RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion

Muhammad Sarmad
KAIST
South Korea
sarmad@kaist.ac.kr

Hyunjoo Jenny Lee*
KAIST
South Korea
hyunjoo.lee@kaist.ac.kr

Young Min Kim*
KIST, SNU
South Korea
youngmin.kim@snu.ac.kr

A. Implementation Details

The suggested RL-GAN-Net architecture combines three fundamental building blocks as shown in Sec. 3 of the main article. The general form of the architecture is provided in Fig. 1. In this section, we provide details of our implementation for each network.

A.1. AE Details

The AE is composed of an encoder that converts input points P_{in} into GFV, and a decoder network that reverts GFV back to the point cloud domain, as shown in Fig. 1a. The input and output points are an unstructured list of 2048 3D coordinates that is sampled from the underlying 3D structure. The encoder network consists of five 1 D convolution layers with 64, 128, 128, 256, 128 channels respectively, while the decoder consists of FC layers 256, 256 and 6144 channels respectively. Each layer is followed by ReLu. The bottleneck size for AE is 128. We trained the AE to reduce the Chamfer distance (Eq.(1) of the main article) between the input and output point cloud. The Chamfer distance calculation is imported from the implementation[1] of Li et al [3].

A.2. l-GAN Details

l-GAN is composed of the encoder block of AE and a generator and a discriminator, as shown in Fig. 1b. For the generator and the discriminator pair, we adapted the main architecture of the GAN from Zhang et al. [8] and applied in the latent space acquired by the AE. The detailed network architecture for our modified l-GAN pipeline has been shown in Table 1 and 2 respectively. We trained the GAN using WGAN-GP [4] adversarial loss with $\lambda_{gp} = 10$. The total number of iterations was one million. As a typical GAN training, we updated discriminator 5 times for every update of the generator. We used Adam optimizer [5] with β_1=0.5 and β_2=0.9. The learning rate for both generator and discriminator was set to 0.0001. Batch size was set to 50 and number of workers were set to 2. We did not use any learning rate decay.

We selected the dimension of z-vector to be 1. We did this to limit the dimensions of action space for the agent. All the experiments conducted were with a single dimension. We also tested with 6 and 32 dimensions but there was no change in the performance of the GAN or the agent in either case. Therefore we kept the dimension of the z-vector to 1.

We trained the GAN using the dataset generated by passing the ShapeNet point cloud dataset through the encoder of the AE. Therefore, the output dimension of our generator is the same as the bottleneck size of our AE (128).

For l-GAN training, we adopted the self-attention GAN open source code[2] [8].

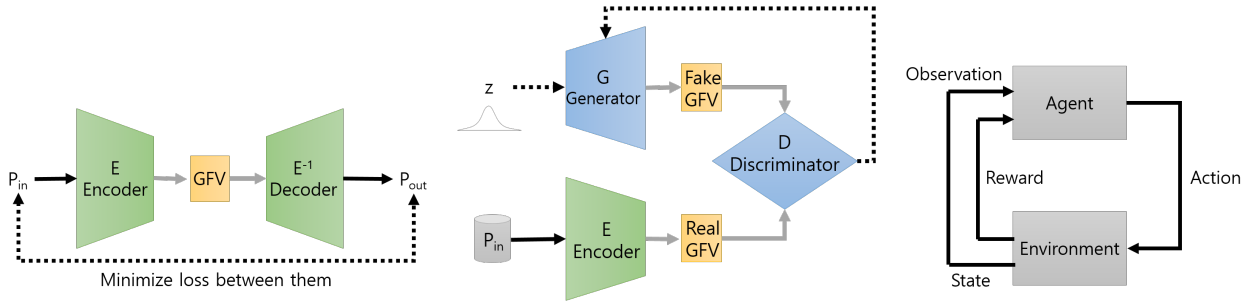
A.3. RL Agent Details

The third element of the basic architecture is RL. The basic RL framework is composed of an agent and the environment as in Fig. 1c. Among many possible variations of the RL agent, we used the actor-critic architecture to enable continuous control of the l-GAN.

Actor and Critic Architecture The actor and critic networks are chosen to be fully connected (FC) layers. The actor has four FC layers with 400, 400, 300, 300 neurons with ReLu activation for the first three layers and $\tanh$ for the last layer respectively. The input to the actor is a 128-dimensional GFV. The output is a single dimension z-vector. The critic also has four FC layers with 400, 432, 300, 300 neurons with ReLu activation for the first two layers respectively.

Reward Function Hyper-parameter In Eq.(5) of the main article, the multiplicative weights of w_{CH}, w_{GFV}, and w_D are assigned to the corresponding loss functions. The weight values are chosen such that, when combined,

*co-corresponding authors
[1]https://github.com/lijx10/SO-Net
[2]https://github.com/heykeetae/Self-Attention-GAN



(a) AE (b) l-GAN (c) RL

Figure 1: Network architecture of the three fundamental building blocks of RL-GAN-Net.

Name	Kernel	Stride	Padding	InpRes	OutRes	Input	Activation	Norm
convtr2d-layer1	4x4	-	-	50x32x1x1	50x256x4x4	z-vector	ReLu	SN,BN
convtr2d-layer2	3x3	2	2	50x256x4x4	50x128x5x5	convtr2d-layer1	ReLu	SN,BN
convtr2d-layer3	3x3	2	2	50x128x5x5	50x64x7x7	convtr2d-layer2	ReLu	SN,BN
Self-Atten[8]	-	-	-	50x1x7x7	50x1x7x7	convtr2d-layer3	-	-
convtr2d-last	2x2	2	1	50x64x7x7	50x1x12x12	Self-Atten	-	-
reshape1	1x1	-	-	50x1x12x12	50x144	convtr2d-last	-	-
convtrans1d	1x1	-	-	50x144	50x128	convtr2d-last	-	-

Table 1: The network architecture of the generator. convtr2d = 2D transposed convolutional layer, convtrans1d = 1D transposed convolution, SN = spectral normalization[8] and BN = batch normalization

the effects of individual terms are not out of proportion or dominant in any way. In other words, the total loss is within range for the RL agent to learn useful information for all of the terms of L_{CH}, L_{GFV}, and L_D. For example, if the value for the Chamfer loss was approximately 1000 and the GFV loss was 10, then they are normalized by dividing by 100 and 1 respectively. After consulting the range of raw loss values of multiple trials, we set $w_{CH} = 100$, $w_{GFV} = 10.0$, and $w_D = 0.01$ for all our experiments.

