

# Improving Generalization of DeepSDF

Marina Aoki (03773384)\*      Soundous Chamrah (03782724)\*  
Leonhard Chen (03711258)\*      Karim ElBowety (03784175)\*

Technical University of Munich

## Evaluation Criteria

### 1. Presentation Quality (10 points)

- **Clarity of Information:**  
Is the problem/task properly introduced?  
Is the information conveyed clearly to the audience?
- **Key Contributions:**  
Does the presentation include key contributions such as modifications and their results?
- **Time Management:**  
Is the presentation time used effectively? Don't run overtime - please practice your presentation :)
- **Use of Visual Aids:**  
Preference for tables (e.g., method comparison, ablation study), figures (e.g., method pipeline, visual result), and videos (e.g., interactive result, shape interpolation) over text.

### 2. Methodology / Results (10 points)

- **Technical Difficulty:**  
Do the modifications have proper technical difficulties?
- **Improvement Over Baseline:**  
Do the modifications improve over the baseline qualitatively and quantitatively? If not, do you provide a scientific analysis and future directions for improvement?

### 3. Q&A (5 points)

- **Response to Questions:**  
Are the questions properly answered?

### 4. Report (15 points)

- **Length:**  
The length of the report should not exceed 4 pages

(not including references).

- **Structure and Formatting:**

The report should be structured and formatted according to a CVPR paper.

### Shape Generation - Project 3

Paper: *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation* [2]

(<https://arxiv.org/pdf/1901.05103>)

Dataset: Objaverse

(<https://objaverse.allenai.org>)

Modifications:

- Explore various ways of improving the generalization ability across different categories, e.g., adding class embedding, and text feature descriptors.
- Adding additional test-time optimization loss for better shape fitting especially for objects with thin structure
- Add color code to shape latent code, also learn view-independent color field given colored-mesh

---

\*Equal contribution

## Abstract

*In the past years there have been many advances in representing 3D geometry using machine learning approaches. One of such is DeepSDF which learns a continuous Signed Distance Function (SDF) for representing a class of shapes. In this project we aim to further improve generalization with the following 3 changes. First we propose to extend generalization by adding more shape classes. Next we propose texturing the shape representation by using an additional latent vector. Finally we propose using multiple forward and backward passes to improve test time optimization.*

## 1. Introduction

Advancements in machine learning on 3D geometry have brought significant progress in representing complex 3D shapes accurately and efficiently. DeepSDF[2] has emerged as a novel approach using an implicit representation. This project is conducted as part of the Machine Learning for 3D Geometry course and our primary goal is to enhance the existing DeepSDF model by addressing specific limitations.

While DeepSDF represents a significant advancement in 3D geometry, it still faces notable limitations that restrict its broader applicability. The model currently struggles to generalize across multiple classes, which reduces its flexibility in handling diverse datasets. Additionally, DeepSDF lacks the capability to texture the output shapes, a feature that is essential for creating realistic and visually detailed 3D models. Moreover, the model's performance declines when tested on corrupted data, such as thin or intricate geometries, resulting in less reliable outcomes.

To overcome these limitations, we propose several enhancements to the DeepSDF model. We first introduce class embeddings using one-hot encoding to improve generalization across multiple classes, enabling the model to effectively manage a wider variety of shapes. Next, we incorporate a latent vector that encodes color information, allowing the model to generate textured 3D shapes and thus produce more realistic outputs. Finally, we implement test-time optimization, involving multiple forward and backward passes, to enhance the model's robustness, particularly when dealing with thin geometries and other challenging data.

## 2. Related Work

We briefly review on related works from the paper.

**Representations for 3D Shape Learning** Point Clouds are efficient for 3D learning, but struggle with representing complex topologies and watertight surfaces.

Mesh-based representations can model similarly shaped objects with predefined templates or learned parameterizations and although effective for certain tasks, that can face the same issues as mentioned for point-clouds.

Voxel-based representations, despite being a natural extension of 2D techniques, face memory and resolution challenges. Octrees attempt to alleviate these issues but still fall short in producing fine details and smooth surfaces.

Implicit surface representations in DeepSDF addresses these issues by using a continuous SDF, which can produce complex watertight surfaces.

**Representation Learning Techniques** General Adversarial Networks can learn deep embeddings of target data by training discriminators adversarially against generators. These networks can generate realistic and complex objects, but the training is known to be unstable. While DeepSDF is also generative model, it takes a different approach by learning a latent space where each point corresponds to a shape.

Auto-Encoders learn efficient representations of data by compressing the input into a lower-dimensional latent space and then reconstructing it. Variational Auto-Encoders further more impose a probabilistic distribution on the latent space allowing for generating new samples with smooth and continuous interpolations between them. DeepSDF also utilizes a latent space to represent 3D shapes, but only has a decoder to reconstruct the SDF values of the shape.

