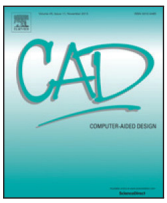
Computer-Aided Design 171 (2024) 103707

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad



Research Paper

Latent Diffusion Models for Structural Component Design

Ethan Herron, Jaydeep Rade, Anushrut Jignasu, Baskar Ganapathysubramanian, Aditya Balu, Soumik Sarkar, Adarsh Krishnamurthy *

Iowa State University, Ames, IA, United States of America

ARTICLE INFO

Keywords:
Diffusion models
Generative design
Topology optimization

ABSTRACT

Recent advances in generative modeling, namely Diffusion models, have revolutionized generative modeling, enabling high-quality image generation tailored to user needs. This paper proposes a framework for the generative design of structural components. Specifically, we employ a Latent Diffusion model to generate potential designs of a component that can satisfy a set of problem-specific loading conditions. One of the distinct advantages our approach offers over other generative approaches is the editing of existing designs. We train our model using a dataset of geometries obtained from structural topology optimization utilizing the SIMP algorithm. Consequently, our framework generates inherently near-optimal designs. Our work presents quantitative results that support the structural performance of the generated designs and the variability in potential candidate designs. Furthermore, we provide evidence of the scalability of our framework by operating over voxel domains with resolutions varying from 32^3 to 128^3. Our framework can be used as a starting point for generating novel near-optimal designs similar to topology-optimized designs.

1. Introduction

Computer-aided design (CAD) software allows engineers and designers to represent their designs digitally. Previously, engineers and designers relied on manual methods of representing and testing their designs, i.e., hand-drawn designs, physical prototypes, and manually calculating relevant computations. CAD software dramatically simplifies the engineering process by representing designs digitally and enabling designs to be easily shared among large teams, eliminating the need for elementary computations and, later, enabling simulations, reducing the number of physical prototypes required. However, the initial design phase is a specific bottleneck in the rapid design process. Engineers and designers often create designs for a given system with only the general requirements in mind. Completing designs can also be incredibly time-consuming, particularly when designers can only create a few designs simultaneously. Multiple methods have been developed to address this bottleneck; two of the most common methods are procedural modeling and topology optimization. Both methods have unique pros and cons, but they have the same goal: to improve the effectiveness of engineers and designers to speed up the design cycle.

Recent advances in generative models, a subset of machine learning algorithms, provide the foundation for new methods in generative design. These new methods may build off of procedural modeling and topology optimization, allowing users to generate a set of potential candidate designs that may vary in overall design while maintaining structural performance. Multiple previous works have utilized generative models, Generative Adversarial Networks [1–3] (GANs) in particular, for problems in design. Most of these works aim to develop surrogate models for current methods in topology optimization and do not address problem-specific exploratory design, i.e., generating a set of potential candidate designs for a given set of initial conditions. This reduced capacity to generate sets of designs may be attributed to the generative algorithm used. GANs, in particular, are difficult to optimize and susceptible to mode collapse, potentially outputting a single design for arbitrary initial conditions regardless of its inherent stochasticity.

Diffusion Models [4,5] (DMs) are a recent advance in generative modeling, which learns to denoise a data sample according to a predefined diffusion/noising process. DM's do not suffer from mode collapse at the expense of increased computational costs compared to GANs. In this work, the core of our framework is a Latent Diffusion Model introduced by Rombach et al. [6] (LDM). This architectural choice is expanded upon in Section 3, but simplifies training while also reducing the computational costs typically incurred by DMs. To train an LDM for problem-specific exploratory design, a distribution of designs parameterized in some manner is required. The optimally trained LDM can sample from this distribution of potential designs. Because our proposed work aims to accelerate the design process by increasing the speed and efficiency of design engineers, the dataset chosen must reflect this. The dataset must contain samples of the initial conditions

* Corresponding author.
E-mail address: adarsh@iastate.edu (A. Krishnamurthy).

https://doi.org/10.1016/j.cad.2024.103707
Received 18 September 2023; Received in revised form 17 February 2024; Accepted 18 March 2024
Available online 22 March 2024

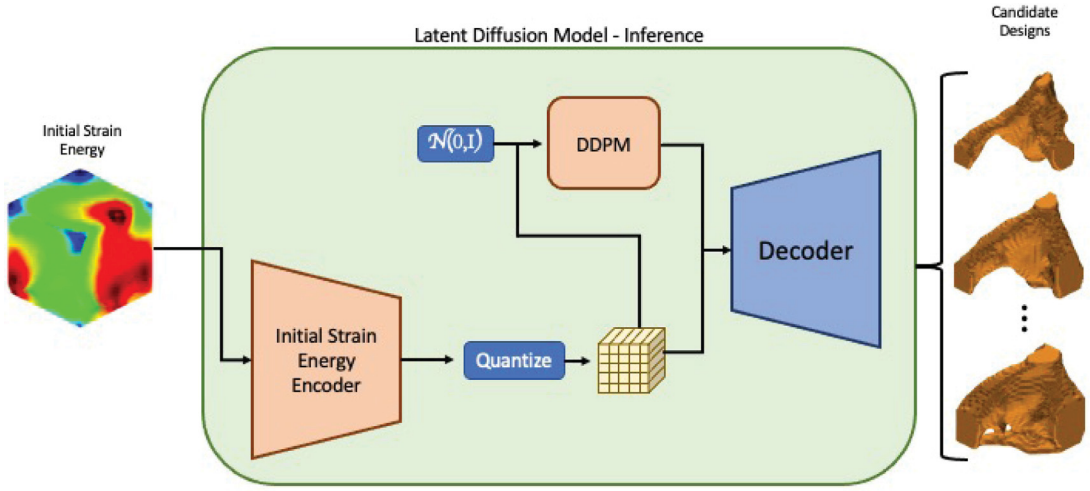


Fig. 1. An overview of the proposed LDM framework for generating new designs (inference). Initial conditions in the form of a strain energy map are encoded to latent representations. The encoded initial conditions are used to condition the latent representations generated by the Diffusion Model. The encoded initial conditions and the generated latent representation are concatenated and decoded to render a potential candidate design. Different generated latent representations by the Diffusion Model will produce different candidate designs for a single set of initial conditions.

of a given problem, as well as a corresponding design that satisfies these unique sets of initial conditions. With this in mind, we created a dataset by randomly generating initial forcing conditions on a cube and performed structural finite element analysis to obtain a strain energy map. We also performed structural topology optimization using these initial conditions to get a corresponding design. These corresponding designs were generated with the solid isotropic material with penalization (SIMP) topology optimization algorithm. The inherent stochasticity of the LDM enables it to generate a set of potential candidate designs conditioned on these initial conditions. A fortunate consequence of using a dataset generated using a topology optimization algorithm is that the designs generated by the proposed framework will be inherently near-optimal. We support this claim in Section 6 while elaborating on the standard practices of evaluating generative models.

This work addresses three major problems: reliably generating *multiple* realistic candidate designs for a set of initial conditions, editing a preexisting design, and reducing the computational costs of operating on 3D voxel grids. Our contributions reflect these major challenges and are as follows:

1. Develop a 3D generative design framework with Latent Diffusion Models, which enables the generation of structural components and is flexible enough to edit preexisting designs.
2. Train a series of Latent Diffusion Models on datasets of approximately 66k 32^3, 25k 64^3, and 10k 128^3 samples, encompassing diverse initial conditions and varying domain sizes.
3. Demonstrate the framework's ability to generate a set of diverse candidate structures for a given set of initial conditions with near-optimal structural performance.

The paper is organized as follows. In Section 2, we provide a background on generative design for structural components and generative modeling in terms of machine learning. Following this, we explain the mathematical formulations of Variational Autoencoders and Diffusion Models, the two primary components of the proposed framework in Section 3. These formulations of Variational Autoencoders and Diffusion Models are then put into context in Section 4, where the overall framework is further elaborated. Finally, we include sections on data generation and analysis of our results in Sections 5 and 6, respectively.

2. Background

One of the main pillars of the engineering design process is the conceptualization step, which includes design ideation and exploration. While most of this is currently a manual process, such as sketching, considerable effort has been dedicated to automating this process. One such method to automate this is topology optimization. Topology optimization (TO) is a methodology that poses the design generation process as an optimization. Typically, the objective function used in a TO framework is to minimize some performance metrics, such as minimizing overall compliance. Most TO algorithms are deterministic; for a set of initial conditions, they will only generate a single design. Due to this deterministic nature, a definitive difference exists between generative design and TO. This is where our proposed framework fits in, a methodology capable of generating *multiple* different candidate designs for a given set of initial conditions. We simultaneously leverage the performance of TO algorithms by training our framework on TO-generated designs, enabling our framework to generate designs that are inherently near optimal for the given objective of minimizing overall compliance.

Current methods of generative shape design for structural components are dominated by these TO algorithms which are computationally expensive [7–9], such as Solid Isotropic Material with Penalization (SIMP) method [10,11], Level Sets [12,13], and Genetic algorithms [14,15]. Typically, the objective function for which these methods optimize is the design's total strain energy or compliance accompanied by user-defined constraints. These methods are computationally expensive due to the iterative requirements and finite element (FE) evaluations required to compute the objective function. Out of these methods, genetic algorithms are the only ones to provide multiple different candidate designs for a set of initial conditions, while the other methods are deterministic. Regardless, generating new potential designs on the fly is infeasible due to the FE solution calls. Reducing the computational costs of these methods would provide substantial value to design engineers from any domain.