The RL agent was adopted from the open source implementation of the DDPG algorithm.[3].

Training Details The training of the agent can be divided into two parts. The first part is the collection of experience. The second part is the training of the actor and critic network in accordance with the DDPG algorithm as outlined in the previous work [6].

For the first part, we refer the readers back to Fig. 3 of the main article. It shows the mechanism by which the replay buffer **R** is filled continuously with useful experiences. We fill the memory with one input at a time. This implies that the batch size for this case is one. Our task is episodic, which means that after each episode we collect a reward. The number of episodes is equal to the maximum number of allowed iterations. In each episode, the agent is allowed to take a single action after which the episode terminates. The sequences of state, action and reward tuples are then stored in the replay buffer.

The second part, i.e., training the actor and critic in accordance with DDPG, is performed by keeping the batch size equal to one hundred. This means that a batch of a 100 memories from the replay buffer is picked randomly to train the actor and critic networks according to the DDPG algorithm. The evaluation of the policy was carried out after 5000 iterations. The number of dimensions of state is 128, which is basically the noisy GFV obtained by encoding the incomplete point cloud. The action dimension is determined by the dimension of the GAN's z-space, which is 1. The action space is kept to unity to achieve better performance by the agent. We also tested with 32 dimensions for z space but it did not have any noticeable effect on the performance of GAN or the agent.

We list the parameter values used for the training with DDPG algorithm in Table 3.

B. Additional Results

In this section, we provide enlarged images of the experiments in Sec. 4 of the original document and include some additional results that were omitted due to the page limit.

[3]https://github.com/sfujim/TD3

Name	Kernel	Stride	Padding	InpRes	OutRes	Input	Activation	Norm
convtrans1d	1x1	-	-	50x128	50x144	input	-	-
reshape1	-	-	-	50x144	50x12x12	convtrans1d	-	-
conv2d-layer1	3x3	2	2	50x12x12	50x64x7x7	reshape1	ReLu	SN,BN
conv2d-layer2	3x3	2	2	50x64x7x7	50x128x5x5	conv2d-layer1	ReLu	SN,BN
conv2d-layer3	3x3	2	2	50x128x5x5	50x256x4x4	conv2d-layer2	ReLu	SN,BN
Self-Atten[8]	-	-	-	50x256x4x4	50x256x4x4	conv2d-layer3	-	-
conv2d-last	4x4	-	-	50x256x4x4	50x1	Self-Attention	-	-

Table 2: The network architecture of the discriminator. conv2d = 2D convolutional layer, convtrans1d = 1D transposed convolution, SN = spectral normalization [8] and BN = batch normalization

Parameter	Value
maximum number of iterations	1e6
exploration noise	0.1
batch size from **R** for the actor training	100
discount γ	0.9
speed of target value updates τ	0.005
noise added to policy during critic update	0.2
range to clip noise policy	0.5
frequency for delayed policy update	2

Table 3: The parameter values used to train the RL agent.

Input	V 32^3	V 128^3	AE	RL-GAN-Net
0.0688	0.169	0.162	0.0531	0.0690

Table 4: The Chamfer distance compared to the ground truth. There are two volumetric approaches compared against two point cloud based approach. V 32^3 is the results using encoder-decoder based network in voxel space at the resolution of 32 per dimension, and V 128^3 shows the distance after the full pipeline including patch-based synthesis in [2]. AE and RL-GAN-Net are the point cloud based approaches of [1] and ours.

B.1. Shape Completion Results

The examples of shape completion results for point cloud missing 20% and 70% of its original points are enlarged in Fig. 2 and Fig. 6. In addition, we provide the examples of results for remaining data sets we used, which are missing 30%, 40% and 50% of the original points as shown in Fig. 3, 4 and 5 respectively. It is clear that the performance of our pipeline is prominent as the percentage of missing portion increases.

B.2. Robustness Results

The robustness test results with the different dataset provided by Dai et al [2] are included in Fig. 7 and Fig. 8. Our result is almost not affected by the jitter, and the completed shape is semantically similar to its original shape.

For the cases where there is no jitter, we also include the completion results of Dai et al [2]. Their approach works in a different domain (voxel grid) but we are including a comparison as there is not much prior work in point cloud space. To briefly describe, their approach used an encoder-decoder network in 32^3 voxel space followed by an analytic patch-based completion in 128^3 resolution. Their results of both resolutions are available as distance function format. We converted the distance function into a surface representation using the MATLAB function *isosurface* as they described, and uniformly sampled 2048 points to compare with our results. By comparing against the ground truth model, ours is superior to their approach in terms of the Chamfer distance as shown in Table 4. It should be noted here that the Chamfer distance between the input and ground truth is comparable to autoencoder. This is expected because the dataset provided by Dai et al.[2] does not have any drastic level of incompletion in many cases. We also refer the reader back to the Fig.5a of the main article where clearly the Chamfer distance of the input point cloud compared to the ground truth was even lower than an AE for missing data percentages less than forty.

We present the qualitative visual comparison in Fig. 9. The results of encoder-decoder based network (refered as Voxel 32^3 in the figure) are smoother than point clouds processed by AE as the volume accumulation compensates for random noise. However, the approach is limited in resolution and washes out the local details. Even after the patch-based synthesis in 128^3 resolution, the details they could recover are limited. On the other hand, our approach robustly preserves semantic symmetries and completes local details in challenging scenarios. It should be noted that we used only scanned point data but did not incorporate the additional mask information, which they utilized.