Optimizing Latent Vectors is the fundamental aspect of DeepSDF's methodology for representing and reconstructing 3D shapes. Compact data representations are learned by training decoder-only networks by simultaneously optimizing the latent vectors and decoder weights.

**Shape Completion** 3D shape completion aims to infer unseen parts of given sparse or partial input observations. Classical methods are only to model a single shape, while most data-driven approaches use encoder-decoder architectures to reduce partial input to latent vectors for shape completion.

**DeepSDF Model** DeepSDF introduces a novel method for representing and generating 3D geometry using continuous SDFs.

The architecture of DeepSDF revolves around a fully connected neural network that serves as decoder. The decoder approximates the SDF, enabling the reconstruction of the 3D shape from a latent vector. The input is a 3D coordinate and a latent vector that represents the shape and the output is the corresponding SDF value at the queried coordinate. Each shape in the dataset is associated with a unique latent vector, which captures the shape’s intrinsic features. These latent vectors are stored and optimized during the training process compressing the shape into a fixed-size representation.

One of the key distinctions of DeepSDF is its ability to operate without the explicit encoder network. Instead of using an encoder to map input shapes to latent vectors, it directly optimizes the latent vectors during training. These latent vectors, treated as free parameters, are randomly initialized and then adjusted to minimize the reconstruction error of the Signed Distance Function (SDF) for each shape. This approach eliminates the need for an encoder and allows for a more flexible and potentially more accurate representation of the shape space, as the optimization process can focus entirely on refining the decoder’s ability to accurately reconstruct the 3D geometry from these latent vectors.

The training of DeepSDF involves optimizing both the parameters of the decoder network and the latent vectors associated with each shape in the training set. DeepSDF treats these latent vectors as free parameters that are directly optimized. The primary objective during training is to minimize the difference between the predicted SDF values and the true SDF values for a set of sampled points in space.

### 3. Method

In this section, we detail our data preparation process and the architectural modifications to improve the DeepSDF model. This includes the introduction of class

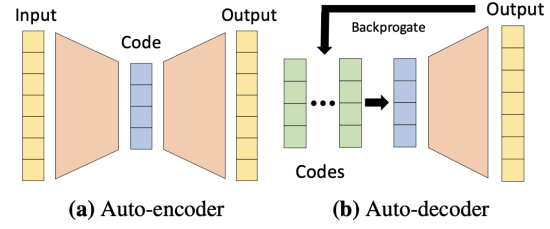


Figure 1. Illustration comparing an auto-encoder with the auto-decoder architecture used in DeepSDF. The decoder learns to predict SDF values given the corresponding 3D coordinates and the encoded shape.

embeddings, view direction extraction and color encoding for texturing, as well as the integration of test-time optimization through multiple forward and backward passes. We aim to demonstrate how these innovations address the limitations of the original DeepSDF model, particularly in generalization across multiple classes, texturing output shapes, and dealing with thin geometry.

#### 3.1. Data Preprocessing

In our pre-processing workflow, we modified the ”mesh-to-sdf” Python library to sample 200,000 points from each of the 3D models, following a method inspired by the original DeepSDF paper. It converts 3D mesh surfaces into a continuous Signed Distance Field by calculating the minimum distance from each sampled point in space to the nearest surface of the mesh. This library was selected over the provided pre-processor due to its compatibility with modern systems.

To calculate the SDF values for a given mesh, it is first transformed to fit inside a unit sphere. Then, laser scans are generated from virtual cameras set up around the mesh. Using these cameras, we introduce a custom method to extract the viewing direction as Euler angles consisting of  $\theta$  and  $\phi$ . The color is also extracted in this step to enable more detailed color coding. From the depth buffer of the laser scans, the library then calculates a surface point cloud and determines the SDF for each query point using a kd-tree.

To meet the model’s input requirements, we reformatted the output from two arrays — initially containing all positions and their corresponding SDF values, into two distinct arrays: one for positive SDF values and one for negative SDF values. Each array was struc-

tured to include the x, y and z coordinates followed by the respective SDF value, the corresponding color, and the viewing directions. Given the class imbalance in our dataset, we ensured uniformity by randomly selecting 80 samples from each of the four classes. Additionally, in order to maintain compatibility during network testing, we converted the original mesh files from .glb to .obj format.

### 3.2. Class Embeddings

To implement class embeddings in DeepSDF, we used a one-hot encoding. Each class of 3D shapes is represented by a unique binary vector where the value corresponding to the class is set to 1 and all other elements are set to 0. This one-hot vector is concatenated with the input to the decoder network. By including class information directly the model can better differentiate between shapes of different classes, thereby improving its ability to generalize across multiple classes. This approach allows the network to learn distinct features for each class, enhancing its performance in recognizing and reconstructing shapes from various categories and reducing confusion between similar shapes from different classes.

