# Bridging the world of 2D and 3D Computer Vision

## Self-Supervised Cross-modality Feature Learning through 3D Gaussian Splatting

**Andrei Simionescu**
**Supervisor: Dr. Xucong Zhang**

EEMCS, Delft University of Technology, The Netherlands

June 23, 2024

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: Andrei Simionescu
Final project course: CSE3000 Research Project
Thesis committee: Dr. Xucong Zhang, Dr. Michael Weinmann

## Abstract

*Current robotic perception systems utilize a variety of sensors to estimate and understand a robot's surroundings. This paper focuses on a novel data representation technique that makes use of a recent scene reconstruction algorithm, known as 3D Gaussian Splatting, to explicitly represent and reason about an environment using only a sparse set of camera views as input. To achieve this, I generate and analyze the first cross-modal dataset consisting of 3D Gaussians and views taken around ten household objects. I introduce the resulting 3D Gaussians and images to a self-supervised feature learning network, that learns robust 2D and 3D embedding representations, by optimizing for the cross-view and cross-modality correspondence pretext tasks. I experiment with several 3D Gaussian features as input to the model and two point sub-network backbones, and report results on the two pretext tasks. The learned features are subsequently fine-tuned for the 2D and 3D shape recognition tasks. Moreover, by leveraging the fast scene reconstruction capabilities of the algorithm, I propose the use of rendered views as a visual memory aid to support downstream robotic tasks. The proposed networks achieve comparable results to state-of-the-art methods for point and image processing. The code associated to this paper is available at* https://github.com/SimiOy/Self-Supervised-Learning-for-3DGS.


## 1. Introduction

Robotic perception encompasses the study of how robots perceive, understand, and reason about their surrounding environment [9]. The representation of the observed environment through the use of sensors (*e.g*. RGBD, LiDAR, Sonar *etc*.) is arguably the most important component of a perception system, as it dictates the limits of the reasoning capabilities of a robot. Sünderhauf *et al*. [15] present several challenges in robotics that require the reasoning about object and scene geometry *e.g*. 3D object structure estimation, pose estimation or object-based simultaneous localization and mapping (SLAM) [3]. The success of such tasks is greatly influenced by the robot's internal representation of the environment.

This paper introduces a novel method for representing the perceived environment through the use of 3D Gaussian Splatting (3DGS) [6], a scene reconstruction technique, and proves that this technique can be used to learn cross-modality features in a self-supervised manner. 3DGS provides an explicit representation of the scene that allows a robot to visualize and *remember* what parts of the environment look like, enabling unprecedented reasoning capabilities *e.g*. retrieving a cup is more likely to be successful if the agent knows a priori what a cup looks like (shape recognition), which room is the kitchen (semantic segmentation), and how to get there (path planning).

The proposed self-supervised framework follows closely the work of Jing *et al*. [5], which introduce an optimization technique for multi-modal feature learning for point clouds and images. The authors train two sub-networks to recognise whether a set of points and a view belong to the same object (cross-modality correspondence), and whether two images were taken around the same subject (cross-view correspondence). I follow a similar approach (Fig. 1), but make some adjustments to the point sub-network to allow for the processing of 3D Gaussians. Using the learned network weights and feature embeddings, I optimize for the shape recognition task, and then compare these results with state-of-the-art (SOTA) benchmarks on both image and point processing. Findings suggest that the proposed data representation is capable of achieving comparable performance on the evaluated dataset, without requiring an initial mesh (*i.e*. using only camera views), while also doubling as an efficient *memory module* for an autonomous agent. To summarize, the main contributions of this research are as follows:

- The first multi-modal data points consisting of 3D Gaussians and distinct views taken around an object is generated and an extensive analysis is performed on the dataset.
- The introduction of 3D Gaussians to a self-supervised cross-modal deep-neural network for Multi-View Learning (MVL), and comparison to SOTA models in this field. Experiments are performed for different sampling techniques, model backbones, and input features considered.
- The proposal of a new task, *Memory-based Vision*, which facilitates the rendering and reasoning about a previously observed environment or object. The encoder sub-networks are evaluated for shape recognition using both views obtained at inference time, and *remembered* views (*i.e*. using reconstructed images by the 3DGS algorithm).

## 2. Related Work

**Multi-View Learning:** MVL is concerned with the problem of learning a representative common feature space from multi-modal data [18]. Although the multi-modal information captured from multiple sources often represents complementary views of the same data (*e.g*. different views around the same object, mesh and render of an object, text and audio *etc*.), this information has been used in the past to enhance the accuracy of the task or the explainability of the underlying method. One such example are Multi-View CNNs (MVCNN) [14], introduced by Su *et al*., which combine multiple views of a single 3D shape into a single shape descriptor. The authors proposed a multi-view image-based CNN architecture that pools together the image features from distinct views into a single CNN which predicts the class labels. Their findings suggest that the multi-modal

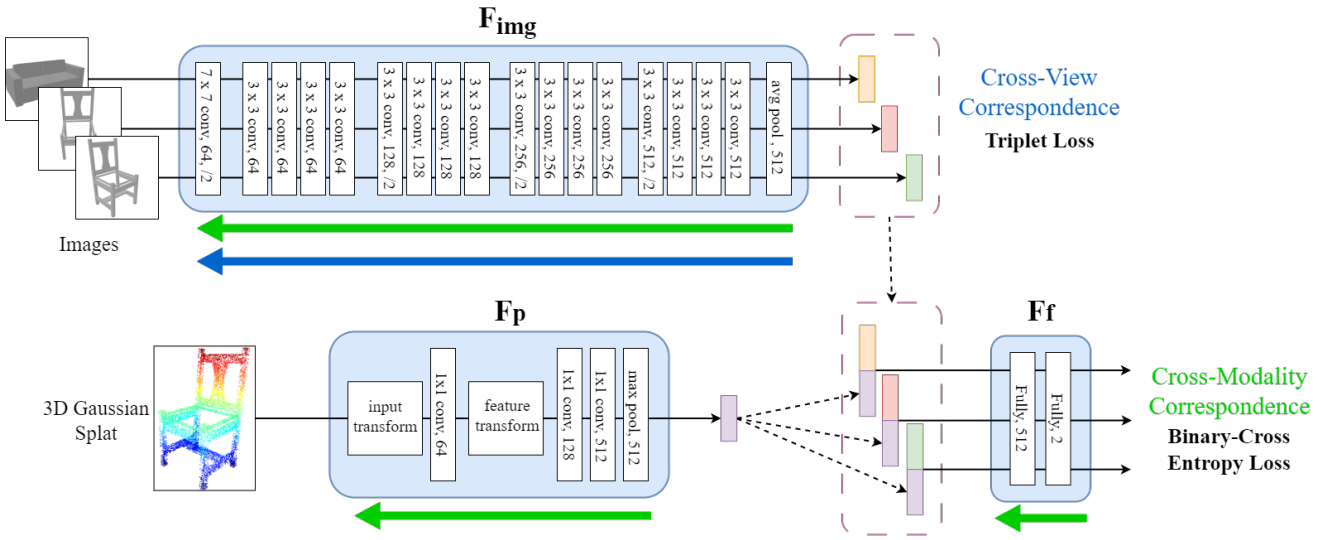


Figure 1. The proposed self-supervised network architecture (figure adapted from Jing *et al*. [5]), featuring three sub-networks. The first two are feature extractors for the different modalities: a ResNet-18 ($F_{img}$) that takes the different views and trains for detecting when two views belong to the same object (*i.e*. cross-view correspondence, blue arrow), and a PointNet model ($F_p$) model that processes and embeds the unordered Gaussian point cloud data into a global feature vector. The embeddings of the different modalities are then concatanated and used by the last sub-network ($F_f$) that jointly optimizes all three sub-networks for predicting whether the two cross-modal inputs belong to the same object (*i.e*. cross-modality correspondence, green arrow).

information encoded in different views achieves better performance than existing 3D shape descriptors at the time.

**Scene Reconstruction Techniques:** Recently, a lot of advancements have been made in MVL aimed to solve the problem of scene reconstruction *i.e*. novel-view synthesis of an observed environment from a sparse set of input images. Mildenhall *et al*. [8] propose a Neural Radiance Field (NeRF) scene representation algorithm that uses a deep neural network to approximate a view-dependent, continuous volume of space. However, since the scene is represented implicitly, the rendering process is very slow, and it's difficult to modify it after the optimization process is completed.