B.3. Classifier Details

We have trained the PointNet [7] classifier to distinguish the four categories that our RL-GAN-Net was trained on. After training, it classifies the shapes with 99.36% of accuracy on the full data set with a complete point cloud.

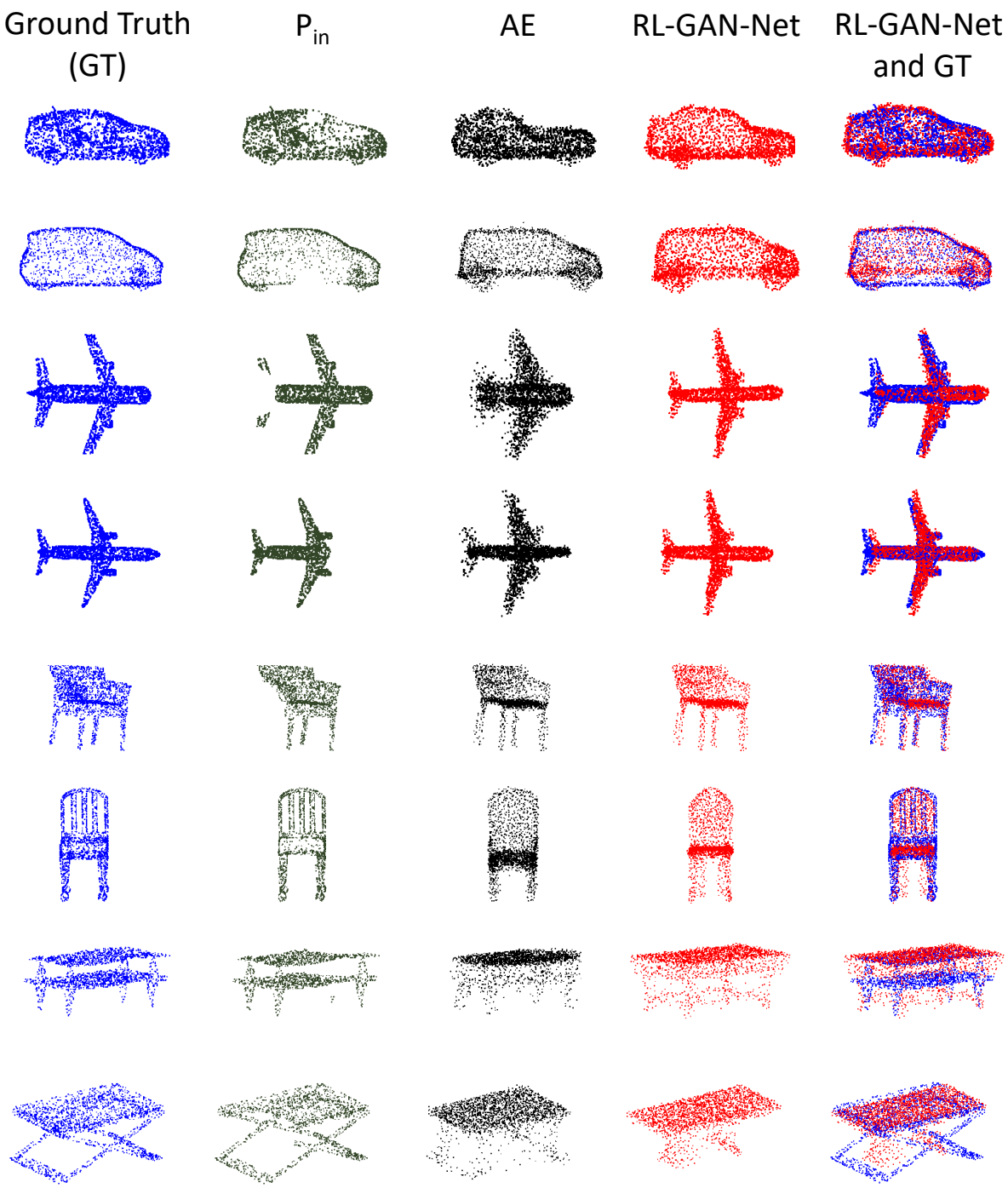


Figure 2: Qualitative results of point cloud shape completion missing 20% of its original points.

At test time, we consider the three scenarios as shown in Fig. 10, namely using the raw partial input, and using the shapes processed and completed by AE and RL-GAN-Net. The three pipelines are tested with the incomplete point cloud dataset for classification accuracy. For the cases that more than 30% of the original shape data is missing, the classification accuracy is boosted when the shapes are pre-processed with shape completion pipeline. And our suggested pipeline is superior to AE and robust to large missing regions. The detailed classification results are shown in Table 5.

