# Deep Learning on Point Clouds for 3D Shape Generation

**Max Ohm**
Yale University

**Lukass Kellijs**
Yale University

**Jacqueline Wang**
Yale University

## Abstract

We present a deep learning framework for 3D shape generation using signed distance functions (SDFs). Our model learns a compact latent representation of object geometry from point clouds, enabling the generation of novel shapes with smooth, continuous surfaces. We propose a variational autoencoder architecture that combines PointNet++-style hierarchical feature extraction with a DeepSDF-inspired decoder, trained on sampled SDF values from car models in the ShapeNet dataset. The model achieves low reconstruction error and produces plausible, diverse shapes from sampled latent vectors. We further explore the structure of the learned latent space and discuss potential extensions for improved geometry encoding, simulation-aware design, and shape optimization.

## 1 Introduction and motivation

With the growing prominence of machine learning, more and more techniques from this field are being used in computational engineering. A particularly interesting application is the use of deep learning models for generating new engineering designs, enabling faster exploration of optimal design and parameter space. Such generative methods could also provide the benefit of training models that can conduct simulations "in parallel" with design generation. This new paradigm of engineering could drastically change how modern engineering is conducted, especially in complex fields that require expensive simulations and detailed optimization—such as the design of airplane wings, car chassis, or fusion reactor cores. Many emerging companies in this field aim to leverage machine learning to accelerate and optimize design.

We explored machine-learning-driven generation of 3D designs. Specifically, we trained a model on 3D geometry mesh data to learn a compact latent representation of object shape and, from that latent code, to generate new, similar designs. We have considered various 3D representations (meshes, point clouds, voxel grids) and will use Signed Distance Functions (SDFs) both as input and output. An SDF is a function that, for any point in space, returns the distance to the nearest surface of the object.

## 2 Background and related work

### 2.1 Design Representation Methods

First and foremost, an appropriate design representation must be chosen. Various design representations for generative models in engineering design are present in the literature [Regenwetter et al., 2022]. Perhaps the purest form of representation is that of a triangular mesh as this is the format directly used in many engineering software tools and simulations. It can be considered a special type of graph and can leverage graph operators like graph convolutional layers (GCN). Zhou et al. [2020] developed a convolutional mesh autoencoder using such layers. Meshes, however, are challenging to generate directly and are limited by the expressivity of GCNs.

A somewhat simpler representation is the point cloud. Point clouds are unordered sequences of points in 3D space and are often the natural output of 3D scanning software. They can also be easily

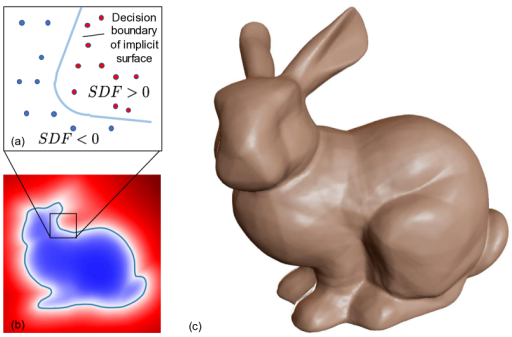


Figure 1: Visualization of an SDF function [Park et al., 2019].

generated from meshes by sampling points on the mesh surface (interpolating between vertices). However, they are not particularly useful on their own for downstream tasks. Many architectures have been proposed for processing and generating point clouds like that of Luo and Hu [2021]. These architectures must effectively handle the irregular spatial distribution and unordered nature of the data. A particularly prominent class of architectures for point cloud processing is PointNet [Qi et al., 2017a] and PointNet++ [Qi et al., 2017b].

The PointNet++ architecture organizes points into a hierarchy of neighborhoods: it repeatedly samples a subset of "centroid" points, groups nearby points based on distance, and applies small shared multi-layer perceptrons to each local patch before aggregating features via max pooling. Through this strategy, PointNet++ combines fine-grained local context with global shape encoding, resulting in significantly better performance on both classification and segmentation benchmarks.