Explicit scene representation has also seen an increase in popularity lately. Kerbl *et al*. [6] introduce a fast differentiable rendering and optimization algorithm (3DGS) that uses a set of points, called 3D Gaussians, to approximate the environment. The optimization process projects the 3D Gaussians into 2D, by splatting them into the camera frustrum. The generated image is then compared to the original view and a reconstruction loss is calculated, which is used to optimize the 3D Gaussians. In particular, these points have the following properties that they optimize for: position $\in \mathcal{R}^3$, rotation $\in \mathcal{R}^4$, scale $\in \mathcal{R}^3$, opacity $\in \mathcal{R}$, and spherical harmonics coefficients $\in \mathcal{R}^{44}$ (used to represent the view-dependent coloring of a Gaussian). The rendering process is highly parallelizable and produces high-fidelity reconstructions, while also allowing for the manipulation of the resulting Gaussian point cloud, *e.g*. scaling or adding/removing a subset of points, which makes it an incredible multi-modal data point. To my knowledge, nobody has yet attempted to directly use the 3D Gaussians in a deep-neural network for Multi-View Learning.

**3D Point Cloud Processing:** To process a Gaussian point cloud, there is a need of a network that accepts an unordered set of vectors as input and embeds these into a feature space. The revolutionary work of Qi *et al*. [10], the PointNet model, is a simple, yet highly efficient solution that directly processes a set of points. The network consumes a subset of points sampled at the surface of a 3D mesh, and predicts an affine transformation matrix of the input space, where these points are invariant to translation, rotation or scale. The transformed input is then fed through a series of multi-layer perceptrons and a feature transformation layer to obtain global and per-point feature vectors of the entire point cloud. The network achieves remarkable performances on several computer vision tasks *e.g*. shape classification, part segmentation and scene semantic segmentation.

However, Qi *et al*. [11] recognizes that the PointNet algorithm is limited by its ability to capture only global or per-point features, and proposes a neighbourhood-based improvement that exploits the underlying metric space. PointNet++ works by hierarchically partitioning the points and applying the PointNet abstractions for each neighbourhood resolution. Similar to CNNs, the model learns features at varying abstraction levels, yet the grouping of samples is not given by a kernel size (an image can be thought of hav-

ing a uniform density of pixels), but by a metric distance (a point cloud has a varying density *e.g*. LiDAR).

**3D Self-supervised Feature Learning:** An obvious challenge of this research is the lack of labeled data-points, since it proposes a new form of data representation (3D Gaussians) to perform Computer Vision tasks in. To mitigate this issue, I suggest switching the learning paradigm to self-supervised learning (SSL), where no labeled data is required to train a model. As opposed to completely unsupervised learning which discovers patterns into the data without any explicit guidance, SSL optimizes for a set of supervisory signals (*i.e*. pretext tasks) that act as feedback during the training process. Jing *et al*. [5] propose a self-supervised cross-modal feature learning algorithm that is trained for the *cross-view* and the *cross-modality correspondence* pretext tasks. The network architecture consists of two encoder sub-networks, one for the point and one for the image modalities respectively. The image sub-network is trained to recognize whether two views were taken around the same object (cross-view correspondence), while both sub-networks are jointly optimized to distinguish whether a point cloud and an image belong to the same object (cross-modality correspondence). These two pretext tasks are used as supervision signals for the network to learn robust cross-modality embeddings, which are later fine-tuned for specific tasks *e.g*. shape recognition, retrieval and segmentation.

## 3. Deep Learning on 3D Gaussian Splats

The proposed method is working on a new type of data, and thus, the process of constructing and validating the dataset is essential to the success of the model. This section introduces the first cross-modal dataset made from multi-view images taken around an object, and the resulting 3D Gaussian point clouds. Then, I present the self-supervised network architecture, discuss its parameterization and training.

### 3.1. Data Generation

To train the model, the two types of data-points are generated from 3D objects. First, each 3D object is loaded in the scene, a texture is applied and its scale is normalized, such that all the objects have the same dimensions. Additionally, three light sources facing the object are placed around the scene to ensure a consistent lighting setup from all angles. Similar to the works of Mildenhall *et al*. [8], a virtual camera is pivoted around the object to capture $V$ views, and during its rotation, its vertical position is interpolated between two points, leading to a spiralling motion around the object of interest. These $V$ frames, together with the camera matrix transformations are used as input for the 3D Gaussian Splatting algorithm [6]. Since the complexity of the scene is low (only one object is placed at a time), the reconstruction algorithm only runs for a fixed 15000 iterations for each object. In constructing the dataset, there was no significant improvement observed to using the Structure-from-Motion (SfM) [13] initialization for the Gaussians over the random one. Thus, I opted for using the latter, as it proved more time efficient as well. The models are loaded from the ModelNet10 dataset [17], and the official train/test split is kept. Figure 2 showcases an example of a generated cross-modal data point featuring a bathtub. See supplementary (Sec. 8) for a detailed analysis of the dataset.

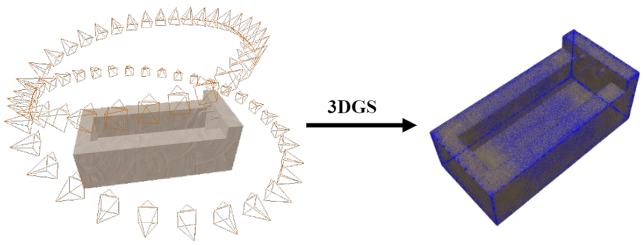


Figure 2. The camera trajectory (Left) which takes 64 800x800 images around the subject in a spiralling motion, and the resulting 3D Gaussian splat render and point cloud overlayed (Right). The 64 distinct views taken around the object, and the resulting Gaussian point cloud represent the cross-modal data-point. The bathtub mesh is up-scaled, and the point cloud (in blue) is down-scaled for visualization purposes.

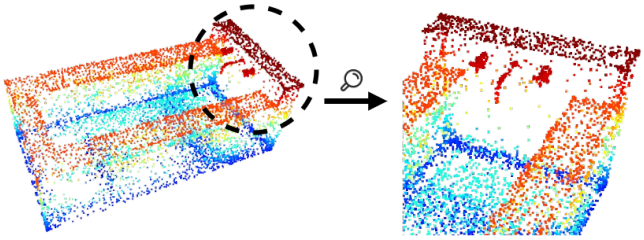

Figure 3. Visualization of the 3D Gaussian point cloud positions of a bathtub model (Left), and zoomed in on the reconstruction of its faucet and wall (Right). The faucet is an example where 3DGS uses more Gaussians to represent complex geometries.

Since the PointNet model [10] expects an unordered set of $N$ points, and each 3DGS optimization results in a different number of Gaussians, the resulting point clouds are also filtered using an opacity threshold, and then $N$ points are sub-sampled. Two distinct approaches for the sub-sampling of points are considered:

- *Uniform sampling:* Since the scene reconstruction uses more Gaussians to represent intricate areas, a uniform sub-sampling will also preserve a higher concentration of points in more distinctive areas of an object. Conversely, flat surfaces, where Gaussians can more easily be stretched to approximate the object's face, require less descriptive features (and thus less points are used). Figure 3 shows a highly concentrated area of Gaussians (the faucet of the bathtub), and a less concentrated surface (the wall of the bathtub). Section 4.1 analyses whether a

higher concentration of distinctive features aids the model in classifying the point-cloud better.

- *Farthest Point Sampling (FPS):* FPS is a greedy algorithm that iteratively selects the furthest away point from the current selection. The resulting subset better preserves the geometry and shape of the object and the features learned could prove more stable for fine-tuning. Figure 4 illustrates the sampling difference between the two proposed methods for the same 3D Gaussian point cloud.

