Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction

Bo Yang [1] **Sen Wang** [2] **Andrew Markham** [1] **Niki Trigoni** [1]

[1]University of Oxford
[2]Heriot-Watt University
firstname.lastname@cs.ox.ac.uk

Abstract

We study the problem of recovering an underlying 3D shape from a set of images. Existing learning based approaches usually resort to recurrent neural nets, *e.g.*, GRU, or intuitive pooling operations, *e.g.*, max/mean pooling, to fuse multiple deep features encoded from input images. However, GRU based approaches are unable to consistently estimate 3D shapes given the same set of input images as the recurrent unit is permutation variant. It is also unlikely to refine the 3D shape given more images due to the long-term memory loss of GRU. The widely used pooling approaches are limited to capturing only the first order/moment information, ignoring other valuable features. In this paper, we present a new feed-forward neural module, named **AttSets**, together with a dedicated training algorithm, named **JTSO**, to attentionally aggregate an arbitrary sized deep feature set for multi-view 3D reconstruction. AttSets is permutation invariant, computationally efficient, flexible and robust to multiple input images. We thoroughly evaluate various properties of AttSets on large public datasets. Extensive experiments show AttSets together with JTSO algorithm[1]significantly outperforms existing aggregation approaches.

1 Introduction

Given a set of images, to recover a geometric representation of the 3D world is classically defined as multi-view 3D reconstruction in computer vision. Traditional pipelines such as Structure from Motion (SfM) [20] and visual Simultaneous Localization and Mapping (vSLAM) [3] typically rely on hand-crafted feature extraction and matching across multiple views to reconstruct the underlying 3D model. However, if the multiple viewpoints are separated by large baseline, the feature matching approach is extremely challenging due to significant appearance changes or self occlusions [18]. Furthermore, the reconstructed 3D shape is usually a sparse point cloud without geometric details.

Recently, a number of deep learning approaches, such as 3D-R2N2 [6], LSM [15], DeepMVS [11] and RayNet [21] have been proposed to estimate the 3D dense shape from multiple images and have shown encouraging results. Both 3D-R2N2 [6] and LSM [15] formulate multi-view reconstruction as a sequence learning problem, and leverage RNNs, particularly GRU, to fuse the multiple deep features extracted by a shared encoder for input images. However, there are three limitations. First, the recurrent network is permutation variant, as the order of the input sequence matters [27]. Therefore, inconsistent 3D shapes are estimated from the same image set with different permutations. Second, it is difficult to capture long-term dependencies in the sequence because of gradient vanishing or exploding [2][16], so the estimated 3D shapes are unlikely to be refined even if more images are given during training and testing. Third, the RNN unit is inefficient as each element of the input sequence must be sequentially processed without parallelization [19], so is time-consuming to generate the final 3D shape given a sequence of images. The recent DeepMVS [11] applies max pooling to aggregate deep features across a set of unordered images for multi-view stereo reconstruction, while RayNet [21] takes use of average pooling to aggregate the deep features corresponding to the same voxel

[1]Code and data are available at *https://github.com/Yang7879/AttSets*

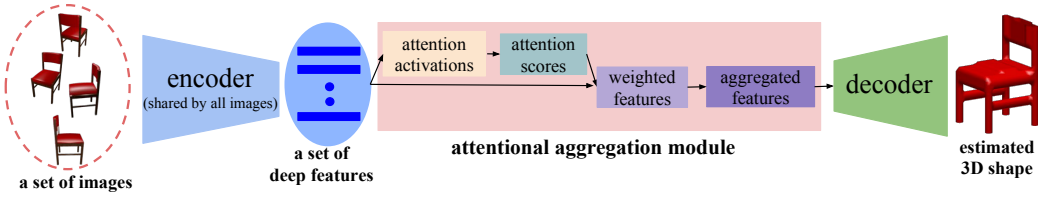

Figure 1: Overview of an attentional aggregation module for multi-view 3D reconstruction.

from multiple images to recover a dense 3D model. Although max and average poolings do not suffer from above limitations of RNN, they only capture the first order or moment information from the large deep feature set, totally ignoring other features which might be valuable for accurate 3D shape estimation.

In this paper, we introduce a simple yet efficient attentional aggregation module, named **AttSets**, that can be easily included in an existing multi-view 3D reconstruction network to aggregate an arbitrary number of elements of a deep feature set, completely replacing the RNN module or max/average pooling operations. Inspired by the attention mechanism which shows great success in natural language processing [1][23], image captioning [29], *etc.*, we design a feed-forward neural layer that can automatically learn to aggregate each element of the input deep feature set. In particular, as shown in Figure 1, given a variable sized deep feature set, which are usually learnt view-invariant visual representations from a shared encoder [21], our AttSets module firstly learns an **attention activation** for each latent feature through a standard neural layer (*e.g.*, a fully connected layer, a 2D or 3D convolutional layer), after which an **attention score** is computed for the corresponding feature. Subsequently, the attention scores are simply multiplied by the original elements of the deep feature set, generating a set of **weighted features**. At last, the weighted features are aggregated across different elements of the deep feature set, producing a fixed size of **aggregated features** which are then fed into a decoder to estimate 3D shapes. To enable AttSets to learn the desired attention scores for deep feature sets, we further propose a **joint-training and separate-optimizing (JTSO)** algorithm that decouples the base encoder-decoder to learn deep features, while dedicating the AttSets module to learning attention scores for feature sets.

Our AttSets is designed with the following desirable properties and advantages over existing approaches for multi-view 3D reconstruction:

- Compared with RNN approaches, AttSets is permutation invariant and is able to capture value information across a large number of elements of a set. AttSets consists of standard forward neural nets, and therefore can be parallelized instead of sequentially computed.
- Compared with max/average pooling, AttSets learns scores for all elements of the deep feature sets, attentionally aggregating useful features instead of simply capturing the first order/moment information.
- In addition, AttSets is flexible to embed either standard fully connected, or 2D/3D convolutional layers and can be easily plugged into an existing encoder-decoder net to estimate 3D shapes from a variable number of images without increasing notable memory and computation cost.

2 Related Work

(1) Multi-view 3D Reconstruction. 3D shapes can be recovered from multiple color images or depth scans. To estimate the underlying 3D shape from **multiple color images**, classic SfM [20] and vSLAM [3] algorithms firstly extract and match hand-crafted geometric features [10] and then apply bundle adjustment [26] for both shape and camera motion estimation. Ji *et al.* [14] use "maximizing rigidity" for reconstruction, but this requires 2D point correspondences across images. Recent deep neural net based approaches tend to recover dense 3D shapes through learnt features from multiple images and achieve compelling results. To fuse the deep features from multiple images, both 3D-R2N2 [6] and LSM [15] apply the recurrent unit GRU, resulting in the networks being permutation variant and inefficient for aggregating long sequence of images. Recent SilNet [28] and DeepMVS [11] simply use max pooling to preserve the first order information of the deep features of multiple images, while RayNet [21] applies average pooling to reserve the first moment information of multiple deep features. MVSNet [31] proposes a variance-based approach to capture the second moment information for multiple feature aggregation. These pooling techniques only capture partial information, ignoring the majority of the deep features. Recent SurfaceNet [13] and

SuperPixel Soup [17] can reconstruct 3D shapes from two images, but they are unable to process an arbitrary number of images. As to **multiple depth image** reconstruction, the traditional volumetric fusion method [7] integrates multiple viewpoint information by averaging truncated signed distance functions (TSDF). Recent learning based OctNetFusion [24] also takes a similar strategy to integrate multiple depth information. However, this integration might result in information loss since TSDF values are averaged [24].