Generative modeling is a statistical term for modeling a conditional probability distribution, $P(X|Y)$, with the observable variable being Y and the target variable being X. This can be extremely difficult to model, particularly as the sample size defining the distribution of X grows, quickly making it intractable. In this work, *generative modeling* refers to methods that use neural networks to approximate the conditional probability distribution, $P(X|Y)$.

Three of these generative modeling methods are GANs, Variational Autoencoders [16] (VAE), and Diffusion Models (DM), more formally, Denoising Diffusion Probabilistic Models (DDPMs). From a high level, these methods fall into two categories of generative modeling: explicit

and implicit density estimation. VAEs and DMs explicitly define the conditional probability distribution, optimizing the neural networks to maximize the variational lower bound. GANs implicitly model the conditional distribution by optimizing a secondary neural network to learn the difference between samples in the target distribution and those out of the target distribution. Irrespective of the objective function formulation, each method learns a mapping from a known prior distribution $P(X)$ to the unknown, high dimensional, and complex target distribution $P(Y)$. Typically, a Normal Gaussian distribution is chosen as the prior distribution, and the target distribution is problem-specific. Each of these methods has its unique pros and cons.

GANs have shown remarkable results and are computationally economical but are prone to mode collapse and instabilities during training. Recent works have created methods to reduce these issues but have not eliminated them. VAEs and DMs do not suffer from mode collapse or instabilities during training. VAEs often fall to the mean of the dataset and generate samples dominated by lower-frequency features, resulting in blurry samples (in the case of image generation). DMs can model extremely high-frequency features at the expense of increased computational costs. This trade-off comes from the iterative refinement nature of DMs. DMs are trained to remove isotropic Gaussian noise from a noisy data sample iteratively. During inference, the DM starts from pure isotropic Gaussian noise, iteratively removing noise until it reaches an approximate sample from the target distribution. A consequence of this iterative refinement structure is the DM can denoise from any arbitrary point in the diffusion/noising process [17]. In this work, denoising a partially noised sample amounts to editing an existing design, a desirable functionality for designers.

Current state-of-the-art (SOTA) generative models for image generation unite VAEs and DMs in LDMs. LDMs are trained with a two-stage process. First, a VAE is trained on the target dataset, learning a lower-dimensional representation of the target distribution in its latent space. Second, a DM is trained to denoise latent samples from the pretrained VAE. So, the ground truth samples used for the DM's training are encoded latent representations from the pretrained VAE. This LDM formulation reduces the computational costs incurred from running the DM while improving the performance of the VAE.

Machine learning algorithms have been used extensively for problems related to structural design, typically through the lens of TO. Most works aim to develop fast alternatives to the computationally demanding TO algorithms. Such methods take as input initial conditions, various boundary conditions, and user-defined constraints such as the volume fraction to predict the final optimized topology [18–25]. In Banga et al. [26], they run the SIMP algorithm for a small number of iterations as a starting point for an ML model to predict the final optimized topology. Another approach uses supervised learning methods to predict several iterations of the TO algorithm's optimization trajectory [27], which can then be directly manufactured using 3D printing [28]. These works focus mostly on 2D datasets, with a few using 3D datasets, over resolutions ranging from 32^2 to 128^2 of standard TO problems such as the cantilever. Additionally, the ML algorithms used vary from standard CNN architectures trained in a supervised fashion to conditional GANs (CGANs) optimized to minimize the Wasserstein distance between the predicted topology and a ground truth topology or even a classifier-guided DM. Other works look to directly integrate ML models into the overarching TO framework by representing the topology with a neural network or predicting potential changes in the topology [29–33]. In Guo et al. [34], they take the inverse approach by using a VAE to compress design attributes to a reduced latent space that the TO algorithm operates in, effectively reducing computation without a loss in performance. Several other works leverage the reduction in computational costs of ML-integrated TO algorithms for generative design. To encourage the generation of multiple *different* designs, these methods either add a filtering process to remove similar designs or generate multiple designs with an ML model that are then passed on to a TO algorithm [35–38].

Our proposed work further builds on the works mentioned above by operating over a large-scale 3D design space with variable boundary conditions. The proposed work also removes the need for computationally expensive TO algorithms to generate unique families of near-optimal candidate designs.

3. Mathematical formulation

The core of our framework is a Latent Diffusion Model. LDMs are composed of two main components, an external VAE, and an internal DM, which operate in the latent space of the VAE, hence *Latent* Diffusion Model. The following section covers the mathematical preliminaries of VAEs followed by DMs.

3.1. Variational Autoencoders

Unlike traditional Autoencoders, which map an input $\mathbf{x}$ to a latent representation $\mathbf{z}$, VAEs map an input $\mathbf{x}$ to a probability distribution. The VAE's encoder defines this probability distribution by predicting its mean and variance. The latent variable $\mathbf{z}$ is extracted from this distribution by randomly sampling a standard normal Gaussian, multiplying it with the predicted variance, and shifting that product by the predicted mean. More formally:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and $\odot$ is element-wise multiplication. The VAE's decoder then predicts $\hat{\mathbf{x}}$, a reconstruction of the original sample $\mathbf{x}$, given the random sample $\mathbf{z}$ from the distribution defined by the encoder. In summary, the VAE is composed of an encoder $p_\theta(\mathbf{z}|\mathbf{x})$, the random latent sample $\mathbf{z}$ which comes from a prior distribution $p_\theta(\mathbf{z})$, and a decoder $p_\theta(\mathbf{x}|\mathbf{z})$. The encoder and decoder are both neural networks parameterized by θ. By formulating the VAE such that the latent variable $\mathbf{z}$ is sampled from some prior distribution, during inference, the decoder can then randomly sample from the target distribution by decoding a latent variable $\mathbf{z}$, which is drawn from a standard normal distribution. This ability qualifies the VAE as a generative modeling algorithm.

During training, the VAE is optimized with two different loss terms; the first is a simple reconstruction loss, similar to a traditional autoencoder, and the second term is for the latent variable $\mathbf{z}$. This second term is used to regularize the latent variable sampled from the distribution defined by the encoder towards the prior distribution, a standard normal Gaussian. This is accomplished by minimizing the KL divergence between the prior distribution, $p_\theta(\mathbf{z})$, and the distribution defined by the encoder, $p_\theta(\mathbf{z}|\mathbf{x})$. Thus, our VAE loss function is:

$$\mathcal{L}(\theta; \mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x})}\Big[\|\mathbf{x} - p_\theta(\mathbf{x}|\mathbf{z})\|\Big] + KL\Big(p_\theta(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\Big) \tag{2}$$

3.2. Vector-Quantized Variational Autoencoders

A Vector Quantized VAE (VQ-VAE) [39] is a VAE that uses a discrete codebook of embeddings to quantize the continuous latent space learned by a standard VAE. Opposed to a normal Gaussian distribution defining the VAE's latent prior distribution, $p(\mathbf{z})$, the discrete codebook of embeddings defines the VQ-VAE's prior distribution. The discrete codebook contains K embeddings, each of dimension $\mathbb{R}^D$, that act as the nearest discrete approximation to the continuous latent space. During the encoding step, the input $\mathbf{x}$ is first mapped to the continuous latent space by the encoder function $q(\mathbf{z}|\mathbf{x})$. Then, the continuous latent variable is quantized by finding the nearest embedding, by Euclidean distance, in the codebook. The decoder function $p(\mathbf{x}|\mathbf{z}_q)$ decodes the embeddings, $\mathbf{z}_q$, to reconstruct the input $\mathbf{x}$. The decoder will only ever see embeddings from the discrete codebook, but the combination of discrete codes will vary across different samples. Overall, the VQ-VAE can be seen as a trade-off between the expressive power of a continuous latent space and the efficiency of a discrete codebook.

The VQ-VAE is trained with the same loss function as a VAE, except instead of minimizing the KL-Divergence between the predicted latent variable $\mathbf{z}$, the VQ-VAE is optimized to minimize the Euclidean distance between the predicted latent codes and its nearest, in Euclidean distance, discrete code.

3.3. Denoising Diffusion Probabilistic Models (DDPM)

Diffusion models comprise two distinct parts: the forward and backward diffusion processes. The forward diffusion process is simply the iterative addition of Gaussian noise to a sample from some target distribution. This forward process may be carried out an arbitrary number of times, creating a set of progressively noisier and noisier samples of the original data sample. The following Markov chain defines this process:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{3}$$

where $\mathbf{x}_0$ is the sample from some target distribution $q(\mathbf{x})$, and a variance schedule defined as $\{\beta_t \in (0,1)\}_{t=1}^{T}$. The reverse diffusion process would then iteratively remove the noise added during the forward diffusion process, i.e., $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

In practice, it is not feasible to sample $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ since we do not know the entire distribution. Therefore, we approximate the conditional probabilities via a neural network, $G_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, where the parameters of G, and θ, are updated via gradient-based optimization. Training a diffusion model then consists of optimizing a neural network, $G_\theta(\mathbf{x_t}, t)$, to reconstruct the random normal noise used in the forward diffusion process to transform the original sample into a noisy sample ($\mathbf{x_t}$) at an arbitrary timestep in the diffusion process. This objective function is defined as:

$$\|z - G_\theta(\mathbf{x}_t, t)\|^2 = \|z - G_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}z, t)\|^2 \tag{4}$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \Pi_{s=1}^{t}\alpha_s$ and $z \sim \mathcal{N}(0, I)$.

In summary, the neural network is given a noisy data sample at some arbitrary timestep in the diffusion process and predicts the randomly sampled normal Gaussian noise used for the forward diffusion process. Since the KL divergence between two standard normal Gaussian distributions is the $\mathcal{L}_2$ loss, we can optimize our neural network using a simple Mean Squared Error reconstruction loss.