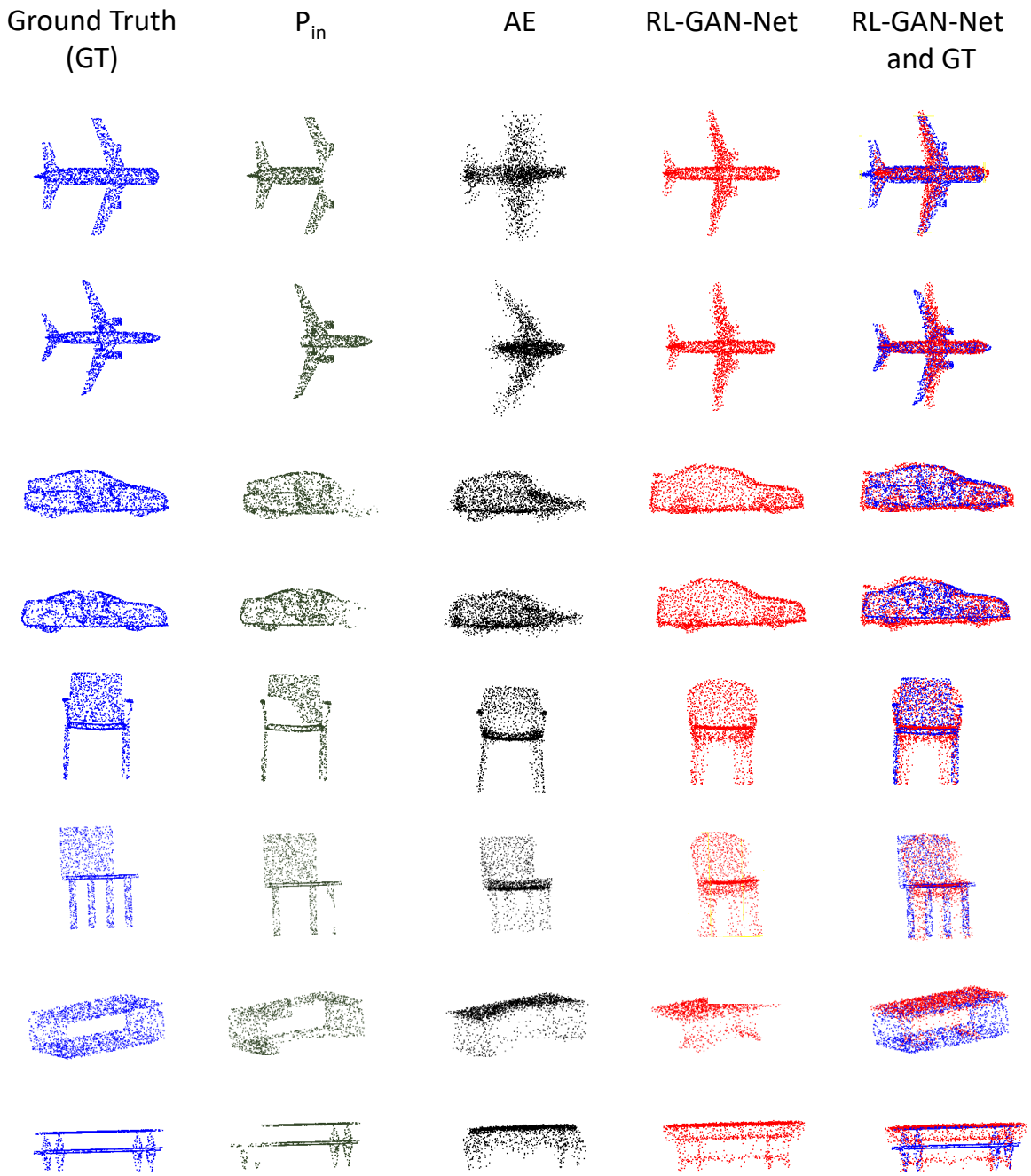


Figure 3: Qualitative results of point cloud shape completion missing 30% of its original points.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Representation learning and adversarial generation of 3d point clouds. *CoRR*, abs/1707.02392, 2017. 3, 7

[2] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. *CoRR*, abs/1612.00101, 2016. 3, 9, 10, 11

[3] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *CoRR*, abs/1612.00603, 2016. 1

