SUGAR: Pre-training 3D Visual Representations for Robotics

Supplementary Material

In Section 1, we provide more implementation details for SUGAR pre-training and downstream adaptation. Then in Section 2 we present additional quantitative results. We further perform real robot experiments in Section 3 to demonstrate the effectiveness of SUGAR pre-training for robotic manipulation in the real world. Finally, we discuss limitations and future work in Section 4.

1. Implementation Details

1.1. Pre-training

Network details. We set the number of points $N = 4096$, the number of key points $N_e = 256$ and the group size $S_e = 32$ to obtain the point cloud input tokens. The SUGAR encoder and decoder contains $L = 12$ transformer blocks with hidden size $d = 384$ and 6 attention heads per block.

Training details. We pre-train two sets of models according to the pre-training data: 'SN' uses objects only in ShapeNet, and 'Ens' uses the ensembled four datasets. For the 'SN' model, we train 100K iterations on the single-object dataset with learning rate 1e-4 and 100K iterations on the multi-object dataset with learning rate 1e-5 and batch size 128. For the 'Ens' model, we train 300K iterations on the single-object dataset and 200K iterations on the multi-object dataset using the same learning rate and batch size as in 'SN' models. The pre-training is performed on one NVIDIA-A100 GPU, taking 50 hours for the 'SN' model and 130 hours for the 'Ens' model.

1.2. Referring expression grounding

For the OCID-Ref dataset, we fix the point cloud encoder and only finetune the prompt-based decoder. We finetune the model with a batch size of 64 and learning rate of 1e-4 for 20 epochs. For the RoboRefit dataset, we finetune the full model with a batch size of 16 and learning rate of 4e-5 for 50 epochs. We use the AdamW optimizer with cosine learning rate scheduler.

1.3. Language-guided robotic manipulation

Experimental setup. Our experimental setup on RLBench [31] 10 tasks is the same as previous works [7, 23]. Specifically, we use three cameras located on the left shoulder, right shoulder and wrist of the robot with known camera intrinsics and extrinsics. Each camera produces an RGB-D image with image resolution of 128×128 at every step. A merged point cloud can be obtained given the camera parameters. Following [7], we only keep points inside the robot's workspace by using a fixed bounding box around the table. We use voxel downsampling to uniformly downsample the point cloud with 0.5cm grid size. For robotic control, we use keysteps [7, 23, 45] - key turning points in action trajectories where the gripper changes its openness state or velocities of joints are close to zero. The control policy should predict a position (3D), rotation (4D represented by quaternion) and openness state (1D) of the gripper for the next keystep. The default motion planner in RLBench is used to find a trajectory between two keysteps.

Model details. Figure 1 illustrates the policy network in detail. We first combine the action prompt embedding y_1^L and point embeddings $\{x_i^L\}_{i=1}^{N_e}$ to compute a heatmap over all the key points, which denotes the importance of the key points for action prediction. We then average the point embeddings and position of key points respectively using the heatmap. The averaged point embedding is concatenated with y_1^L to regress the position offset relative to the averaged key point position, a rotation vector and an openness state. The policy is trained by behavior cloning, with MSE loss for position and rotation, and BCE loss for openness state.

Training details. We use a batch size of 8 to train the model for 200K iterations for the 10 RLBench tasks. We adopt a learning rate of 5e-5 for the model trained from scratch, while a lower learning rate of 2e-5 for the model initialized from SUGAR pre-training.

2. Additional Results

Zero-shot object recognition. Though we consider the Ensembled w/o LVIS setup to be better suited for evaluating the generalization ability of models, we include results with LVIS training in Table 1 for complete comparison with prior work [44]. Training with LVIS split improves performance on the LVIS dataset but does not impact much on the other two datasets. Our model still outperforms the SoTA method [44] under this setup.

Referring expression grounding. We provide an additional variant SUGAR (Ens_s) in Table 2, which is pre-trained on single objects of the ensembled dataset. To be noted, we only initialize the point cloud encoder for SUGAR variants pre-trained on single objects as we find initializing both encoder and decoder deteriorates the performance. As the decoder in single-object pre-training focuses on the overall scene for cross-modal learning, we hypothesize that the learned cross-modal attentions can suffer from recognition of local objects. As shown in Table 2, the single object pre-training on the ensembled dataset does not benefit the generalization on unseen cluttered scenes in testB split, demonstrating the importance of pre-traning on multi-object scenes.

