# 2020-1 Reinforcement Learning Final Project
# RL-GAN-Net++

Hyeontae Son Seongju Kang

## Abstract

*Point cloud is one of the most wisely used data representation generated by 3D measurement instruments. Despite of the progress of the measurement technology, raw point cloud often suffers from noise and missing data. In this project, we deal with the point cloud completion, which predicts the complete point cloud from the partial shape. We adopted the framework of RL-GAN-Net which utilizes the reinforcement learning to infer the complete shape fast, but changed the formulation of completion process more realistic. With this natural environment setting, we suggest* ***RL-GAN-Net++*** *applying the novel RL algorithm and results show that ours can improve the performance of the existing method.*

## 1. Introduction

3D sensors such as LIDARs, RGB-D cameras, and 3D scanners generate the raw measurements in the format of point clouds. These are often sparse, noisy, and have large missing shapes due to the lack of the view points. Since many applications need complete 3D geometry, many researchers attempt to 3D reconstruction, in specific, point cloud completion to utilize the raw measured data directly.

We adopted the *RL-GAN-Net* framework, which combines the unsupervised learning and reinforcement learning techniques. However, they used the environment which always ends after only one step for each episode to make the completion task done by one RL agent's action. Due to its unnatural formulation, we changed the ending condition environment more realistic. With this natural environment setting, we applied the novel RL algorithm named *Soft Actor Critic (SAC)*, which makes the results better than baseline algorithm (*Deep Deterministic Policy Gradient (DDPG)*).

## 2. Related Works

**Deep learning for Point Clouds** PointNet[9] pioneered the deep neural networks for point clouds. The most difficult problem for dealing with point clouds is that they are unordered. Thus, networks for point clouds should be able to extract *permutation invariant features*, and PointNet showed that combination of multi-layer-perceptrons(mlp) and a symmetric function can do it. **Point Cloud Generation** Achlioptas et al. [1] suggested various deep neural networks generating point clouds and metrics for measuring performance of the networks. They suggested autoencoder structure for point clouds, which uses PointNet[9] for encoding point sets to global feature vector (GFV) and fully connected layers for decoding the GFV to a point set. They also experimented GANs[3], and showed that *l*-GAN which learns the distribution of the GFV and combining it to the pretrained decoder often makes more plausible outputs than *r*-GAN of which generator outputs point cloud directly.

**Point Cloud Completion** PCN[11] pioneered to solve the point cloud completion in a supervised manner. They made a dataset comprised of paired data, partially observed point set and complete point set. However, it is hard to get a large database of paired point sets in the real world. So recent researches[4, 10, 2] attempts to point completion task in unsupervised manner with unpaired point sets. [4] showed optimization with *l*-GAN framework learning the semantic prior of the complete shapes can handle the completion successively. However, the optimization process needs many iterations, so *RL-GAN-Net* replaced it with RL agent for real-time completion. Environment in the *RL-GAN-Net*[10], however, always ends after one step for each episode to make the inference fast with only one action. In this project, we aims to change the ending condition of the environment to natural way in the [4], and improved the algorithms for RL agent using the popular *Soft Actor Critic*[5] instead of *Deep Deterministic Policy Gradient*[8]. Detailed methods will be covered in the next section.

## 3. Methods

### 3.1. Autoencoder

First, we trained the autoencoder for complete point cloud to extract the GFV from point cloud. Loss function for autoencoder should be the distance between two point sets. We used the Chamfer Distance(CD) which is widely used distance function for point clouds. Definition of CD is

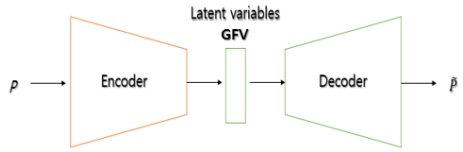


Figure 1. Autoencoder architecture. We trained autoencoder with 2048 points and used chamfer distance as the loss function

