Attentional Convolutional Occupancy Networks

xxxxx *, Lucas Brunner†, xxxxx ‡ and xxxxx§
*†§Department of Computer Science, ‡Department of Mathematics
ETH Zurich, Switzerland
Email: *xxxxx@ethz.ch, †brunnelu@ethz.ch, ‡xxxxx@ethz.ch, §xxxxx@ethz.ch

***Abstract*—We present a new state of the art model for learning-based 3D reconstruction. Our model builds on convolutional occupancy networks, an architecture for the implicit representation of 3D objects, and extends it with various attention mechanisms at different locations in the network. Results showed that our method can capture complex and detailed structures and outperforms the current state of the art models on single object benchmark datasets.**

***Index Terms*—3D, Attention, ML, Occupancy, Convolution, Graph, Meshes, ShapeNet,**

I. Introduction

The field of 3D reconstruction has recently achieved incredible progress. While the community has agreed on a representation in the 2D images, there is no representation of 3D objects yet that is memory efficient and can be efficiently inferred by data. On the one hand, voxels, which are a generalization of pixels, have a major drawback in terms of memory requirements for high resolutions. On the other hand, point clouds discard topological relations. Hence, to extract 3D geometry from the model, one needs to perform extra post-processing steps. Furthermore, mesh-based representations are hard to be predicted reliably using feed-forward networks. In 2019, Mescheder et al. [1] have introduced one of the first implicit representations where 3D structures are described by the decision boundary of an occupancy network, i.e. the model predicts the occupancy probability for every 3D point. As this decision boundary is continuous, the 3D space is not discretized and thus, there are no topology restrictions on the generated 3D shapes. As the initial occupancy network architecture was limited by its simple fully-connectedness, the integration of local information in the observations, and the incorporation of inductive biases was not possible. Hence, the initial approaches using occupancy networks mainly focused on single objects and did not scale to larger scenes. As a result, Peng et al. [2] introduced convolution occupancy networks, in which they combine the complementary strengths of convolutional neural networks of integrating inductive biases and encoding information in a hierarchical way and the strengths of occupancy networks of implicitly representing 3D structures.
The rising popularity of attention mechanisms gave rise to the proposal of the following improvements of the convolutional occupancy networks. First, due to the hard attentive nature of basic pooling operations (sum, max, average), we replace them with robust attentional aggregation mechanisms [3]. Second, we replace the feature plane decoding U-Net, which improved the occupancy networks in the first place, with an attentional U-Net. Third, we replace the positional encoding network with a graph neural network yielding an embedding based on nearest neighbors. The illustration of our ideas is shown in Fig. 1.

II. Models and Methods

In the following sections, we first present the baseline model as well as the different attention mechanisms and optimization procedures, explaining the rationale behind them.

A. Baseline

To evaluate our models performing object-level 3D reconstruction from point clouds, we use as a baseline Peng et al.'s [2] best convolutional occupancy network based on their paper. Their model produces a feature encoding for all the input points using a shallow PointNet [4] with local pooling for 3D points. Then, they orthographically project the encoded points onto three canonical planes discretized by a grid of a resolution of $H \times W$ pixels. Pixels being projected into the same grid cell are aggregated using average pooling. The resulting planar features are then independently processed by a 2D convolutional hourglass (U-Net) network [5]. The processed feature planes are then aggregated using sum pooling. For the actual occupancy prediction, the project every point $\mathbf{p}$ in 3D space onto the three canonical planes and compute the corresponding feature vector using bilinear interpolation. Given the point $\mathbf{p}$ and its feature vector, they predict the point's occupancy using the similar small fully-connected occupancy network consisting of multiple ResNet blocks to the one Mescheder et al. [1] used in the initial occupancy network paper.

B. Self Attention

As self attention layers have improved many machine-learning models [6]–[10], we experimented with self attention layers at different locations in the encoder network. Integrating it after the first ResNet block of the initial PointNet encoder architecture produced the best results.
Due to explicit input shape requirements of Pytorch and the subsequent axes permutations of the tensors, we had to drastically reduce the batch size and the number of points per point cloud in order to stay in the memory bounds given by the Leonhard GPUs. This experiment was not mentioned in the project proposal.

C. Robust Attentional Aggregation

A recurrent problem in the convolutional occupancy network paper is the aggregation of a set of deep features. There are two ways how one can aggregate a set of features: by treating the set as a sequence and processing the sequence with a recurrent network, or by performing pooling operations. However, recurrent networks have major drawbacks for this task. They are permutation variant, long term dependencies in the sequence might be hard to capture due to the vanishing and exploding gradient problem and the sequential processing of the inputs is inefficient. Pooling operations, on the other side, tend to be 'hard-attentive', i.e. they do not learn to attentively preserve useful information [3].

The two main occurrences of pooling operations are the following: First, after processing the 3D input points with a shallow PointNet [4] with local pooling, Peng et al. [2] perform an orthographic projection of every embedded point onto a canonical plane of discrete resolution. All the feature points which are projected onto the same grid cell are aggregated using average pooling. Second, after the projections, the three generated feature planes are all separately processed by a multi-plane decoder using 2D U-Nets. The three resulting feature maps are aggregated using sum-pooling. As opposed to our initial proposal of aggregating the three feature planes **before** being

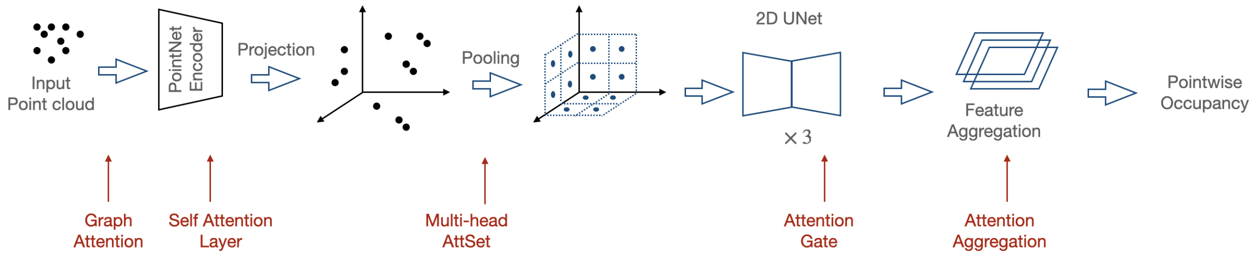

Fig. 1: Our added attention mechanism on the original network structure

processed by the U-Net, we decided to aggregate them **after** the U-Net. The main reason for this was the assumption that the model should be capable of extracting more information when applying the convolutional 2D U-Net to all three feature planes independently and thus extracting local and global information contrary to applying it only once to an aggregated feature plane.