(2) Deep Learning on Sets. In contrast to traditional approaches operating on fixed dimensional vectors or matrices, deep learning tasks defined on sets usually require learning functions to be permutation invariant and able to process an arbitrary number of elements in a set [32]. Such problems are widespread. Zaheer *et al*. introduce general permutation invariant and equivariant models in [32], and they end up with a **summation pooling** for permutation invariant tasks such as population statistics estimation and point cloud classification. In the very recent GQN [8], summation pooling is also used to aggregate an arbitrary number of orderless images for 3D scene representation. Gardner *et al*. [9] use **average pooling** to integrate an unordered deep feature set for classification task. Su *et al*. [25] use **max pooling** to fuse the deep feature set of multiple views for 3D shape recognition. Similarly, PointNet [22] also uses max pooling to aggregate the set of features learnt from point clouds for 3D classification and segmentation. In fact, the above summation, average, and max pooling techniques are the most common aggregation operators on sets in mathematics [4]. However, these pooling operations ignore a majority of the information of a set and they do not have trainable parameters available for the network to learn.

(3) Attention Mechanism. The attention mechanism was originally proposed for natural language processing [1]. Being coupled with RNNs, it achieves compelling results in neural machine translation [1], image captioning [29], image question answering [30], *etc*. Little work has been done to explore attention mechanisms for learning tasks on sets, which usually requires permutation invariance and adaptability to variable cardinality. Compared with the original attention mechanism, our AttSets does not couple with RNNs. Instead, AttSets is a simplified feed-forward module which shares similar concepts with [23][12]. However, [23] aims to solve the long-term memory problem for sequence learning, while [12] focuses on multiple instance learning for classification. By contrast, our AttSets and the dedicated JTSO algorithm are designed for general learning tasks on sets.

3 AttSets

3.1 Problem Definition

This paper considers the problem of aggregating an arbitrary number of elements of a set $\mathcal{A}$ to a fixed dimensional output $\boldsymbol{y}$. Usually, each element of set $\mathcal{A}$ is a feature vector extracted from a shared encoder, while the fixed dimensional $\boldsymbol{y}$ is fed into a subsequent decoder, such that the whole network can process an arbitrary number of input elements with a fixed and predefined network architecture.

Given N elements in the input deep feature set $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_n \in \mathbb{R}^{1\times D}$, where N is an arbitrary value, while D is fixed for a specific encoder, and the output $\boldsymbol{y} \in \mathbb{R}^{K\times D}$, where K is also fixed and predefined for the subsequent decoder, our task is to design an aggregation function f with learnable weights $\boldsymbol{W}$: $\boldsymbol{y} = f(\mathcal{A}, \boldsymbol{W})$, which should be permutation invariant, *i.e*., for any permutation π:

$$f(\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}, \boldsymbol{W}) = f(\{\boldsymbol{x}_{\pi(1)}, \boldsymbol{x}_{\pi(2)}, \cdots, \boldsymbol{x}_{\pi(N)}\}, \boldsymbol{W}) \tag{1}$$

Basically, the max/mean/sum pooling operations are the simplest instantiations of function f where $\boldsymbol{W} \in \varnothing$. However, these pooling operations are predefined to capture partial information without trainable weights, which is unable to unleash the power of a standard neural net.

3.2 AttSets Module

The basic idea of our AttSets module is to learn an attention score for each latent feature of the whole deep feature set. In this paper, each latent feature refers to each entry of an individual element of the feature set, with an individual element usually represented by a latent vector, *i.e*., $\boldsymbol{x}_n$. The learnt scores can be regarded as a mask that automatically selects useful latent features across the set. The selected features are then summed across multiple elements of the set.

As shown in Figure 2, given a set of features $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_n \in \mathbb{R}^{1\times D}$, AttSets aims to fuse it into a fixed dimensional output $\boldsymbol{y}$, where $\boldsymbol{y} \in \mathbb{R}^{1\times D}$, *i.e*., we set $K = 1$ for simplicity.

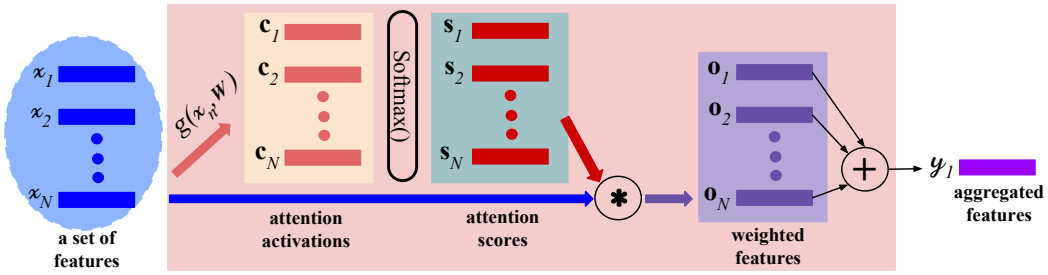

Figure 2: Attentional aggregation module on sets.

First of all, we feed each element of the feature set $\mathcal{A}$ into a shared function g which can be a standard neural layer, *i.e.*, a linear transformation layer optionally followed by a non-linear activation function. Here we use a fully connected layer followed by a $tanh$ layer as an example, the bias term is dropped for simplicity. The output of function g is a set of learnt attention activations $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_N\}$, where

$$\boldsymbol{c}_n = g(\boldsymbol{x}_n, \boldsymbol{W}) = tanh(\boldsymbol{x}_n \boldsymbol{W}), \qquad (\boldsymbol{x}_n \in \mathbb{R}^{1\times D}, \quad \boldsymbol{W} \in \mathbb{R}^{D\times D}, \quad \boldsymbol{c}_n \in \mathbb{R}^{1\times D}) \tag{2}$$

Secondly, the learnt attention activations are normalized across the N elements of the set, computing a set of attention scores $\mathcal{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_N\}$. We choose $softmax$ as the normalization operation, so the attention scores for the n^{th} feature element are

$$\boldsymbol{s}_n = [s_n^1, s_n^2, \cdots, s_n^d, \cdots, s_n^D], \qquad s_n^d = \frac{e^{c_n^d}}{\sum_{n=1}^{N} e^{c_n^d}}, \qquad c_n^d \text{ is the } d^{th} \text{ entry of } \boldsymbol{c}_n. \tag{3}$$

Thirdly, the computed attention scores $\mathcal{S}$ are multiplied by their corresponding original feature set $\mathcal{A}$, generating a new set of deep features, denoted as weighted features $\mathcal{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_N\}$, where

$$\boldsymbol{o}_n = \boldsymbol{x}_n * \boldsymbol{s}_n \tag{4}$$

Lastly, the set of weighted features $\mathcal{O}$ are summed up across the total N elements to get a fixed size feature vector, denoted as $\boldsymbol{y}$, where

$$\boldsymbol{y} = [y^1, y^2, \cdots, y^d, \cdots, y^D], \qquad y^d = \sum_{n=1}^{N} o_n^d, \qquad o_n^d \text{ is the } d^{th} \text{ entry of } \boldsymbol{o}_n. \tag{5}$$

In above formulation, we show how AttSets gradually aggregates a set of N feature vectors $\mathcal{A}$ into a single vector $\boldsymbol{y}$, where $\boldsymbol{y} \in \mathbb{R}^{1\times D}$. If we want to learn larger size of aggregated features $\boldsymbol{y}$ where $K > 1$, simply add another parallel branch of layers to learn another set of different attention scores for the original set of features. In this case, the number of learnable weights $\boldsymbol{W}$ and the capacity of AttSets increases accordingly, but the AttSets module is still parallelly computed.