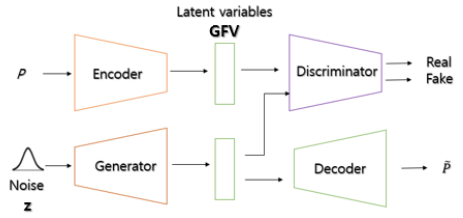


Figure 2. *l*-GAN architecture. *l*-GAN learns the distribution of the latent space. For shape generation, sampling the noise $z$ and pass it to the pretrained generator and decoder.

as follows.

$$CD(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\| + \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|.$$

We used same autoencoder structure in the [1], which uses PointNet[9] as encoder($E$), and fully connected layers as decoder($E^{-1}$) to generate 2048 points. We used the best model in the validation set later.

### 3.2. GAN for the latent space (*l*-GAN)

We trained GAN to learn the distribution of the extracted GFVs from the pretrained encoder. This is called *l*-GAN[1] for the latent space of the point clouds. We followed the setting of the *RL-GAN-Net*[10] to use the self-attention mechanism in [12] for this part. Generator($G$) learns to make a fake GFV from the noise $z$, and discriminator($D$) learns to discriminate the GFV as real or fake.

### 3.3. Reinforcement Learning

We followed the shape completion task formulated in a RL framework[10]. The environment is the pretrained autoencoder and *l*-GAN, and the reward will be calculated with them. The observed state is the GFV encoded from the partial point cloud. The agent takes an action to pick the correct seed in the $z$-space of the generator.

Our environment setting emulates the optimization process of the [4]. Ultimate goal for the optimization is to find the appropriate input $z$ of the generator and make plausible point cloud ($E^{-1}(G(z))$ There are three loss term for the target function to minimize in the paper. They are as follows: For the input point cloud $P_{in}$,

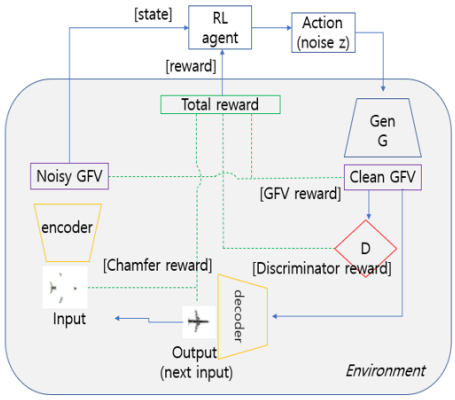


Figure 3. RL-GAN-Net++ framework. Noisy GFV is the input state of the RL agent and the action is the seed for generator of the *l*-GAN. Following the ending condition of the optimization process in the [4], environment ends when the discriminator judges that previous state is more plausible.

- Chamfer loss:

$$L_{CH} = CD(P_{in}, E^{-1}(G(z))) \tag{1}$$

- GFV loss:

$$L_{GFV} = \|G(z) - E(P_{in})\|_2^2 \tag{2}$$

- Discriminator loss:

$$L_D = \|-D(G(z))\| \tag{3}$$

They set the objective function as linear combination of the three terms with hyperparameter coefficients as

$$L_{total} = w_{CH} L_{CH} + w_{GFV} L_{GFV} + w_D L_D \tag{4}$$

and minimize it with many iterations using Adam optimizer[6]. We used reward term as

$$reward = -L_{total}$$

so we can emulate the optimization process.