[4] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017. 1

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 1

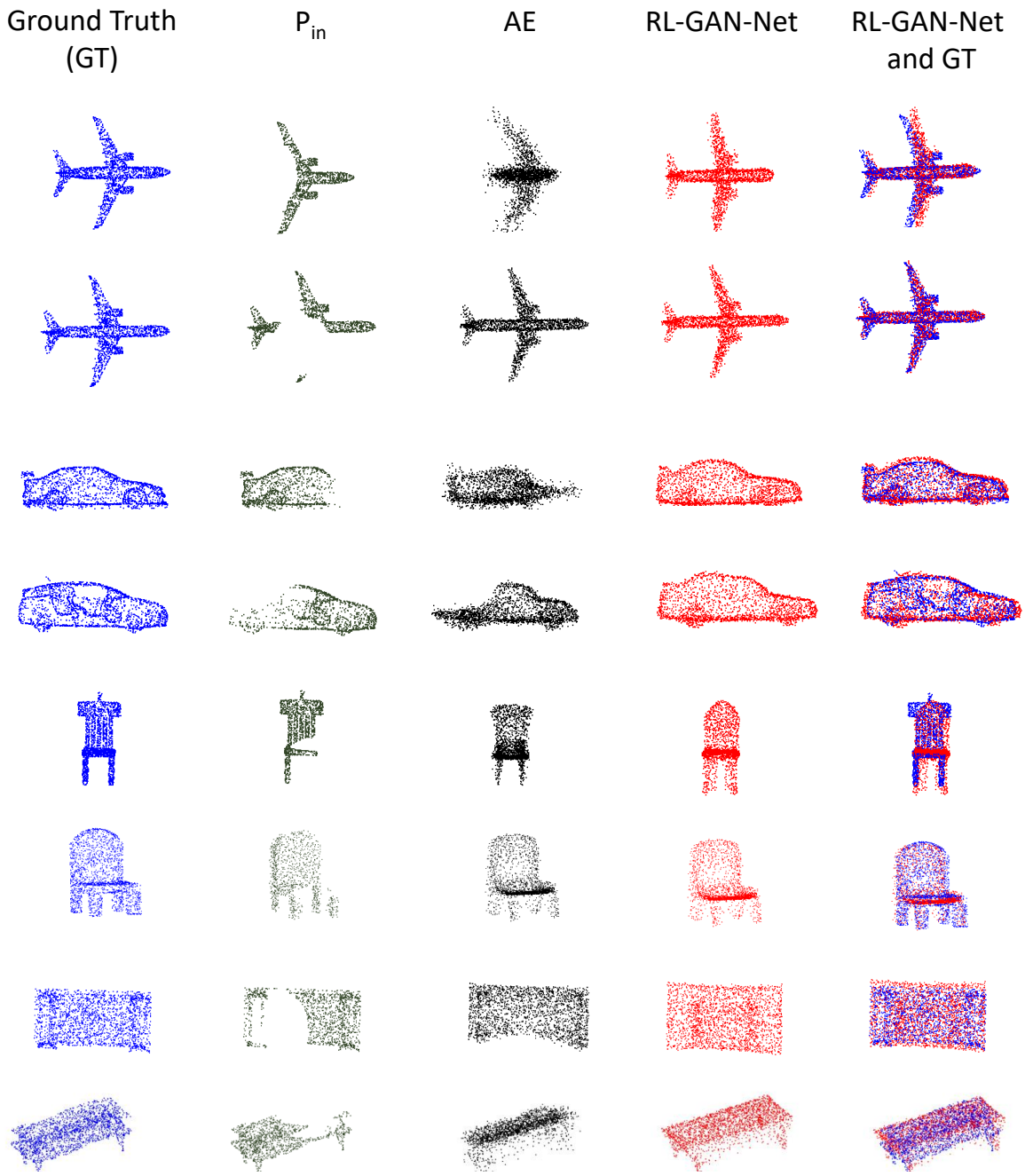


Figure 4: Qualitative results of point cloud shape completion missing 40% of its original points.

[6] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015. 2

[7] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. 3, 7

[8] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *CoRR*, abs/1805.08318, 2018. 1, 2, 3

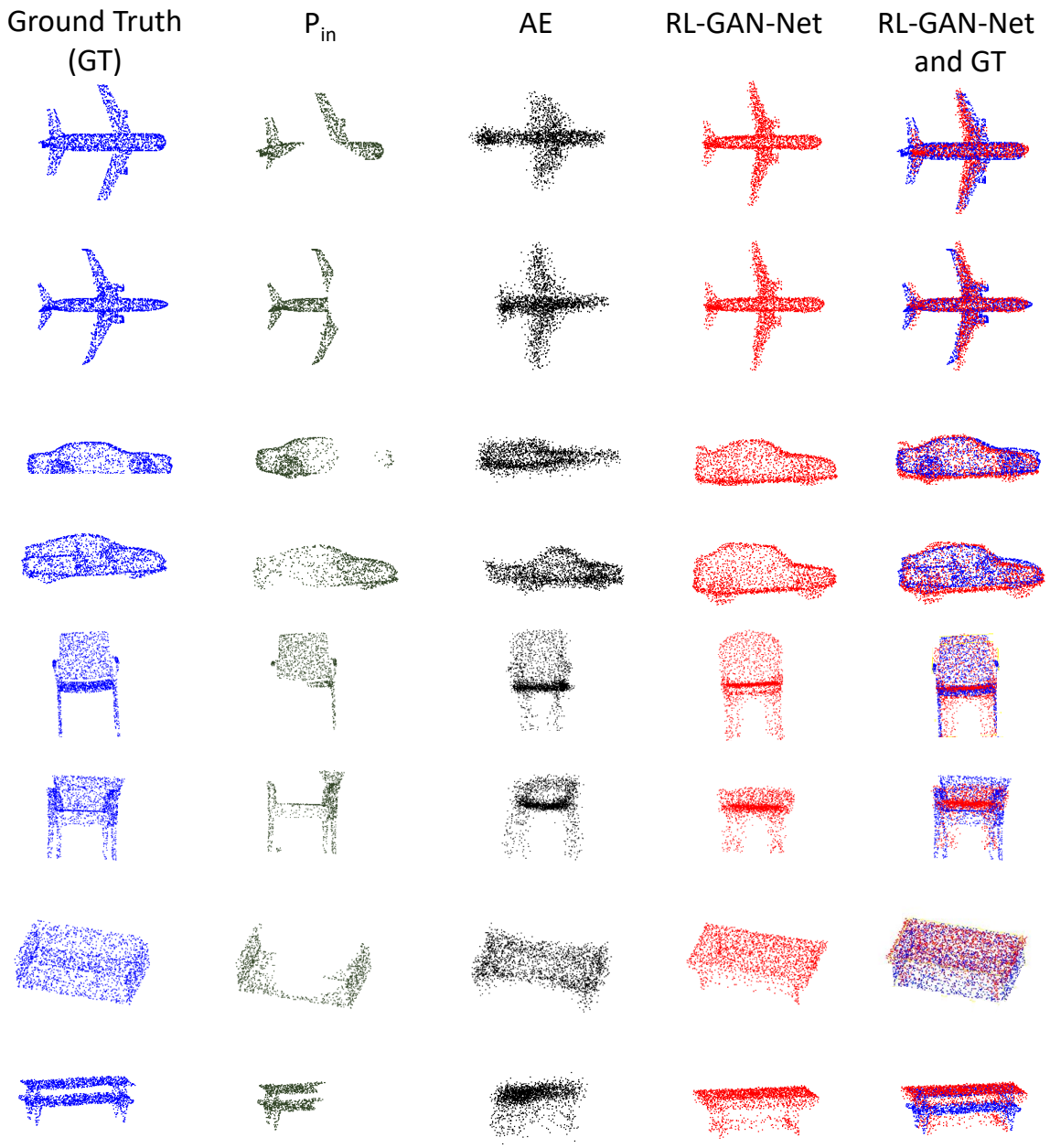


Figure 5: Qualitative results of point cloud shape completion missing 50% of its original points.