1) Attention Sets: As a first approach, we used AttSets as introduced by Yang et al. [3]. Given a feature set of N elements $\{x_1, \ldots, x_N\}$, where $x_i \in \mathbb{R}^D$, the goal is to find a permutation invariant function with learnable weights that aggregates all the elements of the set into one single element $x_{agg} \in \mathbb{R}^D$. AttSets learns an attention score for every latent feature, i.e. for every dimension of every element of the set. It is important to distinguish between the features x_i representing the elements of the feature set, and the latent features corresponding to the D entries of every feature vector x_i.

First, every element x_i is processed by a linear layer with no activation function, yielding a set of learned attention activations $C = \{c_1, \ldots, c_N\}$, where $c_i \in \mathbb{R}^D$

$$c_i = g(x_n, W) = W \cdot x_n \tag{1}$$

Next, in order to compute the attention scores, the learned attention activations are being normalized by a softmax layer across the N elements of the set, yielding a set of attention scores $S = \{s_1, \ldots, s_N\}$, where $s_i \in \mathbb{R}^D$ and

$$s_n^d = \frac{e^{c_n^d}}{\sum_{j=1}^N e^{c_j^d}} \tag{2}$$

The superscript d indicates the entry of the vector. In the next step, the elements of the feature set are reweighted by the computed attention scores, yielding a set of reweighted features $O = \{o_1, \ldots, o_N\}$, where

$$o_n = x_n *_{comp} s_n \tag{3}$$

The component-wise multiplication operation is represented by $*_{comp}$. Finally, the N reweighted features are summed up yielding the aggregated element $x_{agg} \in \mathbb{R}^D$.

$$x_{agg}^d = \sum_{n=1}^N o_n^d \tag{4}$$

Note that the softmax function is applied for one latent feature across all the elements of the set, i.e. the latent feature x_{agg}^d for $d \in \{1, \ldots, D\}$ is a convex combination of $x_1^d, \ldots, x_n^d$. An illustration of the explained procedure can be seen in Fig. 2.

2) Multi-Headed Set Attention: Inspired by Vaswani et al.'s [6] Multi-Head Attention, we extended the simpler AttSets module for the feature plane generation with a similar approach in order to combine the best of both worlds. We want to incorporate the idea of the model having multiple attention heads as opposed to only one. Therefore, we collect the points being projected onto the same grid cell and apply the AttSets module h-times producing h independent aggregated features, similar to the multi-head attention idea. The results are then concatenated and transformed into a single aggregated feature using a linear layer. The rationale behind this was that having multiple AttSet-heads would increase the power of our model. We found h=8 to be a good number of heads. A visualization of this approach is shown in Fig. 3. The colors of the different components are matching in Fig. 2 and 3.

D. Attention U-Net

In the original Convolutional Occupancy Networks, 2D convolutional hourglass networks (U-Net) were used to process the feature planes from the encoder. 2D U-Net composed of a contracting path (4 down-conv layers) and an expansive path (4 up-conv layers) with skip connections to integrate both local and global information (Ronneberger et al. [5]). Inspired by Oktay et al. [11], we add an attention block to the original U-Net structure, where attention gates (AGs) are incorporated into the standard U-Net architecture to highlight prominent features from irrelevant responses that are passed through the skip connections. We use additive attention [12] to get the attention coefficients $\alpha_i \in [0, 1]$, which identifies whether features are salient or not. Additive attention is more computationally expensive, but is experimentally proved to work well in this case, as from Oktay et al. [11]. In layer l, AGs output element-wise multiplication of the input feature-maps and attention coefficients. The calculation path in attention gates works as follows:

$$q_{att}^l = \phi^T \left(\sigma_1 (W_x^T x_i^l + W_g^T g_i + b_g) \right) + b_\phi \tag{5}$$

$$\alpha_i^l = \sigma_2 (q_{att}^l (x_i^l, g_i; \Theta_{att})) \tag{6}$$

$$\hat{x}_{i,c}^l = x_{i,c}^l \cdot \alpha_i^l \tag{7}$$

where gating vector $g_i \in \mathbb{R}^{F_g}$ is taken from the next lowest layer in the expansive path. The vector has smaller dimensions and better feature representation, given that it comes from deeper into the network. Feature vector $x_i^l \in \mathbb{R}^{F_l}$ is from skip connection at layer l. The set of parameters Θ_{att} contains: linear transformations $W_x \in \mathbb{R}^{F_l \times F_{int}}$, $W_g \in \mathbb{R}^{F_g \times F_{int}}$, $\phi \in \mathbb{R}^{F_{int} \times 1}$ and bias term $b_\phi \in \mathbb{R}$, $b_g \in \mathbb{R}^{F_{int}}$, where $\mathbb{R}^{F_{int}}$ denotes an intermediate space. The i and c in $x_{i,c}^l$ denote spatial and channel dimensions respectively. The linear transformations are computed by adding channel-wise $1 \times 1 \times 1$ convolution layers on the input tensors.