4. Methodology

In this work, we develop a framework that maps the initial conditions to a diverse range of potential candidate designs, contingent upon the specified constraints. This problem is particularly well-suited for a generative modeling algorithm, given that multiple near-optimal or satisfactory solutions may exist. We choose to employ a Latent Diffusion Model owing to its advantages in training stability, distribution coverage (emphasizing mode coverage), and computational efficiency. The decision to use a Latent Diffusion Model was further reinforced at the beginning of this work when implementations of a StyleGAN architecture [40] failed to converge. As previously discussed, the Latent Diffusion Model comprises two distinct components: an external Variational Autoencoder (VAE) and an internal Diffusion Model (DM). In the following subsections, we provide a detailed account of the design choices made for each component and an overview of the framework's capabilities, thereby demonstrating its suitability for application in structural component design.

The final implementation of our framework may be seen in Fig. 1. At inference, the input to the framework is the problem-specific initial conditions in the form of a 3D voxel grid. The output is a binary 3D voxel grid, where 1s denote material presence, and 0s denote no material presence. The multiple candidate designs emphasize the ability of the framework to generate different structures for the same initial conditions given different initial random tensors. Fig. 2 outlines the training process of the VAE where the input is a tuple of the initial conditions and SIMP-Optimized design, both of which are 3D voxel grids of the same size. In this figure, the VAE is trained to reconstruct the SIMP-Optimized design used as input to the *Optimized Design Encoder*. Lastly, a visual of the LDM training loop may be found in Fig. 3 where the Multi-Headed VAE's Decoder is absent, and the Diffusion Model is optimized to invert the noising process used on the *Optimized Design Encoder's* latent representation.

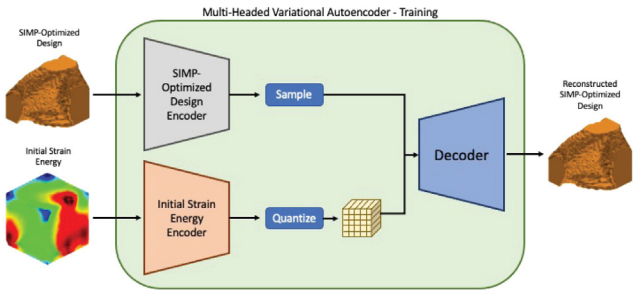


Fig. 2. An overview of training the external Multi-Headed VAE.

The dataset employed for training and validation comprises pairs of initial conditions and corresponding ground truth: designs optimized using the Solid Isotropic Material with Penalization (SIMP) method. The initial conditions are represented as a single voxel grid, where each voxel contains continuous normalized strain energy values. A comprehensive explanation of the dataset generation process can be found in Section 5.

4.1. Multi-headed Variational Autoencoder

The external VAE of our framework is composed of two different encoder networks and a single decoder network. We further refer to the external VAE as a multi-headed VAE. One encoder (head) compresses the Initial Strain Energy into its respective latent representation, while the other compresses the SIMP-Optimized design into its respective latent representation. Each latent representation is a tensor of size $[4, 8, 8, 8]$. The decoder upsamples the concatenated latent representations (total size of $[8, 8, 8, 8]$) back to the voxel space. When given the Initial Strain Energy and SIMP-Optimized design, the entire multi-headed VAE is optimized to reconstruct the SIMP-Optimized design, essentially a traditional VAE conditioned on the Initial Strain Energy. This training process is shown in Fig. 2. This Multi-Headed VAE framework was developed to efficiently integrate the initial conditions, Initial Strain Energy, into the latent representation the decoder was trained to reconstruct. Traditionally, the external VAE of a LDM is not explicitly trained with conditional information. One reason for this is the VAE mainly serves as a nonlinear dimensionality reduction technique to ease the compute of the generative model operating in the reduced latent space. The conditional information is then only utilized by the generative model in the latent space. Since our framework is aimed at a very specific application of generative design for structural components we decided to implement this Multi-Headed VAE framework to directly integrate conditional information into the external VAE's latent representation. From preliminary experiments this framework was the simplest implementation we found. This conditional information is also integrated into the DM and is expanded upon in Section 4.2.

In our multi-headed VAE formulation, we implement two different regularization loss terms, one for each encoder's respective latent representation. The Initial Strain Energy Encoder utilizes a Vector-Quantized bottleneck (VQ-VAE). In contrast, the Optimized Design Encoder uses the original KL-Divergence formulation for its latent regularization term. Weighting terms of 10^-5 and 0.25 were used for the Optimized Design KL-Divergence regularization term and VQ regularization term,

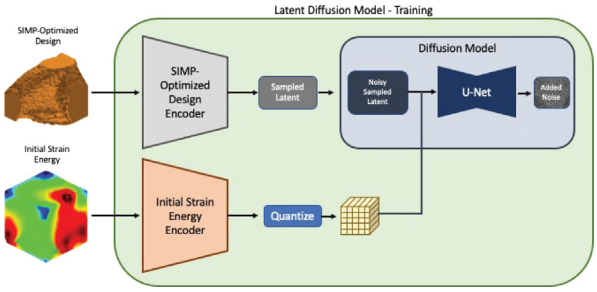


Fig. 3. An overview of training the Latent Diffusion Model.

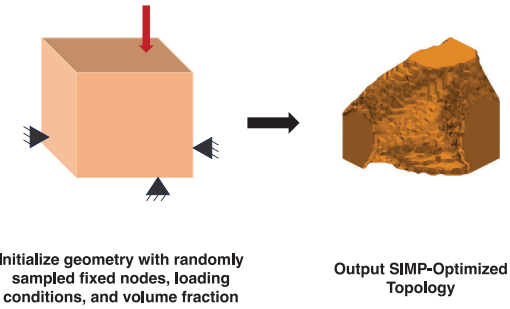


Fig. 4. Data generation process.

respectively. These weighting terms are so small because we only want to bias the latent representations towards a normal Gaussian or codebook entry while maintaining a premium on the overall reconstruction loss. Experimentally, we found that implementing small biases towards the given regularization terms assisted the DM's optimization. Using such small weighting terms is common practice in Latent Diffusion Models, specifically seen in the original LDM paper [6].

Preliminary experimental results found the VQ-KL formulation of the multi-headed VAE, VQ bottleneck for initial conditions and KL bottleneck for Optimized Designs, produced better results in terms of connected components and binarized designs, e.g., VQ bottlenecks for each encoder was prone to generating designs with floating artifacts and KL bottlenecks for each encoder was prone to generating designs with non-binary outputs.

4.2. Latent diffusion model

During inference, our framework may only have access to the Initial Strain Energy when generating a new candidate design. A generative model is then required to generate a latent representation that appears to come from the posterior distribution defined by the Optimized-Density Encoder's latent representation. This requirement motivates using a DM in the latent space of the multi-headed VAE. The generated latent representation is concatenated with the latent representation from the Initial Strain Energy Encoder and decoded back to the voxel space, rendering a potential candidate design.

This DM is trained and sampled like a traditional DDPM, learning to reconstruct the pure white noise used to compute the input noisy sample. Since this is a DM operating in the latent space of the VAE, the ground-truth samples are the compressed latent representations by the **pretrained** multi-headed VAE. The DM is trained to reconstruct the pure white noise used to compute the noisy latent representations sampled from the Optimized Design Encoder's posterior distribution. Additionally, the DM is conditioned on the latent representation from the Initial Strain Energy encoder. We condition this DM by concatenating the noisy Optimized Design latent with the unperturbed Initial Strain Energy latent as input to the DM. The DM is then learning a mapping of, $\mathbb{R}^{[8,8,8,8]} \rightarrow \mathbb{R}^{[4,8,8,8]}$, where the input is a clean conditioning sample concatenated with the noisy Optimized Design sample to predict the pure noise used to get the noisy Optimized Design sample. The noising schedule used to train the DM was a linear beta schedule with 1,000 time steps and start and end variances of $\beta_1 = 1.5 \cdot 10^{-3}$ and $\beta_T = 1.55 \cdot 10^{-2}$, respectively. Recently, conditional DMs have used classifier-free guidance [41]. When training a DM with classifier-free guidance the conditional signal will undergo a form of dropout to remove portions of the signal. Then during inference to generate a single sample the DM will actually execute two different forward passes, one with full conditioning and one with no conditioning. The subsequent denoising step taken will use a noise sample that is derived by linearly interpolating between the two predicted noises. During inference the user is then able to define how much guidance they want the DM to use during inference. Due to the specific application of our proposed framework, we do not implement any traditional form of guidance or conditioning, i.e., classifier or classifier-free. We found that always providing the DM with the full conditional signal of the initial conditions satisfied our goal of generating multiple candidate designs. Additionally, this allows us to execute a single forward pass for each denoising step, instead of two forward passes for each denoising step used in classifier-free guidance schemes. Fig. 3 shows the training process of the LDM.

4.3. Design generation and translation

Our framework offers two primary capabilities: design generation and design translation. Design generation is the direct inference process described earlier, wherein an initial strain energy is input, an optimized design latent is generated, and a candidate design is subsequently decoded from this generated latent representation.

Design translation serves as a secondary capability of our framework, necessitating no additional training, and addresses the question: Given an initial design, can we generate several inherently similar yet distinct designs? In this process, the initial strain energy and design are mapped to their respective latent representations using their corresponding encoders. The Diffusion Model is subsequently employed to iteratively denoise a *partially* noised latent representation of the initial design. It is important to note that the underlying structure is preserved by only partially noising the latent representation. The variability in the edited latent representation is approximately proportional to the degree of initial noise introduced; a noisier latent representation results in a more distinct translated design. For a visual example of this process, please refer to Fig. 14.

