Utilising 3D Gaussian Splatting for PointNet object classification

Exploring the potential of volume rendering techniques without using meshes

Dirk van Dale

Supervisor(s): Dr. Xucong Zhang

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Dirk van Dale
Final project course: CSE3000 Research Project
Thesis committee: Michael Weinmann, Xucong Zhang

Abstract

The demand for high-quality 3D visualizations has surged across various professional fields, prompting significant advancements in computer graphics. One such advancement is 3D Gaussian Splatting, a technique evolving from Lee Westover's splatting concept introduced in 1990. This study investigates the potential of 3D Gaussian Splatting to enhance PointNet classification techniques for 3D objects. Utilizing the Princeton ModelNet10 dataset, I convert 3D models into point clouds, applying 3D Gaussian Splatting to generate optimized Gaussian representations. These representations are used to train PointNet++ models with various feature configurations, including position, scale, rotation, opacity, and spherical harmonics. Our findings reveal that while 3D Gaussian Splatting enables effective 3D object classification, it does not outperform traditional methods that utilize ground truth data directly sampled from object surfaces. Nonetheless, the method demonstrates comparable accuracy, suggesting its viability in scenarios where initial meshes are unavailable. This research highlights the potential and limitations of 3D Gaussian Splatting in advancing 3D object classification.

1 Introduction

In recent years, the need for 3D visualizations has been rapidly increasing across a multitude of professional fields, including geospatial analysis, meteorology and medical imaging. Zhou *et al.* states "medical experts are often not aware of the many advanced three-dimensional (3D) medical image visualization techniques that could increase their capabilities in data analysis and assist the decision-making process for specific medical problems" [1]. This growing demand for high-quality and performance 3D rendering techniques has resulted in significant advancements in the field of Computer Graphics.

One such advancement is the development of 3D Gaussian Splatting, an evolution of the splatting technique initially introduced by Lee Westover in 1990 [2]. Westover's original concept involved using small pre-integrated kernels to represent volumetric elements and calculate their contribution to the final image. Over the years, the technique has seen substantial improvements, primarily due to technological advancements and increased computational power. In the paper on 3D Gaussian Splatting by Kerbl *et al.* [3], the technique is extensively explained. In their implementation, which is based on an alternative method called Neural Radiance Fields (NeRF) [4], they managed to achieve high-quality rendering results comparable to other well-known methods, while having significantly lower training times.

The next challenge lies in further utilizing the high-quality 3D scenes obtained via Gaussian Splatting, such as segmenting and classifying different objects using available Machine- or Deep-Learning techniques. Studies on this already exist, such as the paper "Segment any 3D Gaussians" by Cen *et al.* [5], but advancements regarding classification have yet to be further explored. This research aims to dive deeper into this particular topic by applying PointNet classification techniques onto 3D Gaussian Splatted common household objects.

The research question can be defined as such: *Can 3D Gaussian Splatting be used to improve the performance of training a PointNet classification model?*. Being able to achieve higher classification accuracy would be of great interest to a wide range of scientific fields, such as medical imaging [1].

The paper is structured as follows: Section 2 will describe the work related to this research, including 3D Gaussian Splatting, PointNet and its extension, PointNet++. Section 3 will then explain the methodology used to obtain the results, which can subsequently be found in section 4. Section 5 will discuss the ethical aspects of the research and on its reproducibility. Section 6 will reflect on the results and the work done and discuss further possible improvements on the research. Finally, section 7 will present the conclusion, and subsequently answer the research question using the results.

2 Related Work

This research builds upon the work of Kerbl *et al.* and Qi *et al.*, who developed 3D Gaussian Splatting and the PointNet architecture, respectively. This section will briefly cover these two concepts including their papers, and describe why they will be used for this research.