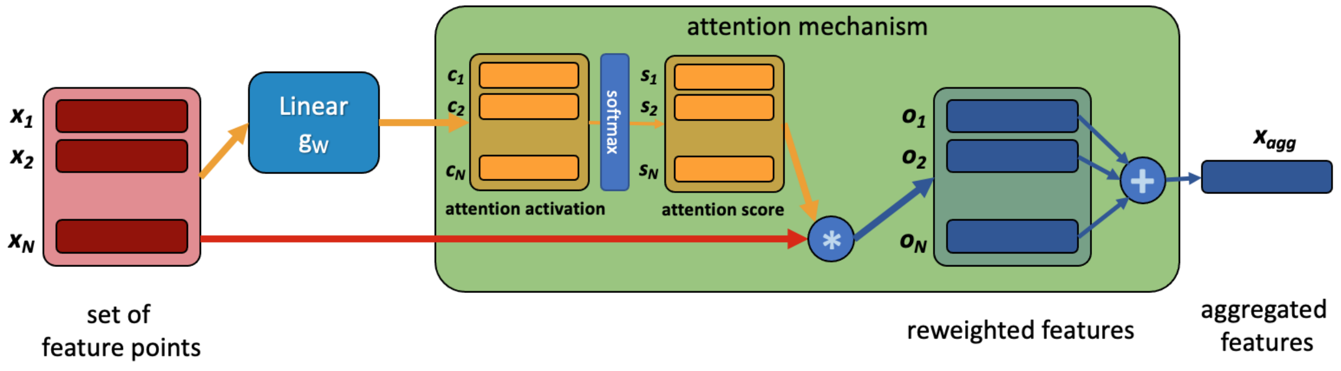

Fig. 2: Attention Set

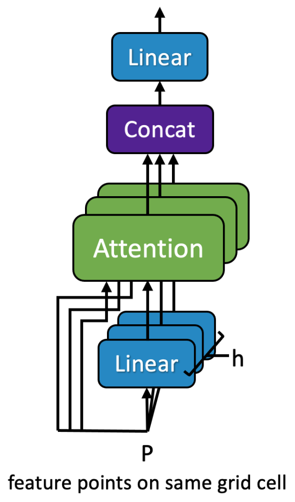

Fig. 3: Multi-Head Attention Set

E. Graph Neural Network

Graph neural networks (GNNs) have made it possible to train better models on point cloud data [13]–[15]. The main motivation behind using a GNN is that it enables the computation of a point embedding based on the relations of the nodes in the graph. In order to make use of the graph structure, one first needs to decide, how the points of a point cloud should be connected and what kind of GNN model one wants to use. Due to the good results in the SuperGlue paper [15] using fully-connected graphs combined with attention, we base our implementation on the implementation in the SuperGlue paper, where the attentional graph neural network consists of multiple layers of self and cross attention is applied to a pair of key-point sets from two images. However, in our base graph model we only consider one point cloud (not a pair of point clouds) and only use self attention. Besides choosing the structure of the graph, one also has to decide how to actually model the nodes of the graph, i.e. choosing the location of the graph layer within the initial convolutional occupancy network architecture. We experimented with two different models. On the one hand, we added a graph layer at the very beginning of the encoder pipeline, treating every 3D point as a node of the complete graph. This graph layer can be seen as an extension of the linear layer being responsible for the positional encoding in the baseline model architecture. On the other hand, we built a graph treating the embedded points as nodes. Both graph layers did not replace any layers but can be seen as an enrichment of the model. Because of GPU memory limitations, we only experimented with 1(-4) graph attention layer in both models, no edge features were used and the number of points per point cloud as well as the batch size had to be reduced drastically.

For future work, one could try to follow the alternating scheme of self and cross attention of the SuperGlue [15] more closely, by using the 3D input points as the first set of nodes and the grid points as a second set of points. The goal of this approach would again be the computation of a point embedding based on information of neighboring nodes, but this time sharing information between two layers.

Besides using a fully-connected graph, we also experimented with a k-nearest-neighbor graph, based on Euclidean distance, i.e. as opposed to the complete graph, nodes are only connected to their k nearest neighbors. The intuition behind this approach is that this should lead to better local features and therefore better models. Building the knn-graph during runtime is more costly but seemed more promising as one can keep the rest of the input pipeline as is. Opposing dependency requirements made this approach infeasible and replacing the input pipeline would have led to models deviating too strongly from the baseline, making a direct model comparison harder. Hence, we omit this approach in the rest of this report.

III. Results

In the following section, we first give a brief overview over the used datasets and the metrics used for the evaluation of the different approaches. We then present the results of the different experiments that we conducted in comparison to the given baseline.

A. Dataset

As a dataset, we use the ShapeNet [16] subset of Choy et al. [17] which contains 13 classes of subjects and contains a training, validation and testing split. As every model had to be trained for $\sim$48 hours and due to long waiting periods on the cluster, we were not able to test our model on other datasets of larger scenes.

B. Metrics

We use Volumetric Intersection over Union (IoU), Chamfer L_1 Distance, Normal Consistency and the F score as metrics for evaluation.

IoU: Let $\mathcal{M}_{\text{pred}}$ and $\mathcal{M}_{\text{GT}}$ be the set of all points that are inside or on the surface of the predicted and ground truth meshes. The IoU is the volume of the two meshes' intersection divided by the volume of their union. The IoU ranges from 0, where there is no intersection with the ground truth, to 1, where we have a perfect match.

$$\text{IoU}(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) \equiv \frac{|\mathcal{M}_{\text{pred}} \cap \mathcal{M}_{\text{GT}}|}{|\mathcal{M}_{\text{pred}} \cup \mathcal{M}_{\text{GT}}|} \tag{8}$$