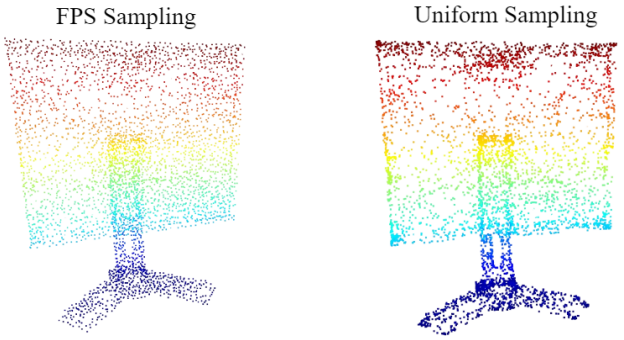


Figure 4. Visualization of the 3D Gaussian point cloud positions of a monitor model under the two proposed sub-sampling techniques (for 4096 points). The Furthest Point Sampling (Left) preserves the geometric structure of the object better, while the Uniform Sampling (Right) preserves a higher concentration of Gaussians in more intricate areas *e.g.* the stand of the monitor.

**Notation:** To ensure consistency, a similar notation to the works of Jing *et al.* [5] is used throughout this paper: Let $\mathcal{D} = \{sample^{(1)}, sample^{(2)}..., sample^{(N)}\}$ denote the training data of size $N$. Each $sample^{(i)} = \{g^{(i)}, v_1^{(i)}, v_2^{(i)}, v_3^{(i)}, y_1^{(i)}, y_2^{(i)}, y_3^{(i)}\}$, where $v_1^{(i)}, v_2^{(i)}$ and $g^{(i)}$ represent two distinct random views taken around the same object and the sampled Gaussian point cloud, and $v_3^{(i)}$ is a rendered image from a different mesh. Note that the input images used for training the model are the same set of images used for optimizing the 3D Gaussian splats, and novel views are used only during testing. The labels $y_j^{(i)} \in \{0, 1\}$ denote whether the Gaussian point cloud $g^{(i)}$ and the rendered view $v_j^{(i)}$ belong from the same object, where 1 indicates the same object, and 0 a different one. In practice, since the first two views $v_1^{(i)}, v_2^{(i)}$ are taken around the same object, and the last one is taken from a different one, the labels become $y^{(i)} = \{1, 1, 0\}$ for each $sample^{(i)}$.

### 3.2. Network Architecture

Following the works of Jing *et al.* [5], the proposed network architecture uses two pretext tasks for self-supervised learning: *cross-view correspondence* and *cross-modality correspondence*. As presented in Figure 1, to optimize for the cross-view task, an image sub-network $F_{img}$ is employed to extract semantic features for each view. For the cross-modality task, an additional point sub-network $F_p$ is used to extract global information about the Gaussian point-cloud. The output of the image sub-network $F_{img}$ is concatenated to the output of the point sub-network $F_p$, and fed though a block of two fully connected layers $F_f$. The output of the final block $F_f$ is a binary classification value that is optimized for the cross-modal similarities $y^{(i)}$.

For the image sub-network $F_{img}$, a ResNet18 [4] is used, where the last fully-connected layer is removed such that the output becomes a 512-dimensional vector. For the point-based sub-network $F_p$, I considered the PointNet [10] and the improved PointNet++ [11] architectures, where the classification and segmentation heads are removed. To ensure that the sub-networks can also capture the additional features (scale, rotation, and spherical harmonics) that each Gaussian has, both models were adapted to work with any number of input dimensions per point. Additionally, after the last max pooling layer, both point sub-networks are modified to output 512-dimensional embeddings instead of the original 1024. Finally, the output of the last 512-dimensional layers from both the image sub-network $F_{img}$ and the point-based sub-network $F_p$ are concatenated into a 1024-dimensional vector and fed through the final block $F_f$, that aims to predict whether the two input modalities belong to the same object.

### 3.3. Model Parameterization

In the self-supervised learning framework proposed by Jing *et al.* [5], two types of supervision signals are used during training to help the model learn whether two distinct views belong to the same object, and whether a rendered image and a point cloud originated from the same mesh. This paper uses the same tasks and shows that a similar model parametrization can be extended to a Gaussian cloud obtained from input images, instead of point clouds sampled at the mesh level.

**Cross-view correspondence:** The network encounters images of the same object from different angles, and it must learn similar semantic features for these views. Thus, this task is modelled as a metric learning task, and uses the triplet loss [12] to minimize the distances between positive pairs (*i.e.* views belonging to the same object), and maximize the distance between negative ones (*i.e.* views belonging from different objects). The loss is then given by:

$$\begin{aligned} L_{triplet} = max(&\|F_{img}(v_1^{(i)}) - F_{img}(v_2^{(i)})\|^2 \\ &- \|F_{img}(v_1^{(i)}) - F_{img}(v_3^{(i)})\|^2 + \alpha, 0) \end{aligned} \tag{1}$$

where $v_1^{(i)}$ is the anchor used to compare against the positive $v_2^{(i)}$ and negative $v_3^{(i)}$ samples, and $\alpha$ is the margin hyperparameter that controls the minimum distance between the

anchor and the negative samples. The effect of this loss is that it brings closer embeddings belonging to the same object, and pushes away embeddings from different ones.

**Cross-modality correspondence:** To learn cross-modal features the network must be able to distinguish whether an input view and a sampled Gaussian point cloud belong to the same original object. Following the works of Jing *et al*. [5], this task is modelled as a binary classification task, optimized using the Binary Cross-Entropy loss. As opposed to the previous loss which was used to train only the image network, this loss jointly trains all three sub-networks:

$$L_{cross} = -\sum_{j=1}^{3} \Big[ y_j^{(i)} \log \Big( F_f(F_{img}(v_j^{(i)}), F_p(g^{(i)})) \Big) + (1 - y_j^{(i)}) \log \Big( 1 - F_f(F_{img}(v_j^{(i)}), F_p(g^{(i)})) \Big) \Big] . \quad (2)$$

The cross-modal pretext task is much more difficult for the network to learn then the cross-view one, so an additional regularization term $\beta$ is used to balance the influence of the two losses. Finally, the joint loss is given by:

$$L_{self} = L_{triplet} + \beta L_{cross} \quad \text{,where} \quad \beta > 1. \quad (3)$$

### 3.4. Optimization

The network is optimized using the Stochastic Gradient Descent (SGD) optimizer with an initial learning rate of $0.001$, momentum of $0.9$, and weight decay of $0.0005$. Each model is trained for $100$ epochs, with $\beta = 3$, and every $40$ epochs the learning rate is decreased by $90\%$. During training, the images are augmented by randomly cropping and randomly flipping with a $50\%$ probability on the horizontal axis; the Gaussian clouds do not go through any augmentation process because the 3DGS algorithm produces indeterministic results *i.e*. the high variation of the Gaussians acts as regularization for the deep-neural network. Figure 5 showcases the CM and CV losses and success metrics during training.

The metrics are computed on the test set, using a single pass *i.e*. the number of samples in the data loader is equal to the number of test models, and might show variation across epochs depending on which views were sampled for each object. Nevertheless, the network is clearly learning to optimize for both supervision signals, as the losses gradually decrease closer to 0, and the measured success for each task increases. Note that the model achieves remarkable performance even after only 25 epochs, but letting the model train for longer decreases the variation in reported performance.

## 4. Experiments

This section presents the experiments conducted to assess the performance of the self-supervised network. The model

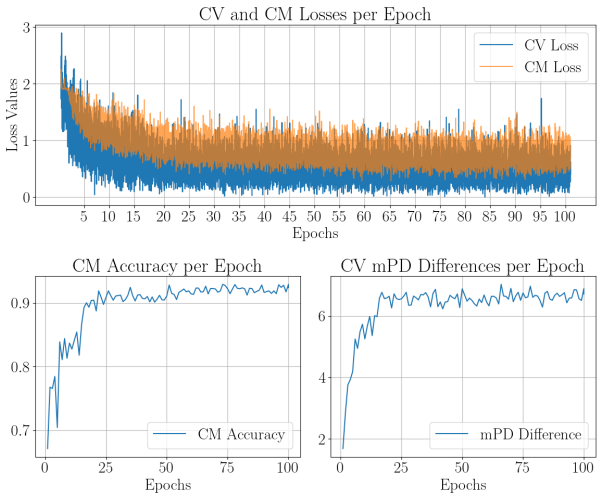


Figure 5. Training losses (Top) and pretext task metrics (Bottom) using the *PointNet* point sub-network backbone. The model was trained using only the point information, uniformly sampled. The CM accuracy (Left) and the CV mean pair distance (mPD) (Right) are computed on a single pass on the test set. For visualization purposes, in the CV task, only the difference between the negative mPD and positive mPD is reported.

is quantitatively and qualitatively evaluated on the two pretext tasks it was trained on. Then it's fine-tuned for the memory-based and real-time inference recognition tasks.