Finally, recent works have explored using Signed Distance Functions for geometry representation. An SDF (Signed Distance Function/Field) is a parameterization method that consists of a (typically 3D) functional map from a coordinate point to an SDF value. In this way, it can also be interpreted as a point cloud with a single scalar feature indicating the distance to the nearest point on the object's surface, with the sign indicating whether the point lies inside or outside the object.

Our project was in part inspired by the growing use of SDFs in engineering and modeling. Many up-and-coming companies, such as nTop and PhysicsX, which specialize in computational engineering and machine-learning-based methods, use such functions for design representation. These so-called implicit geometries, as opposed to traditional boundary representations (B-Reps, i.e., meshes), offer several advantages: they enable smooth interpolation between shapes, support gradient-based optimization and differentiable simulation, and avoid common issues associated with B-Rep meshing.

### 2.2 Generative Design Architectures

The choice of architecture for representing, encoding (i.e. compressing), and generating new designs largely depends on the representation method selected.

Since we aim to generate the SDF representation of a geometry, we focus primarily on methods tailored to this representation. An SDF is a continuous function $f : \mathbb{R}^3 \to \mathbb{R}$, where $f(\mathbf{x})$ gives the signed distance to the nearest surface. Neural networks that represent such field functions are known as *neural fields* Xie et al. [2022] or *coordinate-based neural networks*. These models map continuous spatial (or spatiotemporal) coordinates to signal values. They are particularly useful for representing geometry, physical fields, or signals like images and sound in a compact and differentiable way. A closely related type of neural field is the *occupancy network*, which learns a function $f_\theta(x) \in [0, 1]$ that represents the probability of a point lying inside the surface of a 3D object.

A canonical paper for SDF representations is *DeepSDF* by Park et al. [2019]. In their architecture, they use an autodecoder (i.e., just the decoder part of an autoencoder) parameterized by a learned embedding vector. This embedded vector encodes a specific shape and thus conditions the DeepSDF network. The network concatenates this conditioning vector with a query coordinate and outputs the corresponding SDF value after passing through multiple fully connected layers. A closely related occupancy network is IM-NET [Chen and Zhang, 2019]. This kind of latent representation of

geometry is something we aim to learn, as it could more efficiently encode key features of designs and enable accelerated simulation and surrogate modeling.

However, such approaches are often limited in their expressiveness. Pooling information from point clouds is inherently difficult, and fundamentally, design generation involves learning an entire distribution rather than producing individual samples.

We sought to explore novel methods for effectively aggregating global features of a design and producing a compact latent encoding. One such method involves using an attention-based aggregation mechanism. Existing state-of-the-art methods such as *3DShape2VecSet* typically use a set of feature vectors as the latent representation [Zhang et al., 2023]. This set encodes a specific shape and is then used to condition a decoder network. Each vector in the set can be treated as a token, enabling the use of transformers to model interactions between them. Such a vector set has also been shown to outperform concatenation for conditioning neural fields [Rebain et al., 2022].

Alternatively, it could be argued that many of these methods lean too heavily on the analogy between SDFs and point clouds. Fundamentally, an SDF is a continuous function, not a discrete set of points. Ideally, then, one would use an architecture that is natively suited to processing functions. This led us to explore the field of *neural operators*—neural networks designed to approximate not just functions, but operators. These models map functions to functions and are widely used in scientific machine learning, particularly in solving partial differential equations (PDEs) and related problems in physics and fluid dynamics. We aimed to draw inspiration from this domain in designing our own network.

In light of this, we began by experimenting with a *VecSet Transformer* architecture. The key novel idea behind this model is the use of central points that aggregate information from neighboring points via cross-attention. However, this architecture did not perform well on our generated dataset. We then turned to neural operator methods [Li et al., 2020], specifically exploring the combination of *Graph Neural Operators* and *Fourier Neural Operators* [Zhou et al., 2021], [Li et al., 2021], [Li et al., 2023] to help embed information about the SDF function. These layers, however, were not inherently designed for compression. They required additional feedforward networks for pooling, were overly complex and memory-intensive for our application, and also suffered from vanishing gradient issues.

## 3 Methods

Based on our exploratory work, we developed a VAE-like network that leverages layers from the PointNet++ architecture as an encoder and a DeepSDF-inspired decoder.