Chamfer-L_1: Measures the distance of points from two point clouds. This measurement approaches 0 if there are perfect matches for all

category	IoU							Chamfer-$L_1 \cdot 10^{-2}$						
	Baseline	SelfAtt	AttSet	AttUNet	UNetAgg	GraphAtt	AttUAgg	Baseline	SelfAtt	AttSet	AttUNet	UNetAgg	GraphAtt	AttUAgg
airplane	0.840	0.837	0.850	0.845	0.848	0.839	**0.863**	0.353	0.360	0.335	0.344	0.336	0.351	**0.309**
bench	0.821	0.826	0.831	0.827	0.829	0.825	**0.841**	0.366	0.357	0.349	0.355	0.351	0.357	**0.332**
cabinet	0.936	0.936	0.940	0.935	0.938	0.940	**0.943**	0.492	0.496	0.467	0.492	0.475	0.464	**0.451**
car	0.882	0.881	0.886	0.883	0.885	0.884	**0.889**	0.773	0.785	0.744	0.765	0.757	0.745	**0.731**
chair	0.865	0.869	0.872	0.868	0.872	0.867	**0.880**	0.474	0.466	0.458	0.469	0.454	0.466	**0.437**
display	0.924	0.930	0.927	0.926	0.926	0.927	**0.934**	0.372	0.360	0.366	0.369	0.368	0.363	**0.348**
lamp	0.776	0.773	0.786	0.738	0.780	0.776	**0.798**	0.604	0.626	0.580	0.721	0.582	0.573	**0.549**
speaker	0.914	0.917	0.920	0.914	0.914	0.919	**0.922**	0.663	0.667	0.632	0.670	0.661	0.623	**0.613**
rifle	0.840	0.834	0.844	0.841	0.831	0.833	**0.858**	0.290	0.298	0.285	0.285	0.301	0.300	**0.257**
sofa	0.932	0.934	0.938	0.934	0.935	0.938	**0.941**	0.436	0.427	0.409	0.426	0.422	0.407	**0.397**
table	0.882	0.883	0.889	0.885	0.885	0.887	**0.900**	0.399	0.397	0.380	0.395	0.386	0.382	**0.362**
telephone	0.949	**0.956**	0.951	0.954	0.953	0.952	0.955	0.283	**0.263**	0.273	0.271	0.271	0.272	0.265
vessel	0.863	0.860	0.870	0.863	0.862	0.863	**0.878**	0.437	0.446	0.419	0.441	0.443	0.436	**0.397**
mean	0.879	0.880	0.885	0.878	0.881	0.881	**0.893**	0.457	0.458	0.438	0.462	0.447	0.442	**0.419**

category	Normal Consistency							F-score						
	Baseline	SelfAtt	AttSet	AttUNet	UNetAgg	GraphAtt	AttUAgg	Baseline	SelfAtt	AttSet	AttUNet	UNetAgg	GraphAtt	AttUAgg
airplane	0.928	0.926	0.931	0.929	0.932	0.929	**0.934**	0.961	0.960	0.968	0.962	0.966	0.963	**0.973**
bench	0.918	0.919	0.921	0.919	0.921	0.920	**0.923**	0.960	0.963	0.965	0.963	0.965	0.963	**0.970**
cabinet	0.955	0.954	0.957	0.954	0.956	0.955	**0.958**	0.951	0.948	0.956	0.950	0.954	0.957	**0.959**
car	0.891	0.890	0.892	0.891	0.893	0.890	**0.894**	0.842	0.840	0.852	0.845	0.849	0.849	**0.856**
chair	0.941	0.941	0.943	0.941	0.944	0.941	**0.946**	0.934	0.937	0.942	0.935	0.944	0.936	**0.947**
display	0.968	0.968	0.969	0.967	0.969	0.968	**0.970**	0.971	0.972	0.973	0.970	0.972	0.973	**0.976**
lamp	0.898	0.898	0.902	0.888	0.903	0.900	**0.905**	0.884	0.881	0.895	0.855	0.893	0.888	**0.905**
speaker	0.937	0.937	0.940	0.937	0.937	0.939	**0.940**	0.887	0.888	0.897	0.885	0.888	0.895	**0.898**
rifle	0.924	0.930	0.928	0.930	0.931	0.924	**0.932**	0.976	0.979	0.979	0.979	0.977	0.975	**0.983**
sofa	0.957	0.956	0.959	0.957	0.958	0.958	**0.960**	0.948	0.949	0.955	0.950	0.952	0.955	**0.958**
table	0.958	0.957	0.960	0.958	0.959	0.959	**0.961**	0.964	0.964	0.970	0.965	0.967	0.970	**0.974**
telephone	0.982	0.982	0.983	0.982	0.983	0.982	**0.983**	0.987	0.989	0.989	0.987	0.989	0.989	**0.990**
vessel	0.917	0.917	0.919	0.918	0.921	0.917	**0.923**	0.928	0.929	0.936	0.928	0.930	0.930	**0.941**
mean	0.936	0.937	0.939	0.936	0.939	0.937	**0.941**	0.938	0.938	0.944	0.937	0.942	0.942	**0.949**

Table I: 3D Reconstruction performance metrics of all our models vs. baseline model. AttSet and UNetAgg refer to Section II C, AttUNet refers to Section II D, GraphAtt refers to Section II E, AttUAgg refers to a combination of AttSet and UNetAgg. Note for some methods, we trained different variants and only put the best score here.

points in the two point clouds. Let $\partial\mathcal{M}_{\text{pred}}$ and $\partial\mathcal{M}_{\text{GT}}$ be the surfaces of the meshes.

$$\text{Chamfer-}L_1(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) \equiv \frac{1}{2\partial\mathcal{M}_{\text{pred}}} \int_{\partial\mathcal{M}_{\text{pred}}} \min_{q\in\partial\mathcal{M}_{\text{GT}}} \|p-q\| dp + \frac{1}{2\partial\mathcal{M}_{\text{GT}}} \int_{\partial\mathcal{M}_{\text{GT}}} \min_{p\in\partial\mathcal{M}_{\text{pred}}} \|p-q\| dq \tag{9}$$

Normal Consistency: Measures the consistency of normal vectors. Let $n(p)$ and $n(q)$ be the unit normal vectors on the mesh surface $\partial\mathcal{M}_{\text{pred}}$ and $\partial\mathcal{M}_{\text{GT}}$ respectively, and $\text{proj}_2(p)$and $\text{proj}_1(q)$ denote the projections of p and q onto $\partial\mathcal{M}_{\text{GT}}$ and $\partial\mathcal{M}_{\text{pred}}$ respectively .