### 4.1. Evaluation of learned features

The network is trained under 12 different configurations with varying number of input features, sampling methods, and backbone architectures. More specifically, I considered training the model using only the Gaussian positions, then adding the scale & rotation, and finally including also the spherical harmonics. For every feature, I also investigate the effect of using uniform sampling as opposed to FPS. As anticipated in Sec. 3.2, the point sub-network can either be the PointNet or the PointNet++ model. For the CV task, I report the mean pair distance (mPD) between positive and negative sample embeddings. Since the samples drawn from $\mathcal{D}$ contain random views for each object, the evaluation is computed on a test size 10 times larger than the test split. This aims to minimize the variance in the reported results, since some drawn samples are more difficult for the model to classify than others (*e.g*. two desks vs. a monitor and a desk). Table 1 and 2 showcase the results obtained for the CM and CV correspondence pretext tasks, under all configurations.

The results show that the point sub-network performs better with FPS than with uniform sampling, indicating that the model is more reliant on the geometrical structure of the object than on the distinctive features. A reasonable ex-

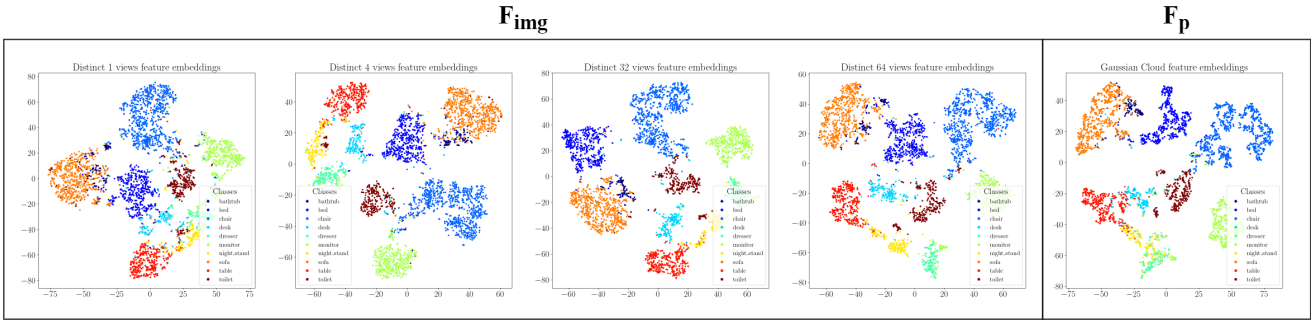


Figure 6. TSNE [16] visualization of the learned features on the image and point sub-networks. All images belong to a network with a *PointNet* backbone, using only the 3D Gaussian's positions, sampled with FPS. For the image sub-network, multiple views are considered. As the number of views increases, the class clusters are more clearly defined, and approach the ones formed in the point sub-network.

| CM accuracy (%) | PointNet | | PointNet++ | |
|---|---|---|---|---|
| | FPS | Unif | FPS | Unif |
| Positions | 0.941 | 0.924 | **0.952** | 0.931 |
| + Scale & Rotation | 0.938 | 0.942 | 0.938 | 0.906 |
| + Spherical Harmonics | 0.943 | 0.943 | 0.939 | 0.918 |

Table 1. Performance comparison for the cross-modality pretext task for *PointNet* and *PointNet++* as point sub-network backbones, for the two sampling techniques (Uniform and Furthest Point Sampling), and varying number of features. + indicates the accumulation of the features for each row *e.g.*, for sh. harmonics the model uses position, scale & rotation and spherical harmonics. The network achieves overall better performance with FPS.

| CV mPD | FPS | | Unif | |
|---|---|---|---|---|
| | Pos | Neg | Pos | Neg |
| Positions | **5.34** | **13.57** | 4.94 | 11.65 |
| + Scale & Rotation | 4.94 | 12.07 | 5.05 | 12.6 |
| + Spherical Harmonics | 5.12 | 12.91 | 5.1 | 12.79 |

Table 2. Performance comparison for the cross-view pretext task, for the two sampling techniques (Uniform and Furthest Point Sampling), and varying number of features. + indicates the accumulation of the features for each row *e.g.*, for scale & rotation the model uses position and scale & rotation. The performance is measured in mean pair distance (mPD) *i.e.* the mean distance between positive and negative sample embeddings respectively. The network achieves the best performance when only the Gaussian positions are considered as input.

planation would be that not all objects in the dataset have complex geometries (as in in Fig. 3) that the network could optimize for, and a more uniform approximation of the underlying mesh leads to a better cross-modal accuracy.

Across the two backbones, there is a noticeable gain to using the improved PointNet++ model across all features and sampling techniques considered. Moreover, it seems that adding more information to the model decreases the reported performance in the CM and CV tasks, and the best model is obtained when just the point positions are used.

Additionally, I also evaluate qualitatively the performance of the learned features. To this end, I use T-distributed Stochastic Neighbor Embedding (TSNE) [16] to visualize the embeddings of both sub-networks. For the image model, each feature is max-pooled from the extracted features of the $v$ views considered. Figure 6 shows class clusters forming, meaning that the model has learned shape recognition and retrieval without any explicit signals. Note that the mistakes that the network makes are *geometrically correct e.g.* it sometimes confuses the table with the desk, since they have a similar shape and appearance. A comparison of the learned features using the two proposed sampling techniques and model backbones is available in the supplementary material (Sec. 9).

### 4.2. Applications

The performance for the multi-view 2D and 3D shape recognition tasks is evaluated by attaching a Support Vector Machine, with one class linear kernel, on the image and point cloud features extracted from the pre-trained $F_{img}$ and $F_p$ networks respectively. For the image network, when multiple views are available, each feature is max-pooled from the extracted features of the $v$ views considered. The recognition accuracy is reported on the generated dataset test-split when 1, 4, 32 or all 64 views are available.

**Memory-based Recognition:** Humans have incredible recall abilities that allow them to visualize and remember previously observed environments or objects *e.g.* babies develop object permanence at around 5 months old, when they realize that objects exist even though they can no longer sense them. Some robots lack such an ability and usually make decisions based on the information received through their sensors at inference time. The proposed 3D Gaussian cloud used as a data-modality for the networks is primarily

a novel-view synthesis technique that enables a robot to *remember* (visualize) what environments look like. I coin this novel task *Memory-based Vision i.e.* images are rendered from memory, instead of storing them on disk. It can be extended to other tasks such as object detection or segmentation, but for the purpose of this research, I focus on classification. The implications in robotics for such a family of tasks are numerous, ranging from enhanced navigation and path planning to increased human-agent collaboration.

**Real-time Recognition:** Direct perception through the use of onboard sensors (*e.g.* RGB cameras) is equally important for a robot to be able to make decisions on-the-fly. This is commonly referred to as 2D or 3D shape recognition in literature. For this task, the recognition accuracy is reported on the original views taken to construct the 3D Gaussian clouds, which were also used to train the models.

| Features | # Views | Accuracy (%) | |
|---|---|---|---|
| | | PointNet | PointNet++ |
| Positions | 1 | 0.86 | 0.86 |
| | 4 | 0.93 | 0.93 |
| | 32 | 0.95 | 0.95 |
| | 64 | 0.95 | **0.96** |
| | 64∗ | **0.96** | **0.96** |
| + Scale & Rotation | 1 | 0.87 | 0.84 |
| | 4 | 0.93 | 0.92 |
| | 32 | 0.95 | 0.95 |
| | 64 | **0.96** | 0.95 |
| | 64∗ | **0.96** | 0.95 |
| + Sh. Harmonics | 1 | 0.85 | 0.83 |
| | 4 | 0.93 | 0.91 |
| | 32 | **0.96** | 0.95 |
| | 64 | **0.96** | 0.95 |
| | 64∗ | **0.96** | 0.95 |

Table 3. Performance comparison for the 2D shape recognition accuracy (%) for the image sub-network $F_{img}$ under varying number of features for the *PointNet* and *PointNet++* point backbones, using FPS. $*$ indicates that the evaluation is performed on reconstructed (remembered) views. $+$ indicates the accumulation of the features for each set of rows *e.g.*, for sh. harmonics the model uses position, scale & rotation and spherical harmonics. Increasing the amount of features considered does not result in a better accuracy.

