventional optimization, which must resolve each new scenario from scratch, the cGAN uses learned patterns to generate viable design information for unseen loading conditions. By deploying cGANs in this context, we aim to complement established analytical methods with a data-driven tool that can quickly produce stress maps to inform adaptive modifications in response to stress demands.

Recent advances in machine learning for structural design provide a foundation for this work. Generative models, including GANs, have been successfully used to create design candidates in architecture and engineering. A cGAN extends this concept by conditioning the generation process on specific inputs (in our case, height maps with force annotations, and stress maps that act as design constraints).

Prior studies have shown that cGANs can capture latent design features across multiple optimized structures, inspiring our application of this approach to continuous systems. We hypothesize that a cGAN trained on pairs of stress fields and optimal material configurations will learn to predict effective material redistribution for new stress scenarios. The following sections detail our framework, from

dataset curation through model training and deployment, and evaluate its performance using a representative shell case study.

2. Background

There is a lag in the canonical digital workflow's ability to aid in designing architecture that can respond to a variety of stimuli over time. While architectural systems continuously engage with variability (forces, flows, interactions), our normative design tools often externalize these constraints rather than internalize them.

In several veins of digital practice and theory, form is being challenged to move away from its permanent, static condition toward one that is contingent on changing environments. This repositions architecture as something that should operate through continuous interaction, rather than being a fixed geometry subject to evaluation only at the end of the design process or the building's lifecycle.

Extending this idea beyond physical form, we can apply similar logic to the procedural workflows that shape our designs. If force and motion are not inherently operative in shaping a system but are instead layered on later as visualized analysis, then adaptability is relegated to an aesthetic rather than a structural principle. As Greg Lynn noted, "Force and motion are eliminated from form only to be reintroduced, after the fact of design, through concepts and techniques of optical procession" [1, p.11].

Reframing adaptability as an internal computational condition, rather than just a formal response, pushes design toward systems capable of real-time feedback and evolution.

Computational infrastructures, such as Grasshopper and its associated plugins, are the backbone of our contemporary digital workflows. Its plugin ecosystem offers components to facilitate the creation of simulation and analytical scripts for structural and environmental evaluations. However, when the output is realized, the system does not automatically return and rerun based on the results. This process reveals a weak point of our current analytical workflows, which is that they produce terminal outputs. In this context, terminal is used to describe the computational characteristic of a process that produces a fixed output that is not reintegrated into the computational system that introduced it. The terminal aspect is the end of a one time operation. The results are frozen and must be evaluated completely externally by the author without any internal adjustments made by the system.

There is no internal mechanism that allows for it to revise its own operations and, therefore, its future generations. To re-engage with the script, the designer will have to adjust and rerun the script entirely, as the script does not retain memory, nor does it reason based on its past operations.

While this blank-slate structure allows for parametric flexibility, it does not support memory, hierarchy, or cross-instance reasoning. As such, insights are difficult to scale beyond isolated simulations. In high-throughput design settings, such as performance-based or simulation-driven workflows, this lack of internal referentiality hinders rather than enables deeper structural understanding.

This limitation underscores a critical gap in current parametric workflows: the absence of embedded mechanisms for self-reference and recursive learning. Addressing this gap is essential to enabling structural systems that evolve in response to performance constraints rather than merely reacting retroactively.

2.1. Neural Networks as Structural Interpreters

We can instead use neural networks as a generative tool to help us produce analytical structural mappings. Their capacity to generate visual outputs without a predefined semantic filter allows for the revelation of latent structural patterns embedded within a design.

With a specific dataset, the user can achieve a more transparent understanding of the generated output because the output is relational to the constraints created for the training dataset. Neural networks can

serve as generative instruments capable of revealing latent structural information through image-based outputs that reflect the conditions of their training data. When carefully curated, the dataset itself guides the network toward outputs that expose relational qualities embedded in design constraints. This capacity makes neural networks particularly valuable for analytical simulations, as their architectures have demonstrated the ability to extract visual features with classification performance exceeding that of human perception [2].