$$\text{Norm-Con}(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) \equiv \frac{1}{2\partial\mathcal{M}_{\text{pred}}} \int_{\partial\mathcal{M}_{\text{pred}}} |\langle n(p), n(\text{proj}_2(p))\rangle| dp + \frac{1}{2\partial\mathcal{M}_{\text{GT}}} \int_{\partial\mathcal{M}_{\text{GT}}} |\langle n(\text{proj}_1(q)), n(q)\rangle| dq \tag{10}$$

F-Score: As Tatarchenko et al. [18], define recall as the percentage of points on GT mesh lying within a certain distance to the reconstructed mesh, and precision as the percentage of points on the reconstructed mesh that lies within a certain distance to the GT. F-score is defined as the harmonic mean of the two.

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

The results of the different experiments can be seen in Table I. All the models were trained for 500k iterations. The `AttSet` model, corresponding to the base architecture enriched with a multi-headed (8 heads) AttSets module for the orthographic projections of the embedded points and the `UNetAgg` model, corresponding to the base model enriched with an AttSets module for the aggregation of the processed feature planes after the U-Net, both outperform the baseline model. One can see that the model `AttUAgg`, a combination of the `AttSet` and the `UNetAgg` model, clearly outperforms all the other models on all the metrics.

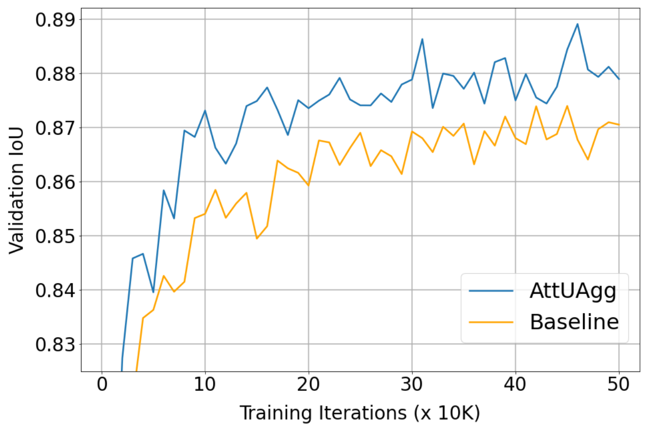

Fig. 4: Validation IoU Graph

A plot of the validation IoU metrics of our best approach and the baseline can be seen in Fig. 4. Fig. 7 - 9 in the appendix show a comparison of the meshes generated by our different models, the ones generated by the baseline model and the ground truth meshes.

IV. Discussion

In the following section, we discuss the results and implications of our models.

In table I, one can see that three of our models outperform the baseline model. `AttUAgg`, the combination of the models `AttSet` and `UnetAgg` even outperforms the baseline as well as the other approaches on all categories and metrics by a significant margin.

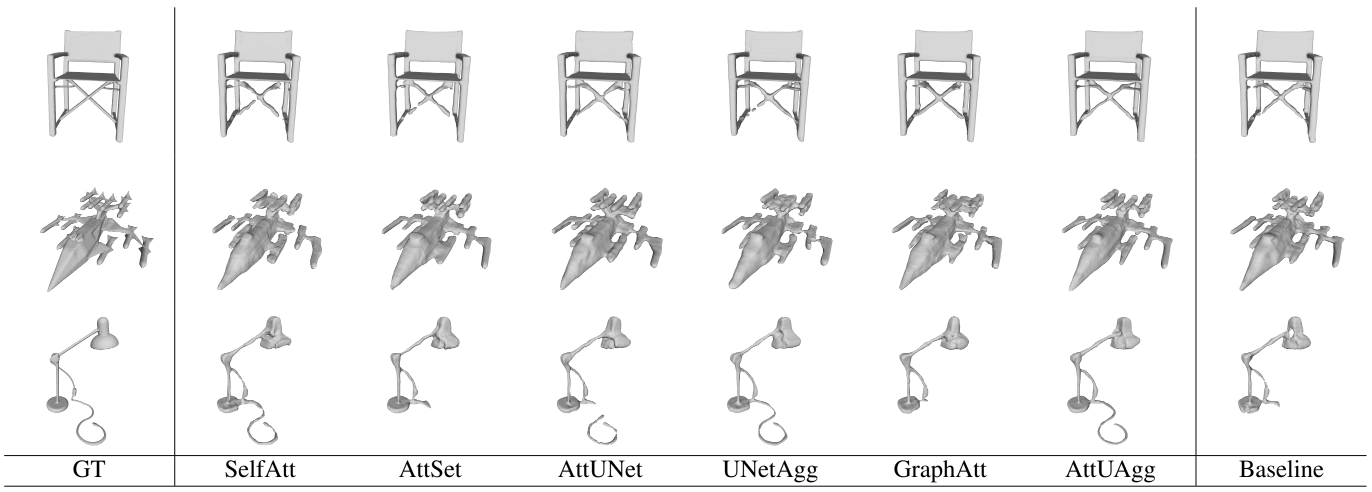

Fig. 5: A comparison of generated meshes vs. ground truth mesh on ShapeNet

During training, our models did not only reach a higher validation IoU, but they also reached higher values than the baseline much faster i.e. after only one third of the total iterations, this can be seen in Fig. 4.

The models `SelfAtt`, `AttUNet` and `GraphAtt` did not manage to exceed the performance of the baseline but performed equally well. However, `SelfAtt` and `GraphAtt` reached their peak validation IoU in about half of the time, compared to the baseline. `AttUNet` uses additive attention instead of multiplicative attention, which is computationally more expensive. The validation IoU was still going up at around 500k iterations. One could argue that with more training time the result would have gotten even better. This leaves us with the following insight: Using graph layers and self attention layers in the encoder network and using an attention U-Net for the processing of the generated feature planes does not benefit the final results evidently, but they have some positive impacts as mentioned above. Replacing the hard-attentive pooling operations with robust attentional aggregation mechanisms, however, significantly improved the performance of convolutional occupancy networks, which is also the reason why we did a new experiment with the combination of the two.