As shown in Table 3 and 4, the pre-trained image and point networks achieve very high accuracy on the 2D and 3D shape recognition tasks. When using the *PointNet* backbone, with just Gaussian positions, $F_p$ obtains $89\%$ accuracy, and when switching to the improved *PointNet++* backbone, the performance increases to $90\%$, indicating that the accuracy can be improved by scaling up the network. Similar to the findings reported in Sec. 4.1, the model's performance does not increase when more of the Gaussian's features are considered. Note that this could be in part caused by the way the dataset was constructed. The spherical harmonics have a small influence in this experiment, as all models in the dataset have the same texture applied and are lit in exactly the same way (and thus have similar view-dependent color). Similarly, the scale does not contribute significantly since all models have been resized to identical dimensions.

| Features | Accuracy (%) | |
|---|---|---|
| | PointNet | PointNet++ |
| Positions | 0.89 | **0.9** |
| + Scale & Rotation | 0.87 | 0.88 |
| + Sh. Harmonics | 0.88 | 0.87 |

Table 4. Performance comparison for the 3D shape recognition accuracy (%) for the point sub-network $F_p$ under varying number of features for the *PointNet* and *PointNet++* point backbones, using FPS. $+$ indicates the accumulation of the features for each row *e.g.*, for scale & rotation the model uses position and scale & rotation. The network achieves overall higher accuracy when the *PointNet++* backbone is used.

The image model outperforms the point one, obtaining $96\%$ recognition accuracy when 32 or all $64$ views are available. Increasing the number of views from one to all leads to a $10$-$12\%$ boost in performance (an observation which is constant across all combinations of features, and point backbone architectures considered). Moreover, using reconstructed images from the 3DGS algorithm does not impact the classification accuracy at all, suggesting that novel views could be used to augment the performance of downstream tasks. Note that given the low complexity of the considered scenes, the reconstruction loss is minimal, and thus has little influence on the quality of the rendered images.

## 5. Results

The performances of the proposed methods are evaluated against a set of point-based and image-based SOTA model baselines. The point sub-network $F_p$ is compared to the PointNet [10] and PointNet++ [11] models; and the image sub-network is compared to the MVCNN [14] model. To ensure a fair analysis between the type of networks, they each share a common ground: the image networks use a ResNet-18 as a backbone architecture, and are trained using all $64$ available views; the point networks share the same input feature dimensionality, and are trained on the position of the points/Gaussians, using $2048$ points sampled with FPS.

The networks are evaluated on the ModelNet10 dataset [17] using the official train-test split. The proposed methods are trained on the Gaussian-views cross-modality dataset in-

| Training Data | Modality | Network | Accuracy (%) |
|---|---|---|---|
| 100 % | Points | PointNet [10] | 0.93 |
| | | PointNet++ [11] | 0.95 |
| | Gaussians | $F_p*$ | 0.89 |
| | | $F_{p++}*$ | 0.9 |
| | Images | MVCNN [14] | **0.98** |
| | | $F_{img}*$ | 0.96 |
| | | $F_{img++}*$ | 0.96 |
| 50 % | Points | PointNet [10] | 0.92 |
| | | PointNet++ [11] | 0.93 |
| | Gaussians | $F_p*$ | 0.88 |
| | | $F_{p++}*$ | 0.9 |
| | Images | MVCNN [14] | **0.96** |
| | | $F_{img}*$ | 0.95 |
| | | $F_{img++}*$ | 0.95 |
| 10 % | Points | PointNet [10] | 0.9 |
| | | PointNet++ [11] | 0.91 |
| | Gaussians | $F_p*$ | 0.87 |
| | | $F_{p++}*$ | 0.89 |
| | Images | MVCNN [14] | 0.91 |
| | | $F_{img}*$ | **0.93** |
| | | $F_{img++}*$ | 0.92 |
| 1 % | Points | PointNet [10] | 0.6 |
| | | PointNet++ [11] | 0.75 |
| | Gaussians | $F_p*$ | 0.76 |
| | | $F_{p++}*$ | 0.75 |
| | Images | MVCNN [14] | 0.49 |
| | | $F_{img}*$ | **0.77** |
| | | $F_{img++}*$ | 0.76 |

Table 5. Classification accuracy comparison with SOTA models on ModelNet10 dataset [17], under different amounts of training data available. The proposed methods (marked with $*$) are trained using only the Gaussian positions, with FPS. $++$ indicates that the *PointNet*++ backbone was used during SSL and/or fine-tuning. The 3D Gaussian modality achieves comparable performance to the points sampled at the surface of the meshes.

troduced in Sec. 3.1, and the same images are used to train the MVCNN model as well. The point-based baselines are trained using the true point positions sampled from the mesh of each object in the dataset. Since the sampling of points uses the actual mesh of an object, these points are considered the ground truth, while the Gaussians represent only an approximation. I also evaluate the performance of the learned self-supervised features by reducing the amount of training data available for fine-tuning. Table 5 reports the recognition accuracy when $100\%$, $50\%$, $10\%$ and $1\%$ of the data is available to train the baselines or to fine-tune the proposed methods.

With the entire dataset available for training, the MVCNN [14] model achieves a remarkable $98\%$ recognition accuracy, closely followed by the two image-based proposed methods at $96\%$ accuracy. The point modality baselines obtain a higher performance ($+3$ to $5\%$) than the Gaussian-based models. Although different learning strategies were employed, most of the reflected difference is caused by how the point clouds were constructed. Since the *PointNet* models use the ground-truth coordinates sampled at the surface of the mesh, these obtain better recognition scores than an approximate representation thereof.

When reducing the amount of available training data by half, the self-supervised models show only a $1$ to $2\%$ decrease in reported performance, and when considering only $10\%$ of the data, the accuracy further drops again by another $1$ to $2\%$. This suggests that the proposed methods have learned robust cross-modality features that enables them to overcome the challenge of sparsely labeled datasets. The advantage of the self-supervised methods becomes evident when less than $10\%$ data is available for training, as the proposed networks outperform the SOTA supervised classifiers for both the image and point modalities. With just $1\%$ data available, the image and point networks obtain scores of $77\%$ and $76\%$ recognition accuracy respectively.

## 6. Conclusion & Future Work

In this work, I introduced a novel data-modality for jointly learning 2D and 3D features requiring only a set of sparse camera views as input. The 3D Gaussians used serve both as an explicit representation of the observed environment, and as input to the proposed self-supervised networks. Initial experimental results on the considered dataset indicate that Gaussian-based models perform better when considering only the point positions and when the point cloud is sampled with FPS. Moreover, I introduce a new task, *Memory-based Vision*, which facilitates lossless 2D reasoning about a previously observed scene or object, via novel-view renderings of the obtained 3D Gaussian point cloud. These findings are subjected to the limits of the generated dataset. The contributions show a promising direction of research that could enhance current robotic perception systems.

This paper has presented a comparison between two different point-based modalities: points sampled at the surface of a mesh, and 3D Gaussians. The former is guaranteed to obtain a better performance since it represents the ground truth, yet it is often inaccessible in real-world applications. A valuable study would be a comprehensive comparison between current data representation techniques in perception systems (*e.g.* LiDAR, Sonar, RGBD), which could advance the practical applications of 3D Gaussians in this field.

## 7. Responsible Research

This section reflects on the reproducibility and integrity of the research conducted. The proposed method does not utilize any sensitive information and the construction of the dataset is completely unbiased, and thus there are no moral ethical concerns associated with this research.

### 7.1. Reproducibility

**Dataset generation:** The cross-modal dataset was generated and published following the *FAIR* data management principles. It can be found and publicly accessed through the 4TU website [1]. To ensure interoperability, the dataset is formatted in a widely-used format: *.png* for images and *.ply* for point cloud files. For rendering the images, I used Blender 2.8 [1] with the fast render engine Eevee, and the *.off* file loading add-on which can be downloaded from this GitHub repository [2]. Moreover, reproducibility is ensured by including in the same repository [1]: the Blender scene, the texture utilized [3], and the Python script used to pivot the camera around the objects and to render the views. As presented in Sec. 3.1, the main logic of the script is an extension of the NeRF script used in generating spiralling camera trajectories provided by Mildenhall *et al*. [8]. The script can easily be extended to generate identical views for all remaining 30 classes in the ModelNet40 dataset [17]. For each model, the script generates 64 800x800 images and a set of camera matrix transformations. These are subsequently used to generate the 3D Gaussian point clouds. The 3DGS algorithm [6] is ran using default hyper-parameters, random initialization of the 3D Gaussians, for 15000 iterations. The final *.ply* model files are also made available under 4TU's website [1].