Network	20%	30%	40%	50%	70%
PointNet[7](Fig. 10a)	**98.6**	95.4	85.2	73.9	50.2
AE[1] + PointNet (Fig. 10b)	98.5	96.0	89.6	80.4	69.6
RL-GAN-Net (vanilla) + PointNet (Fig. 10c)	97.7	96.7	95.0	92.7	82.5
RL-GAN-Net (hybrid) + PointNet (Fig. 10c)	98.1	**97.2**	**95.5**	**93.3**	**83.8**

Table 5: Classification accuracy of point cloud input processed by RL-GAN-Net compared to vanilla and AE for various percentage of missing data points

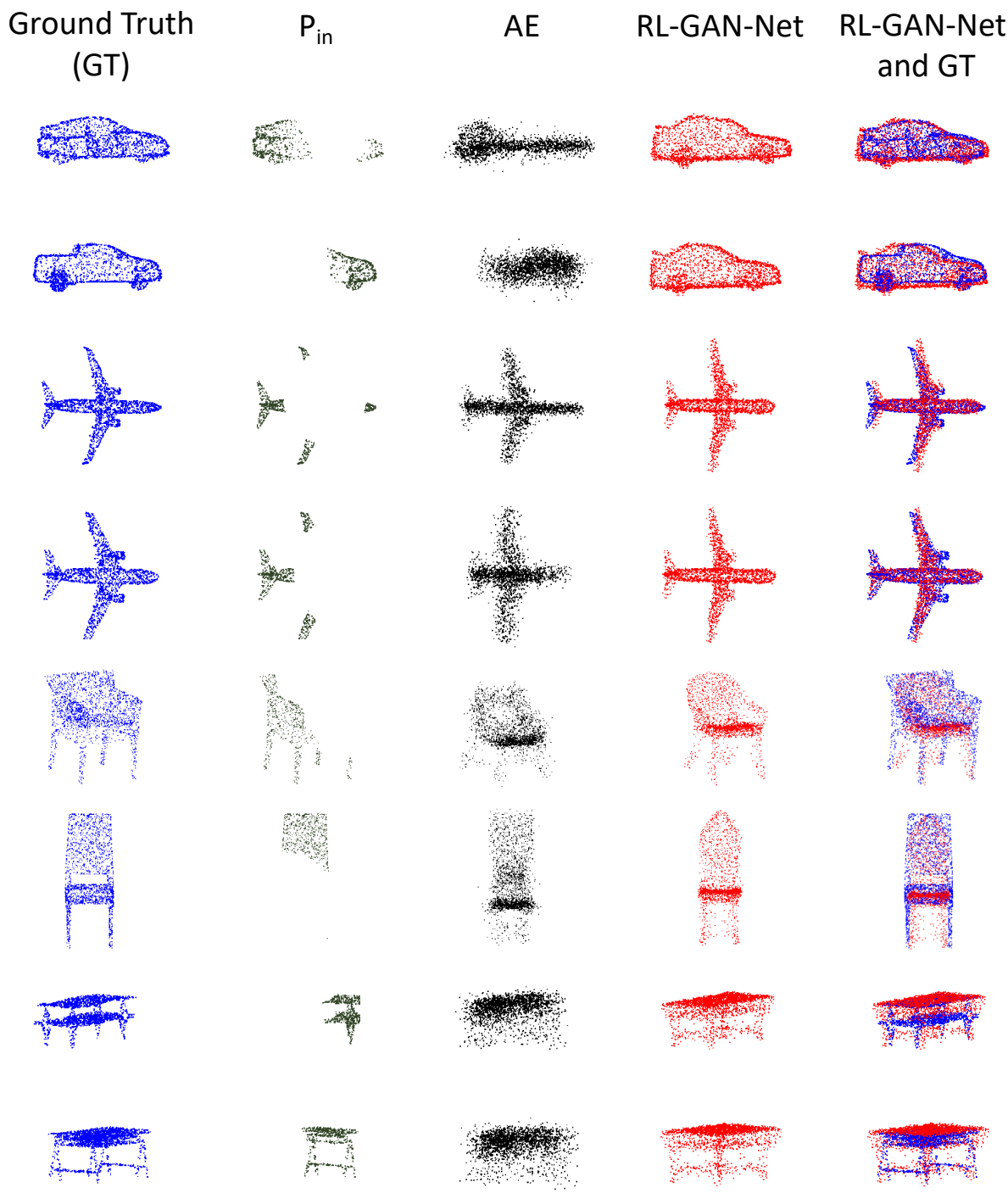


Figure 6: Qualitative results of point cloud shape completion given input data missing 70% of its original points.