2.1 3D Gaussian Splatting

As mentioned in the introduction, this research is based upon a recently developed novel volumetric rendering technique introduced as "3D Gaussian Splatting" in a paper by Kerbl *et al.* [3]. In the paper, they describe the elements used to achieve their state-of-the-art optimization and rendering algorithm. In essence, 3D Gaussian Splatting builds upon similar volumetric rendering methods such as Neural Radiance Fields (NeRF) [4], which optimize a Multi-Layer Perceptron using volumetric ray marching for novel-view synthesis of scenes. This synthesis is done by using multiple images from different angles of the scene, otherwise known as photogrammetry, and producing Structure-from-Motion sparse point clouds. Where 3D Gaussian Splatting differentiates from other volume rendering techniques, is that it transforms each point into a Gaussian kernel. Known primarily for its applications in mathematics and statistics, the Gaussian function describes a type of probability distribution characterized by a peak at its mean, with values tapering off symmetrically as they move away from the

mean. A 3D Gaussian extends this concept into three dimensions, forming an ellipsoid where the density decreases with distance from the center. Kerbl *et al.* describe 3D Gaussians as being an excellent choice for scene representation, since they are a differentiable volumetric representation allowing for gradient-based optimization, supporting the computation of gradients with respect to their parameters. These parameters, mainly including position, scale, rotation, opacity and spherical harmonics, characterize 3D Gaussians in their respective space, and allow for smooth and continuous 3D scene representation.

The splatting part of the process refers to the process of projecting these 3D Gaussians onto a 2D image plane to generate a rendered image. The contribution of each Gaussian to the final image is computed based on its parameters and the camera view. The second key element of the technique consists of the following optimization process, where the generated image is compared to the original view and a reconstruction loss is calculated. This loss is subsequently used to calculate new weights for the different parameters, to eventually get the most optimal representation of the scene. The optimization also utilises "interleaved density control", which consists of occasionally adding and removing 3D Gaussians present in the cloud. Finally, the paper explores the development of a "fast visibility-aware rendering algorithm that supports anisotropic splatting and both accelerates training and allows real-time rendering", making use of the optimized 3D Gaussians, including spherical harmonics for visibility-awareness. The rendering process is parallelizable allowing for high quality reconstructions which can be interacted with in real-time. The resulting optimization and rendering algorithms allow for state-of-the-art visual quality while maintaining low training times.

2.2 PointNet/PointNet++

Classification of three-dimensional models has been researched using several techniques, including mesh-based methods (MeshCNN [6]), voxel-based methods (NormalNet [7]), and multi-view based methods (MVCNN [8]). However, none of these methods methods use point clouds as input directly. PointNet, introduced by Qi *et al.* in 2017 [9], is a deep learning architecture designed specifically for directly processing point clouds. This offers several advantages over the aforementioned traditional methods for 3D object classification and segmentation. One of these advantages is preserving the permutation invariance, meaning it is designed to be invariant to the order of points in the input. Since point clouds are unordered sets of points, this property is crucial. PointNet achieves this by using symmetric functions to aggregate features from individual points, ensuring that the network's output is not affected by the order of the input points.

PointNet++ is an extension of the original PointNet architecture, developed by the same group of researchers [10]. PointNet++ expands upon the original framework by addressing its limitations, particularly the handling of local structures at different scales within a point cloud. PointNet treats each point independently, which means it does not explicitly capture these local structures formed by neighboring points. This can be a limitation when dealing with complex geometries where local context is important. PointNet++ introduces a hierarchical learning approach similar to how convolutional neural networks operate on images. This hierarchy allows the network to capture fine-grained local features and combine them to form higher-level features. For this reason, I will be using the PointNet++ architecture for this research.

3 Methodology

The methodology will mostly cover the classification aspect of the research, with less emphasis on the 3D Gaussian Splatting performed on the data. The aim of the research is to find out what the implications of this technique, which has been researched extensively in the paper by Kerbl *et al.* [3] and briefly explained in section 2.1, are on object classification.

Data for the purpose of applying 3D Gaussian Splatting and subsequently classifying three-dimensional objects should comply to a set of pre-determined specifications. For 3D Gaussian Splatting, a dataset that consists of 3D Models or point clouds is needed. On top of that, the data should contain sufficient enough object examples for the purpose of training a classification model. In the end, a dataset containing hundreds of 3D models or point clouds, categorized into multiple classes, each split into a training and test set is needed.

The Princeton ModelNet10 dataset [11] is a subset of the ModelNet dataset specifically designed for 3D object recognition and classification tasks. Described in a paper by Wu *et al.* on the representation of volumetric shapes [11], the dataset was manually constructed with the goal of providing researchers in computer graphics with a clean collection of 3D CAD models. The dataset contains hundreds of said models, categorized into ten object classes of the most common objects in the world, including chairs, tables, bathtubs, etc. A sample of each can be found in fig. 1. Each category is divided into a training and test set, for the purpose of training object recognition models, this split was defined by Wu *et al.* in their paper [11] and can be considered as the default split for training a model using the dataset. I will therefore also be using this same split. The exact amount of models per category and this split can be found in table 1.