**Results:** To ensure transparency and reproducibility of the reported results, the (fully-commented) code associated with this paper, the trained model weights, the experiments performed, and their logs are available in the GitHub repository [4]. The model weights can be loaded to reproduce any of the results reported in the experiments, and the logs associated ensure the full transparency of the research process. The code base makes use of the PointNet and PointNet++ PyTorch implementations available at the following GitHub repository [5]. The training and fine-tuning of the models was performed on the Delft High Performance Computing Centre [2] using NVidia V100S GPU nodes. To ensure replicability of the results, the slurm job configurations used to train the networks are also made public under the same repository [4].

### 7.2. Integrity

This research adheres to the 5 integrity principles outlined in the Netherlands Code of Conduct [6]: honesty, scrupulousness, transparency, independence, and responsibility. All results were reported in a truthful manner, logs were made available [4], and limitations of the method are discussed. To ensure the transparency of the research and of the reported results, this section describes in detail the training of the point- and image-based baselines.

The implementation used for the MVCNN [14] model can be found at the following Github repository [7]. The network is optimized using the Adam optimizer [7] with an initial learning rate of $0.0001$. The model is trained using the original $64$ generated views, with a batch size of $16$, for $100$ epochs, and every $30$ epochs the learning rate is decreased by $90\%$. As presented in Sec. 5, the model is trained $4$ times, under varying amounts of training data available. To ensure a fair comparison, both the image-based proposed network and the baseline are trained using identical dataset splits (*i.e*. the same data subset is used when reducing the amount of training data available). The same data subsets are also used to train the point- and Gaussian-based models.

The PointNet [10] and PointNet++ [11] models [5] are trained using $2048$ points sampled with FPS at the surface of the object's meshes. The networks are optimized using the SGD optimizer with an initial learning rate of $0.001$, momentum of $0.9$ and weight decay of $0.0001$. The models are trained with a batch size of $24$, for $100$ epochs each.

## References


[1] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 9

[2] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024. 9

[3] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4


[1] https://data.4tu.nl/private_datasets/q32Led--j18SvCZ_X4-RLQBsZRy5Ded3Pf6EbdkzXJg

[2] https://github.com/alextsui05/blender-off-addon

[3] https://polyhaven.com/a/patterned_clay_plaster

[4] https://github.com/SimiOy/Self-Supervised-Learning-for-3DGS

[5] https://github.com/yanx27/Pointnet_Pointnet2_pytorch/tree/master

[6] https://www.nwo.nl/en/netherlands-code-conduct-research-integrity

[7] https://github.com/RBirkeland/MVCNN-PyTorch/tree/master

[5] Longlong Jing, Ling Zhang, and Yingli Tian. Self-supervised feature learning by cross-modality and cross-view correspondences. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1581–1591, 2021. 1, 2, 3, 4, 5

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 1, 2, 3, 9

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 9

[8] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2, 3, 9

[9] Cristiano Premebida, Rares Ambrus, and Zoltan-Csaba Marton. Intelligent robotic perception systems. *Applications of mobile robots*, pages 111–127, 2018. 1

[10] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2, 3, 4, 7, 8, 9

[11] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2, 4, 7, 8, 9

[12] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 4

[13] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. 3

[14] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 1, 7, 8, 9

[15] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International journal of robotics research*, 37(4-5):405–420, 2018. 1

[16] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008. 6, 1, 5

[17] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 3, 7, 8, 9, 1

[18] Xiaoqiang Yan, Shizhe Hu, Yiqiao Mao, Yangdong Ye, and Hui Yu. Deep multi-view learning methods: A review. *Neurocomputing*, 448:106–129, 2021. 1

# Bridging the world of 2D and 3D Computer Vision

## Self-Supervised Cross-modality Feature Learning through 3D Gaussian Splatting

## Supplementary Material

## 8. Dataset analysis

The dataset is comprised of ten household items. The classes are not equally represented as some classes can have up to 6 times more models than others. Table 6 shows the average number of 3D Gaussians used to reconstruct each of these models. The 3DGS algorithm will use more points to represent intricate or complex gemetries, and as a result, some models will produce more Gaussians during optimizations. As observed in Fig. 7, on average, the simplest to represent object is the bathtub, and the hardest is the night stand. On closer inspection of the dataset, most bathtub models are simple convex shapes, or a hallowed out rectangular cuboid. On the other hand, most night stands have handles and support feet which are more complex geometries that require more 3D Gaussians.

Section 3.1 briefly discussed the problem of selecting a representative sample of Gaussians from which the sampling is performed. Besides positions, scale, rotation and spherical harmonics, the 3D Gaussians also have an opacity associated to them. Since the opacity is mostly used during the rendering process to accumulate the color, I decided to leave it out from the input feature consideration of the proposed networks, and instead use it as a measure of the Gaussian importance during sampling. A small opacity would mean that the point has small influence during the rendering process, and thus, intuitively, should also have smaller influence in the deep neural network.

To be sure that we don't leave out any important points, the opacity threshold is an important hyper-parameter of the proposed method. Table 7 shows the minimum number of 3D Gaussians that a model has under different opacity thresholds. Moreover, Fig. 8 and 9 illustrate the distribution of all points above a certain opacity threshold. As it can be observed, there are some extreme cases (*e.g*. one simple bed model) where there are very few 3D Gaussians used, and using a high opacity threshold limits the number of points available for sampling drastically. The threshold used by this paper is 0.3, as it still preserves most of the points for each object, but filters out less relevant ones. Observations have shown that thresholds too high impact the quality of the sampled point cloud, and negatively affect the reported results (*i.e*. smaller recognition accuracy).

| Classes | # Models | Average number of 3D Gaussians |
|---|---|---|
| Bathtub | 156 | 77327 |
| Bed | 615 | 115809 |
| Chair | 989 | 105881 |
| Desk | 286 | 102624 |
| Dresser | 286 | 123331 |
| Monitor | 565 | 91509 |
| Night stand | 286 | 145933 |
| Sofa | 780 | 85950 |
| Table | 492 | 99062 |
| Toilet | 444 | 105715 |

Table 6. Average number of 3D Gaussians that were used to represent each model in the ModelNet10 dataset [17]. The object that requires the least amount of points on average is the bathtub, and the one that requires the most amount of points on average is the night stand. Most bathtubs have simple geometries (simple convex shapes), whereas the night stands usually have handles which are more complex and require more 3D Gaussians to represent.

| Opacity threshold | $\geq 0.3$ | $\geq 0.5$ | $\geq 1.0$ | $\geq 2.0$ |
|---|---|---|---|---|
| Number of 3D Gaussians | 1288 | 1062 | 641 | 153 |

Table 7. Minimum number of 3D Gaussians that a model has under different opacity thresholds. The count is based on the number of 3D Gaussians for all classes that exceed the considered opacity threshold.