We assumed the environment is Markov, and we can find the optimal action by solving MDP problem. However, in the original *RL-GAN-Net* paper, they complete point cloud by using only one action and do not feed-back the new state(synthesized GFV) to the RL agent. We followed the original optimizing process setting of the [4]. We adopted the **Soft Actor Critic**[8] to enable the elastic exploration. For calculating critic loss, we used mse loss of the current $Q_\phi$ and the target $\hat{Q}$ as:

$$J_Q(\phi) = \mathop{\mathbb{E}}_{(s_t, a_t) \sim D} \left[ Q_\phi(s_t, a_t) - \hat{Q}(s_t, a_t) \right]^2 \tag{5}$$

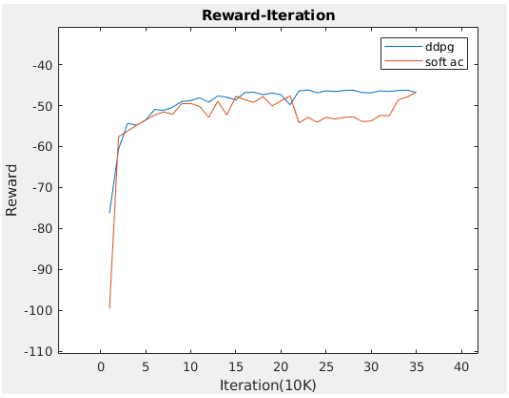


Figure 4. Training curves. Reward is evaluated on the validation set.

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma(1-d)\big(Q_{\phi_{targ}}(s_{t+1}, a') - \alpha \log \pi_\theta(a'|s_{t+1})\big) \tag{6}$$

when $a' \sim \pi_\theta(\cdot|s_{t+1})$. For actor loss,

$$J_\pi(\theta) = \mathop{\mathbb{E}}_{s_t \sim D, \xi \sim \mathcal{N}} \big[Q_\phi(s_t, \tilde{a}_\theta(s_t)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s_t)|s_t)\big] \tag{7}$$

when $\tilde{a}_\theta(s_t)|s_t)$ is obtained by reparameterization trick [7] with $\xi$ sampled in standard normal distribution $\mathcal{N}$. Unlike the original SAC paper, we used constant value 0.1 for $\alpha$. Also, we used only one Q function approximator, not using the min-double-Q trick. Detailed algorithm is shown in the **Algorithm 1**.

## 4. Results

**Dataset** We used the same dataset from the [1], which consists of the point clouds consisting of 2048 number of points, and the incomplete point cloud dataset with same algorithm of the [10], but used only the missing 70% points to tackle the most hard case. We splitted the incomplete point cloud dataset for training set and validation set with ratio 19:1 and used the validation set for policy evaluation.

**Policy Evaluation** Fig.4 shows the total average reward of evaluation during training for DDPG and SAC. The evaluation is performed every 10K training step. The result shows that SAC and the DDPG converges with similar learning speed and performance. The objective function of the SAC intends the expected future entropy to be maximized which could enable the policy spread more, making exploration more elastic and policy, so reward of SAC has larger oscillation compared to that of the DDPG after 200K training steps. DDPG algorithm also has stochastic property

**Algorithm 1:** RL-GAN-Net++

**Definition:**
**E**: pretrained encoder
$\mathbf{E}^{-1}$: pretrained decoder
**G**: pretrained generator of the *l*-GAN
**D**: pretrained discriminator of the *l*-GAN
$P_0$: input partial point cloud

**Agent Input:**
State ($s_t$): $s_t = GFV_t = \mathbf{E}(P_t)$
Reward ($r_t$): $r_t = -L_{total,t}$
The total loss $L_{total,t}$ is calculated by Eq.4

**Agent Output:**
Action ($a_t$): $a_t = z$
$GFV_{t+1} = \mathbf{G}(a_t)$
$P_{t+1} = \mathbf{E}^{-1}(GFV_{t+1})$

**procedure** ENV($P_t, a_t, isFirstStep$)
Set the pretrained networks: **E**, $\mathbf{E}^{-1}$, **G**, **D**
Get state: $s_t \leftarrow$**E**($P_t$)
Implement action: $GFV_{t+1} \leftarrow \mathbf{G}(a_t)$
Get intermediate points $P_{t+1} \leftarrow \mathbf{E}^{-1}(GFV_{t+1})$
Calculate reward: $r_t = -L_{total,t}$ using Eq.4
Next state: $s_{t+1} \leftarrow \mathbf{E}(P_{t+1})$
**if** $isFirstStep = True$ **then**
  $done \leftarrow False$
**else**
  **if** $L_D < prevL_D$ **then**
    $done \leftarrow True$
  **else**
    $done \leftarrow False$
  $prevL_D \leftarrow L_D$
**return** $s_{t+1}, P_{t+1}, r_t, done$