This ability to extract visual patterns can expose hard to discern relationships within the design's structural and spatial system that might not be obvious on manual observation done by the viewing of one instance of conditions at a time, as we do with traditional scripting analysis. Instrumentalizing neural networks as a synthesis tool for multidimensional data to create interpretable visual outputs from complex systems (spatial, structural, or material), more legible across a wide range of conditions in a reciprocal manner. If machine learning models can parse and sort through datasets that have measurable constraints and are performance-oriented, then there is value in exploring how to expose dynamics within a form. Specifically, shell structures as they are contingent on the efficient dissipation of force on a structure. In the past, we had a restricted version of this capacity, one that compartmentalized analysis as an afterthought, which disconnects it from the larger and nonlinear nature that is the design process.

	Finite Element Analysis Approach	**cGAN Based Approach**
Method	Parametric and numerically based	Generalizable, probabilistic pattern recognition
Data Cost	High-resolution meshes	Training data from FEA outputs
Computational Intensity	Robust scripting, longer load times	High upfront training cost, faster iteration cycles
Adaptability	Based on manually changing script	Self-referential

Figure 1: Table of Parametric FEA versus cGAN-based approach

2.2. Precedents: Graphical Thinking and Form-Finding

Our work builds on a long tradition of graphic statics. Originally introduced by Carl Culmann (1821–1888) in his monograph *Die Graphische Statik*, graphical statics provided a geometric framework for analyzing structural equilibrium. Rather than relying solely on numeric methods, graphical statics uses geometric diagrams to assess the stability of forces acting on a structure. The method offered coherence between design and calculation because "the clarity of graphical techniques has a high didactic value, since interdependencies, e.g., between forces and structural geometry, can be directly experienced visually" [3, p. 997].

The technique lowers the barrier of entry for designers to execute quick structural assessments. It demonstrates the relation of geometric arrangements, allowing for the analysis of spatial relationships in a simplified plane. It relies on the diagrams of the reciprocals of force polygons and funicular polygons. Graphical statics bypasses the numerical strategy of structural analysis in favor of an image based analysis methodology to determine structural stability through geometric relations. This is particularly useful in design contexts where iteration and intuitiveness are key, as it allows for visual and conceptual modifications to be completed quickly.

The mapped images, the diagrams, indicate visually, through the polygon closure, whether a system is in equilibrium via its observable condition. The reciprocal diagrams further the stability of the graphic based structural analysis by demonstrating the internal balance of forces by pairing each structural element with a corresponding force.

The iterative capacity from working with graphical statics is that it allows for designers to see the resulting impacts from geometric modifications within a graphical environment. In this context, stability is found in the visual arrangement of forces within a form, which aids in situating it as an accessible method of structural design, as it visually compiles complex structural situations in a more digestible format. Graphical statics soon faded out as analytical methods of analysis became favored, however, with the rise of computation, these structural engineering practices are able to regain a level of design influence. Computer-aided graphical statics is aiding in the reintroduction of the unity of design and engineering via image based methods. "It would facilitate the lost 'statical feeling' at the interface of design and construction becoming second nature again to the user: science would become sensory and poetry would become scientific. It could break the established role behavior of structural engineers and architects" [3, p.1005-1006]. With aid from digital software, "graphical statics" can now incorporate more complex problems in a mature framework.

2.3. From Form-Finding to Neural Feedback

As Oxman notes, "Tectonics is a seminal concept that defines the nature of the relationship between architectural design and its structural properties" [4, p.173]. Antoni Gaudí's (1852–1926) work is often cited as an early example of analog form-finding. His hanging chain models for the Sagrada Família embodied a method of determining geometry based on physical response to gravity. Weighted chains, suspended from above, naturally formed funicular curves, ideal compressive paths for masonry. Altering a single chain modified the entire model's equilibrium, demonstrating a parametric-like relationship between local conditions and global form. Inverting these geometries yielded optimized arches and vaults. This analog form-finding process was materially grounded and structurally performative: the geometry *was* the analysis. This physical methodology contrasts with later diagrammatic, image-based systems like graphical statics, but both share the ambition of fusing design and structural behavior. Similarly, machine learning (particularly generative neural networks) can compress such behavior into predictive, image-based systems. While form-finding relies on gravity as a solver, a conditional GAN (cGAN) relies on learned associations between force conditions and geometric outcomes. A cGAN can be seen as an abstract successor to form-finding: when given a force-annotated image, the network adapts its output in response, predicting structural adjustments akin to a new equilibrium.