## 9. Qualitative analysis of learned features

Although TSNE [16] evaluation cannot be quantitatively assessed, it can still provide a reasonable understanding on the quality of the learned features. In the shape recognition setting, a good representation of the emeddings would have clear compact clusters for each class label. Such self-supervised features would indicate that the model requires less training samples during fine-tuning (*i.e*. if the model knows what monitors look like, but doesn't know what a

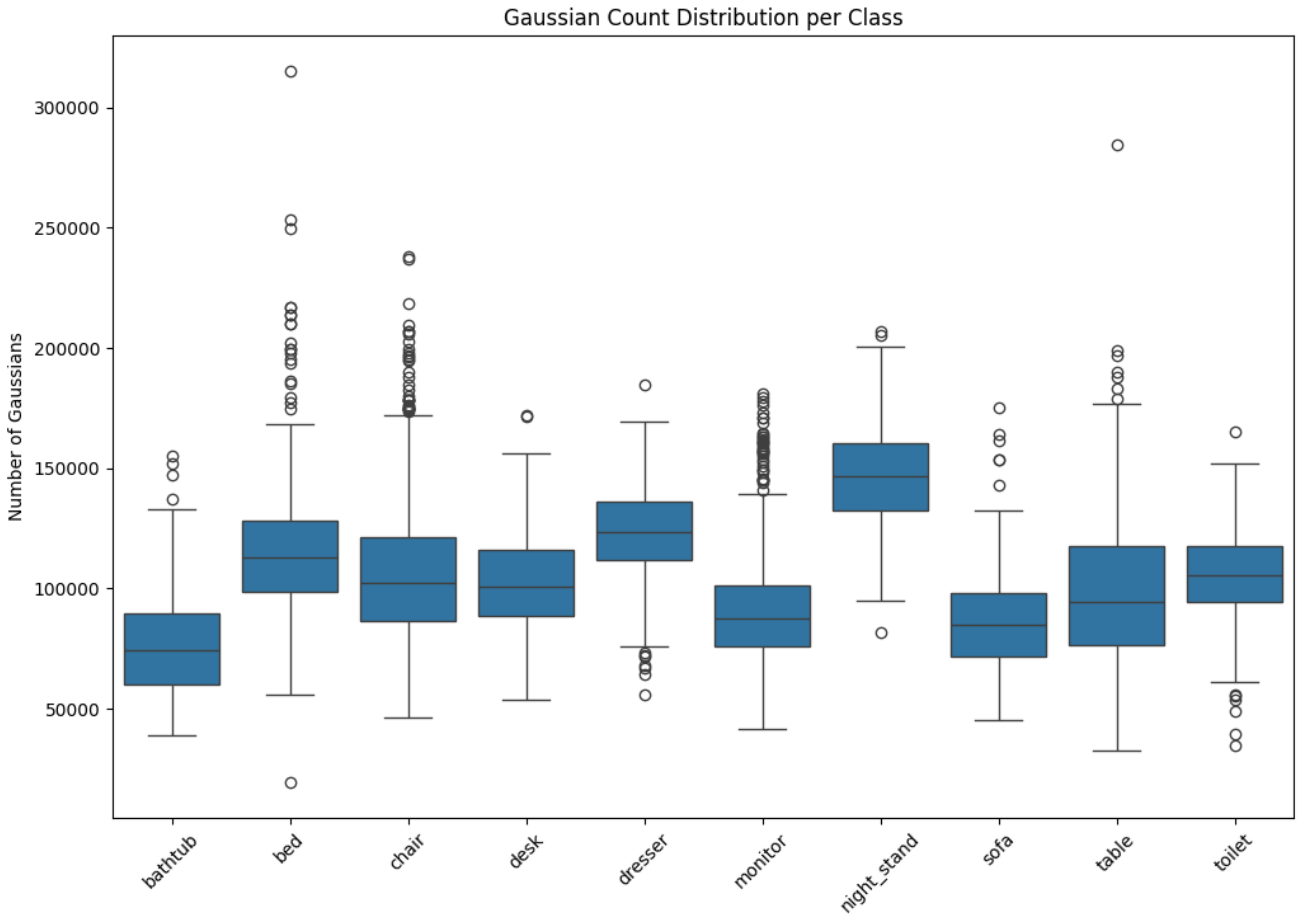


Figure 7. Boxplot of the count of 3D Gaussians per class. Overall all models have between 50000 and 150000 3D Gaussians each. The bed class has both the simplest and the most complex model, because some 3D models were canopy beds or came with pillows, while others had a simple rectangular geometry. On average the night stand object required the most amounts of 3D Gaussians, while the bathub required the least. The toilet class exhibits the least amount of variation between models, requiring around 100000 points each.

monitor is, it would have to see fewer labeled monitors to be able to correctly identify the rest).

Figure 10 illustrates a comparison between the two sampling techniques and model backbones, for networks trained using only the Gaussian positions. This experiment confirms again that FPS outperforms uniform sampling, showing more distinctive cluster shapes forming. Between the two point backbones, the difference is minimal, and since the TSNE algorithm is not deterministic, a definitive conclusion between the two is difficult to draw.

Across all models, there exists some common confusion. For example, the table and the desk, or the dresser and the night stand are often mistaken for one another. That is in part to their similar geometrical appearance but also to their similar visual appearance (the 2D appeareance of the objects still contributes during the self-supervised learning framework). Since all models have the exact same texture applied, are lit in an identical fashion, and have the same scale, it's much more difficult to differentiate between such shapes. Perhaps the model would benefit more from a realistic dataset, where sizes and colors have meaning.

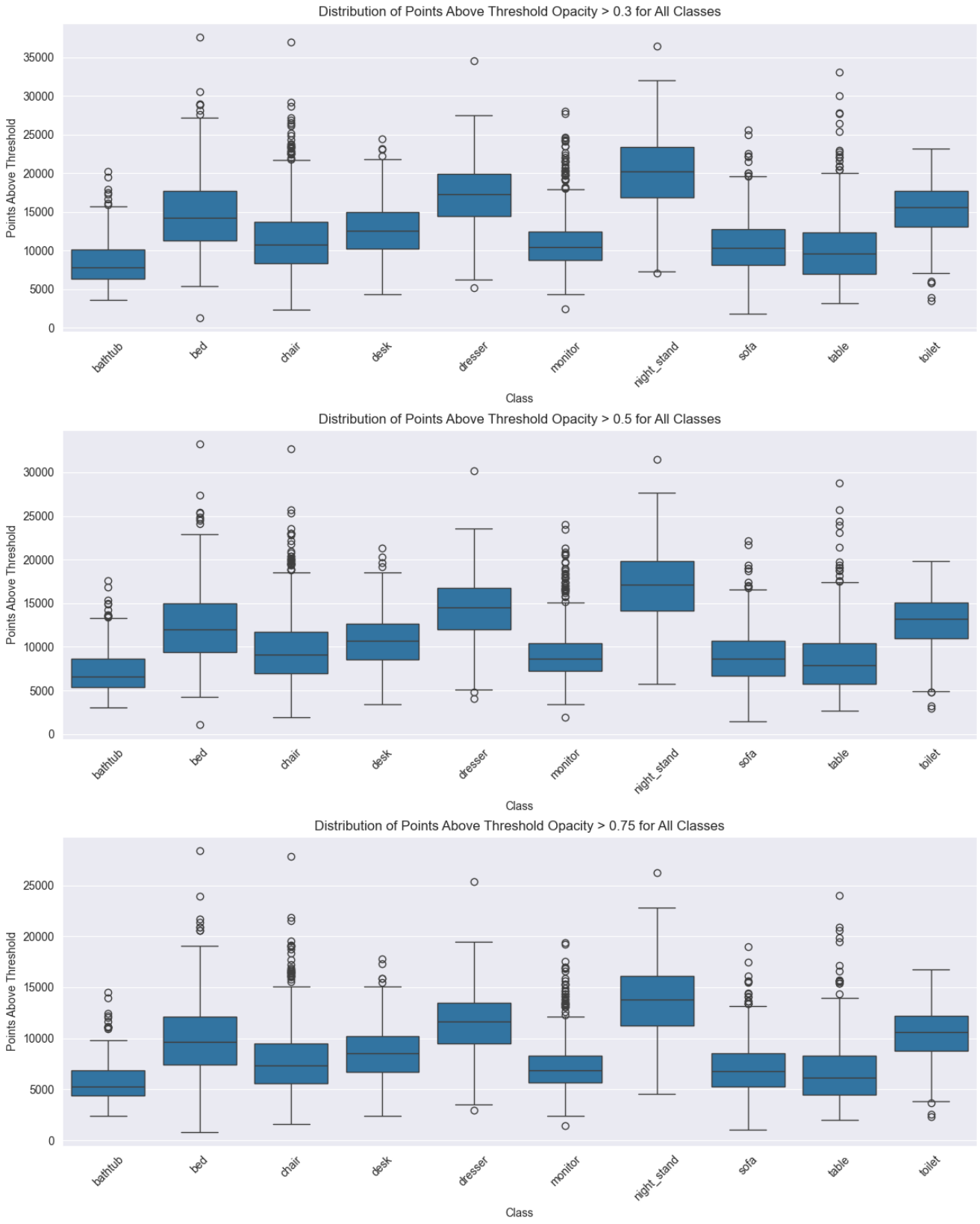


Figure 8. Boxplots for the number of 3D Gaussians above a certain opacity threshold for each class. As the threshold gets increased to 0.75, a simple bed model has only 800 3D Gaussians left. The overall distribution of the number of points remains constant.