3.3 Permutation Invariance

The output of AttSets module $\boldsymbol{y}$ is permutation invariant with regard to the input deep feature set $\mathcal{A}$. Here is the simple proof where $\boldsymbol{y}$ is a single vector.

$$[y^1, y^2, \cdots, y^d, \cdots, y^D] = f(\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n, \cdots, \boldsymbol{x}_N\}, \boldsymbol{W}) \tag{6}$$

In above Equation 6, the d^{th} entry of the output $\boldsymbol{y}$ is computed as follows:

$$\begin{aligned} y^d &= \sum_{n=1}^{N} o_n^d = \sum_{n=1}^{N} (x_n^d * s_n^d) = \sum_{n=1}^{N} \left(x_n^d * \frac{e^{c_n^d}}{\sum_{n=1}^{N} e^{c_n^d}} \right) = \sum_{n=1}^{N} \left(x_n^d * \frac{e^{tanh(\boldsymbol{x}_n \boldsymbol{w}^d)}}{\sum_{n=1}^{N} e^{tanh(\boldsymbol{x}_n \boldsymbol{w}^d)}} \right) \\ &= \frac{\sum_{n=1}^{N} \left(x_n^d * e^{tanh(\boldsymbol{x}_n \boldsymbol{w}^d)} \right)}{\sum_{n=1}^{N} e^{tanh(\boldsymbol{x}_n \boldsymbol{w}^d)}}, \qquad \boldsymbol{w}^d \text{ is the } d^{th} \text{ column of the weights } \boldsymbol{W}. \end{aligned} \tag{7}$$

Both the denominator and numerator are a summation of a permutation equivariant term in above Equation 7. Therefore the value y^d, also the full vector $\boldsymbol{y}$, is invariant to different permutations of the deep feature set $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n, \cdots, \boldsymbol{x}_N\}$ [32]. For the aggregated feature $\boldsymbol{y} \in \mathbb{R}^{K\times D}$, where $K > 1$, AttSets module is still permutation invariant because all the aggregated feature vectors are parallelly and independently learnt.

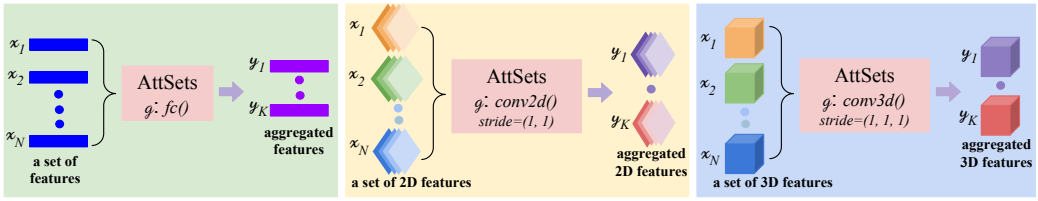

Figure 3: Implementation of AttSets with fully connected layer, 2D ConvNet, and 3D ConvNet.

3.4 Implementation

In above section 3.2, our AttSets aggregates a set of arbitrary number of vector features into a fixed number of vectors, where the attention activation learning function g embeds a fully connected (fc) layer. AttSets is also flexible and able to be easily implemented with both 2D and 3D convolutional neural layers to aggregate both 2D and 3D deep feature sets. Particularly, as shown in Figure 3, to aggregate a set of 2D features, *i.e.*, a tensor of $(width \times height \times channels)$, the attention activation learning function g embeds a standard $conv2d$ layer with a stride of (1×1). Similarly, to fuse a set of 3D features, *i.e.*, a tensor of $(width \times height \times depth \times channels)$, the function g embeds a standard $conv3d$ layer with a stride of $(1 \times 1 \times 1)$. Compared with fc enabled AttSets, the $conv2d$ or $conv3d$ based AttSets tends to have less learnable weights. Note that both the $conv2d$ and $conv3d$ based AttSets are still permutation invariant, as the function g is shared across all elements of the deep feature set and it does not depend on the order of the elements [32].

We already show the implementation of a fc based AttSets to aggregate vector features in previous section 3.2. Similarly, $conv2d$ or $conv3d$ based AttSets can be plugged into a 2D encoder or a 3D decoder to fuse 2D/3D feature sets with minimal deployment cost, which are studied in section 4.2.

3.5 Training Algorithm

Our AttSets module can be easily plugged in an existing encoder-decoder multi-view 3D reconstruction network, replacing the RNN unit or pooling operation. Basically, in an AttSets enabled encoder-decoder net, the encoder-decoder serves as the base architecture to learn visual features for shape estimation, while the AttSets module learns to assign different attention scores to combine those features instead of learning visual features concurrently. As such, the base network tends to have generality with regard to different input image content, while the AttSets module tends to be general regarding arbitrary number of input images.

To train an AttSets enabled network, a naive approach is applying a unified end-to-end training strategy, treating AttSets as a standard layer in the middle. However, the unified training paradigm may optimize the whole network with regard to the statistics of training image batches, resulting in less generality overall. Therefore, we propose a **joint-training separate-optimizing (JTSO)** approach for an AttSets enabled network. In particular, the trainable weights of an encoder-decoder base network are denoted as $\mathbf{\Theta}_{base}$, and the trainable weights of AttSets module are denoted as $\mathbf{\Theta}_{att}$, while the loss function of the whole network is represented by ℓ which is determined by the specific supervision signal of the base network. Our JTSO is shown in Algorithm 1.

Algorithm 1 Joint-training separate-optimizing of an AttSets enabled network. M is batch size, N is image number, k_1 and k_2 are hyperparameters. We use $k_1 = k_2 = 1$ in our experiments.

for number of training iterations **do**

- Sample M sets of images $\{\mathcal{I}_1, \cdots, \mathcal{I}_m, \cdots, \mathcal{I}_M\}$ and sample N images for each set, *i.e.*, $\mathcal{I}_m = \{\boldsymbol{i}_m^1, \cdots, \boldsymbol{i}_m^n, \cdots, \boldsymbol{i}_m^N\}$. Sample M 3D shape labels $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m, \cdots, \boldsymbol{v}_M\}$.

for k_1 steps **do**

- Update the base network by ascending its stochastic gradient:

$$\nabla_{\Theta_{base}} \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left[\ell(\hat{\boldsymbol{v}}_m^n, \boldsymbol{v}_m)\right], \ where\ \hat{\boldsymbol{v}}_m^n \text{ is the estimated 3D shape of single image } \{\boldsymbol{i}_m^n\}.$$

for k_2 steps **do**

- Update the AttSets module by ascending its stochastic gradient:

$$\nabla_{\Theta_{att}} \frac{1}{M} \sum_{m=1}^{M} \left[\ell(\hat{\boldsymbol{v}}_m, \boldsymbol{v}_m)\right], where\ \hat{\boldsymbol{v}}_m \text{ is the estimated 3D shape of the image set } \mathcal{I}_m.$$

The gradient-based updates can use any algorithm.

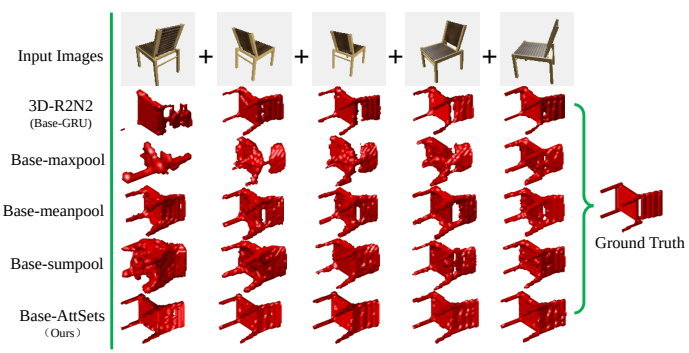

Figure 4: Qualitative results of multi-view reconstruction from different approaches.

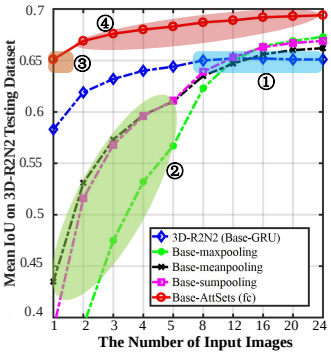

Figure 5: Mean IoU of different approaches.