The use of machine learning in design gained traction only after significant technical evolution. While the term "Artificial Intelligence" was first formalized by John McCarthy in 1956, early symbolic approaches emphasized rule-based logic, insufficient for handling complex spatial and material systems. It wasn't until Geoffrey Hinton's development of backpropagation in 1986 that neural networks (connectionist AI) re-emerged with the ability to learn from data. This enabled the rise of deep learning: multi-layered networks that can abstract features across many dimensions of input.

Among the architectures emerging from this era, Generative Adversarial Networks (GANs) introduced by Ian Goodfellow in 2014 [5] are particularly relevant. GANs consist of two competing networks: a generator that synthesizes data, and a discriminator that evaluates its realism. Through this adversarial process, the generator improves until its output becomes indistinguishable from real samples. Standard GANs generate data from noise vectors, limiting user control. In contrast, Conditional GANs (cGANs) introduce structured inputs (such as stress maps or class labels), so the generated output responds to a specific design condition. This conditionality allows cGANs to serve as a feedback-based design vehicle: they respond to structural stimuli with geometric adaptation, effectively folding analysis into generation. In this sense, cGANs do not merely visualize performance but *simulate it*, offering designers a predictive tool that learns to "reason" about form through structural precedent encoded in data.

3. Methodology

A cGAN extends a standard GAN by generating outputs based on a specific input condition, rather than from random noise alone. In our case, this condition is a stress distribution map. The generator uses this

input to produce structurally responsive material layouts, while the discriminator evaluates how realistic these generations are. The conditional setup ensures that each output is directly tied to the structural logic of the input, enabling design adaptations that are both targeted and informed.

This paper explores a modified workflow from Gabriella Rossi and Paul Nicholas in their paper "Encoded Images: Representational Protocols for Integrating cGANs in Iterative Computational Design Processes," [6] for the expansion of use cases of an FEA analysis incorporated into a cGAN model.

Pix2Pix [7] [8] is available online through Google Colab and TensorFlow, which allows for people to write Python code on their browser. When uploading a dataset for training on Pix2Pix, a "single" item in the dataset is 256 pixels by 512 pixels, which are the input and output images side by side (each image comprising 256 pixels by 256 pixels alone). This is because Pix2Pix is a supervised image translation model, one image correlates to another, and having the images placed together allows the pixels to correlate directly with one another (Figure 2). During the training process, the "single" item is split in half, and the input goes into the generator and the output, or target image, goes to the discriminator. The dataset is most conducive to be formatted into RGB as the Pix2Pix model has an output channel of three, meaning the pixels produced by the generator will have a value of triple digits for each channel, ranging from 0 to 255.

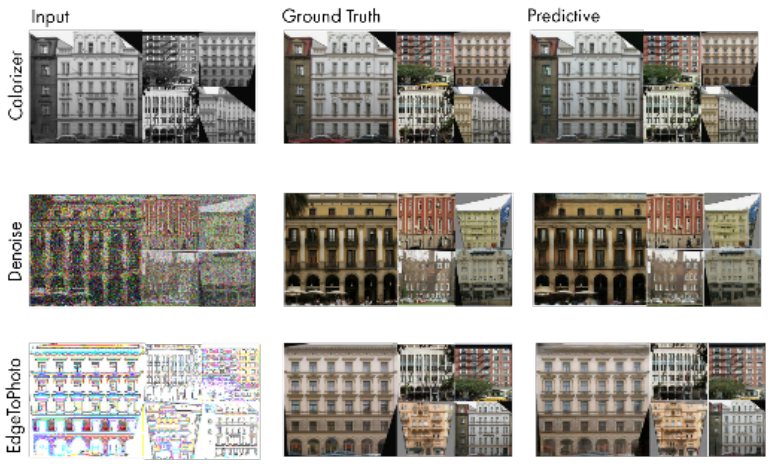

Figure 2: Results from test datasets to verify the cGAN's operation. Top: Black and white to color, Middle: Noise image to clear image, 3: Outline photos to full image. Adapted from Pix2Pix source images using the TensorFlow Pix2Pix Colab workflow. Available: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/pix2pix.ipynb [8]

Figure 2 displays the result of three test datasets. The facade images were downloaded from the TensorFlow's Pix2Pix dataset, specifically the "facade" category available on the Google Colab demo. These images were converted into black and white, edge maps, and random noise. Each modified image was paired with its original counterpart for each individual dataset in the standard 256 pixel by 512 pixel format.