The 3D Gaussian Splatting optimizer described in the paper by Kerbl *et al.* [3] requires point clouds as input, not meshes. To account for this, their repository[1] provides a converter to extract undistorted images and Structure-from-Motion information from input images. These images are obtained by executing a Python script on each model in Blender [13]. This script first loads the mesh

[1] https://github.com/graphdeco-inria/gaussian-splatting

Class Name	Train	Test	Total
Bathtub	106	50	156
Bed	515	100	615
Chair	889	100	989
Desk	200	86	286
Dresser	200	86	286
Monitor	465	100	565
Nightstand	200	86	286
Sofa	680	100	780
Table	392	100	492
Toilet	344	100	444

Table 1: Amount of models per category in the Princeton ModelNet10 [11] dataset

Figure 1: A sample model from each category of the ModelNet10 dataset [12]

into the scene, applies a texture and then normalizes the scale, to keep the data as consistent as possible. Multiple light sources are placed in the scene, allowing for brightness variation per face of the mesh, needed when using the spherical harmonics of the 3D Gaussians as features for training. A camera is then rotated around the model in question in a smooth spiralling motion, while constantly facing the object, capturing a series of views. These views can then be used as input for the converter, resulting in SfM sparse point clouds. Next, the optimizer is run on the resulting sets of point clouds to get the best possible parameters for the 3D Gaussians. Figure 2 shows the result for a sample object, including the Gaussians at different scales.

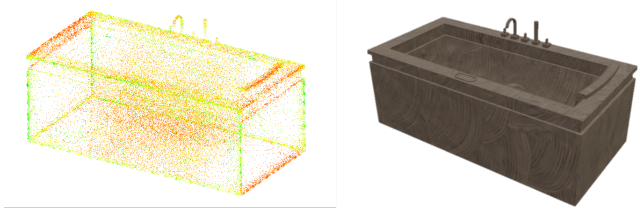

Figure 2: Visualization of 3D Gaussian Splatting point cloud of sample ModelNet10 bathtub object. (Left) The Gaussians at minimal scale. The red colors indicate denser areas of points, usually where more Gaussians are used for complex geometry. (Right) The Gaussians at maximal scale.

PointNet++ [9][10], the chosen classification framework that was covered in section 2.2, also uses sparse point clouds as input. However, the point clouds that 3D Gaussian Splatting produces contains different amounts of points per model, which means they need to be normalized for the classification accuracy to be realistic. This normalization is done via sampling a particular amount of points from every model. While there are multiple different types of sampling to choose from, the Furthest Point Sampling method seemed to be the most fitting for this research. Furthest Point Sampling (FPS) is a sampling method used to select a subset of points from a larger set by iteratively choosing the point that is the farthest away from the already selected points. This ensures that the sampled points are spread out as evenly as possible, covering the entire space of the original point cloud effectively. While this means that the overall shape of the object is captured as best as possible, the complex geometries of smaller elements present in the model might not be captured fully. However, the uniformity ensures that important geometric features are well represented and no region is overly neglected, providing provide a structured representation of the point clouds, which is essential for the effective performance of PointNet. Figure 3 shows the results of the Furthest Point Sampling method with 8192 points on the same bathtub ModelNet10 model.

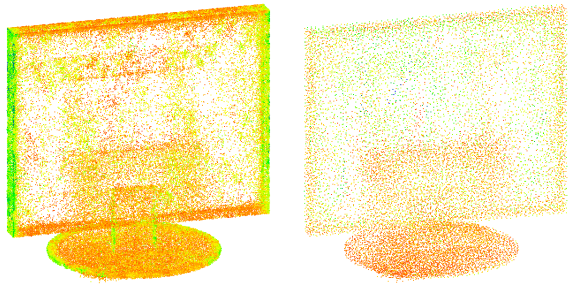

Figure 3: Furthest Point Sampling (FPS) visualized. The left image shows the original 3D Gaussian Splatted point cloud. The right image shows a sampling of 8192 points having been applied.