Besides the numerical results, a clear improvement can also be perceived visually (see Fig. 5, a large comparison can be found in the appendix in Fig. 7-9). As the baseline IoU is already at a very high value, the improvements are especially visible in complex and detailed structures. We particularly noted a large impact on fragile structures which speaks for the attention mechanism. Such detailed structures get easily lost through pooling operations like a max pooling or average pooling. The attention mechanism learns to treat such structures with more care and gives better aggregation results.

Note that during training, we were not able to reach the same performance with the baseline model as reported in the original Convolutional Occupancy paper (see Peng et al. [2] supplementary material). This may have different reasons, one of which being the imprecise information about training time: "*at least* 300k iterations". Therefore, we trained each model for 500k iterations, leading our model to even exceed the reported performance presented in the original paper.

For the approaches `GraphAtt` and `SelfAtt`, one has to note the following: As we used a different number of points per pointcloud than the baseline model that we are comparing our model against, it is hard to directly argue about the performance differences of the two models. However, our model achieved comparable results even though it was trained on $17\% - 30\%$ fewer points.

We hypothesize that the added attention mechanisms would also benefit the other versions of Peng et al.'s convolutional occupancy networks that use local patch encoder and decoder models. The verification of this claim and the application of our model to datasets of larger scenes, such as MatterPort3D [19] or ScanNet v2 [20], are left for future work.

V. Summary

Attention mechanisms have shown promising results in many different areas. We managed to successfully apply different attention modules to the task of learning-based 3D reconstruction resulting in a new architecture that we call attentional convolutional occupancy networks. Our model builds on the implicit representation of 3D objects as introduced in [2]. Due to the attentive extensions of the base model, our methods outperform the state of art in terms of quality and computation time. Our best model shows a large improvement in connecting fragile structures and preserving details more precisely.

References

[1] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.

[2] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger, "Convolutional occupancy networks," *arXiv preprint arXiv:2003.04618*, 2020.

[3] Bo Yang, Sen Wang, Andrew Markham, and Niki Trigoni, "Attentional aggregation of deep feature sets for multi-view 3d reconstruction," *arXiv preprint arXiv:1808.00758*, 2018.

[4] 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*, 2017, pp. 652–660.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[7] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[8] Minh-Thang Luong, Hieu Pham, and Christopher D Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[9] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le, "Massive exploration of neural machine translation architectures," *arXiv preprint arXiv:1703.03906*, 2017.

[10] Jianpeng Cheng, Li Dong, and Mirella Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[11] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert, "Attention u-net: Learning where to look for the pancreas," 2018.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[13] Weijing Shi and Raj Rajkumar, "Point-gnn: Graph neural network for 3d object detection in a point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1711–1719.

[14] Loic Landrieu and Martin Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4558–4567.

[15] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4938–4947.

[16] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al., "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[17] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *European conference on computer vision*. Springer, 2016, pp. 628–644.

[18] Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox, "What do single-view 3d reconstruction networks learn?," 2019.

[19] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017.

[20] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5828–5839.

Appendix

A. *Network Architecture*

We present a complete overview of the model architecture of our best attentional convolutional occupancy network containing two robust attentional aggregation blocks (see Fig 6).

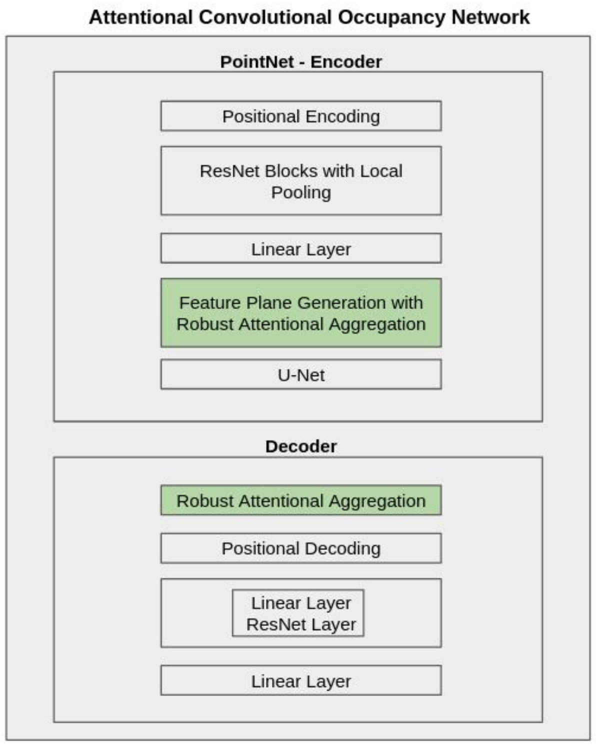
Attentional Convolutional Occupancy Network
PointNet - Encoder
Positional Encoding
ResNet Blocks with Local Pooling
Linear Layer
Feature Plane Generation with Robust Attentional Aggregation
U-Net
Decoder
Robust Attentional Aggregation
Positional Decoding
Linear Layer
ResNet Layer
Linear Layer

Fig. 6: Model Architecture of the Attentional Convolutional Occupancy Network.

B. *Hardware and Training Time*

We train our models on the Leonhard Cluster (ETHZ GPU Cluster). The GPUs used for training are GTX 1080Ti and RTX 2080Ti, one at a time, both with 11Gigabyte Memory which we used to the full extent. For the base model, this allows a batch size of 32 for the biggest model the batchsize has to be smaller than 16. Training time was around 48h for each model. We used Adam optimizer with a learning rate of $1e-4$. All our models lay within $2'000'000 \pm 10\%$ parameters.

C. *Object-level 3D reconstruction comparison*

In this section, we give examples of our best generated meshes versus ground truth meshes and baseline meshes to show our best model can reconstruct details better than the baseline method.