These datasets were then loaded into the Pix2Pix infrastructure to test how a custom dataset could be integrated into the code environment with new code cells interacting with the original TensorFlow code cells. There were roughly twenty items of input-output objects in each dataset, however, the results proved that for this task, twenty items were robust enough to see substantial results.

3.1. cGAN-Based Structural Adaptation Framework

Figure 3 presents the structured pipeline developed for training the cGAN on stress-informed geometry adaptations. The method begins with modeling a base geometry in Rhino. Then, using Grasshopper, a color-coded "height map" is generated onto the surface to show elevation changes across the object. Separately, a finite element analysis is generated using Karamba3D. Here is where the constraints of the "images" are produced. If the constraints of the image are understood when going into the dataset, then more information can be retrieved from the cGANs' output generations.

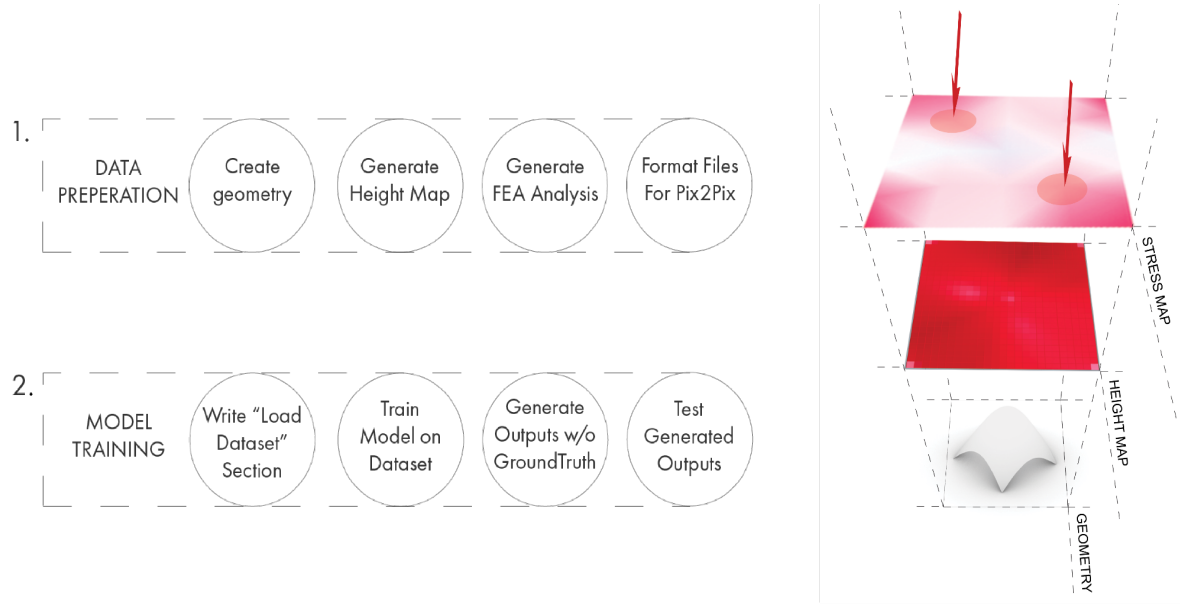

Figure 3: Left: Workflow diagram from creating the dataset to training the cGAN model, Right: Relational diagram of the dataset information

In the FEA script, the supports, gravity load, force magnitude, material, and display scale are decided. The height maps are manually marked in Photoshop to show force location (with a circle) and force magnitude (which correlates to the color of the circle). The force annotation is matched with the manually scripted annotation of force on the FEA stress map. The model then extracts and resizes the images, and the generator network is built to help produce the output images and the discriminator network is built to judge the generator's performance.