Table 1: Mean IoU for multi-view reconstruction of all 13 categories in 3D-R2N2 testing dataset.

	1 view	2 views	3 views	4 views	5 views	8 views	12 views	16 views	20 views	24 views
3D-R2N2 (Base-GRU) [6]	0.583	0.619	0.632	0.640	0.644	0.650	0.652	0.652	0.651	0.651
Base-maxpooling	0.246	0.392	0.475	0.532	0.567	0.623	0.650	0.664	0.669	0.673
Base-meanpooling	0.435	0.531	0.573	0.597	0.610	0.635	0.647	0.656	0.660	0.662
Base-sumpooling	0.389	0.516	0.568	0.596	0.611	0.639	0.653	0.663	0.667	0.669
Base-AttSets(Ours)	**0.651**	**0.669**	**0.676**	**0.680**	**0.683**	**0.687**	**0.689**	**0.692**	**0.693**	**0.694**

Table 2: Per-category mean IoU for single view reconstruction on 3D-R2N2 testing dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
3D-R2N2 (Base-GRU) [6]	0.531	0.456	0.728	0.811	0.487	0.501	0.398	0.669	0.554	0.635	0.521	0.688	0.540
Base-maxpooling	0.261	0.093	0.318	0.364	0.194	0.175	0.327	0.341	0.151	0.146	0.223	0.258	0.252
Base-meanpooling	0.404	0.269	0.564	0.663	0.378	0.328	0.373	0.555	0.290	0.392	0.398	0.408	0.386
Base-sumpooling	0.348	0.240	0.552	0.584	0.356	0.298	0.319	0.542	0.193	0.387	0.336	0.383	0.329
Base-AttSets(Ours)	**0.607**	**0.569**	**0.788**	**0.848**	**0.571**	**0.572**	**0.452**	**0.723**	**0.610**	**0.712**	**0.597**	**0.758**	**0.609**

4 Evaluation

To evaluate the performance and various properties of AttSets, we choose 3D-R2N2 [6] as the base network. The original 3D-R2N2 consists of (1) a shared ResNet-based 2D encoder which encodes a size of $127 \times 127 \times 3$ images into 1024 dimensional latent vectors, (2) a GRU module which fuses N 1024 dimensional latent vectors into a single $4 \times 4 \times 4 \times 128$ tensor, and (3) a ResNet-based 3D decoder which decodes the single tensor into a $32 \times 32 \times 32$ voxel grid representing the 3D shape. The released dataset consists of 13 categories of $43,783$ common objects with synthesized RGB images from the large scale ShapeNet 3D repository [5]. For each 3D object, 24 images are rendered from different viewing angles circling around. The train/test dataset split is $0.8 : 0.2$. Intersection-over-Union (IoU) is used to evaluate the reconstruction performance [6].

4.1 Comparison with GRU and Pooling Operations

To compare with the existing GRU module [6][15] and the widely used max/mean/sum pooling operations [28][11][21], we replace the GRU module of 3D-R2N2 by our fc based AttSets and the three max/mean/sum poolings, keeping all other neural layers untouched. Architecture details are in the Appendix A. All networks are trained from scratch, with the image number N=24 and learning rate $= 0.0001$ which are the same as in 3D-R2N2 [6], and the batch size M=2, on a single Titan X GPU. As there is no validation dataset split, to calculate the IoU scores, we independently search the optimal binarization threshold value from $0.2 \sim 0.8$ with a step 0.05 for all approaches for fair comparison. In our experiments, we found that all optimal thresholds of different approaches end up with 0.3 or 0.35. We use the author released well-trained 3D-R2N2 weights to calculate its IoU.

Aggregation Performance. Table 1 shows the mean IoU scores of different approaches on all 13 categories in 3D-R2N2 testing dataset, while Figure 5 shows the trends of IoU changes. Table 2 highlights per-category IoU scores for single view reconstruction. During testing, permutation of the images are the same for different approaches for fair comparison. Figure 4 shows the estimated 3D shapes regarding an increasing number of images for different approaches. Our AttSets based approach outperforms all others by a large margin for either single view or multi view reconstruction, and generates much more compelling 3D shapes.

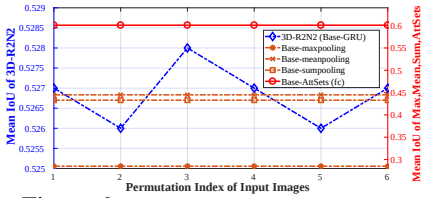

Figure 6: Mean IoU for different permutations.

Table 3: Mean time consumption for a single object estimation from different number of images (milliseconds).

number of input images	1	4	8	12	16	20	24
3D-R2N2(Base-GRU) [6]	6.9	11.2	17.0	22.8	28.8	34.7	40.7
Base-maxpooling	6.4	**10.0**	**15.1**	20.2	**25.3**	**30.2**	**35.4**
Base-meanpooling	**6.3**	10.1	**15.1**	**20.1**	**25.3**	30.3	35.5
Base-sumpooling	6.4	10.1	**15.1**	**20.1**	**25.3**	30.3	35.5
Base-AttSets(Ours)	7.7	11.0	16.3	21.2	26.3	31.4	36.4

Analysis. (1) GRU based approach can generate reasonable 3D shapes given few images, but the performance saturates quickly after being given more images, *e.g.*, 8 views, because the recurrent unit is hardly to capture features from longer image sequences (Figure 5 ①). (2) All pooling based approaches are able to estimate satisfactory 3D shapes until being given enough images, *e.g.*, 12 views, but they are unlikely to learn reasonable shapes given only few images (Figure 5 ②), because the pooled features from fewer images are unlikely to be as general and representative as pooled features from more images. (3) Our JTSO algorithm completely decouples the base network to learn visual features for accurate single view reconstruction (Figure 5 ③), while the trainable weights of AttSets module are separately responsible for learning attention scores for better multi-view reconstruction (Figure 5 ④). Therefore, the whole network does not suffer from limitations of GRU or pooling approaches, and can achieve superior performance for either fewer or more image reconstruction. More results are in the Appendix B.1.

Permutation Invariance. To evaluate the permutation invariance of different approaches, we choose the $bench$ category in 3D-R2N2 testing dataset for experiments. In particular, for each object, we randomly select 3 images with 6 different permutations in total for testing. As shown in Figure 6, the mean IoU scores of 3D-R2N2 fluctuates regarding different image permutations, although the model has been officially trained with various image permutations by authors [6]. In contrast, the AttSets based approach is completely not sensitive to input image permutations, and the mean IoU score consistently achieves 0.601. The pooling approaches are also permutation invariant, but their mean IoU scores are all below 0.5. Qualitative results are in Appendix C.

Computation Efficiency. To evaluate the computation and memory cost of AttSets, we implement all nets in Python 2.7 and Tensorflow 1.6 with CUDA 9.0 and cuDNN 7.1 as the back-end driver and library. All approaches share the same base network and run in the same Titan X and software environments. Table 3 shows the average time consumption to reconstruct a single 3D object given different number of images. Our AttSets based approach is as efficient as all pooling based methods, while 3D-R2N2 takes more time when processing an increasing number of images due to the sequential computation mechanism of its GRU module. In terms of the total trainable weights, all pooling based approaches have 16.66 million, while AttSets based net has 17.71 million. By contrast, the original 3D-R2N2 has 34.78 million. Overall, our AttSets module is able to replace the recurrent unit or pooling operations without incurring notable computation and memory cost.