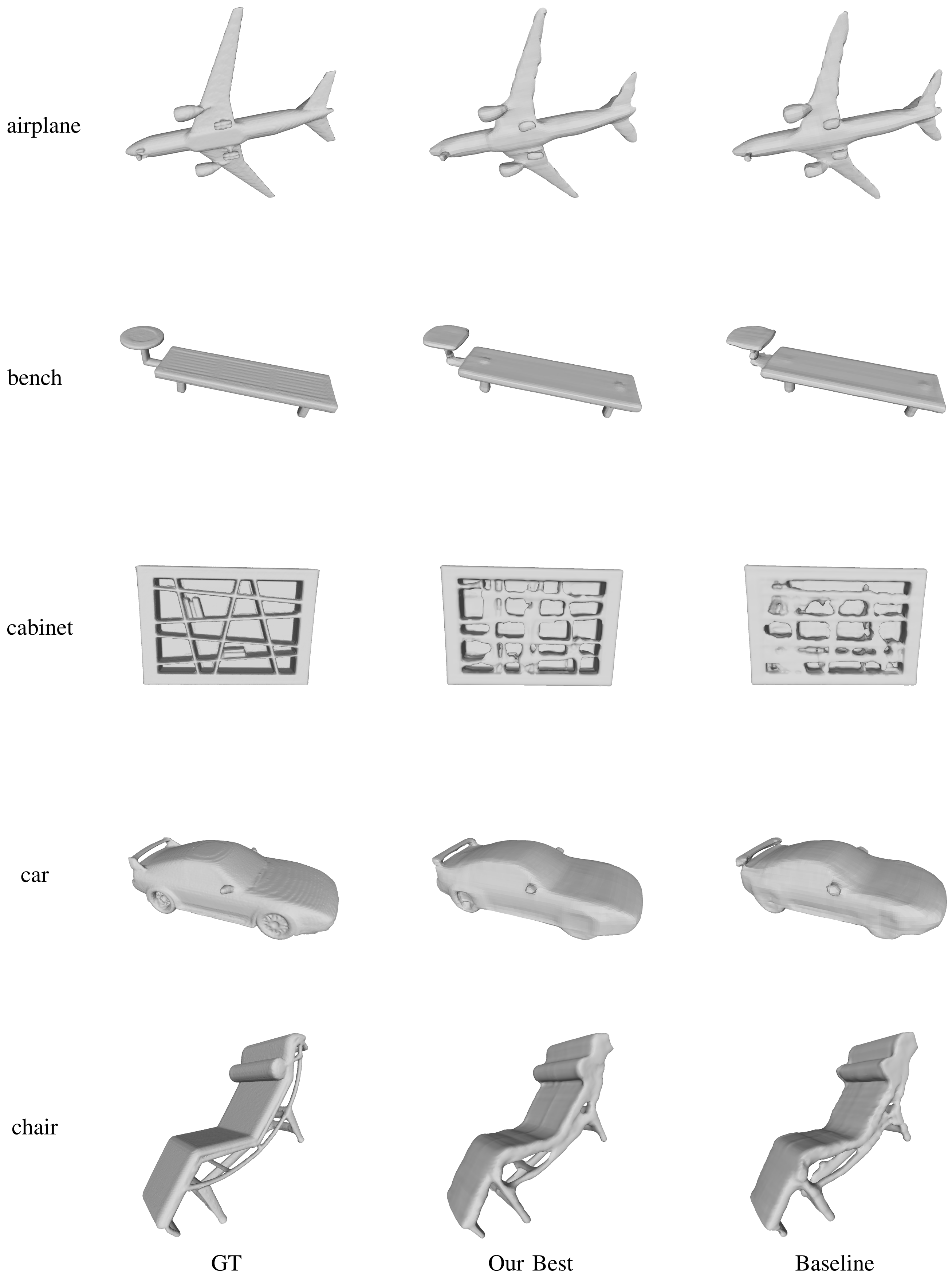

Fig. 7: A comparison of generated meshes of all the 13 classes

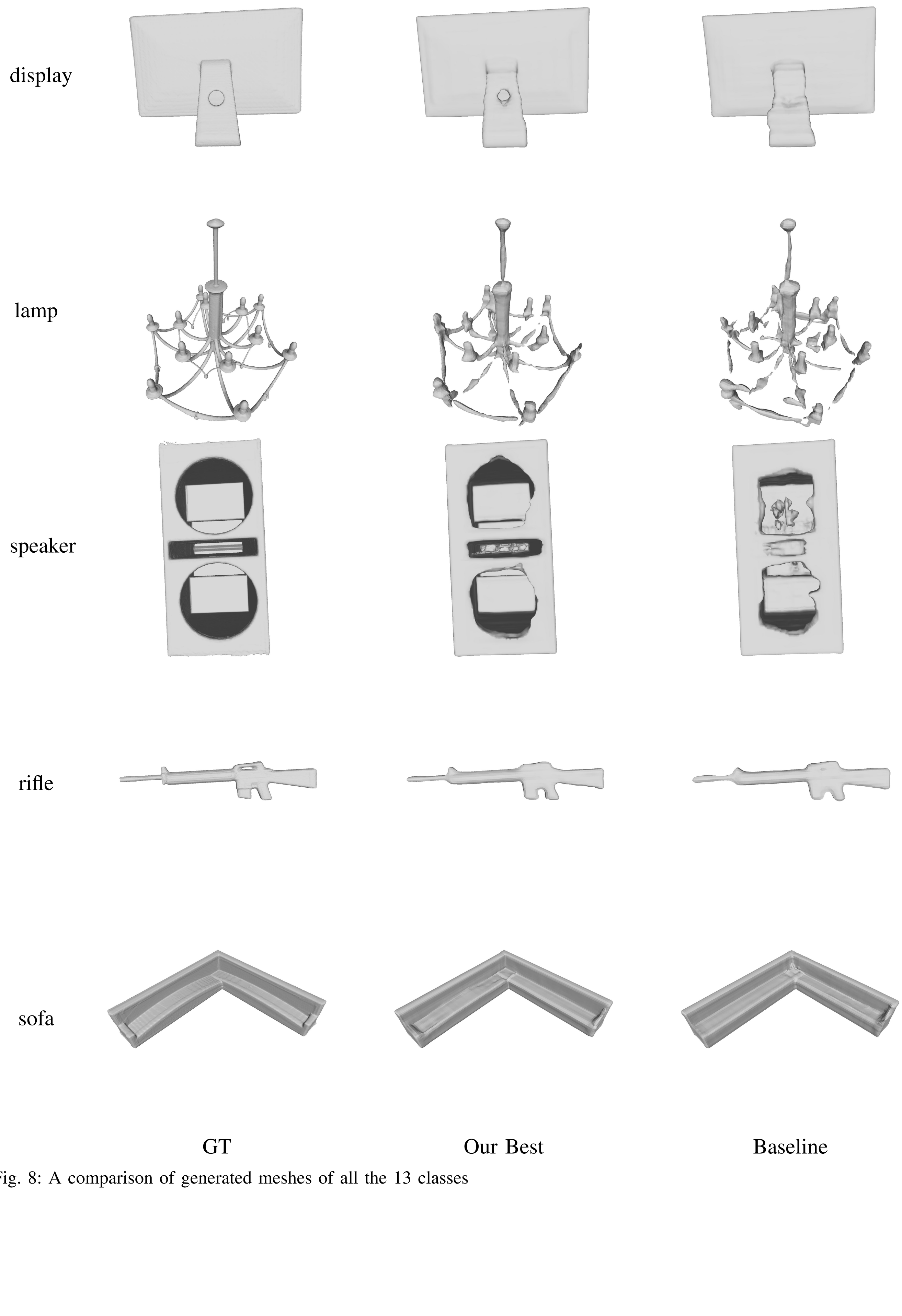