5. Data generation

We use the ANSYS Mechanical APDL v19.2 software to generate the dataset, which uses the SIMP method for performing structural topology optimization. We use a mesh of a cube with a length of 1 meter as an initial geometry. The mesh of the cube consists of 31093 nodes and 154,677 elements with 8 degrees of freedom. To guarantee that the dataset contains diverse shapes originating from the cube, we use a set of different loading and boundary conditions available in ANSYS, like Nodal Force, Surface Force, Remote Force, Pressure, Moment, and Displacement. To create a single topology optimization geometry, we use a random selection process to choose three nodes that are not collinear and designate them as fixed points by assigning them zero displacements. This process is shown in Fig. 4. The type of load to be applied is also randomly selected from the available loads within the ANSYS software, and the location on the mesh where the load is to be applied is chosen randomly as well. The specific value of the load is then determined by sampling from the predefined range assigned to that particular load type. Additionally, to generate topologies of varying shapes, we utilize a range of target volume fractions with a

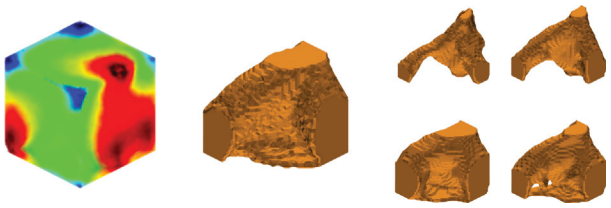


Fig. 5. Multiple candidate designs for a given Initial Strain Energy Map. From left to right, Initial Strain Energy, SIMP-Optimized Design, and four different generated designs by the proposed framework.

minimum value of 10% and a maximum of 50%, at increments of 5%. A rejection sampling strategy to ensure that sampled topologies are unique was employed. In our dataset, the topology optimization took an average of 13 iterations with the fewest being 6 and the most being 72. The duration of the topology optimization trajectory depends on several factors such as the mesh resolution, boundary conditions, and the target volume fraction. In total we generated 66,000 (66K) samples by utilizing the randomization above scheme.

We save the initial strain energy and optimal topology in a mesh format throughout the data generation phase. However, since our framework operates over voxel grids and not meshes, it is necessary to transform the mesh data into a voxel representation. We adopt the voxelization process outlined in Rade et al. [27] to convert all samples to voxel-based representations. We choose three different resolutions for voxelization: 32^3, 64^3, and 128^3. The dataset is divided into training and testing sets, consisting of 80% and 20% of all samples, respectively. The voxelization process, which takes around 15 min to complete for a single geometry, was sped up significantly by using GNU parallel to perform the voxelization of all samples on the cluster.

6. Results

In the following section, we present several different modes of analysis. These analyses are conducted on a small subset of the test set, consisting of 100 sample pairs of Initial Strain Energy and SIMP-Optimized designs. For each of the 100 different samples, we generated 20 unique designs corresponding to their respective Initial Strain Energy. This evaluation set includes 2,000 LDM-Generated designs and 100 SIMP-Optimized designs.

Additionally, we would like to highlight common practice metrics for generative modeling algorithms. Due to the nature of generative modeling (approximating intractable high-dimensional probability distributions), there is not a clean or easily computable performance metric. It is then common practice to evaluate the performance of generative models with the *Frechet Inception Distance* [42] (FID). The FID score is obtained by comparing the mean and standard deviation of two datasets, generated and ground truth images. Each sample in each dataset is the output of the deepest layer in a pretrained Inception-v3 [43] model. Inception-v3 model is pretrained as an ImageNet classifier, meaning any design, ground truth, or generated model would be Out of Distribution (OOD), as well as 3D, not a 2D image, rendering the FID score meaningless for evaluating our work.

To quantitatively evaluate our model we train an evaluation network that predicts the total amount of strain energy for a given design, specific details are discussed in Section 6.2. The evaluation network serves as an application-specific 'Inception-v3' model, providing a scalar metric to compare the generated designs with ground truth designs.

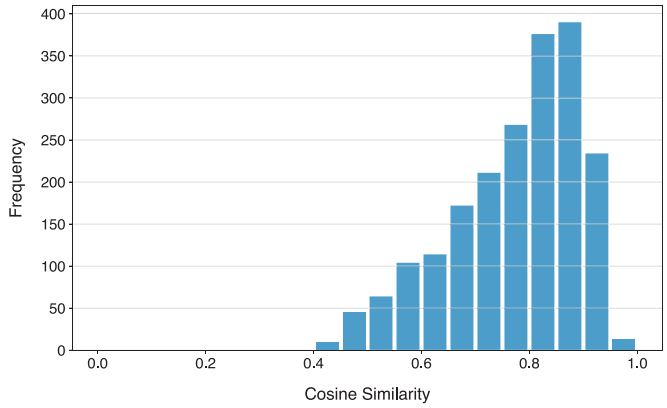


Fig. 6. The distribution of cosine similarity between the SIMP-Optimized design and LDM-Generated designs for a given Initial Strain Energy.

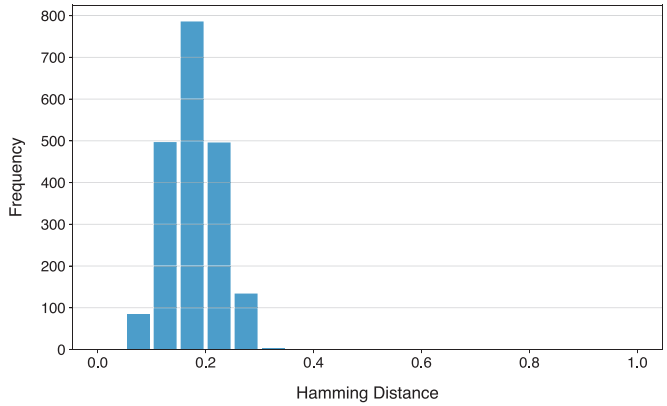


Fig. 7. The distribution of the Hamming distance between the SIMP-Optimized design and LDM-Generated designs for a given Initial Strain Energy.

6.1. Design generation

As a reminder, the design generation task is the fundamental capability of our framework, which maps a given Initial Strain Energy to potential candidate designs. An example set of generated designs with the corresponding Initial Strain Energy and SIMP-Optimized design is shown in Fig. 5. Due to the stochasticity of the DM, every different initial random sample the DM starts with will render a unique design. We evaluate the performance of our framework by comparing the distributions of generated designs and SIMP-Optimized designs across a few different metrics. As the main contribution of this paper is generative design, we define the success of our framework by how closely the distribution of a given metric aligns with the SIMP-Optimized designs, while simultaneously how different individual samples are. The two different performance metrics we want the proposed framework to capture are the distribution of predicted strain energy from the evaluation network and the distribution of volume fraction. Each of these performance metrics are elaborated on further in subsequent sections. To evaluate the variability in generated designs compared to SIMP-Optimized designs, we compute the Cosine Similarity between the 20 LDM-Generated designs and their corresponding SIMP-Optimized design for all 100 test samples. This histogram is shown in Fig. 6, where we can conclude the LDM consistently generates unique designs and does not simply reconstruct the corresponding SIMP-Optimized design. To further support this claim we compute the Hamming distance between the 20 LDM-Generated designs and their corresponding SIMP-Optimized design for all 100 test samples, see Fig. 7. The Hamming

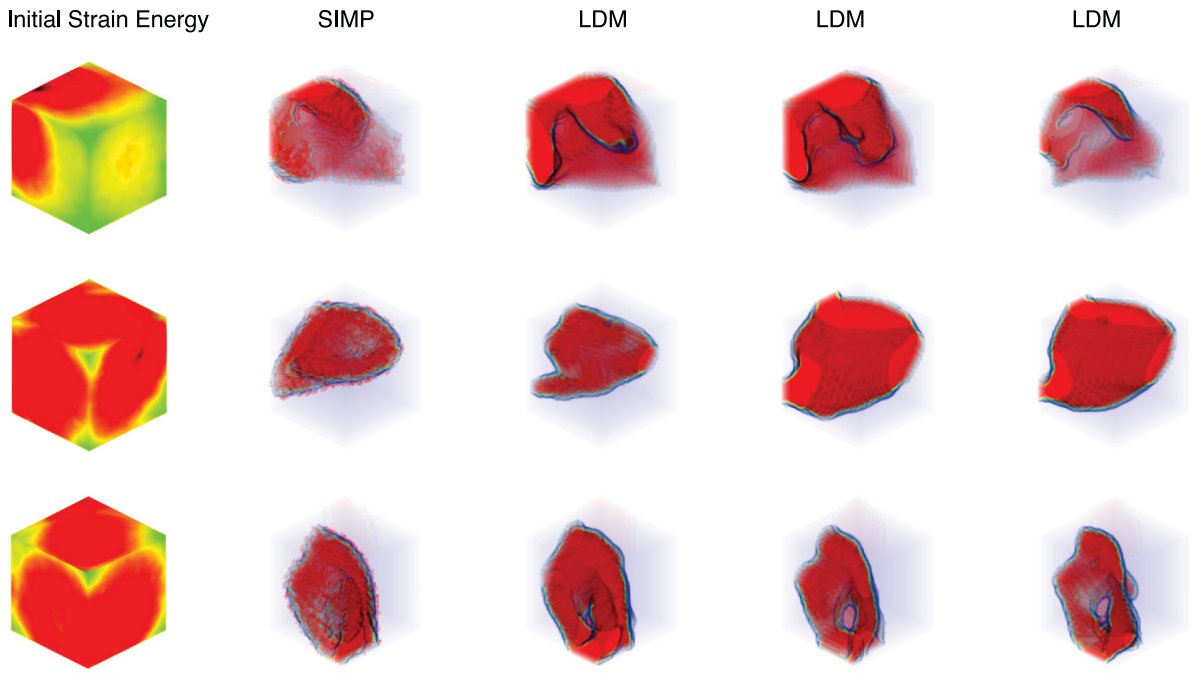


Fig. 8. Each row are designs generated by the column label method with respect to the Initial Strain Energy, the leftmost column. The second column are SIMP-Optimized designs for the respective Initial Strain Energy. The following three columns are designs generated by the proposed framework all with respect to the left-most Initial Strain Energy. The variability in the three designs is because of the random initial starting points and other noise used during the reverse diffusion process.

