

Code for General Policy Composition

We introduce a novel policy composition approach, General Policy Composition (GPC), which composes distributional scores from multiple pre-trained diffusion policies, enabling significant performance improvement without the need for additional training.

Note: This repository as well as the following guidelines are based on [RoboTwin](#). We thank the authors of Robotwin for their open source resources.

Installation & Data Preparation

Please carefully follow the guidelines in [RoboTwin](#) for installation and data generation.

Usage

Step 1. Prepare the pre-trained single modality-based diffusion policies

1. Task Running and Data Collection

Running the following command will first search for a random seed for the target collection quantity (default is 100), and then replay the seed to collect data.

```
bash run_task.sh ${task_name} ${gpu_id}
```

2. Training the diffusion policy as well as the 3D diffusion policy

(1) Diffusion Policy

The DP code can be found in [policy/Diffusion-Policy](#).

Process Data for DP training:

```
python script/pkl2zarr_dp.py ${task_name} ${head_camera_type} ${expert_data_num}
# As example: python script/pkl2zarr_dp.py dual_bottles_pick_hard L515 100, which
indicates preprocessing of 100 dual_bottles_pick_hard task trajectory data using
L515 camera.
```

Then, move to [policy/Diffusion-Policy](#) first, and run the following code to train DP:

```
bash train.sh ${task_name} ${head_camera_type} ${expert_data_num} ${seed}
${gpu_id}
```

```
# As example: bash train.sh dual_bottles_pick_hard L515 100 0 0
```

(2) 3D Diffusion Policy

The DP3 code can be found in [policy/3D-Diffusion-Policy](#).

Process Data for DP3 training:

```
python script/pkl2zarr_dp3.py ${task_name} ${head_camera_type} ${expert_data_num}
# As example: python script/pkl2zarr_dp3.py dual_bottles_pick_hard L515 100
```

Then, move to [policy/3D-Diffusion-Policy](#) first, and run the following code to train DP3:

```
bash train_ddpm.sh ${task_name} ${head_camera_type} ${expert_data_num} ${seed}
${gpu_id}
# As example: bash train_ddpm.sh dual_bottles_pick_hard L515 100 0 0
```

Step 2. Compose the pre-trained policies via distribution-level composition

Move to [policy/3D-Diffusion-Policy](#) first, then run the following code to evaluate MCDP for a specific task for 100 times:

```
bash eval_composed.sh ${task_name} ${head_camera_type} ${expert_data_num}
${checkpoint_num} ${seed} ${gpu_id} ${dp_w} ${dp3_w}
# As example: # bash eval_composed.sh dual_bottles_pick_hard L515 100 3000 0 0 0.3
0.7
```

Core Code of GPC

```
dp_w = args['dp_w']
dp3_w = args['dp3_w']

## init noise
trajectory = torch.randn(
    size=condition_data1.shape,
    dtype=condition_data1.dtype,
    device=condition_data1.device,
    generator=generator)

# set step values
dp_scheduler.set_timesteps(num_inference_steps)
dp3_scheduler.set_timesteps(num_inference_steps)
```

```
for t in dp3_scheduler.timesteps:

    # 1. apply conditioning
    trajectory[condition_mask2] = condition_data2[condition_mask2]

    # 2. predict model output
    model_output_dp = dp_model(trajectory, t,
                               local_cond=local_cond1, global_cond=global_cond1)

    model_output_dp3 = dp3_model(trajectory, t,
                                 local_cond=local_cond2, global_cond=global_cond2)

    # 3. distribution-level composition
    model_output = dp_w * model_output_dp + dp3_w * model_output_dp3
    # model_output = model_output_dp3

    # 4. compute previous image: x_t -> x_{t-1}
    trajectory = dp3_scheduler.step(
        model_output, t, trajectory,
        generator=generator,
    ).prev_sample
```