3.2. Magnitude Change

The first test dataset was based on a flat plate surface with digital dimensions of ten feet by ten feet. While the analysis was modeled using Karamba3D, a specific material code (corresponding to 24h glulam timber) was used primarily for consistency with later comparative tests and does not significantly affect the results of this initial magnitude test.

This dataset was designed to assess the cGAN's sensitivity to subtle variations in input. To do this, color-coded circles were applied to indicate changes in force magnitude, serving as a visually distinct and geometrically minimal alteration for the network to interpret (Figure 4).

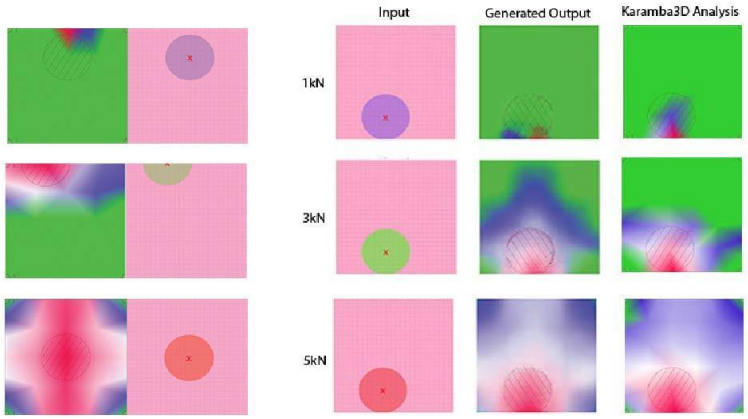

Figure 4: Left: Examples from the training dataset for various magnitude changes, Right: Results from unseen input data compared to manually validated Karamaba3D analysis

Blue indicated the lightest point load, green represented a moderate point load, and red is the highest magnitude point load. All point loads are vertical, and a distributed gravity load was not accounted for in this dataset. The structure had fixed supports at each corner that resisted rotation in all directions. This boundary condition allowed for the point loads to cause more specific and exaggerated results on the surface to see a clear distinction between various inputs and produced outputs.

4. Results

The selected geometry was a traditional shell structure shape and was assigned 24h glulam timber. The base geometry is a continuous, doubly curved surface. Fixed supports were defined at the corners. The forces included a distributed gravity load and a point load that was represented with a 100% scale deformation on a stress/strength analytical map. The results from the cGAN show a general pattern of understanding across the dataset. The cGAN demonstrates an understanding of where the point load locations are and mirrors the general mappings of the FEA (Figure 5).

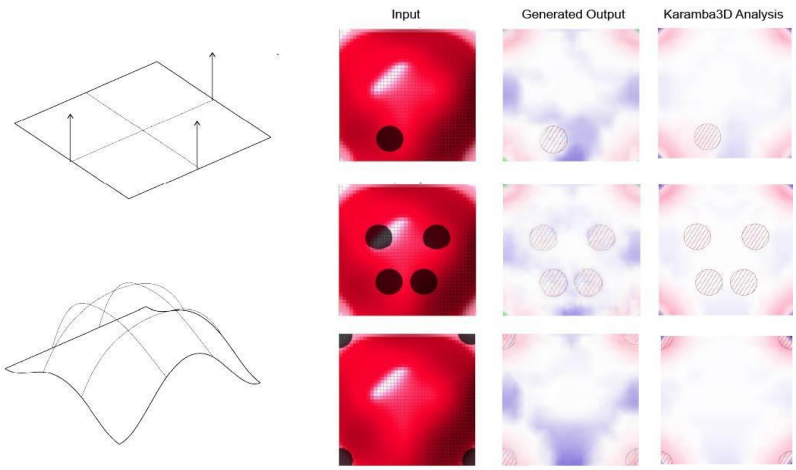

Figure 5: Left: The shell structure shape before becoming segmented, Right: Results from the cGAN based on force location, next to manually validated maps on Karamba3D

The results show less stress at the top of the center of the structure and that the stress moves diagonally across the surface towards the edge conditions. Therefore, we can reduce the amount of material from the top of the structure and utilize it more efficiently at the edges, where there are higher concentrations of stress. The shell is then turned into a discretized gridshell system that is connected at intersecting nodes.