The PointNet++ classification architecture uses an *nx3* array as the input of features by default, corresponding to the *x, y* and *z* positions of the *n* points present in the cloud. However, the amount of columns of this input array can be modified to allow for more than three channels. 3D Gaussians, on top of a position (R^2), also contain a scale (R^3), rotation (R^4), opacity (R^1) and spherical harmonics (R^45), which define how the Gaussian should react to the angle it is viewed from regarding the different light sources in the scene. Combined, these parameters total up to 56 possible features. This means that our final input array becomes *nx56*. To extract these values from the *.ply* files that the 3D Gaussian Splatting optimizer created as output, I use the plyfile[2] module in Python. While the original

[2]https://pypi.org/project/plyfile/

PointNet++ implementation[3] developed by Qi *et al.* [10] uses TensorFlow as the machine learning framework, this research will be using a different implementation, developed by Xu Yan[4] using PyTorch instead. The main reason for this is that I personally am more familiar with the PyTorch framework than I am with TensorFlow, and it is generally better supported, with a large amount of libraries integrating well with it, such as NumPy.

4 Results

Before presenting the results obtained from classifying the 3D Gaussian Splatted point clouds in section 4.2, the specific PointNet configurations used for producing these results will be showcased in section 4.1.

4.1 Training Setup

As mentioned in the section 3, the architecture of the PointNet++ framework uses an *nx3* array as the input of features. However, it allows the user to increment the amount of columns of the array, otherwise known as the channels, corresponding to the amount of features used for training the model. As 3D Gaussian Splatting transforms every point present in the cloud into a Gaussian with extra parameters, these channels need to be increased to allow for the extra features that 3D Gaussian Splatting introduces. Training a classification model on just the position features wouldn't justify the use of 3D Gaussian Splatting. This research thus requires us to train multiple models using these different features obtained by the method.

To answer the research question proposed in the introduction, multiple PointNet++ models will be trained, each using a different amount of parameters present in the Gaussians. Specifically, I will train on four different configurations, each one adding a new set of features on top of the old ones. The goal is to find out which set of features has the most significant impact on the accuracy of the trained models. The performance of these models will be compared to the model trained on point clouds sampled directly from the mesh surfaces of the original ModelNet10 dataset, using Furthest Point Sampling, which can be considered as the baseline. As these point clouds do not make use of 3D Gaussians yet, the only features used for training this model are the *x, y* and *z* positions of the points. To ensure a fair comparison, both the proposed models and the baselines have been sampled using FPS, with every point cloud containing exactly 8192 points.

4.2 Classification Results

As previously mentioned, classification results will be compared to a baseline, which can be found in table 2. This table shows the performance of training a PointNet++ model on point clouds directly sampled from the surfaces of the ModelNet10 meshes as training data, using Furthest Point Sampling. As these point clouds have no notable features other than the *x, y* and *z* positions of the points, the presented accuracy can be considered as a definite baseline.

All of the training done in this research is using 200 epochs and a learning rate of 0.001. These numbers are the default values for PointNet++ training, and no significant changes in the results seem to appear when increasing the amount of epochs or decreasing the learning rate.

Network	Features	Accuracy
PointNet++	Positions	0.952

Table 2: Classification results of a PointNet++ model trained on points sampled directly from the ModelNet10 meshes. These results can be considered as the baseline for this research.

Network	Features	Accuracy
PointNet++	Positions	0.866
PointNet++	+ Scale & Rotation	0.887
PointNet++	+ Opacity	0.890
PointNet++	+ Sp. Harmonics	0.889

Table 3: Classification results of the PointNet++ trained on different configurations of 3D Gaussian Splatted ModelNet10 data

Looking at the results in table 3, one can immediately observe that all of the results from applying PointNet++ classification on 3D Gaussians are significantly lower than that of training on points sampled from the surfaces of the meshes. The reason for this is that the baseline is trained using points sampled directly from the surface of the object mesh, which can be considered as the ground truth. In contrast, the Gaussians obtained from 3D Gaussian Splatting are an approximation of the actual scene. As expected, classification of an approximation will almost never surpass the accuracy of training on the ground truth, sampled or not. Moving on to the performance differences between the different configurations, there is a difference of 0.024 in accuracy between the best and worst performing configurations, the best performing configuration using positions, rotations, scales and opacity as features. The significant deviation between the classification results of the first and second configuration in the table seems to suggest that the scale and rotation of the 3D Gaussians add the most amount of information regarding the general shape of the object. As opacity is eventually used during Gaussian Splatting to fine-tune the rendering, the model seems to be able to use this information for slightly improving the accuracy. Even more remarkable, adding 45 more channels for spherical harmonics as features for training seems to have no effect on the classification accuracy, and even decreases it by 0.001. All models in the dataset have a similar texture applied to them, and the light sources present when capturing the different views are positioned in the exact same location, which means that visibility-awareness does not seem to have as much effect as it would in real-world scenarios. However, the training time is