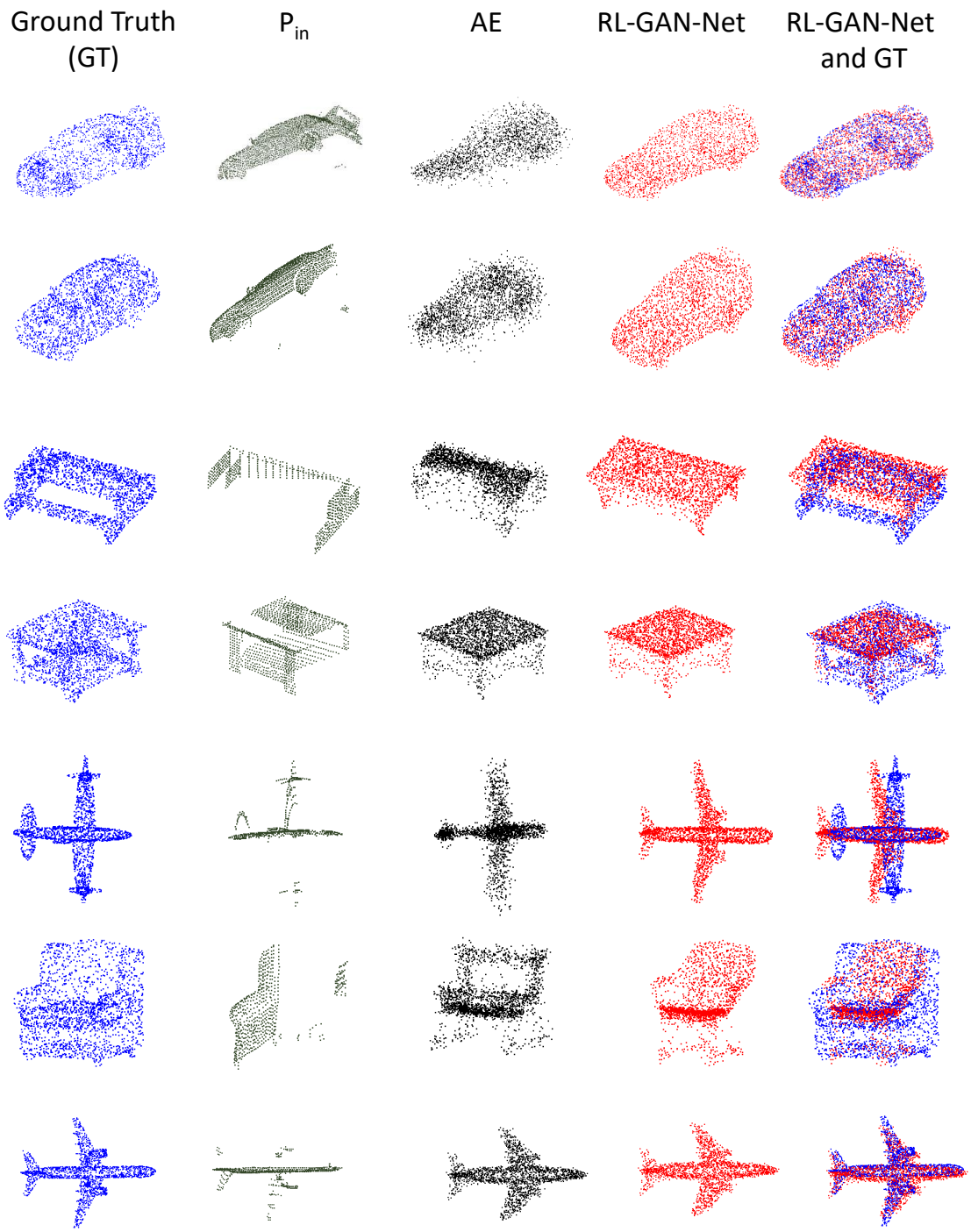


Figure 7: **Robustness test.** We applied our algorithm to the point cloud data provided by [2]. This figure shows examples of shape completion results with the raw scan data provided.

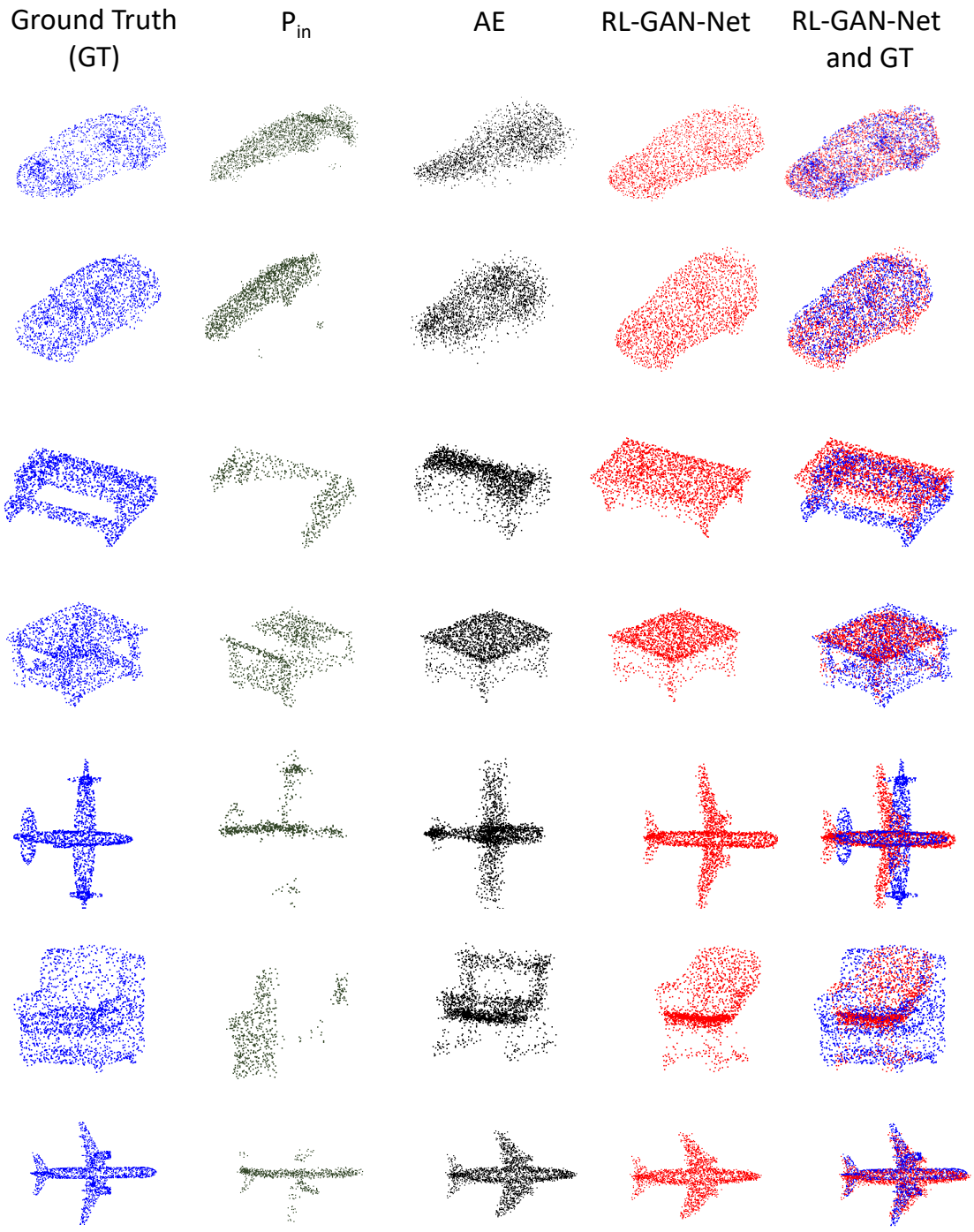


Figure 8: **Robustness test.** We applied our algorithm to the point cloud data provided by [2]. This figure shows results when we added zero-mean Gaussian noise with standard deviation 0.01 (clipped at 0.05).

