







Deep Neural Networks for Geometric Shape Deformation

Aida Farahani , Julien Vitay  , and Fred H. Hamker 

Chemnitz University of Technology, Straße der Nationen 62, 09111 Chemnitz,
Germany
{aida.farahani,julien.vitay,fred.hamker}@informatik.tu-chemnitz.de

Abstract. Geometric deep learning is a promising approach to bring the representational power of deep neural networks to 3D data. Explicit 3D representations such as point clouds or meshes can have varying and often a huge number of dimensions, what limits their use as an input to a neural network. Implicit representations such as signed distance functions (SDF) are on the contrary low-dimensional and fixed representations of the structure of a 3D shape that can be easily fed into a neural network. In this paper, we demonstrate how deep SDF neural networks can be used to precisely predict the deformation of a material after the application of a specific force. The model is trained using a set of custom finite element simulations in order to generalize to unseen forces.

Keywords: Geometric deep learning · Implicit neural representation · Geometric deformation modeling · FEM simulations · 3D data processing · Signed Distance Functions

1 Introduction

Deep neural networks have shown great potential in processing data such as images and videos. In contrast, providing the 3D structures to the neural networks is still challenging. Geometric deep learning (GDL) is a branch of machine learning (ML) that deals with 3D data for different purposes such as classification, compression, and segmentation. Although 3D models are more informative to describe the environment, common 3D representations are unfortunately not easily combined with neural networks, and methods introduced for geometric deformation processing are still under development.

The most common 3D representations are explicit, such as RGB-depth images, voxels, point clouds, and meshes. A RGB-depth map simply concatenates the depth information to the 2D image grid, which makes it Neural Network (NN)-friendly and an ideal data type for 3D pose estimation; however, the 3D modeling is partial and depends on the camera viewpoint.

Point clouds visualize the object's shape with a set of unordered 3D points sampled on the surface. The unstructured point clouds are most favored in the industry and are easily captured by 3D scanners. As a global descriptor of the

shape, they are mostly limited to segmentation and classification tasks [4, 5]. A point cloud does not contain surface information; therefore, dense sampling of points is usually required. Insufficient point sampling of a shape with fine details may cause an incomplete description while increasing the number of samples becomes quickly memory inefficient. By extending the pixels of 2D images to the third dimension, voxels represent the shape structure in a 3D space. Despite the regular structure and arrangement of data units in grids that make them NN-friendly, voxels are memory inefficient and are not suitable for small localized deformations. As for point clouds, the sampling rate can highly affect memory consumption.

Polygon meshes define a shape by a set of vertices on the surface and their neighboring connections as edges. Meshes are generic data structures favored in 3D modeling for being simple, informative, and easy to use. However, bringing them into the deep learning domain either limits us to the datasets with fixed topologies [6], or the input size of the network should be equal to the largest available mesh data sample [1]. In general, methods based on mesh representations suffer from a large set of features provided for the network as the whole structure should be fed simultaneously as input. The difficulty in working with meshes stems from the multitude of possibilities to apply a mesh on a shape, as many different topologies could be defined for one geometry. Therefore for large meshes, the network size drastically increases, and training will be computationally expensive.

Recent advances in GDL and challenges of applying explicit representations to deep networks have shifted attention to implicit ones. For instance, the well-known representation “Signed Distance Functions” (SDF) refers to a regression of the 3D space based on the distance from the shape surface. Here, each point in the 3D space takes a signed value depending on whether they are inside or outside of the shape. The density of these sample points increases the resolution of the final shape and could be justified based on the needs of the problem. SDF representations are a proper feed to deep networks and are highly efficient in terms of memory consumption.

The idea of combining SDF with deep neural models was first introduced in 2019 [3] and has received significant attention since then. First, they trained a network to estimate the SDF value of a shape S for each query input position in the 3D space (Fig. 1-left). This neural network is an embedded representation of a single shape. To generalize the network to multiple shapes, they add an encoded shape S_i as a condition to the network input and estimate the distance from the shape S_i . This shape encoding is implemented using a layered autoencoder architecture. Similar to latent codes in autoencoder architecture, the “codes” of an autoencoder are embedded representations of shapes. For each query point, the network predicts the SDF value corresponding to the provided condition. This continuous interpretation of space makes it possible to reconstruct shapes in any desired resolution and preserves shape deformations without large memory requirements. This method could effectively address the problem of efficient

shape embedding, and the network size is comparable to classical approaches for processing any size of meshes with arbitrary topology.