[3] https://github.com/charlesq34/pointnet

[4] https://github.com/yanx27/Pointnet_Pointnet2_pytorch

significantly longer when using these extra 45 features, implying that the best configuration for our particular dataset seems to be the use of positions, scale, rotation and opacity as features. Further research using non-synthetic data captured in the real world, can be done to determine the impact of spherical harmonics on classification results.

5 Responsible Research

While ethical and responsible aspects of this research may not immediately seem obvious to the reader, there are still certain elements present worth reporting, mainly regarding the data used for this research and its experiments.

5.1 Ethical Research

Ethical aspects of research arise fairly often when Artificial Intelligence is involved. In this research, I have been training multiple deep learning models with the goal of classifying common household objects. The data used for this is completely synthetically generated, which safely eliminates the need for real-world subjects. Keeping in mind that the data used fully consists of non-living objects, privacy and security norms and concerns do not play a significant role in this research. Nonetheless, it is important to remember that further research on this topic may involve actual real-world subjects and scenes, in which case these ethical aspects should be actively taken into consideration and respected. Researchers on this topic must ensure informed consent from all participants, maintain data confidentiality, and implement robust security measures to protect sensitive information. The ethical deployment of machine- and deep-learning technologies in real-world scenarios demands consideration of potential biases in the data and algorithms.

5.2 Data Ethics

While the details of ModelNet10 [11] have already been thoroughly explained, the chosen dataset for this research, in section 3, I have not yet examined the ethical aspects regarding its contents, or data in general, which also apply here. The most prominent aspect considered here is the reusability of the data, as stated by the FAIR principles[5]. These principles provide guidelines for achieving optimal reuse possibilities of data. First of all, the original dataset is easily Findable, as the Princeton ModelNet10 dataset is used by a wide range of similar research, mostly on the classification of 3D objects. In section 3 I explained how this dataset was further processed by applying 3D Gaussian Splatting to every single model. This process was actually performed by another student in the same research project group: Andrei Simionescu, who subsequently uploaded this dataset, around 300GB, to an external data repository[6]. The next principle, Accessibility, is therefore also satisfied. The original ModelNet10 is publicly available for download, with no authentication required whatsoever. Similarly, the 3D Gaussian Splatted dataset is available on the aforementioned external repository. Interoperability of the data, the next principle, is achieved by the ability to be integrated with other data. All the data used in this research is either available in .off or .ply models, both considered as universal 3D model file formats, able to be imported and utilised in most, if not all available 3D modelling, or point cloud manipulation software. The final principle, Reusability, is satisfied with the inclusion of relevant metadata.

[5]https://www.go-fair.org/fair-principles/

[6]https://doi.org/10.4121/3eabd3f5-d814-48be-bbff-b440f2d48a2b

5.3 Responsible Integrity

To achieve optimal reproducibility for this research, code and data used during the experiments are fully provided. On top of that, section 3 has extensively explained the methodology used during the experiments, along with the dataset used to generate synthetic input for the algorithms. This allows for full reproducibility of the experiments and subsequent validation of the results, reinforcing the integrity of my research. All of the code used during this research can be found here:
https://gitlab.ewi.tudelft.nl/cse3000/2023-2024-q4/Zhang/dvandale-Multi-view-image-recognition-through-3D-Gaussian/-/tree/main/.