Table 1. Zero-shot object recognition performance with models trained on Ensembled w/ LVIS dataset.

Method	ModelNet40	ScanObjectNN			Objaverse-LVIS		
		OBJ_ONLY	OBJ_BG	PB_T50_RS	Top1	Top3	Top5
OpenShape [44]	84.4	54.0	59.1	43.6	46.8	69.1	77.0
SUGAR (single)	**84.6**	**65.3**	**67.6**	**49.8**	**49.5**	**72.2**	**78.8**
SUGAR (multi)	84.5	64.9	66.8	48.3	46.8	69.7	76.6

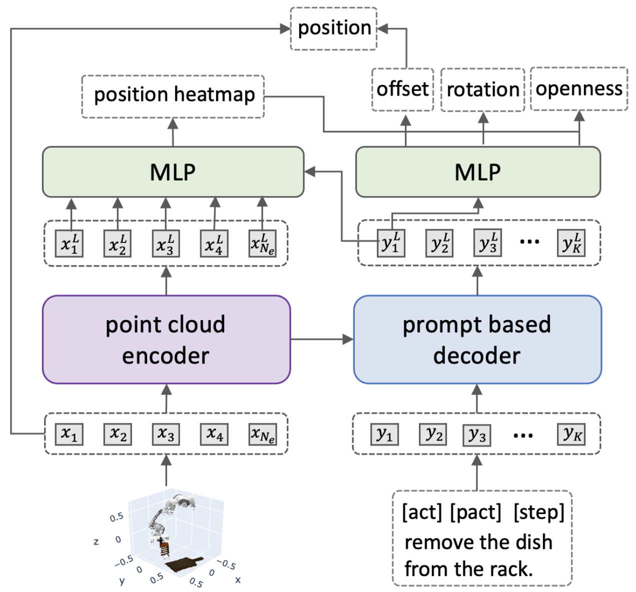

position
position heatmap
offset
rotation
openness
MLP
MLP
x_1^L x_2^L x_3^L x_4^L $x_{N_e}^L$
y_1^L y_2^L y_3^L ... y_K^L
point cloud encoder
prompt based decoder
x_1 x_2 x_3 x_4 x_{N_e}
y_1 y_2 y_3 ... y_K
[act] [pact] [step] remove the dish from the rack.

Figure 1. Network architecture for language-guided robotic manipulation. The point cloud encoder and prompt based decoder can be finetuned from SUGAR pre-training. We use two multi-layer perceptrons modules (MLP) as the action prediction head.

Table 2. Performance of referring expression detection (evaluated by Acc@0.5) and referring expression segmentation (evaluated by mIoU) on the RoboRefit dataset.

Method	testA		testB	
	Acc@0.5	mIoU	Acc@0.5	mIoU
SUGAR (no pre-train)	87.56	81.31	55.62	57.02
SUGAR (Ens_s)	88.11	81.71	52.59	56.57
SUGAR (Ens_m)	**89.47**	**82.11**	**65.04**	**62.80**

Language-guided robotic manipulation. In Table 3, we include both the averaged success rate and standard deviations for the RLBench 10-task experiment. As the 10 RLBench tasks use objects with simple shapes like cups and cubes, pre-training on ShapeNet can be sufficient and thus we do not observe further performance improvement from pre-training on Ens_m. Compared to PolarNet, our model performs slightly worse on Pick & Lift and Push Button though it achieves better performance on average. To be noted, PolarNet employs additional normal and height features in the point cloud, while our method omits those for generalizability in pre-training. As shown in PolarNet, normal and height features benefit some tasks like "Push Button" where the main failure cases are that the gripper does not push down enough to the button. We also notice relatively large variations on individual tasks, and thus we consider the averaged performance is more stable for comparison.

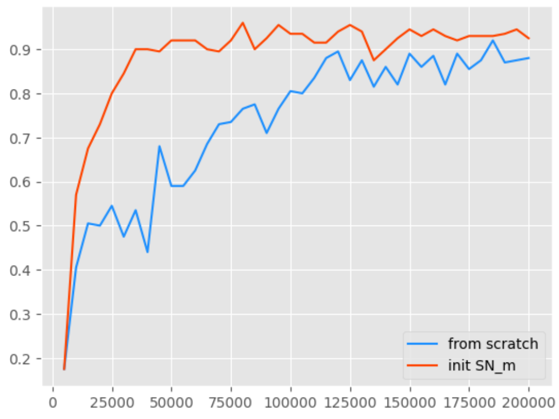


Figure 2. Success rate on RLBench validation split in different training iterations. We compare the policy trained from scratch and the model initialized from SUGAR pre-training.