Fig. 8: A comparison of generated meshes of all the 13 classes

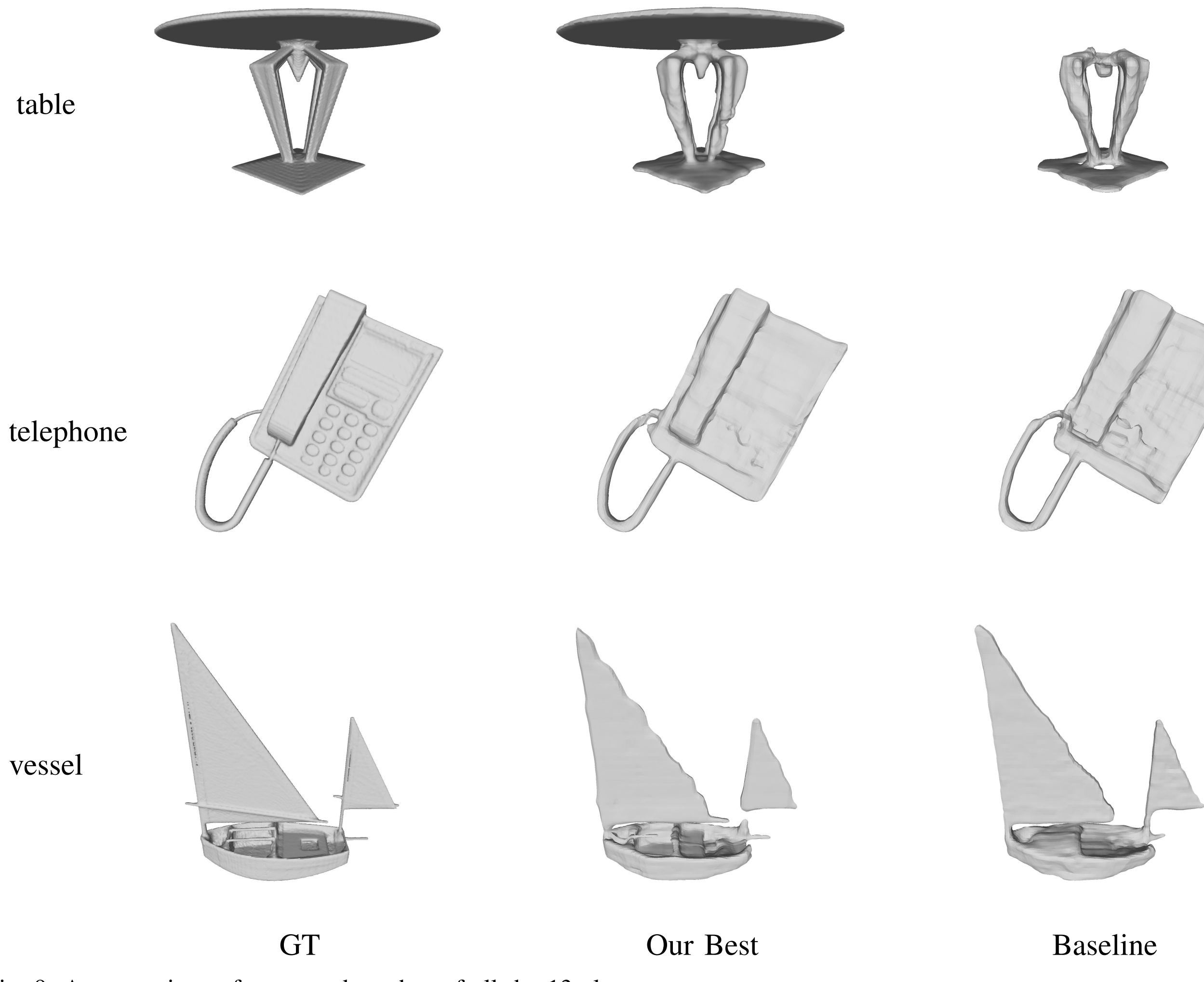

Fig. 9: A comparison of generated meshes of all the 13 classes