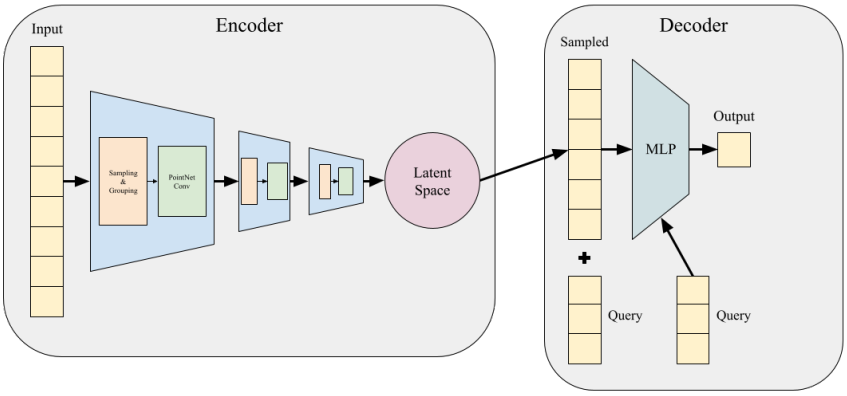


Figure 2: Model Architecture

Our model processes a point cloud composed of a mixture of sampled points. We generated our own dataset from car meshes in the ShapeNet dataset. In total, we created 995 distinct point cloud samplings of cars, consisting of 15,000 points sampled on the mesh surface, 5,000 near-surface points, and 10,000 uniformly distributed points within the normalized bounding cube of each mesh.

The network takes as input 2,048 randomly sampled points from each point cloud. Within the encoder, these points are passed through two hierarchical pooling layers. Each pooling layer downsamples the point set using *farthest point sampling*—a method also used in the *3DShape2VecSet* paper—which selects a subset of points that are maximally dispersed in space. These subsampled points aggregate information from their neighbors using shared fully connected layers. A global max pooling layer is then applied to obtain a single $N = 1024$-dimensional vector representing the shape.

This feature vector is subsequently reduced to 512 dimensions and passed through fully connected layers that parametrize the latent distribution. Sampling from this distribution yields the final latent representation of dimension $N_z = 512$.

The decoder takes as input the concatenation of the latent vector and a query coordinate point (applied in parallel for all query points). This input is passed through a multi-layer perceptron (MLP) with gradually decreasing width. In the middle of the MLP (at the $W = 128$ layer), the coordinate is concatenated again, following an architectural trick from the DeepSDF paper that was shown to improve performance. The final layer is a $64 \to 1$ linear transformation, followed by a $\tanh$ activation.

The motivation behind this architecture that the encoder network learns global features of the design, while the latent distribution enables exploration of the learned design space. This allows us to visually evaluate similarities between car designs and sample novel shapes from the latent space.

The total loss function consists of three terms: the mean squared error (MSE) between predicted and ground truth SDF values at sampled query points, a Kullback-Leibler (KL) divergence regularization term enforcing a Gaussian prior on the latent distribution, and an Eikonal loss term [Gropp et al.] that encourages the network to satisfy the Eikonal equation $\|\nabla f(x)\| = 1$, a property of true SDFs. The full loss takes the form:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{Eik}}\mathcal{L}_{\text{Eikonal}}. \tag{1}$$

We trained the model for 150 epochs and achieved a final MSE loss of $0.0011$, with approximately negligible contributions from the Eikonal and KL divergence terms.