In Figure 2, we present the performance on the RLBench validation split for policies trained from scratch and initialized from SUGAR pre-training. We can see that the policy can converge much faster and achieve better performance with the pre-training.

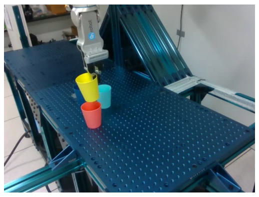
3. Real-world Robotic Manipulation

To evaluate the effectiveness of SUGAR pre-training for real robots, we further perform real world experiments for language-guided robotic manipulation.

To be specific, we use a UR5 robotic arm equipped with a RG6 gripper and set two Intel RealSense D435 RGB-D cameras on the front and lateral sides of the robot's workspace. We adopt 5 real-world tasks including *stack cup*, *put fruit in box*, *open drawer*, *put item in cabinet* and *hang mug* as illustrated in Figure 3. For each task, we collect 20 real-robot demonstrations, where each demonstration consists of RGB-

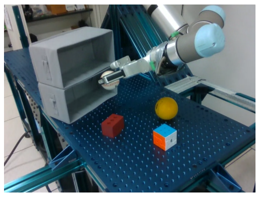
Table 3. Averaged success rate of three runs for multi-task policies on 10 tasks of RLBench simulator.

Method	Pre-train	Avg.	Pick & Lift	Pick-Up Cup	Push Button	Put Knife	Put Money	Reach Target	Slide Block	Stack Wine	Take Money	Take Umbrella
PolarNet [7]	ShapeNetPart	89.8$_{\pm1.5}$	97.8$_{\pm1.4}$	86.0$_{\pm2.1}$	99.6$_{\pm0.4}$	80.5$_{\pm1.1}$	94.1$_{\pm0.8}$	100$_{\pm0.0}$	93.4$_{\pm0.9}$	80.5$_{\pm3.6}$	68.1$_{\pm4.3}$	97.8$_{\pm0.2}$
SUGAR	-	85.9$_{\pm3.9}$	77.7$_{\pm4.9}$	92.7$_{\pm4.2}$	91.7$_{\pm0.9}$	69.4$_{\pm8.0}$	87.7$_{\pm1.2}$	99.7$_{\pm0.4}$	94.3$_{\pm0.4}$	83.1$_{\pm7.8}$	66.8$_{\pm9.2}$	95.7$_{\pm1.6}$
	SN_m	93.0$_{\pm1.0}$	93.1$_{\pm1.3}$	94.5$_{\pm1.0}$	98.9$_{\pm0.8}$	85.4$_{\pm1.4}$	97.8$_{\pm1.3}$	100$_{\pm0.0}$	97.9$_{\pm0.8}$	94.5$_{\pm1.5}$	70.0$_{\pm1.6}$	98.4$_{\pm0.2}$
	Ens_m w/o grasp	92.0$_{\pm1.6}$	93.1$_{\pm1.3}$	93.7$_{\pm1.3}$	98.8$_{\pm1.1}$	85.5$_{\pm0.1}$	92.3$_{\pm5.3}$	99.9$_{\pm0.1}$	97.3$_{\pm1.4}$	93.7$_{\pm0.6}$	68.8$_{\pm4.2}$	97.2$_{\pm0.9}$
	Ens_m	93.0$_{\pm1.7}$	95.8$_{\pm1.3}$	95.7$_{\pm1.6}$	96.1$_{\pm5.1}$	86.5$_{\pm2.7}$	94.2$_{\pm1.6}$	100$_{\pm0.0}$	97.0$_{\pm0.5}$	93.5$_{\pm0.6}$	72.0$_{\pm2.9}$	98.8$_{\pm0.9}$


(a) Stack cup.

(b) Put fruit in box.

(c) Open drawer.


(d) Put item in cabinet.

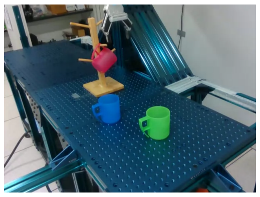

(e) Hang mug.

Figure 3. Illustration of the adopted five real robot tasks.