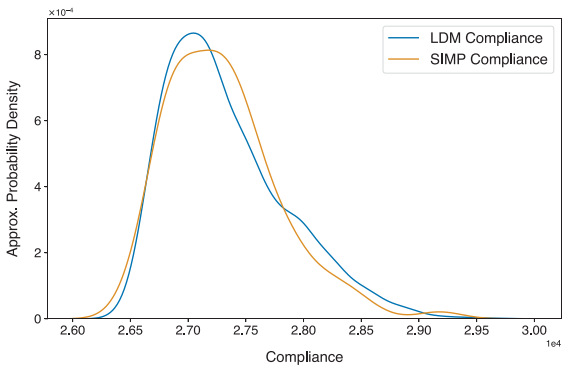


Fig. 9. The distributions of predicted strain energy for SIMP-Optimized designs and LDM-Generated designs.

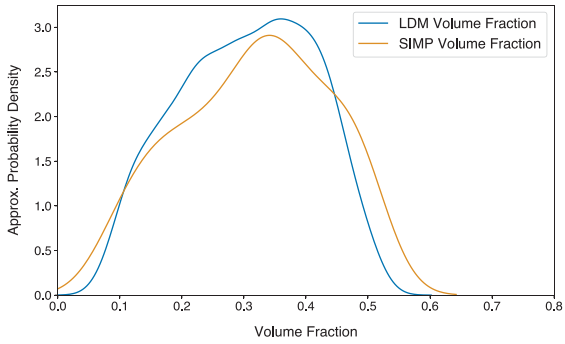


Fig. 10. The distributions of volume fractions for SIMP-Optimized designs and LDM-Generated designs.

distance measures the amount of voxels that differ between the LDM-Generated design and corresponding SIMP-Optimized design. Example generated designs are shown in Fig. 8 and in the Supplement.

6.2. Performance analysis: Strain energy and volume fraction

The added benefit of training the VAE on SIMP-Optimized designs is that the generated designs by our framework are inherently near-optimal. To support this claim, we trained a surrogate neural network, the aforementioned evaluation network, to predict the strain energy of a design given the Initial Strain Energy. To train a surrogate network, we need the dataset of strain energies and topologies at intermediate iterations of the SIMP algorithm. We save these values for a subset of a dataset (around 13.5 K samples) during data generation with an average of 13 iterations per Initial Strain Energy. The network is trained to predict the overall compliance of a design given the design and initial conditions, Initial Strain Energy. This approach of predicting the compliance of a given design is used in Lee et al. [44] where they directly integrate the compliance prediction network into the SIMP optimization algorithm achieving speed-ups in computation with no drop in performance. This compliance prediction network is also used in Rade et al. [27], where they replace the SIMP optimization algorithm with a neural network. This compliance prediction network achieves an R^2 of 0.9521 on the training dataset and 0.9409 on the validation dataset (please see Supplementary material for additional examples).

In Fig. 9, we plot the distribution of predicted strain energy for all 2,000 generated designs against the corresponding 100 SIMP-Optimized designs from the evaluation set. From this figure, we can conclude that, in the aggregate, there is no substantial difference in performance between designs generated by our framework and the SIMP-Optimized designs. Please refer to Fig. 12 for per-sample plots of strain energy distributions. The per-sample plots for the remaining sets of generated designs for the 100 evaluation samples are shown in the Supplementary material. We also highlight a single instance of the predicted compliance for several generated candidate designs and a SIMP-Optimized design for a given Initial Strain Energy in Fig. 11.

Additionally, we evaluate the volume fraction, which is a constraint that specifies the volume of material to be removed from the initial

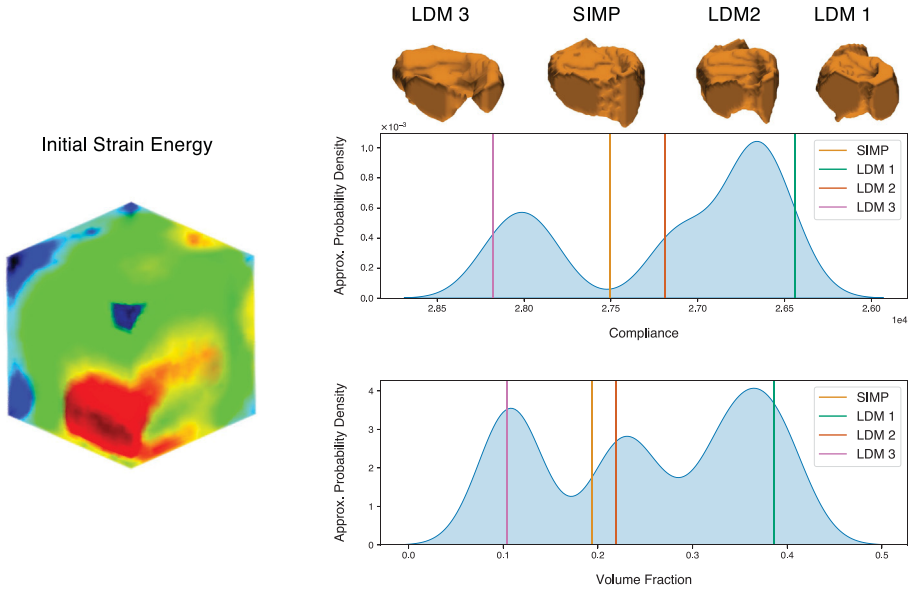


Fig. 11. Single Instance KDE plot comparing potential designs and the SIMP optimized design relative to the distribution of predicted compliance values (top) and volume fractions (bottom). The Initial Strain Energy for this sample is shown on the left. The designs, generated and SIMP-Optimized, are placed according to their predicted compliance values.

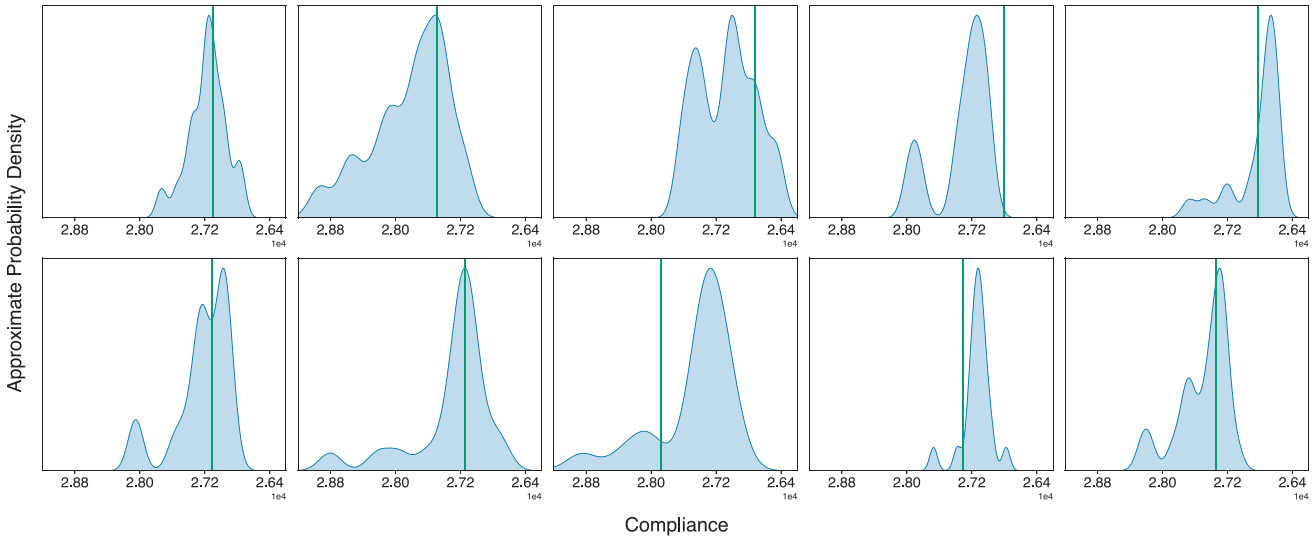


Fig. 12. Each subfigure is the predicted total strain energy distribution for 20 generated designs for a given Initial Strain Energy. The green bar is the predicted total strain energy of the corresponding SIMP-Optimized design.

design. Here, we define the volume fraction as the ratio of occupied voxels to unoccupied voxels. The mean absolute error between the average volume fraction across the 20 generated designs and the corresponding SIMP-optimized design was 0.0989. Fig. 10 compares the distribution of volume fraction across SIMP-Optimized designs and generated candidate designs from the proposed framework.