Generality and Robustness. We further evaluate the generality and robustness of an AttSets enabled network. Particularly, all the well-trained models, *i.e.*, trained on 3D-R2N2 training data split, are tested on the synthesized images in LSM dataset [15], which have totally different camera viewing angles and lighting sources, but both 3D-R2N2 and LSM datasets are generated from the same 3D ShapeNet repository [5], *i.e.*, they have the same ground truth labels regarding the same object. Note that, we only borrow the synthesized images from LSM dataset corresponding to the objects in 3D-R2N2 testing data split, *i.e.*, all the trained models have never seen either the style of LSM synthesized images or the 3D object labels before. We resize the images of LSM dataset from 224×224 to 127×127 through linear interpolation. As shown in Table 4, the IoU scores of all pooling approaches increases given more images, but their overall performance is inferior, primarily because the learnt first order/moment features are unable to generalize well across different styles of synthesized images. 3D-R2N2 has fair generality, but it is unable to fuse more information after being given 8 views because of GRU memory loss. By contrast, our AttSets achieves satisfactory generality though being given a single image, and it can effectively aggregate information from more images without suffering from early saturation. More quantitative results are in the Appendix B.2.

Table 4: Mean IoU for multi-view reconstruction of all 13 categories on LSM dataset.

	1 view	2 views	3 views	4 views	5 views	8 views	12 views	16 views	20 views
3D-R2N2 (Base-GRU) [6]	0.389	0.425	0.440	0.454	0.454	0.461	0.463	0.463	0.463
Base-maxpooling	0.173	0.261	0.310	0.342	0.365	0.404	0.431	0.445	0.453
Base-meanpooling	0.269	0.331	0.361	0.380	0.391	0.411	0.423	0.429	0.434
Base-sumpooling	0.273	0.345	0.380	0.400	0.411	0.432	0.447	0.455	0.459
Base-AttSets(Ours)	**0.406**	**0.447**	**0.467**	**0.479**	**0.487**	**0.500**	**0.506**	**0.511**	**0.514**

Table 5: Mean IoU for AttSets variants on all 13 categories in 3D-R2N2 testing dataset.

	1 view	2 views	3 views	4 views	5 views	8 views	12 views	16 views	20 views	24 views
Base-AttSets ($conv2d$)	**0.651**	0.655	0.660	0.665	0.667	0.675	0.679	0.684	0.685	0.686
Base-AttSets ($conv3d$)	**0.651**	**0.670**	**0.677**	**0.680**	0.682	0.686	0.687	0.689	0.690	0.690
Base-AttSets (fc)	**0.651**	0.669	0.676	**0.680**	**0.683**	**0.687**	**0.689**	**0.692**	**0.693**	**0.694**

Table 6: Mean IoU for different training algorithms on all 13 categories in 3D-R2N2 testing dataset.

	1 view	2 views	3 views	4 views	5 views	8 views	12 views	16 views	20 views	24 views
Base-AttSets (JTO)	0.307	0.437	0.516	0.563	0.595	0.639	0.659	0.673	0.677	0.680
Base-AttSets (JTSO)	**0.651**	**0.669**	**0.676**	**0.680**	**0.683**	**0.687**	**0.689**	**0.692**	**0.693**	**0.694**

We also evaluate all approaches on real world RGB images and our AttSets are fairly robust and outperforms the others. Qualitative results are in the Appendix C.

4.2 Comparison between Variants of AttSets

We further compare the aggregation performance of fc, $conv2d$ and $conv3d$ based AttSets modules. The fc based AttSets net is the same as in section 4.1. The $conv2d$ based AttSets is plugged into the middle of the 2D encoder, fusing a $(N, 4, 4, 256)$ tensor into $(1, 4, 4, 256)$, where N is an arbitrary image number. The $conv3d$ based AttSets is plugged into the middle of the 3D decoder, integrating a $(N, 8, 8, 8, 128)$ tensor into $(1, 8, 8, 8, 128)$. All other layers of these variants are the same. Architecture details are in the Appendix D. Table 5 shows the mean IoU scores of three variants on 3D-R2N2 testing dataset and more results are in the Appendix E. fc and $conv3d$ based variants achieve similar IoU scores for either single or multi view 3D reconstruction, demonstrating the superior aggregation capability of AttSets. In the meantime, we observe that the overall performance of $conv2d$ based AttSets net is slightly less effective compared with the other two. One possible reason is that the 2D feature set has been aggregated at the early layer of the network, resulting in features being lost early. Figure 7 visualizes the learnt attention scores for a 2D feature set, *i.e.*, $(N, 4, 4, 256)$ features, via the $conv2d$ based AttSets net. To visualize 2D feature scores, we average the scores along the channel axis and then roughly trace back the spatial locations of those scores corresponding to the original input. The more visual information the input image has, the higher attention scores are automatically learnt by AttSets for the corresponding latent features. For example, the fourth image has richer visual information than the third image, so its attention scores are higher. More visualization results are in the Appendix F. Note that, for a specific base network, there are many potential locations to plug in AttSets and it is also possible to plug multiple AttSets modules into the same net. To search the optimal location and strategy to integrate AttSets is left for our future work.

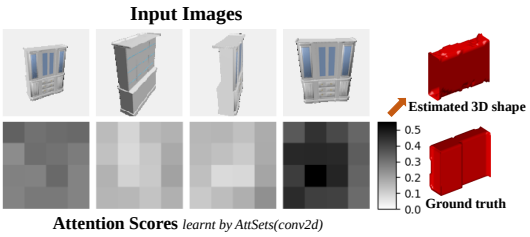

Figure 7: Learnt attention scores for deep feature sets via $conv2d$ based AttSets.

4.3 Impact of JTSO Algorithm

In this section, we investigate the impact of our JTSO algorithm by comparing it with the standard end-to-end joint training and optimizing approach (JTO). Particularly, in JTO, all parameters Θ_{base} and Θ_{att} of the same fc based AttSets net are jointly trained and optimized with a single loss from scratch. As its IoU scores shown in Table 6, the JTO training approach tends to optimize the whole net regarding the training multi-view batches, thus being unable to generalize well for fewer images. Basically, the network itself is unable to dedicate the base layers to learning general features, while the AttSets module to learning attention scores for different elements of a set, if it is not trained with the JTSO algorithm. This issue also exists in the widely used pooling based networks which are unable to wisely aggregate an arbitrary number of deep features. However, the pooling based approaches do not have extra trainable weights to deal with multiple elements of a set.

5 Conclusion

In this paper, we present AttSets module together with JTSO training algorithm to aggregate elements of deep feature sets. AttSets has powerful permutation invariance, computation efficiency, robustness, and flexible implementation properties, along with the theory and extensive experiments to support its performance for multi-view 3D reconstruction. Both quantitative and qualitative results explicitly show that AttSets significantly outperforms other widely used aggregation approaches. Our future work is to integrate AttSets into more multi-view reconstruction networks such as LSM [15]. In addition, we also plan to apply AttSets on other general learning tasks on sets [32].

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *ICLR*, 2015.

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[4] T. Calvo, A. Kolesárová, M. Komorníková, and R. Mesiar. Aggregation Operators: Properties, Classes and Construction Methods. *Aggregation Operators: New Trends and Applications*, 97(1):3–104, 2002.

[5] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv*, 2015.

[6] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. *ECCV*, 2016.

[7] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. *SIGGRAPH*, 1996.

[8] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, and M. Garnelo. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[9] A. Gardner, J. Kanno, C. A. Duncan, and R. R. Selmic. Classifying Unordered Feature Sets with Convolutional Deep Averaging Networks. *arXiv*, 2017.

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[11] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang. DeepMVS: Learning Multi-view Stereopsis. *CVPR*, 2018.

[12] M. Ilse, J. M. Tomczak, and M. Welling. Attention-based Deep Multiple Instance Learning. *ICML*, 2018.

[13] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang. SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis. *ICCV*, 2017.

[14] P. Ji, H. Li, Y. Dai, and I. Reid. "Maximizing rigidity" Revisited: a Convex Programming Approach for Generic 3D Shape Reconstruction from Multiple Perspective Views. *ICCV*, 2017.