Table 4. Success rate of multi-task policies on 5 real-world tasks. We evaluate 10 episodes for each task.

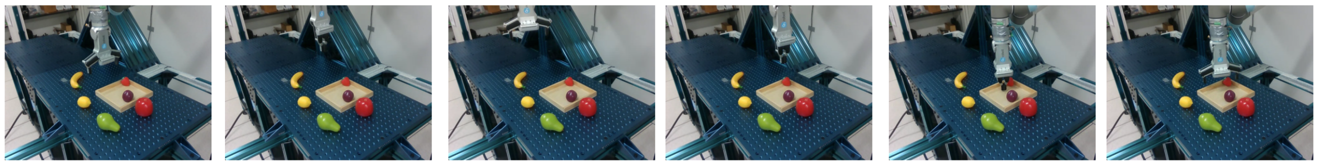
	no pretrain	SUGAR (Ens_m)
Stack cup	0/10	10/10
Put fruit in box	0/10	4/10
Open drawer	0/10	3/10
Put item in cabinet	0/10	9/10
Hang mug	0/10	6/10

D images and proprioceptive information of the gripper at keysteps (typically 3-6 keysteps).

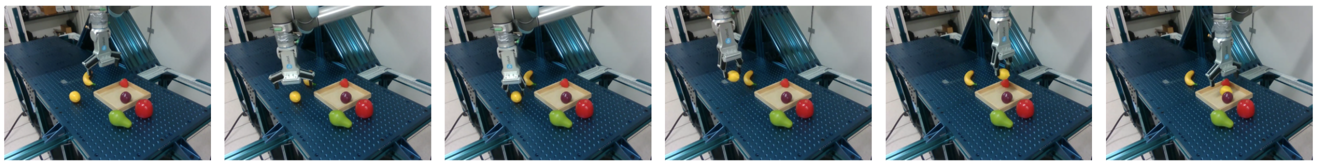
We train a multi-task policy using the collected real-robot data, and evaluate 10 episodes for each task where the object locations and distractor objects are different from the training data. Table 4 presents results of a model trained from scratch on the real robot data and a model initialized from SUGAR pre-training. The model trained from scratch overfits on the limited training data and totally fails in evaluation. As shown in Figure 4a, the model trained from scratch has serious problems of localizing the target object. Our SUGAR pre-training significantly improves the performance for language-guided manipulation in the real world, leading to an average of 64% success rate over the five tasks. Figure 4b presents a successful case of putting lemon in the box. However, we also notice that the model initialized from SUGAR pre-training still has problems in precise object localization in Figure 4c. The problems can result from the sub-optimal network design that largely downsamples the point cloud, the regression action prediction head that is more unstable compared to classification, and the noisy depth sensors. We will investigate more on the policy networks to improve the robotic manipulation performance.

4. Limitations and Future Work

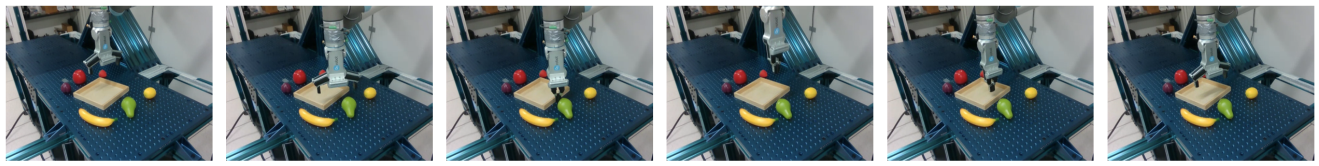
This work only adopt a plain transformer architecture for point cloud encoding, which is computationally expensive. For example, compared to the SoTA method PolarNet [7], our model consists of 4.5x more parameters (65M vs. 14M) and runs 1.3x slower (18h vs. 14h in training on one V100 GPU). This is because PolarNet is based on a UNet backbone which is more efficient. Our vanilla transformer-based backbone alone does not a show clear advantage over the UNet backbone for robotic manipulation as seen in Table 3, although the proposed pre-training significantly boosts the performance. We believe that the proposed pre-training can benefit other architectures and plan to explore more efficient 3D backbones in our future work.



(a) A failure case of the multi-task policy trained from scratch.



(b) A successful case of the multi-task policy initialized from SUGAR pre-training.



(c) A failure case of the multi-task policy initialized from SUGAR pre-training.

Figure 4. Examples of real world execution on the *Put fruit in box* task for different policies.