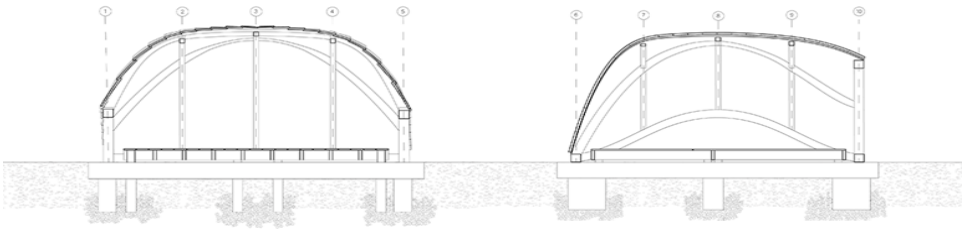

Figure 6: Top: Image depicting the reallocation of material post shell segmentation

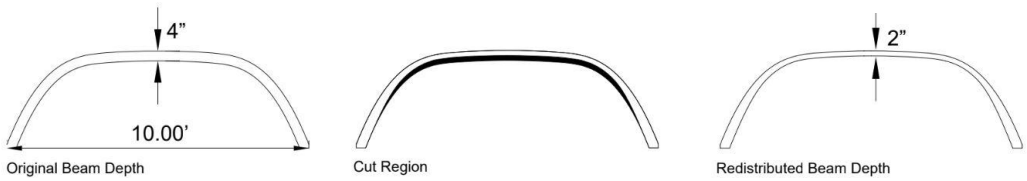

Figure 7: Top: Image depicting the reallocation of material post shell segmentation

The gridshell allows for more manipulation of material as each segment's material depth can be manipulated on its own. The widest beam depth is four inches near the edge boundaries. In low stress zones near the top of the shell, the beam depth is at its thinnest, which is 2 inches. A secondary stress/strength analysis was conducted using Karamba3D to check the efficiency of redistributing the material and breaking the shell into segments. With the beam segmentation and material redistribution, low stress zones became more widespread, and the back beam that runs along the ground was the only segment that retained high levels of stress after dividing the beams.

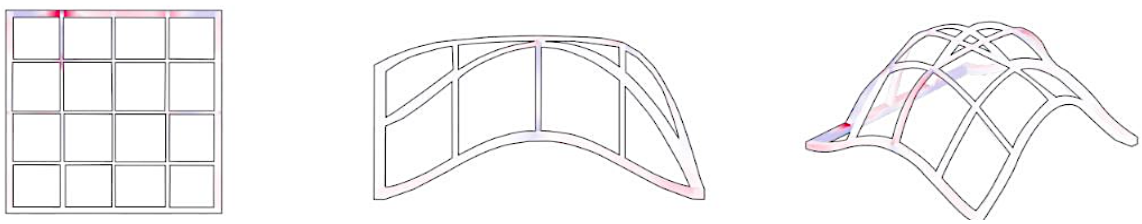

Figure 8: Stress/Strength analysis results post grid segmentation

Uniform beam depth is accommodating for a streamlined fabrication process, however, the specific usage of material depth and size could help with cost savings and minimize unnecessary weight of the structure. The cGAN infrastructure allows for the rigor produced by FEA predictions, as this is what

the training is built on, but with the added benefit of a generative machine learning framework. Due to its adversarial nature, the relationship between the generator and the discriminator is recursive, referencing "itself" (the computational architecture) to correlate relationships between form and performance under load. The generalizable patterns produced by the network allow the user to have an intuitive understanding of a structural system under varied circumstances, within a graphical environment. The training goes through a self correction process in order to better predict future output generations.