6 Discussion

This research is based on the premise that 3D Gaussian Splatting might positively impact PointNet classification due to the increased quality of scenes. The highly realistic scenes produced by 3D Gaussian Splatting create the assumption that it should subsequently improve classification performance. This is particularly relevant because the input is not based on meshes but on point clouds directly obtained using Structure from Motion (SfM) techniques. These point clouds preserve permutation invariance, unlike point clouds obtained from sampling meshes. Our baseline uses Furthest Point Sampling on both the ModelNet10 meshes and the point clouds obtained from 3D Gaussian Splatting, which should have a less negative effect on the 3D Gaussian splatted point clouds compared to the original mesh point clouds. As presented in section 4.2, the baseline training, which is based on the original ModelNet10 3D CAD model meshes and subsequently sampling a set of points from the surface of these meshes, results in a staggering 0.952 classification accuracy on the test data. In contrast, training on 3D Gaussian splatted data results in a performance decrease to between 0.860 and 0.890 test instance accuracy, implying that 3D Gaussian Splatting affects PointNet classification in a negative manner. However, It is crucial to keep in mind that this ground truth data is usually inaccessible in real-world scenarios. 3D Gaussian Splatting is based on approximations. The input data for the process is a series of views of the scene or object, which can usually be obtained ”on-the-go.” The average phone camera can be considered the bare minimum for obtaining the required data for applying 3D Gaussian Splatting. A dataset consisting of actual meshes on the other hand is significantly more difficult to obtain when looking at real-world scenes, especially if it perfectly represents the ground truth. It is therefore crucial that not all focus should be placed on these results, but also on the context surrounding the obtaining of the data.

7 Conclusion

In this research, I explored deep learning possibilities created by the development of 3D Gaussian Splatting. Kerbl *et al.* proposed the use of differentiable volumetric kernels, named Gaussians, for representing a scene, allowing for gradient-based optimization. This new form of volumetric scene representation gave rise to new opportunistic challenges, including the possibility of applying well-known classification techniques on 3D Gaussians. In the end, I proposed the use of the PointNet(++) architecture to classify common household objects represented using 3D Gaussian Splatting, to find out whether the use of Gaussians could improve upon classification performance. Specifically, I performed multiple model training experiments using different configurations of features, including positions, scale & rotation, opacity and spherical harmonics, created and optimized using 3D Gaussian Splatting. The results are then compared to a baseline, consisting of a model trained on points sampled from the surface of the original meshes, where no 3D Gaussian Splatting has been applied. The final results seem to suggest that the use of Gaussians for classification do not directly improve performance, with accuracies hovering between 0.860 and 0.890, with the best performing configuration using positions, scales, rotations and opacity as features. These results are, however, significantly lower than the baseline of 0.952. 3D Gaussian Splatting approximates a scene with differentiable volumetric elements, meaning that it is unlikely to perform better than applying the same classification process on the ground truth of the scene, which samples points directly from the meshes surface. However, the Gaussian classification still achieves results comparable to state-of-the-art trained models, suggesting that using this new representation is not completely unthinkable, especially considering the fact that no initial mesh is required. The results are completely based on synthetically generated views, using methods that can be applied in real-world scenarios as well. From these results it can be safely concluded that 3D object classification using 3D Gaussian Splatting is definitely possible, but it does not achieve optimal performance when compared to non-approximated scenes.

In this research, I managed to find an answer to my research question. However, there are still opportunities available for further research. This research mainly used synthetically generated data, obtained from publicly available 3D Models. To further prove that 3D Gaussian Splatting is a significant advancement in the field of Computer Graphics, real-world data could be used instead. While this would require significantly more time to obtain compared to generating synthetic data, the results would be very valuable for determining the overall utility of 3D Gaussian Splatting.

References

[1] Liang Zhou, Mengjie Fan, Charles Hansen, Chris Johnson, and Daniel Weiskopf. A review of three-dimensional medical image visualization. *Health Data Science*, 2022:1–19, 04 2022.

[2] Lee Westover. Footprint evaluation for volume rendering. *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, 1990.

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[4] 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, 2020.

[5] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3d gaussians. *arXiv preprint arXiv:2312.00860*, 2023.

[6] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics*, 38(4):1–12, July 2019.

[7] Cheng Wang, Ming Cheng, Ferdous Sohel, Mohammed Bennamoun, and Jonathan Li. Normalnet: A voxel-based cnn for 3d object classification and retrieval. *Neurocomputing*, 323:139–147, 2019.

[8] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition, 2015.

[9] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[10] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[11] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes, 2015.

[12] Lorenzo Luciano and A. Ben Hamza. Deep similarity network fusion for 3d shape classification. *The Visual Computer*, 35, 06 2019.

[13] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.