6.3. Design translation

Our evaluation of the Design Translation task is entirely qualitative, but given that the initial translated designs are SIMP-Optimized, the quantitative results shown for design generation should generalize. Fig. 13 demonstrates how varying the level of noise added to the original design latent can impact the amount of variability in the new design. This figure shows that as more noise is added, the DM is given a noisy latent with lower signal-to-noise ratio, the more the design can vary from the original design. Fig. 14, shows one example of design translation where the translated design is similar but has its own unique structure.

6.4. Ablation studies: Multi-Headed VAE Generation and Frozen Initial Strain Energy Encoder

In Fig. 15, we decode several latent representations during the reverse diffusion process undertaken by the DM during inference. If the DM were operating in voxel space, this graphic would show the mapping from pure white noise to a coherent design. Since the DM is operating in the latent space of the multi-headed VAE, we need to decode the latent representations along the reverse diffusion trajectory to visualize this process. In doing so, we can see snapshots from the iterative refinement trajectory taken by the DM. Fig. 15 also highlights the improvement a LDM provides over a traditional VAE. In a traditional VAE, a randomly sampled Gaussian is decoded; this decoded design is the far left figure, which contains a vague, incoherent

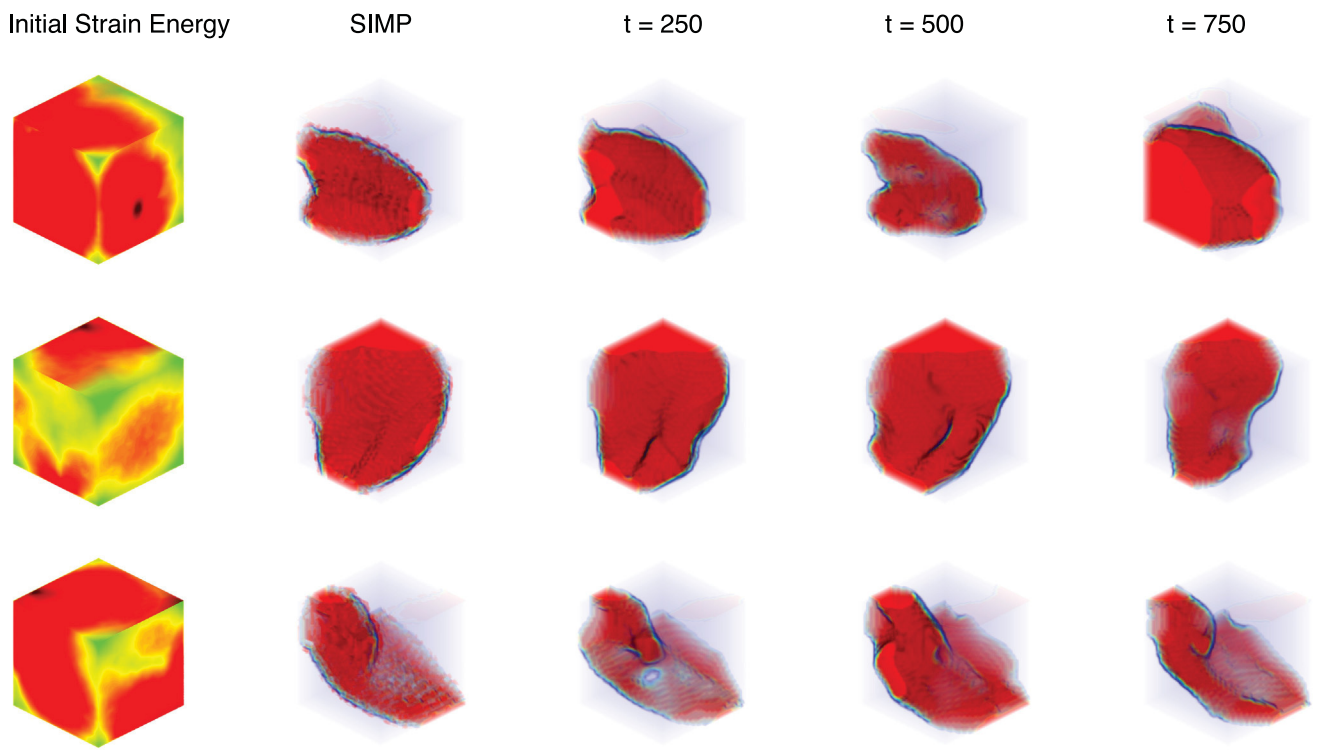


Fig. 13. Each row are different generated candidate structures for the given Initial Strain Energy displayed on the left. The second column are the SIMP-Optimized designs and the subsequent columns denoted by some timestep, t, are designs edited from that timestep in the reverse diffusion process. That is, the Initial Strain Energy and SIMP-Optimized design are input to their respective encoders and the SIMP-Optimized latent representations are noised to their respective timestep according to the forward diffusion process. The DM is then used to denoise the now noisy SIMP-Optimized design latents for the remainder of the reverse diffusion trajectory to render to a new edited design.

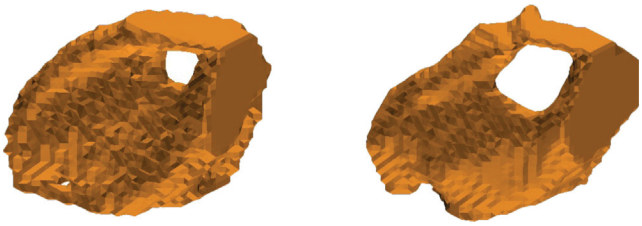


Fig. 14. The framework is also flexible enough to edit pre-existing designs. The trained diffusion model will denoise a partially noised pre-existing design latent from the pre-trained SIMP-Optimized Density Encoder. The SIMP-Optimized density is on the left and the LDM-Translated density is on the right.

Table 1
Mean Absolute Errors over the training and validation datasets of the Multi-Headed VAE and Multi-Headed VAE with a Frozen Initial Strain Energy Encoder.

	Ours	Ablation
Training	0.0171	0.0162
Validation	0.0173	0.0163

structure. Further results for strictly VAE-generated designs are shown in the fourth column of Fig. 16.

We further explore the impact of each encoder head in our proposed Multi-Headed VAE framework. To assess the impact of the Initial Strain Energy encoder on the overall performance on the framework we ran an ablation study in which we **do not** train the Initial Strain Energy encoder at all. That is, for the duration of training the external VAE and internal DM the weights of the Initial Strain Energy encoder are frozen, remaining in their first initialized state. The rest of the training process remains the same. The SIMP-Optimized design encoder is fully trained, as is the single decoder. The DM is then fully trained with the trained SIMP-Optimized encoder and the untrained Initial Strain Energy encoder. The goal of this ablation study is to empirically justify the use of the Multi-Headed VAE.

In Fig. 16 we present a comparison of generated designs between the proposed method, the Multi-Headed Encoder with a randomly sampled Gaussian as the Optimized design latent representation, and a design using the proposed method with an untrained Initial Strain Energy Encoder and a DM-generated Optimized design latent representation. These results show that the proposed work outperforms the naive VAE generation and the untrained Initial Strain Energy encoder variant. The VAE generated designs completely fail to generate a coherent design. The untrained Initial Strain Energy Encoder generates a coherent design, one connected component, but fails to generate a design for the given Initial Strain Energy. While training this Frozen Initial Strain Energy Encoder variant, the Multi-Headed VAE actually attains a lower L1 reconstruction loss Table 1 compared to the fully trained Multi-Headed VAE. This further supports the results showing the Ablation study is able to generate some design that lies in the SIMP-Optimized design data manifold, but fails to generate a design that corresponds to the initial conditions. These results further support our choice of a Multi-Headed VAE, with a DM operating in the latent space. Without the use of a DM the framework generates incoherent designs, and without the use of the Initial Strain Energy Encoder, generated designs are not feasible for the given initial conditions.

6.5. Scaling and compute

Thus far, all analyses described are for generated designs in 32^3 voxel grids (Fig. 17(a)). In Figs. 17(b) and 17(c), we demonstrate the scaling capabilities of our framework for LDM-Generated designs in 64^3 and 128^3 voxel grids. Due to the structure of LDM's, a majority of the learning burden is placed on the external VAE. This means that as long as the VAE achieves high performance on the general reconstruction

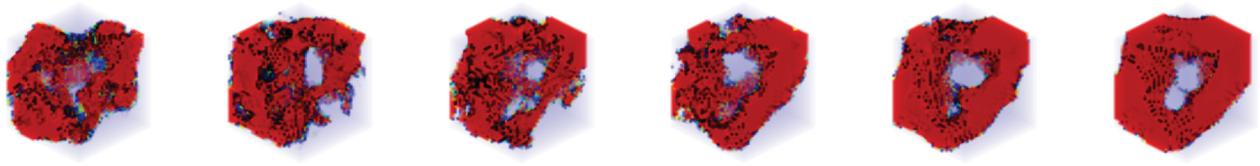


Fig. 15. Decoded posteriors from the DDPM denoising trajectory during inference. Decoding the initial random Gaussian sample (far left design) highlights the improvement our framework provides over a traditional VAE.