### 3.3. Latent Vector Encoding of Shape Texture

To incorporate shape texture into the latent vector, an additional latent vector representing the color is concatenated with the existing latent vector for each shape. The modified architecture can be seen in Figure 2. In addition, the network is provided the viewing direction of the camera as input to guide the color prediction, as suggested by Mildenhall et al. in their paper on Neural Radiance Fields[1]. The network first trains a shared latent code before splitting into the two networks for the shape and color respectively. The color network is provided with the viewing direction and the color latent vector. By integrating color information directly into the latent representation, the model can generate textured 3D shapes with accurate color details. This approach allows the decoder to produce not only the geometric structure but also the color of the shape, improving the realism and fidelity of the rendered shapes. This method performs well in encoding color by leveraging the additional latent dimensions to capture color variations, resulting in more visually appealing and detailed shape reconstructions.

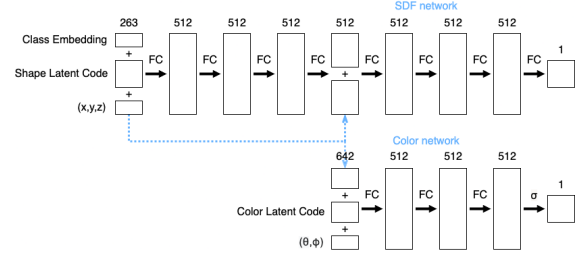


Figure 2. Diagram of our modified DeepSDF Architecture. The auto-decoder is split into two heads which output the predicted SDF and colour value of a given point respectively. The model receives the concatenated class embedding as a one-hot vector, the shape latent vector, the viewing direction and the 3D position of the point as inputs.

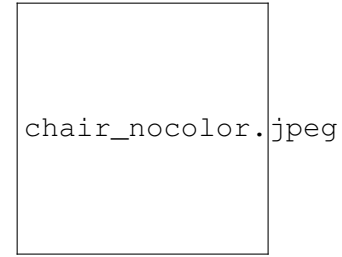


Figure 3. Chair no color


### 3.4. Test-Time Optimization

Test-time optimization involves performing several forward and backward passes through the model to refine the latent vector for a given shape. During this process, an initial latent vector is used to generate a predicted SDF of the shape. The discrepancy between the predicted SDF and the observed data is computed, and the latent vector is updated through gradient descent. Multiple iterations of this forward and backward pass refine the latent vector to better match the observed data, particularly for shapes that are thin or corrupted. This iterative optimization enhances the model’s ability to adapt to variations and incomplete data, leading to more accurate and robust shape reconstructions.

## 4. Results

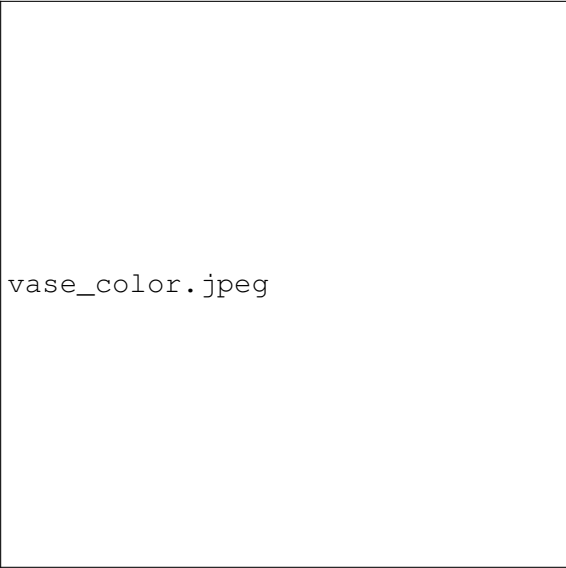
## 5. Conclusion

Overall DeepSDF is a impressive model as it is able to tackle multiple issues when learning 3D shapes. As presented in the paper[2] in table 1, the continuous SDF approach leads to a continuous surface representation. Furthermore it is able to deal with complex topologies,



chair\_color.jpeg

Figure 4. Chair color



vase\_color.jpeg

Figure 5. vase color

while having closed surfaces and surface normals. The model even has a fairly small size compared to other approaches, while being applicable to all evaluation tasks of representing known and unknown shapes as well as shape completion.

All these benefits come at the trade-off with its inference time. The original DeepSDF model in comparison to other approaches on the table has more than 30 times the inference time. This makes it effectively useless for any real-time applications.

When taking our proposed changes into account overall the training time is increased significantly, since we now need to incorporate multiple shape classes as well as the latent code for color. Notably the for allowing shape texturing the network size increases by roughly 50% compared to the original network. Adding the test-time optimization on top, we have multiple forward backward passes at inference further increasing the inference time.

Overall our proposed changes allow for generalization across multiple shape categories while adding shape textures and improving its quality. This on the other hand increases inference time. We think that this trade-off is a worthy compromise as the model is already unsuitable for real-time applications, given the inference time in the base model.

Since we were not able to fully train the model we believe that the improved DeepSDF can perform even better. For future improvements we believe that faster inference time could be worked on, though the model itself is best suited for compressing a class of 3D shapes.

## References

- [1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 4
- [2] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1, 2, 4