## 4 Results

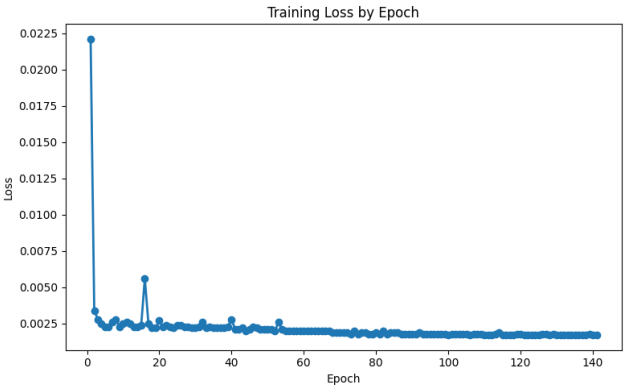


Figure 3: Loss across epochs

We evaluated the ability of our model to generate SDFs both quantitatively and qualitatively. As described above, the model was trained to minimize three loss components: the mean squared error (MSE) between predicted and actual SDF values, the Eikonal loss of the generated SDF, and the KL divergence of the latent space embedding. The model effectively minimized its objective on the training data and demonstrated the ability to generalize to unseen shapes, as evidenced by the decreasing loss over epochs.

A key advantage of variational autoencoders is their ability to produce a smooth latent space—i.e., one in which any sample from the unit normal distribution is likely to correspond to a valid and realistic car shape. Ideally, there should be no latent regions that fail to represent observed samples,


Figure 4: Latent Space Visualization.

and the latent codes should concentrate near the origin rather than being uniformly distributed. To assess whether our model's encoding satisfies these properties, we visualized the latent space using both t-distributed stochastic neighbor embedding (t-SNE) and principal component analysis (PCA), shown in Figure 4. Each plot is approximately colored by point density. Our model's latent embedding appears to form a continuous, smooth latent space, where most vectors sampled from the unit normal are likely to correspond to realistic shapes. However, we observed no strong clustering or interpretable trends in the latent codes, suggesting that the model did not learn to meaningfully organize the latent space.

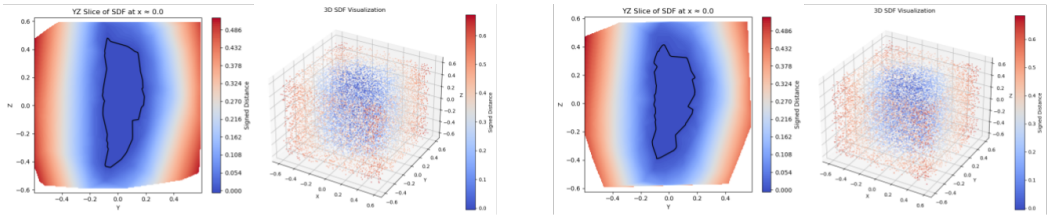


Figure 5: Two generated car SDFs, in 2D and 3D

A major objective of our project was to generate novel shapes by sampling from the VAE's latent space. To test this generative capability, we sampled random latent vectors from the unit normal distribution and passed them through our decoder to produce SDF values at a dense set of query points. Figure 5 displays several resulting shapes. While the generated outputs are not perfect, the model is capable of producing surfaces that are recognizably similar to cars.

## 5 Conclusions and future work

We implemented a proof of concept for an architecture capable of 3D shape reconstruction and generation. The model is trained on point clouds annotated with ground truth SDF values and can predict the SDF value at arbitrary query points. The architecture comprises hierarchical set abstraction layers (inspired by PointNet++), a feature propagation layer, and a variational autoencoder. The decoder performs shape reconstruction by predicting SDF values for sampled coordinates.

Overall, this is a challenging problem, and there are several areas for improvement in future work. One critical factor is the selection of input points. To enable accurate SDF reconstruction, we balance sampling points on the surface, near the surface, and uniformly throughout space. While surface and near-surface points carry important geometric information, their small distance values can lead the model to predict near-constant outputs if not complemented by sufficiently distant points. This issue is particularly relevant under the L2 loss, where small constant errors are weakly penalized. Additionally, the model may be biased toward predicting positive SDF values if too few points are sampled from within the solid volume—especially in the case of hollow objects.

Another important area for improvement is the dataset size. We currently sample 30,000 points per shape, which may be insufficient for capturing fine-grained SDF variations across the car models. A further limitation may stem from the global feature pooling mechanism used. Our current

implementation relies on max pooling to aggregate point cloud features, a method that performs well for classification tasks but may discard important local information in the context of detailed shape reconstruction. A potential improvement would be to use attention-based aggregation to combine the features of the subsampled points more effectively.