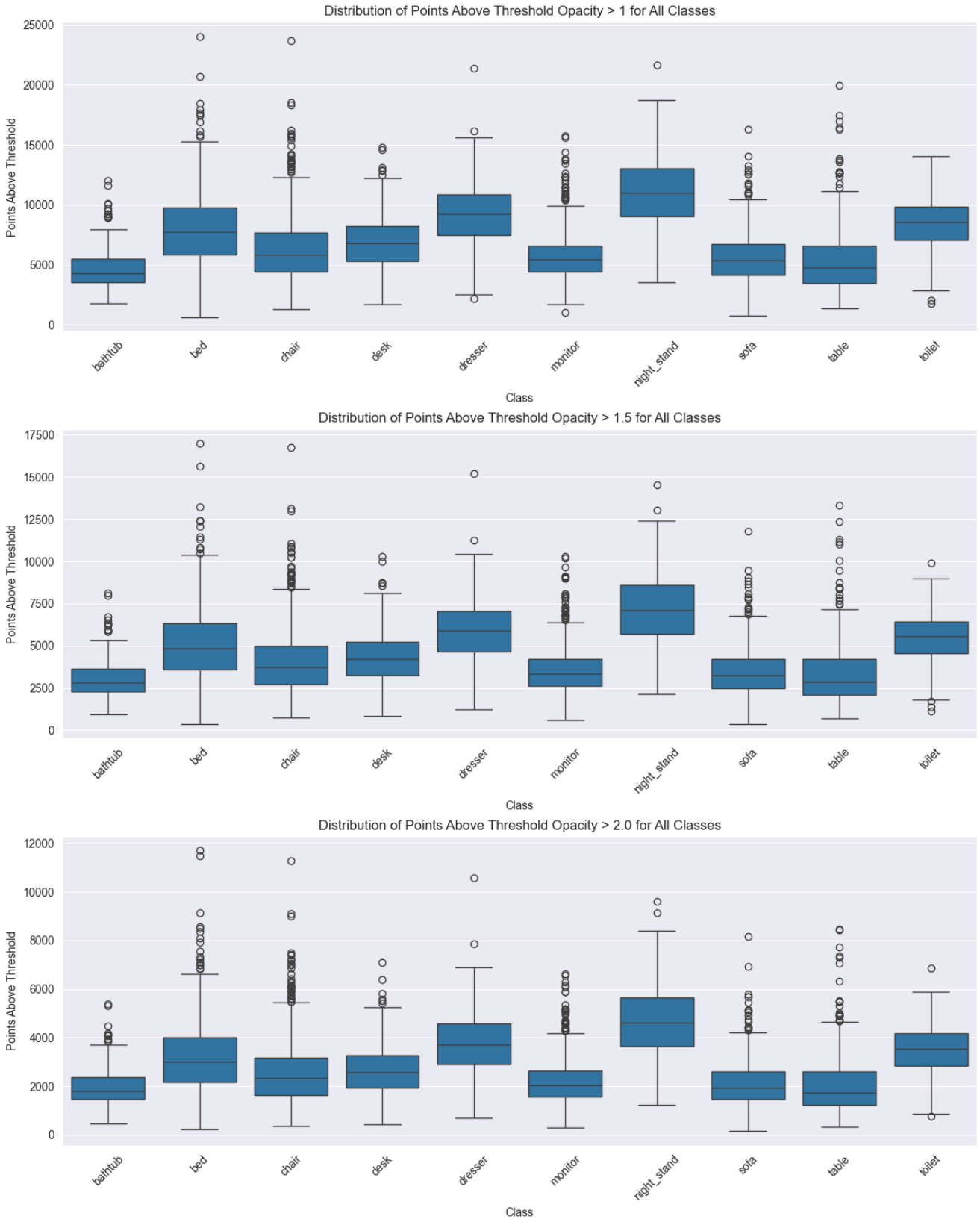


Figure 9. Boxplots for the number of 3D Gaussians above a certain opacity threshold for each class. As the threshold gets increased to 2.0, a lot of the models are left with less than 2000 points, signaling that such threshold are too large to consider.

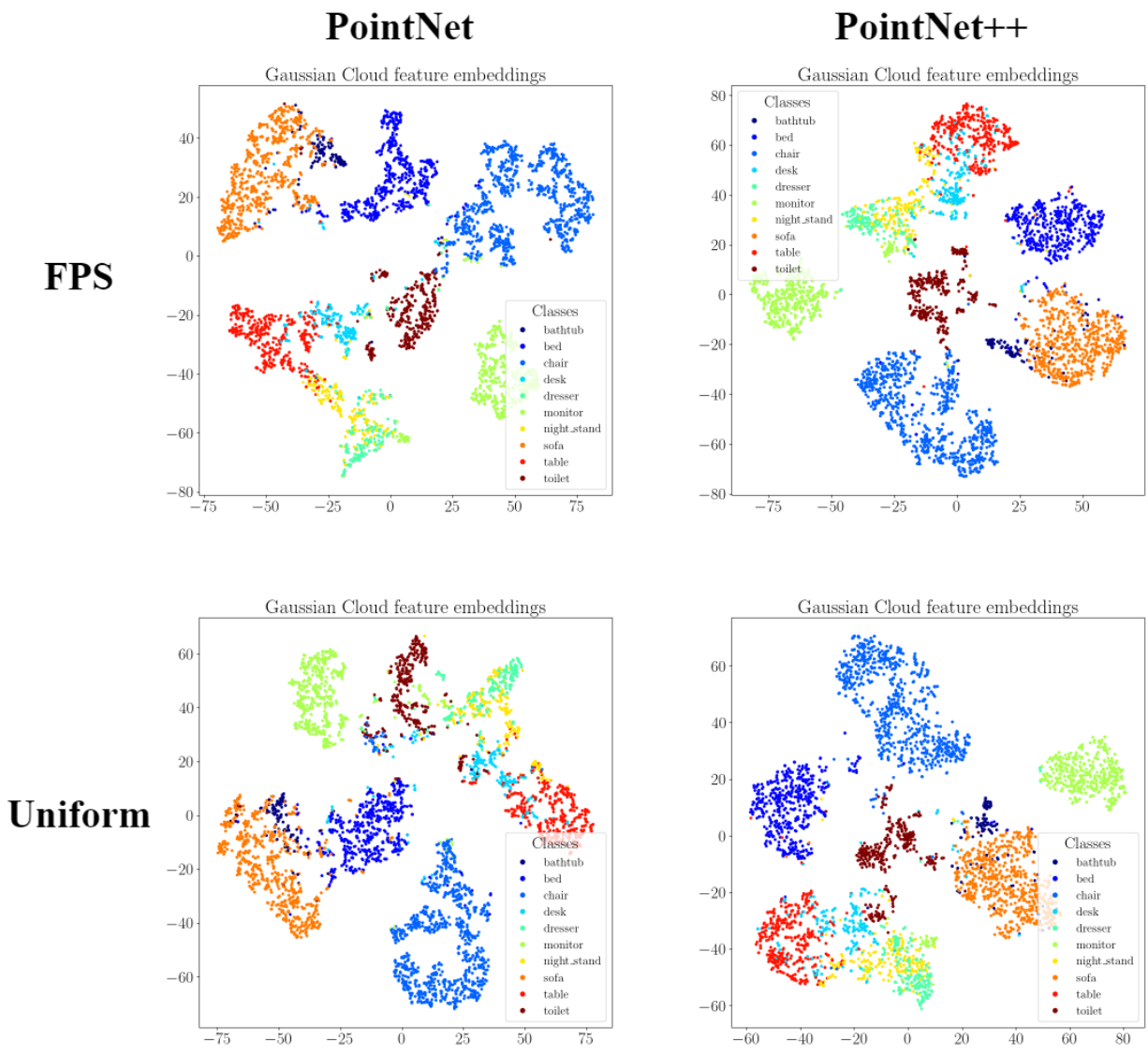


Figure 10. TSNE [16] visualization of the learned features for the Gaussian-based models, for the two network backbones and sampling techniques considered. The embeddings correspond to networks that have been trained using only the 3D Gaussian positions. FPS is better than uniform sampling at making compact class clusters, showing that a better geometrical representation of the objects is favoured. There aren't any noticeable differences in the learned feature visualizations when using the advanced *PointNet++* model.