**Training:**
Initialize the **procedure** ENV
Initialize the policy $\pi$ with **Soft Actor Critic**, actor **A**, critic **C**, and replay buffer **R**
**for** *each iteration* **do**
  Get $P_{in} \leftarrow P_0$
  $s_0 \leftarrow \mathbf{E}(P_{in})$
  t $\leftarrow 0$
  **while** *not done* **do**
    **if** $iter_{curr} > 0$ **then**
      Train **A** and **C** with **R**
    **if** $iter_{curr} < iter_{StartTimeUsingPolicy}$ **then**
      Random Action $a_t$
    **else**
      Use $a_t \leftarrow \mathbf{A}(s_t)$
    **if** $t = 0$ **then**
      isFirstStep $\leftarrow True$
    **else**
      isFirstStep $\leftarrow False$
    resultList $\leftarrow ENV \leftarrow P_t, a_t$, isFirstStep
    $s_{t+1}, P_{t+1}, r_t$, done = resultList
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in **R**
    $P_{in} \leftarrow P_{t+1}$
    $s_t \leftarrow s_{t+1}$
    $t \leftarrow t+1$

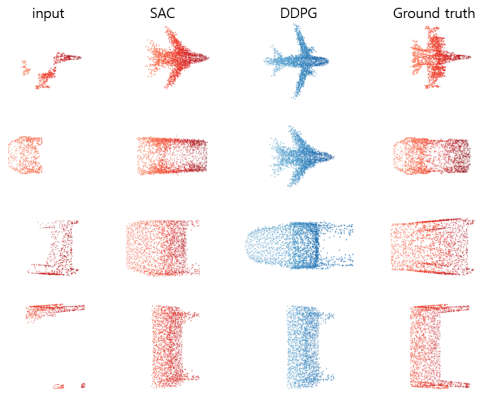


Figure 5. Completion results. Output from the SAC seems to find more plausible category than the DDPG.

| | CD | iteration |
|---|---|---|
| DDPG | 0.096 | 2.438 |
| SAC | **0.088** | **2.109** |

Table 1. Result of the completion task. CD is calculated between the output of the completion network and the ground truth. Lower is better.

which is simply implemented by adding Gaussian noise to the action during the training.

**Completion results** Fig. 5 shows the results of the completion with two RL algorithms. It seems that the output of DDPG fails to find right category as shown in the second row. SAC also finds the optimal action with small iteration and the output is more similar with the ground truth1. It can be explained that the elastic exploration improves the RL algorithm's performance.

## 5. Conclusion

We adopted the *RL-GAN-Net* framework, and changed the setting more natural to the reinforcement learning framework. Also we replaced the DDPG algorithm with SAC algorithm which can improve the performance because of rich exploration. The results of the experiments show that SAC outperforms the average Chamfer Distance with small iterations.

## References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning Representations and Generative Models for 3D Point Clouds. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 40–49, 2018.

[2] Xuelin Chen, Baoquan Chen, and Niloy J Mitra. Unpaired Point Cloud Completion on Real Scans using Adversarial Training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative Adversarial Nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[4] Swaminathan Gurumurthy and Shubham Agrawal. High Fidelity Semantic Shape Completion for Point Clouds using Latent Optimization. *CoRR*, abs/1807.03407, 2018.

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870, 2018.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[7] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[8] 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. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[9] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 77–85, 2017.

[10] Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point Completion Network. In *3D Vision (3DV), 2018 International Conference on*, 2018.

[12] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-Attention Generative Adversarial Networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363. PMLR, 2019.