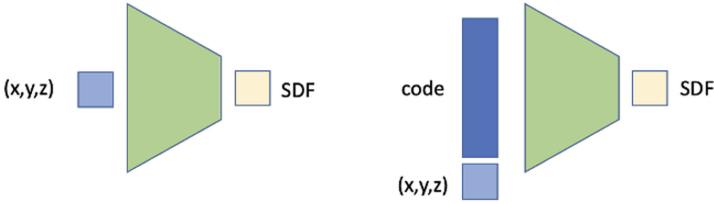


Fig. 1. Left: DeepSDF network for a single shape. Right: DeepSDF for an encoded shape and a 3D query point.

2 Shape Deformation Modeling

The term “Deformation” refers to the changes in the geometry of a structural body. Using Finite Element Analysis (FEA) has been a typical approach for many years that helps the engineers to investigate the simulation results and analyze the final product shape before production. However, the simulation process is time-consuming for large meshes with fine details and requires high computational power to solve the numerical equations. In addition, handmade tuning and re-execution of simulations are required to find the proper process parameters.

Most importantly, finding the optimized input parameters that lead to the desired results in the FEA approaches is only possible through trial and error. Neural networks can optimize input parameters that result in the desired output where cause-and-effect FE methods could not handle this functionality. Our research aims at modeling geometry deformations using neural networks. Inspired by the original DeepSDF paper [3], we designed a model combining SDF representations and deep networks to parameterize the shape deformation based on input conditions. Although the primary goal of the DeepSDF method was the efficient embedding of various shapes from different classes rather than deformations on one shape, our results showed the effectiveness of the approach for the shape deformation task.

2.1 Preparation of the Dataset

In order to train our model, we need a large dataset of deformed shapes. The publicly available datasets are mainly designed for classification or segmentation tasks and do not contain deformations or variations of shapes. Lacking appropriate data, we created our own dataset. For this purpose, we used FreeCAD, which is an open-source application for CAD design, including packages such

as FEA. In FreeCAD, we defined a simple cuboid, set the material properties, and fixed the initial constraints on both ends. We then added force constraints at different positions on the surface. By also varying the force magnitude, we obtained 6228 deformed shapes corresponding to each condition. A mesh sample in the FreeCAD environment is shown in Fig. 2. The generated dataset is freely available for download¹. The meshes then should be converted to the corresponding SDF representation. Each 3D mesh is first scaled into a unit sphere and is virtually rendered from 100 virtual cameras on the sphere surface. Then the distance from the closest mesh triangle is calculated. It is important to sample the points mostly near the surface to have an accurate sampling. We sampled 400000 points for each shape in our dataset.

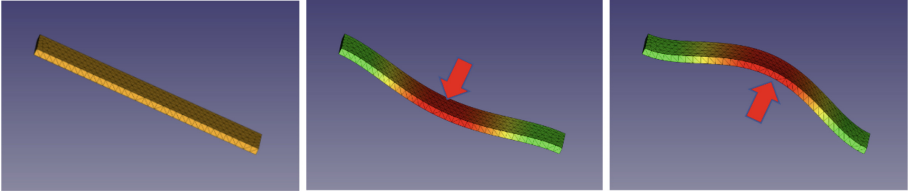


Fig. 2. Left: the initial shape. Middle and right: resulting deformed meshes affected by different forces.

2.2 Implementation Results

The neural network using SDF representations with arbitrary mesh topology can handle large meshes without increasing the network size. We provide the spatial coordinate and the force vector to the network and predict the corresponding SDF value as an output. We train a fully-connected neural network with six inputs (x , y , z coordinates of a sampled point, x , y position of the force applied on the surface, and force magnitude). After a Bayesian hyperparameter search using the optuna library, the neural network is composed of 4 hidden layers (130-118-150-148) using the LeakyRelu activation function and one linear output neuron. The mean square error loss is minimized using the Adam regularizer and a learning rate of 0.0005. The network is trained for 150 epochs, with a validation set composed of 20% of the data. At the end of the training, both the training and validation losses are below 10^{-6} .