[15] A. Kar, C. Häne, and J. Malik. Learning a Multi-View Stereo Machine. *NIPS*, 2017.

[16] J. F. Kolen and S. C. Kremer. Gradient Flow in Recurrent Nets: The Difficulty of Learning Long Term Dependencies. *A Field Guide to Dynamical Recurrent Networks*, 2001.

[17] S. Kumar, Y. Dai, and H. Li. Monocular Dense 3D Reconstruction of a Complex Dynamic Scene from Two Perspective Frames. *ICCV*, 2017.

[18] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[19] E. Martin and C. Cundy. Parallelizing Linear Recurrent Neural Nets over Sequence Length. *ICLR*, 2018.

[20] O. Ozyesil, V. Voroninski, R. Basri, and A. Singer. A Survey of Structure from Motion. *Acta Numerica*, 26:305–364, 2017.

[21] D. Paschalidou, A. O. Ulusoy, C. Schmitt, L. V. Gool, and A. Geiger. RayNet: Learning Volumetric 3D Reconstruction with Ray Potentials. *CVPR*, 2018.

[22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, 2017.

[23] C. Raffel and D. P. W. Ellis. Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems. *ICLR Workshops*, 2016.

[24] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger. OctNetFusion: Learning Depth Fusion from Data. *3DV*, 2017.

[25] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *ICCV*, 2015.

[26] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. *Vision Algorithms: Theory and Practice*, pages 298–372, 1999.

[27] O. Vinyals, S. Bengio, and M. Kudlur. Order Matters: Sequence to Sequence for Sets. *ICLR*, 2016.

[28] O. Wiles and A. Zisserman. SilNet : Single- and Multi-View Reconstruction by Learning from Silhouettes. *BMVC*, 2017.

[29] K. Xu, J. L. Ba, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *ICML*, 2015.

[30] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked Attention Networks for Image Question Answering. *CVPR*, 2016.

[31] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. MVSNet: Depth Inference for Unstructured Multi-view Stereo. *arXiv*, 2018.

[32] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep Sets. *NIPS*, 2017.

Appendix:

A Architecture of 3D-R2N2, Base-max/mean/sum pool, and Base-AttSets

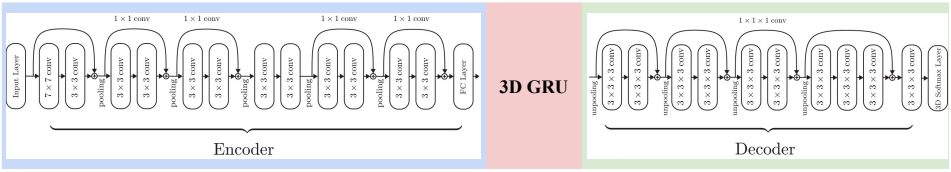

Figure 8: Network architecture of 3D-R2N2. The encoder and decoder parts are extracted from the original paper.

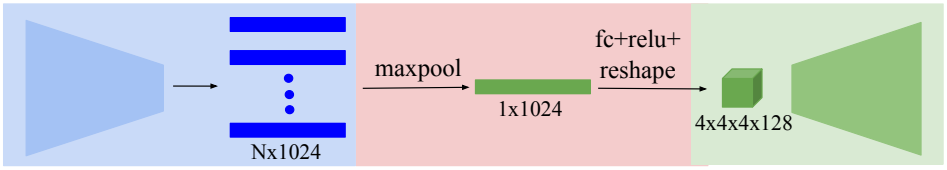

Figure 9: Network architecture of Base-maxpool. The encoder and decoder parts are identical to those in 3D-R2N2.

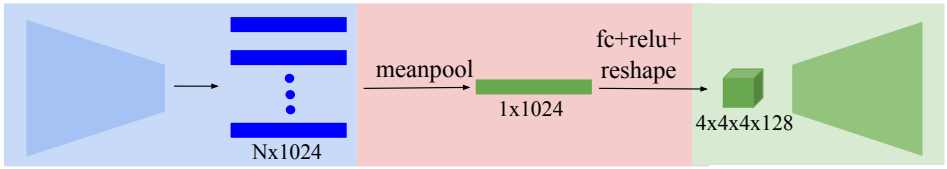

Figure 10: Network architecture of Base-meanpool. The encoder and decoder parts are identical to those in 3D-R2N2.

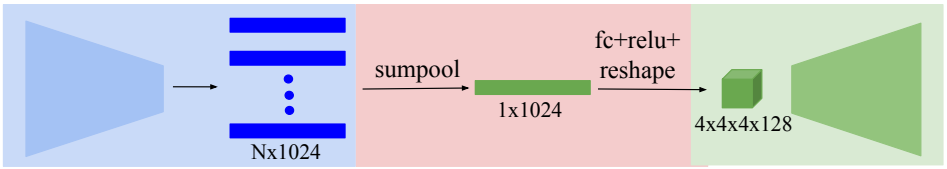

Figure 11: Network architecture of Base-sumpool. The encoder and decoder parts are identical to those in 3D-R2N2.

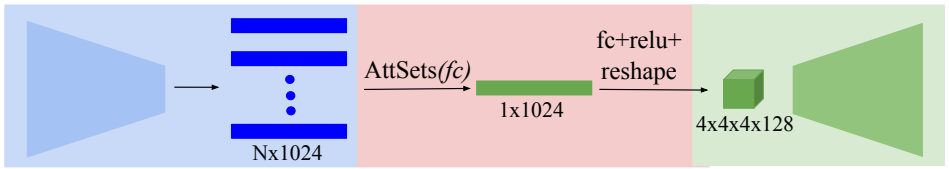

Figure 12: Network architecture of fc based Base-AttSets. The encoder and decoder parts are identical to those in 3D-R2N2.

B Per-category Mean IoU for Multi-view Reconstruction

B.1 The following Table 7, 8 and 9 show the per-category mean IoU scores of different approaches for multi-view reconstruction on 3D-R2N2 testing dataset.

Table 7: Per-category mean IoU for 5-view reconstruction on 3D-R2N2 testing dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
3D-R2N2 (Base-GRU)	0.583	0.541	0.777	0.847	0.563	0.589	0.438	0.723	0.606	0.711	0.589	0.780	0.618
Base-maxpooling	0.515	0.407	0.687	0.798	0.495	0.467	0.410	0.657	0.490	0.573	0.511	0.656	0.567
Base-meanpooling	0.559	0.500	0.744	0.826	0.538	0.529	0.428	0.699	0.545	0.661	0.543	0.719	0.580
Base-sumpooling	0.567	0.498	0.744	0.833	0.533	0.516	0.420	0.698	0.537	0.672	0.546	0.711	0.578
Base-AttSets(Ours)	**0.635**	**0.610**	**0.808**	**0.865**	**0.610**	**0.627**	**0.472**	**0.759**	**0.629**	**0.750**	**0.635**	**0.808**	**0.643**

Table 8: Per-category mean IoU for 12-view reconstruction on 3D-R2N2 testing dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
3D-R2N2 (Base-GRU)	0.592	0.557	0.783	0.851	0.576	0.603	0.442	0.731	0.612	0.724	0.595	0.792	0.627
Base-maxpooling	0.590	0.542	0.782	0.849	0.592	0.550	0.459	0.747	0.600	0.722	0.582	0.761	0.621
Base-meanpooling	0.586	0.551	0.782	0.848	0.592	0.577	0.459	0.751	0.582	0.723	0.573	0.768	0.603
Base-sumpooling	0.593	0.558	0.782	0.854	0.591	0.572	0.457	0.756	0.591	0.728	0.586	0.767	0.609
Base-AttSets(Ours)	**0.643**	**0.622**	**0.814**	**0.868**	**0.619**	**0.639**	**0.475**	**0.764**	**0.635**	**0.759**	**0.639**	**0.820**	**0.648**

Table 9: Per-category mean IoU for 20-view reconstruction on 3D-R2N2 testing dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
3D-R2N2 (Base-GRU)	0.589	0.554	0.781	0.851	0.577	0.605	0.440	0.729	0.611	0.725	0.595	0.790	0.629
Base-maxpooling	0.606	0.574	0.800	0.857	0.625	0.579	0.488	0.776	0.617	0.747	0.593	0.770	0.633
Base-meanpooling	0.592	0.568	0.794	0.853	0.618	0.598	0.486	0.774	0.587	0.742	0.581	0.772	0.609
Base-sumpooling	0.598	0.579	0.791	0.859	0.617	0.595	0.488	0.779	0.599	0.743	0.595	0.777	0.618
Base-AttSets(Ours)	**0.635**	**0.620**	**0.828**	**0.869**	**0.637**	**0.640**	**0.490**	**0.790**	**0.626**	**0.764**	**0.638**	**0.827**	**0.639**

B.2 The following Table 10, 11, 12 and 13 show the per-category mean IoU scores of different approaches for multi-view reconstruction on LSM dataset.