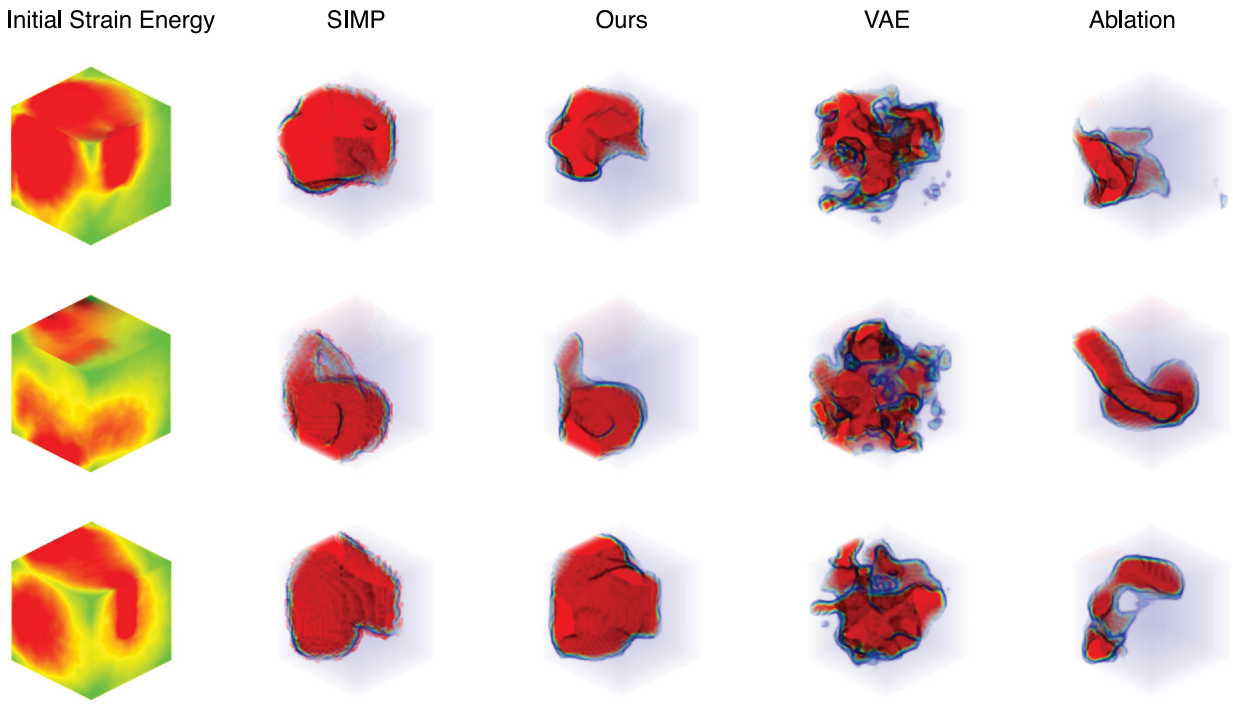


Fig. 16. Each row are designs generated by the column label method with respect to the Initial Strain Energy, the leftmost column. The proposed method is labeled 'Ours', the VAE method is the proposed Multi-Headed VAE, but instead of generating the Optimized-Design latent with a DM a randomly sampled Gaussian. The 'Ablation' column are designs generated with the Frozen Initial Strain Energy Encoder in the proposed framework. The Frozen Initial Strain Energy Encoder is further elaborated on in 6.4.

task, we can confidently train a DM in its latent space. With this in mind, we trained the VAEs for each resolution to have the same size of the latent representation, $[4, 8, 8, 8]$ for each encoder. In doing so, we can use the same model architecture for the internal DM across different resolutions, keeping computational costs in check as we scale.

The VAE training times differ dramatically across resolutions, with 32^3 taking approximately 2 min and 20 s per epoch and 64^3 taking approximately 24 min per epoch. The 128^3 resolution was too large to train from scratch on its own dataset, so we implemented a multi-grid training approach adapted from Rade et al. [45]. In the multi-grid training approach, we initialize the model architecture for the 128^3 domain such that its latent representation will be $[4, 8, 8, 8]$ for each encoder. Next, we pretrain this model on the 32^3 dataset for ten epochs and continue pretraining for one epoch on the 64^3 data. Finally, we train for an additional single epoch on the target 128^3 dataset. Once completed, the model achieves an L_1 reconstruction loss of 0.0255 on the validation set in less than two hours training time. Comparatively, if trained exclusively on the 128^3 dataset, training takes over 10 h to achieve a similar level of performance. Loss values over the training and validation datasets over each resolution are shown in Table 2.

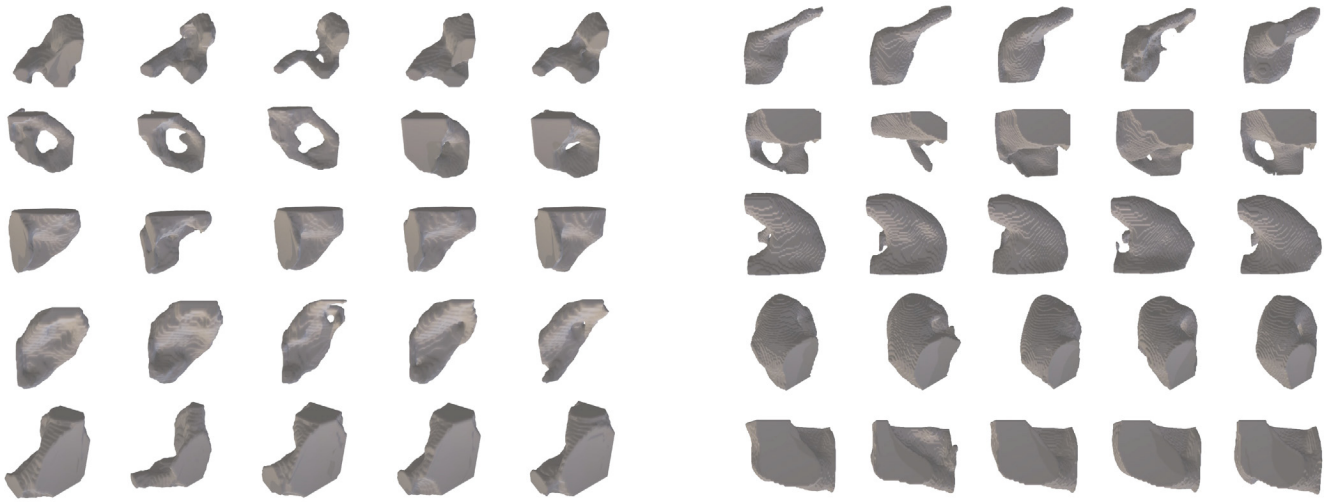
Even though we structured each VAE architecture to have the same size of latent representation across different resolutions, the DM training times differ substantially. During training, it takes roughly 6, 22, and 95 ms per sample for 32^3, 64^3, and 128^3 resolutions, respectively. The difference in training time can be attributed to the resolution of each sample and the reduction in batch size to account for the increased memory requirements for increasing resolution.

Table 2
Mean Absolute Errors over the training and validation datasets at the respective resolutions.

	32^3	64^3	128^3
Training	0.0171	0.022	0.0249
Validation	0.0173	0.0228	0.0255

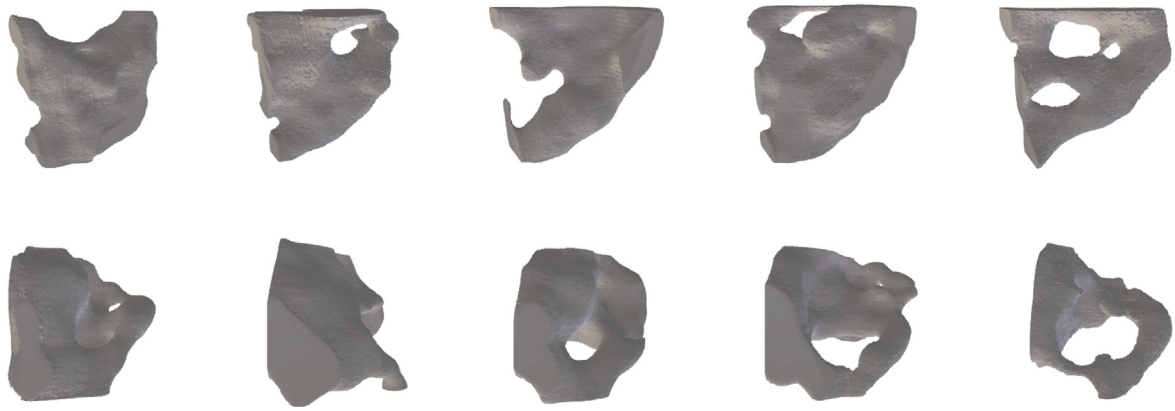
7. Conclusions

We developed a Latent Diffusion Model framework for the generative design of structural components. It comprises an external multi-headed Variational Autoencoder and an internal Diffusion Model. An added benefit of this formulation is the ability to edit an arbitrary initial design according to its accompanying initial conditions. Additionally, given the dataset used to train this model was generated using the SIMP topology optimization algorithm, the generated designs are inherently near-optimal. Future directions for subsequent work may involve adding more conditioning mechanisms to the Latent Diffusion Model to guide the generative process. One such example would be directly providing the desired volume fraction of the generated component, i.e., generating components of user-defined size. Additionally, we believe utilizing a larger dataset composed of optimized designs from different Topology Optimization algorithms with larger variability in loading conditions would be valuable to the community. We hope our



(a) 32^3 *resolution*

(b) 64^3 *resolution*



(c) 128^3 *resolution*

Fig. 17. Example generated designs. Each row corresponds to a single Initial Strain Energy sample, and each column is a different generated design.

framework will be a starting point for using deep learning models for component design during the ideation phase.

CRediT authorship contribution statement

Ethan Herron: Writing – original draft, Software, Methodology, Formal analysis, Conceptualization. **Jaydeep Rade:** Validation, Software, Data curation. **Anushrut Jignasu:** Visualization. **Baskar Ganapathysubramanian:** Writing – review & editing, Supervision, Conceptualization. **Aditya Balu:** Writing – review & editing, Supervision, Conceptualization. **Soumik Sarkar:** Writing – review & editing, Supervision, Conceptualization. **Adarsh Krishnamurthy:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was partly supported by the National Science Foundation, United States under grant LEAP-HI 2053760 and DMREF 2323716. We want to acknowledge NVIDIA Corporation for donating GPUs through their GPU research grant program.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2024.103707.