The network provides a continuous function representing the distance values for each query point in the space, so another step needs to be taken for the final shape retrieval. Marching cube (MC) [2] is the most common approach for extracting the mesh in varying resolutions. By modifying the “cube size” as an input parameter to the algorithm, the SDF values are discretized inside a unit cube to reconstruct the surface of the shape in different resolutions. In Fig. 3,

¹ <https://www.tu-chemnitz.de/informatik/KI/projects/geometricdeeplearning>.

two generated mesh samples and the corresponding ground truth meshes are depicted (for cube count = 95^3).



Fig. 3. The reconstructed mesh from NN prediction (in gray) and the ground-truth mesh (in color) for two different samples. (Color figure online)

We use a popular metric, the Chamfer distance (CD), to evaluate the quality of the reconstructed mesh. This metric represents the difference between two sets of points S_1 and S_2 sampled from both mesh surfaces. In one variation of CD, for all S_1 points, the closest distance to the S_2 points is averaged, and the same process is repeated in reverse. The sum of these two averages is called CD, which should be closed to zero. We randomly chose 116 samples from the test set and reconstructed the mesh from the network predictions. The mean of CDs for 30000 sample points is shown in Table 1. As expected, increasing the number of cubes in the MC algorithm (that leads to finer meshes) reduces the Chamfer distance.

The use of SDF representation has the following advantages: Any simulated mesh or CAD model could be easily converted to an SDF representation so that the existing datasets could be used for training. Contrary to explicit representations, this representation is NN-friendly and handles large-size meshes with arbitrary topology. Also, a large number of shapes could be stored as a trained neural network and save storage. After training the network, less computational power and time are needed to process the large meshes compared with numerical approaches. To our knowledge, this is the first time that this representation is combined with neural networks for processing deformable objects.

Table 1. Chamfer distance metric for different resolutions of the MC algorithm.

Cube count	Chamfer distance
85^3	0.0009595
95^3	0.0009171
105^3	0.0008937
120^3	0.0008754
130^3	0.0008747

Despite these advantages, SDF representations have two major difficulties to deal with: The 3D mesh samples have to be watertight to divide the 3D space into inside and outside regions. Unfortunately, many CAD models are not

watertight, and some modifications in the algorithm are needed to be compatible with non-watertight meshes. The second issue is the additional step added at the end to discretize the space and extract an explicit representation of the shape such as a mesh or point cloud. The final step is, unfortunately, dependent on the required mesh size.

3 Conclusion and Future Work

In this paper, we showed that implicit representations could be effectively combined with neural networks to predict the shape deformations caused by an applied force. The designed network is able to be trained on very large meshes, while the size of the network is kept reasonable. The main advantage is the independence from mesh size and topology that brings the flexibility to process 3D shapes. However, the shapes provided to the network must be watertight. Future work could be suggested to find a solution for non-watertight meshes to generalize the approach. Another improvement could be proposed for the discretization phase in the end to substitute the current marching cube algorithm, which highly depends on the mesh size.

Acknowledgement. This work has been funded by the Federal Ministry of Education and Research (BMBF) - ML@Karoprod (01IS18055) and the German Research Foundation (DFG, 416228727) - SFB 1410 Hybrid Societies.

References

1. Feng, Y., Feng, Y., You, H., Zhao, X., Gao, Y.: Meshnet: mesh neural network for 3d shape representation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8279–8286 (2019)
2. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph Comput. Graph.* **21**(4), 163–169 (1987)
3. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 165–174 (2019)
4. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. [arXiv:1612.00593](https://arxiv.org/abs/1612.00593) [cs] (2016)
5. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: deep hierarchical feature learning on point sets in a metric space. [arXiv:1706.02413](https://arxiv.org/abs/1706.02413) [cs] (2017)
6. Ranjan, A., Bolkart, T., Sanyal, S., Black, M.J.: Generating 3D faces using convolutional mesh autoencoders. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 704–720 (2018)