Table 10: Per-category mean IoU for single view reconstruction on LSM dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
3D-R2N2 (Base-GRU)	0.445	0.293	0.371	0.636	0.318	0.238	0.317	0.378	0.458	0.329	**0.280**	0.326	0.436
Base-maxpooling	0.279	0.080	0.135	0.296	0.112	0.109	0.250	0.151	0.160	0.097	0.099	0.156	0.232
Base-meanpooling	0.310	0.172	0.263	0.453	0.248	0.167	0.265	0.283	0.196	0.206	0.200	0.176	0.289
Base-sumpooling	0.259	0.167	0.365	0.461	0.267	0.196	0.242	0.382	0.128	0.249	0.198	0.227	0.255
Base-AttSets(Ours)	**0.450**	**0.300**	**0.422**	**0.657**	**0.349**	**0.293**	**0.340**	**0.433**	**0.490**	**0.401**	0.267	**0.379**	**0.485**

Table 11: Per-category mean IoU for 5-view reconstruction on LSM dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
3D-R2N2 (Base-GRU)	0.513	0.387	0.385	0.731	0.373	0.253	0.355	0.389	0.508	0.390	**0.360**	0.335	0.520
Base-maxpooling	0.451	0.254	0.288	0.588	0.336	0.215	0.322	0.319	0.398	0.275	0.270	0.278	0.431
Base-meanpooling	0.471	0.312	0.321	0.638	0.367	0.236	0.312	0.342	0.390	0.326	0.290	0.249	0.443
Base-sumpooling	0.466	0.339	0.454	0.670	0.381	0.289	0.303	0.437	0.394	0.360	0.306	0.317	0.440
Base-AttSets(Ours)	**0.525**	**0.396**	**0.477**	**0.753**	**0.427**	**0.359**	**0.390**	**0.480**	**0.556**	**0.494**	0.349	**0.503**	**0.563**

Table 12: Per-category mean IoU for 12-view reconstruction on LSM dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
3D-R2N2 (Base-GRU)	0.528	0.400	0.381	0.740	0.382	0.270	0.358	0.377	0.499	0.404	**0.375**	0.331	0.534
Base-maxpooling	0.491	0.358	0.363	0.676	0.400	0.295	0.355	0.392	0.490	0.382	0.332	0.359	0.457
Base-meanpooling	0.496	0.356	0.362	0.678	0.400	0.268	0.331	0.380	0.426	0.369	0.318	0.289	0.462
Base-sumpooling	0.492	0.391	0.492	0.703	0.424	0.339	0.347	0.473	0.450	0.392	0.340	0.363	0.461
Base-AttSets(Ours)	**0.544**	**0.418**	**0.499**	**0.770**	**0.456**	**0.385**	**0.407**	**0.494**	**0.549**	**0.522**	0.368	**0.531**	**0.571**

Table 13: Per-category mean IoU for 20-view reconstruction on LSM dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
3D-R2N2 (Base-GRU)	0.526	0.401	0.372	0.743	0.381	0.262	0.357	0.373	0.490	0.416	**0.381**	0.325	0.536
Base-maxpooling	0.512	0.400	0.384	0.701	0.426	0.308	0.361	0.409	0.523	0.436	0.346	0.390	0.476
Base-meanpooling	0.507	0.370	0.369	0.694	0.410	0.273	0.333	0.391	0.442	0.390	0.325	0.309	0.474
Base-sumpooling	0.501	0.413	0.453	0.715	0.427	0.331	0.347	0.437	0.469	0.403	0.344	0.388	0.468
Base-AttSets(Ours)	**0.551**	**0.432**	**0.511**	**0.775**	**0.467**	**0.387**	**0.407**	**0.490**	**0.554**	**0.537**	0.374	**0.551**	**0.573**

C Qualitative Results on Real World RGB Images

All models, which are trained on synthetic 3D-R2N2 training dataset, are further tested on real world images crowdsourced from Amazon online shops. As shown in Figure 13, 3D-R2N2 estimates inconsistent 3D shapes given different image permutations, while the pooling approaches and our fc based AttSets net generate permutation invariant 3D shapes. AttSets generates more compelling 3D shapes, demonstrating the stronger generality and robustness to real world images compared with 3D-R2N2 and pooling based approaches. Note that, we manually search for appropriate different thresholds for visualization in favor of different approaches. The same binarization threshold is applied for visualizing all different permutation results of 3D-R2N2.

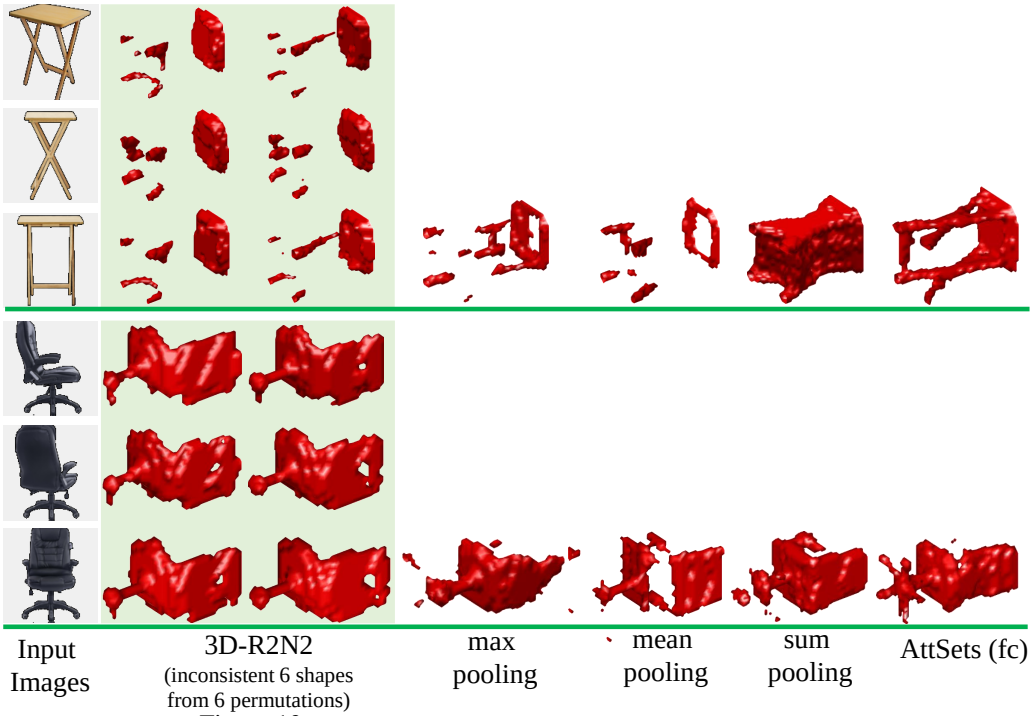

Figure 13: Qualitative results of different approaches on real world RGB images.