5. Discussion and Conclusion

In conclusion, this study presented a cGAN-driven method for stress-informed adaptation in shell structures, demonstrating the feasibility of applying generative models to structural engineering challenges. The cGAN learned from example data to suggest material redistributions that improved structural performance, effectively mimicking and accelerating traditional optimization outcomes. Key contributions include showing a cGAN-optimized model can integrate stress considerations into the design process and providing a workflow for training such a model on simulation data. The results presented improvements in reduced peak stresses and increased stiffness, validating the effectiveness of the approach.

Nonetheless, there are important considerations and limitations. The current cGAN was trained on a relatively limited dataset of simulated scenarios. Its performance may degrade for load cases or boundary conditions that differ significantly from those in the training set.

The methodology proposal of utilizing a cGAN in place of scripting a finite element analysis is not a strict suggestion. In our design tool repertoire, we have many programs and protocols at our disposal. With the continuous development of new technologies, our toolbox is rapidly expanding into more creative applications. The advent of a new tool will inevitably lead to the creation of a new methodology, however, as with all things, there is no end-all be-all tool or workflow. All the processes that we use will have their own unique Achilles heel, and as a result, another tool can step in to fill the gap. In the case of scripting FEA, there are two large problems: 1. The technical debt problem. Many are not familiar with or comfortable with Grasshopper and its auxiliary plugins. Because of this, a large population of designers are left to conduct their analysis elsewhere or not at all. Due to the robustness of Grasshopper, this leaves those who do use it to generally leave the scripting of the analysis until the end of their project, and the analysis remains static as it will not influence the design thereafter. 2. The terminal output problem. The script cannot reference its own system or learn through the continuous iteration of analysis, making it more difficult to create an integrated workflow and creating more blind spots for the designer in regards to their knowledge of the overall structural system. The cGAN model allows for a tighter iterative feedback loop despite its high upfront computational training cost. Further development of this workflow could include a multi-modal cGAN, which could allow the GAN to interpolate other performance constraints, including material selection within the cGAN as opposed to running a separate cGAN model for each material. Interpreting and trust-validating these AI-driven design suggestions remains an area for further research

Future work will build upon this foundation. We plan to expand the training dataset to cover a broader range of geometries and load cases, which will help the cGAN generalize its design suggestions. Incorporating additional constraints (for instance, stability or frequency criteria) directly into the learning process is another priority, ensuring that generated designs are not only strong but also satisfy all practical requirements. Finally, we aim to test the cGAN framework on physical prototypes of hybrid shells, closing the loop between digital prediction and real-world performance. By continuing to refine this approach, we anticipate that AI-assisted design tools will play an increasingly important role in creating efficient, adaptive structures for the built environment.

Acknowledgements

This research was led by Gabrielle Brooking as part of her thesis work, under the supervision of Dr Luis Borunda. Gabrielle Brooking conducted the primary research, dataset preparation, and manuscript drafting. Luis Borunda provided supervisory guidance, contributing to the methodological framing, critical feedback, and revisions of the manuscript.

References

[1] G. Lynn, *Animate Form*. New York, NY, USA: Princeton Architectural Press, 1999.

[2] M. del Campo, *Neural Architecture: Design and Artificial Intelligence*. Novato, CA, USA: Oro Editions, 2022.

[3] R. Gerhardt, S. Huerta, and A. E. Benvenuto, "The methods of graphical statics and their relation to the structural form," in *Proceedings of the 1st International Congress on Construction History*, S. Huerta, Ed. Madrid, Spain: COAM, 2003, p. 997-1006.

[4] R. Oxman, "Morphogenesis in the theory and methodology of digital tectonics," *Journal of the International Association for Shell and Spatial Structures*, vol. 51, no. 3, pp. 173–179, 2010.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[6] G. Rossi and P. Nicholas, "Encoded images: Representational protocols for integrating cGANs in iterative computational design processes," in *ACADIA 2020 Distributed Proximities: Proceedings of the 40th Annual Conference of the Association for Computer Aided Design in Architecture*, vol. 1, pp. 218–227, 2021. [Online]. Available: http://papers.cumincad.org/data/works/att/acadia20_218.pdf

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint*, 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1611.07004

[8] TensorFlow, "Pix2Pix: Image-to-image translation with a conditional GAN," *Google Colaboratory*, n.d. [Online]. Available: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/pix2pix.ipynb