Additionally, we would like to investigate the use of improved positional encodings, such as Fourier feature embeddings, to better capture spatial structure. Another future direction includes incorporating adversarial loss functions, as proposed in recent work on implicit shape representations [Zamorski et al., 2019] [Kleineberg et al., 2020].

A long-term goal of this project is to use the latent space of SDFs for controlled shape manipulation and generation. By interpolating or stepping in specific directions from known latent codes, it may be possible to synthesize novel designs that retain structural similarity to known objects. These synthetic shapes can then be converted to meshes and physically realized through 3D printing or simulation.

We also explored alternative architectures. Specifically, we experimented with a Graph Neural Operator (GNO) for SDF generation. While we were unable to train it to produce sufficiently accurate outputs within our constraints, we believe this direction remains promising with further tuning and additional data.

While prior work has focused on generating 3D point clouds from learned representations Luo and Hu [2021], Zamorski et al. [2019], SDFs offer smoother, more continuous surfaces that are better suited for downstream simulation and manufacturing tasks Park et al. [2019]. Beyond shape generation, we aim to leverage the learned latent codes to train auxiliary networks that predict physical properties (e.g., drag coefficients) directly from the geometric embedding—enabling design optimization in latent space.

## References


Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling, September 2019. URL http://arxiv.org/abs/1812.02822. arXiv:1812.02822 [cs].

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for Learning Shapes.

Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial Generation of Continuous Implicit Shape Representations, March 2020. URL http://arxiv.org/abs/2002.00349. arXiv:2002.00349 [cs].

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural Operator: Graph Kernel Network for Partial Differential Equations, March 2020. URL http://arxiv.org/abs/2003.03485. arXiv:2003.03485 [cs].

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021. URL http://arxiv.org/abs/2010.08895. arXiv:2010.08895 [cs].

Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-Informed Neural Operator for Large-Scale 3D PDEs, September 2023. URL http://arxiv.org/abs/2309.00583. arXiv:2309.00583 [cs].

Shitong Luo and Wei Hu. Diffusion Probabilistic Models for 3D Point Cloud Generation, June 2021. URL http://arxiv.org/abs/2103.01458. arXiv:2103.01458 [cs].

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, January 2019. URL http://arxiv.org/abs/1901.05103. arXiv:1901.05103 [cs].

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, April 2017a. URL http://arxiv.org/abs/1612.00593. arXiv:1612.00593 [cs].

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, June 2017b. URL http://arxiv.org/abs/1706.02413. arXiv:1706.02413 [cs].

Daniel Rebain, Mark J. Matthews, Kwang Moo Yi, Gopal Sharma, Dmitry Lagun, and Andrea Tagliasacchi. Attention Beats Concatenation for Conditioning Neural Fields, September 2022. URL http://arxiv.org/abs/2209.10684. arXiv:2209.10684 [cs].

Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. Deep Generative Models in Engineering Design: A Review. *Journal of Mechanical Design*, 144(7): 071704, July 2022. ISSN 1050-0472, 1528-9001. doi: 10.1115/1.4053859. URL https://asmedigitalcollection.asme.org/mechanicaldesign/article/144/7/071704/1136676/Deep-Generative-Models-in-Engineering-Design-A.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural Fields in Visual Computing and Beyond, April 2022. URL http://arxiv.org/abs/2111.11426. arXiv:2111.11426 [cs].

Maciej Zamorski, Maciej Zięba, Piotr Klukowski, Rafał Nowak, Karol Kurach, Wojciech Stokowiec, and Tomasz Trzciński. Adversarial Autoencoders for Compact Representations of 3D Point Clouds, May 2019. URL http://arxiv.org/abs/1811.07605. arXiv:1811.07605 [cs].

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models, May 2023. URL http://arxiv.org/abs/2301.11445. arXiv:2301.11445 [cs].

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications, October 2021. URL http://arxiv.org/abs/1812.08434. arXiv:1812.08434 [cs].

Yi Zhou, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason Saragih, Hao Li, and Yaser Sheikh. Fully Convolutional Mesh Autoencoder using Efficient Spatially Varying Kernels, October 2020. URL http://arxiv.org/abs/2006.04325. arXiv:2006.04325 [cs].