D Architecture of fc, $conv2d$ and $conv3d$ based Base-AttSets

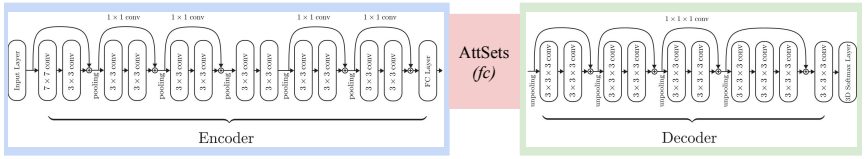

Figure 14: Network architecture of fc based Base-AttSets. The encoder and decoder parts are identical to those in 3D-R2N2.

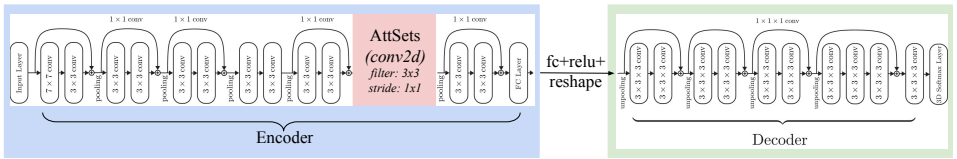

Figure 15: Network architecture of $conv2d$ based Base-AttSets. The encoder and decoder parts are identical to those in 3D-R2N2.

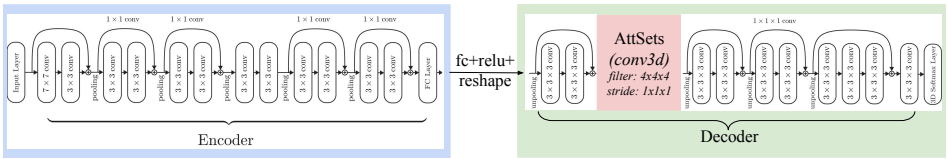

Figure 16: Network architecture of $conv3d$ based Base-AttSets. The encoder and decoder parts are identical to those in 3D-R2N2.

E Per-category Mean IoU of AttSets Variants

The following Table 14, 15 and 16 show the per-category mean IoU scores of fc, $conv2d$ and $conv3d$ based AttSets for multi-view reconstruction on 3D-R2N2 testing dataset.

Table 14: Per-category mean IoU for 5-view reconstruction on 3D-R2N2 testing dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
Base-AttSets ($conv2d$)	0.618	0.588	0.801	0.859	0.586	0.587	0.468	0.738	0.621	0.727	0.620	0.790	0.629
Base-AttSets ($conv3d$)	0.630	0.609	**0.810**	0.864	**0.613**	0.622	0.468	0.755	**0.630**	**0.750**	**0.636**	**0.810**	0.640
Base-AttSets (fc)	**0.635**	**0.610**	0.808	**0.865**	0.610	**0.627**	**0.472**	**0.759**	0.629	**0.750**	0.635	0.808	**0.643**

Table 15: Per-category mean IoU for 12-view reconstruction on 3D-R2N2 testing dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
Base-AttSets ($conv2d$)	0.618	0.605	**0.819**	0.863	0.615	0.604	**0.487**	**0.774**	0.616	0.746	0.628	0.797	0.625
Base-AttSets ($conv3d$)	0.634	0.615	0.814	0.866	**0.619**	0.629	0.469	0.758	0.634	**0.758**	**0.641**	0.816	0.644
Base-AttSets (fc)	**0.643**	**0.622**	0.814	**0.868**	**0.619**	**0.639**	0.475	0.764	**0.635**	**0.759**	0.639	**0.820**	**0.648**

Table 16: Per-category mean IoU for 20-view reconstruction on 3D-R2N2 testing dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
Base-AttSets ($conv2d$)	0.614	0.609	0.827	0.863	0.631	0.623	**0.514**	**0.793**	0.611	0.754	0.629	0.787	0.627
Base-AttSets ($conv3d$)	0.627	0.613	0.827	0.867	**0.634**	0.631	0.485	0.782	0.625	**0.762**	**0.640**	0.815	0.632
Base-AttSets (fc)	**0.635**	**0.620**	0.828	**0.869**	**0.637**	**0.640**	0.490	0.790	**0.626**	**0.764**	0.638	**0.827**	**0.639**

The following Table 17, 18 and 19 show the per-category mean IoU scores of fc, $conv2d$ and $conv3d$ based AttSets for multi-view reconstruction on LSM dataset.

Table 17: Per-category mean IoU for 5-view reconstruction on LSM dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
Base-AttSets ($conv2d$)	0.486	0.386	**0.502**	0.755	0.398	0.334	0.381	**0.503**	0.556	**0.502**	0.343	0.488	0.550
Base-AttSets ($conv3d$)	**0.526**	**0.404**	0.478	**0.768**	**0.436**	0.349	0.385	0.469	**0.571**	0.493	**0.356**	**0.516**	**0.564**
Base-AttSets (fc)	0.525	0.396	0.477	0.753	0.427	**0.359**	**0.390**	0.480	0.556	0.494	0.349	0.503	0.563

Table 18: Per-category mean IoU for 12-view reconstruction on LSM dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
Base-AttSets ($conv2d$)	0.494	0.415	**0.548**	**0.782**	0.447	0.378	0.401	**0.550**	0.543	**0.548**	0.366	**0.552**	0.563
Base-AttSets ($conv3d$)	0.535	0.417	0.498	0.779	**0.463**	0.352	0.402	0.479	**0.562**	0.507	**0.371**	0.551	0.570
Base-AttSets (fc)	**0.544**	**0.418**	0.499	0.770	0.456	**0.385**	**0.407**	0.494	0.549	0.522	0.368	0.531	**0.571**

Table 19: Per-category mean IoU for 20-view reconstruction on LSM dataset.

	plane	bench	cabinet	car	chair	monitor	lamp	speaker	firearm	couch	table	cellphone	watercraft
Base-AttSets ($conv2d$)	0.500	**0.436**	**0.581**	**0.794**	0.462	**0.392**	0.401	**0.556**	0.553	**0.568**	**0.375**	**0.579**	0.570
Base-AttSets ($conv3d$)	0.539	0.421	0.492	0.781	**0.469**	0.353	0.401	0.479	**0.562**	0.512	**0.375**	0.571	0.571
Base-AttSets (fc)	**0.551**	0.432	0.511	0.775	0.467	0.387	**0.407**	0.490	0.554	0.537	0.374	0.551	**0.573**

F Visualization of Learnt Attention Scores

Figure 17 visualizes the learnt attention scores for each latent $1024d$ vector encoded from an input image. The more visual features the input image has, the higher attention scores are automatically learnt for the latent vector of that image overall. For example, the fourth image tends to have more visual information than the third one. Therefore, the fourth column is darker, *i.e.,* higher attention scores, than the third column.

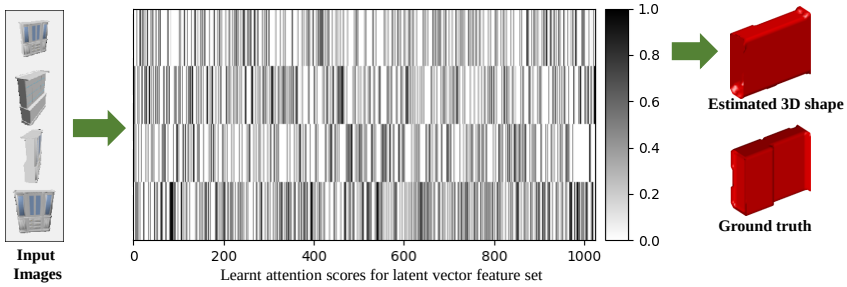

Figure 17: Visualization of the learnt attention scores for each latent $1024d$ vector which is encoded from an input image.