References

[1] Goodfellow Ian, Pouget-Abadie Jean, Mirza Mehdi, Xu Bing, Warde-Farley David, Ozair Sherjil, et al. Generative adversarial networks. Adv Neural Inf Process Syst 2014;3.

[2] Isola Phillip, Zhu Jun-Yan, Zhou Tinghui, Efros Alexei. Image-to-image translation with conditional adversarial networks. In: Computer vision and pattern recognition. CVPR, 2017, p. 5967–76.

[3] Kang Minguk, Zhu Jun-Yan, Zhang Richard, Park Jaesik, Shechtman Eli, Paris Sylvain, et al. Scaling up GANs for text-to-image synthesis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. CVPR, 2023.

[4] Ho Jonathan, Jain Ajay, Abbeel Pieter. Denoising diffusion probabilistic models. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. In: Advances in neural information processing systems, vol. 33, 2020, p. 6840–51.

[5] Nichol Alexander Quinn, Dhariwal Prafulla. Improved denoising diffusion probabilistic models. In: Meila Marina, Zhang Tong, editors. Proceedings of the 38th international conference on machine learning. Proceedings of machine learning research, vol. 139, PMLR; 2021, p. 8162–71.

[6] Rombach Robin, Blattmann Andreas, Lorenz Dominik, Esser Patrick, Ommer Bjorn. High-resolution image synthesis with latent diffusion models. In: Computer vision and pattern recognition. CVPR, 2022, p. 10674–85.

[7] Orme Melissa E, Gschweitl Michael, Ferrari Michael, Madera Ivan, Mouriaux Franck. Designing for additive manufacturing: Lightweighting through topology optimization enables lunar spacecraft. J Mech Des 2017;139(10).

[8] Liu Jikai, Ma Yongsheng. A survey of manufacturing oriented topology optimization methods. Adv Eng Softw 2016;100:161–75.

[9] Rasulzade Jalal, Rustamov Samir, Akhmetov Bakytzhan, Maksum Yelaman, Nogaibayeva Makpal. Computational acceleration of topology optimization using deep learning. Appl Sci 2023;13(1).

[10] Bendsøe Martin Philip, Kikuchi Noboru. Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Engrg 1988.

[11] Bendsøe Martin P. Optimal shape design as a material distribution problem. Struct Optim 1989;1(4):193–202.

[12] Wang Michael Yu, Wang Xiaoming, Guo Dongming. A level set method for structural topology optimization. Comput Methods Appl Mech Engrg 2003;192(1–2):227–46.

[13] Van Dijk NP, Yoon GH, Van Keulen F, Langelaar M. A level-set based topology optimization using the element connectivity parameterization method. Struct Multidiscip Optim 2010;42:269–82.

[14] Hajela P, Lee E, Lin C-Y. Genetic algorithms in structural topology optimization. In: Topology design of structures. Dordrecht: Springer Netherlands; 1993, p. 117–33.

[15] Wang SY, Tai K. Structural topology design optimization using genetic algorithms with a bit-array representation. Comput Methods Appl Mech Engrg 2005;194(36):3749–70.

[16] Kingma Diederik P, Welling Max. Auto-encoding variational Bayes. In: International conference on learning representations. ICLR, 2014.

[17] Saharia Chitwan, Chan William, Chang Huiwen, Lee Chris, Ho Jonathan, Salimans Tim, et al. Palette: Image-to-image diffusion models. In: ACM SIGGRApH. 2022.

[18] Sosnovik Ivan, Oseledets Ivan. Neural networks for topology optimization. Russian J Numer Anal Math Modelling 2019;34(4):215–23.

[19] Yu Yonggyun, Hur Taeil, Jung Jaeho, Jang In Gwun. Deep learning for determining a near-optimal topological design without any iteration. Struct Multidiscip Optim 2019;59(3):787–99.

[20] Dalei Wang, Xiang Cheng, Pan Yue, Chen Airong, Zhou Xiaoyi, Zhang Yiquan. A deep convolutional neural network for topology optimization with perceptible generalization ability. Eng Optim 2021;54:1–16.

[21] Rawat Sharad, Shen MH Herman. A novel topology optimization approach using conditional deep learning. 2019, arXiv preprint arXiv:1901.04859.

[22] Li Baotong, Huang Congjia, Li Xin, Zheng Shuai, Hong Jun. Non-iterative structural topology optimization using deep learning. Comput Aided Des 2019;115:172–80.

[23] Abueidda Diab W, Koric Seid, Sobh Nahil A. Topology optimization of 2D structures with nonlinearities using deep learning. Comput Struct 2020;237:106283.

[24] Takahashi Yusuke, Suzuki Yoshiro, Todoroki Akira. Convolutional neural network-based topology optimization (CNN-TO) by estimating sensitivity of compliance from material distribution. 2019, arXiv preprint arXiv:2001.00635.

[25] Mazé François, Ahmed Faez. Diffusion models beat GANs on topology optimization. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, (8):2023, p. 9108–16.

[26] Banga Saurabh, Gehani Harsh, Bhilare Sanket, Patel Sagar, Kara Levent. 3D topology optimization using convolutional neural networks. 2018, arXiv preprint arXiv:1808.07440.

[27] Rade Jaydeep, Balu Aditya, Herron Ethan, Pathak Jay, Ranade Rishikesh, Sarkar Soumik, et al. Algorithmically-consistent deep learning frameworks for structural topology optimization. Eng Appl Artif Intell 2021;106:104483.

[28] Ghadai Sambit, Jignasu Anushrut, Krishnamurthy Adarsh. Direct 3D printing of multi-level voxel models. Addit Manuf 2021;40:101929.

[29] Chandrasekhar Aaditya, Suresh Krishnan. TOuNN: Topology optimization using neural networks. Struct Multidiscip Optim 2021;63(3):1135–49.

[30] Qian Chao, Ye Wenjing. Accelerating gradient-based topology optimization design with dual-model artificial neural networks. Struct Multidiscip Optim 2020;1–21.

[31] Bujny Mariusz, Aulig Nikola, Olhofer Markus, Duddeck Fabian. Learning-based topology variation in evolutionary level set topology optimization. In: Proceedings of the genetic and evolutionary computation conference. 2018, p. 825–32.

[32] Lagaros Nikos, Kallioras Nikos, Kazakis Georgios. Accelerated topology optimization by means of deep learning. Struct Multidiscip Optim 2020;62.

[33] Oh Sangeun, Jung Yongsu, Lee Ikjin, Kang Namwoo. Design automation by integrating generative adversarial networks and topology optimization. In: International design engineering technical conferences and computers and information in engineering conference, vol. 51753, American Society of Mechanical Engineers; 2018, V02AT03A008.

[34] Guo Tinghao, Lohan Danny J, Cang Ruijin, Ren Max Yi, Allison James T. An indirect design representation for topology optimization using variational autoencoder and style transfer. In: 2018 AIAA/aSCE/AHS/aSC structures, structural dynamics, and materials conference. 2018, p. 0804.

[35] Kallioras Nikos Ath, Lagaros Nikos D. DzAIN: Deep learning based generative design. Procedia Manuf 2020;44:591–8.

[36] Kallioras Nikos Ath, Lagaros Nikos D. MLGen: Generative design framework based on machine learning and topology optimization. Appl Sci 2021;11(24):12044.

[37] Oh Sangeun, Jung Yongsu, Kim Seongsin, Lee Ikjin, Kang Namwoo. Deep generative design: Integration of topology optimization and generative models. J Mech Des 2019;141(11).

[38] Jang Seowoo, Yoo Soyoung, Kang Namwoo. Generative design by reinforcement learning: Enhancing the diversity of topology optimization designs. Comput Aided Des 2022;146:103225.

[39] van den Oord Aaron, Vinyals Oriol, kavukcuoglu koray. Neural discrete representation learning. In: Guyon I, Luxburg U Von, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. In: Advances in neural information processing systems, vol. 30, Curran Associates, Inc.; 2017.

[40] Karras Tero, Laine Samuli, Aila Timo. A style-based generator architecture for generative adversarial networks. In: Computer vision and pattern recognition. CVPR, 2019, p. 4396–405.

[41] Ho Jonathan, Salimans Tim. Classifier-free diffusion guidance. 2022, arXiv preprint arXiv:2207.12598.

[42] Heusel Martin, Ramsauer Hubert, Unterthiner Thomas, Nessler Bernhard, Hochreiter Sepp. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Guyon I, Luxburg U Von, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. In: Advances in neural information processing systems, vol. 30, Curran Associates, Inc.; 2017.

[43] Szegedy Christian, Vanhoucke Vincent, Ioffe Sergey, Shlens Jon, Wojna ZB. Rethinking the inception architecture for computer vision. In: Computer vision and pattern recognition. CVPR, 2016.

[44] Lee Seunghye, Kim Hyunjoo, Lieu Qui X, Lee Jaehong. CNN-based image recognition for topology optimization. Knowl-Based Syst 2020;198:105887.

[45] Rade Jaydeep, Jignasu Anushrut, Herron Ethan, Corpuz Ashton, Ganapathysubramanian Baskar, Sarkar Soumik, et al. Deep learning-based 3D multigrid topology optimization of manufacturable designs. Eng Appl Artif Intell 2023;126:107033.