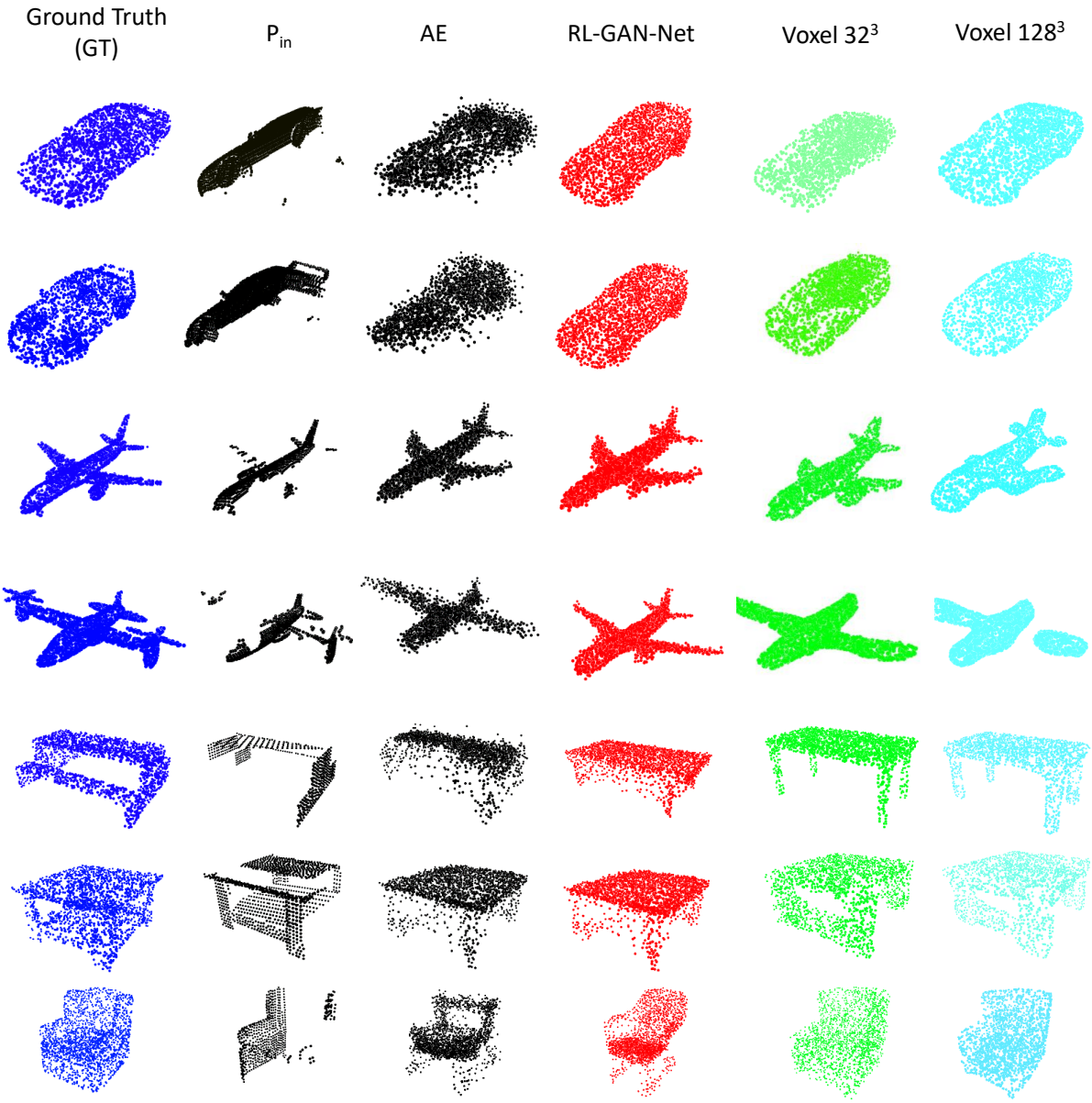
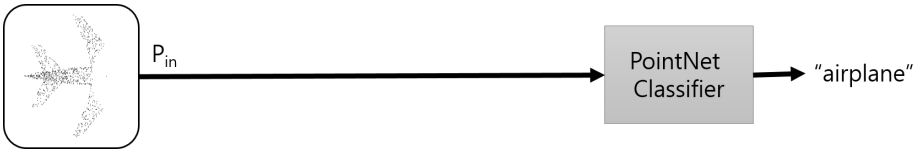
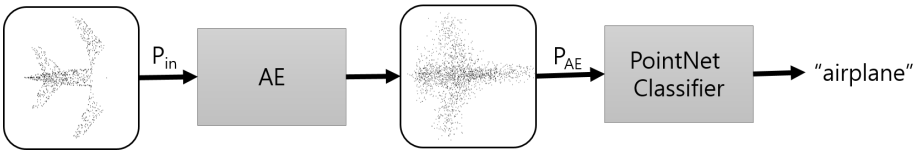


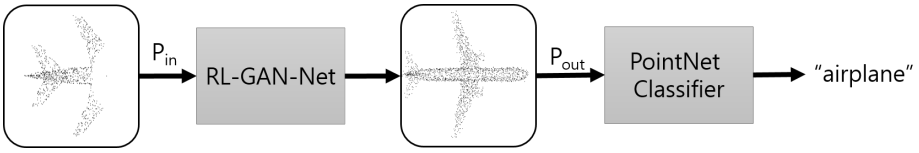
Figure 9: **Performance Comparison.** Comparison of RLGAN-Net vs Dai et al.[2] for their 32^3 and 128^3 resolution results. We converted their distance function output to point cloud domain. It should be noted that they additionally have mask information whereas we operate directly on the scanned points only.



(a) Vanilla PointNet Classifier



(b) AE + PointNet Classifier



(c) RL-GAN-Net + PointNet Classifier

Figure 10: The variations of network architecture for point cloud classification with missing data