Diffusion Policy x Robotics Summer Research Project

Sajjad Hashemi

What did I do this summer?

May

Literature Review

- Diffusion models
- RL and Diffusion models
- Latent Diffusion
- Diffusion Policy

Diffusion Policy Recap

- Diffusion policy:
- Iteratively
    - Predict (noise | state, noisy action)
    - Remove noise from noisy action

June

Implement the Diffusion Policy Paper

- I made a colab notebook that trains diffusion policy models.
    - The user will have to provide the data: robot camera captures (states) and robot actuator values (actions) that led to those states.
    - The notebook will train a diffusion policy model and save it on google drive.

July

- MyCobot
- Robosuite
  - has scripts that help with recording robot demos.
- I made MyCobot in Robosuite
  - Limitations
    - MyCobot is difficult and time-consuming to maneuver.
    - Hard to collect demos.
- I used the Robosuite Panda robot.

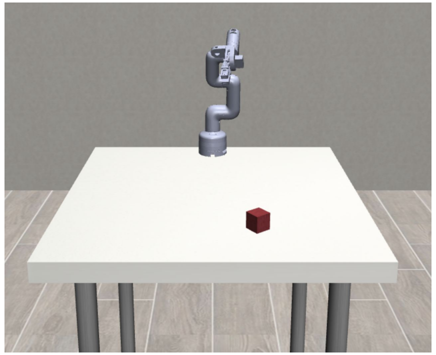


Figure 1: MyCobot robot in Robosuite environment.

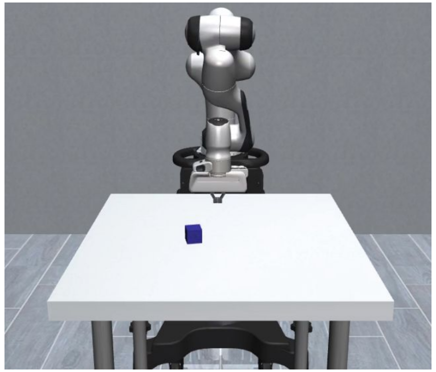


Figure 2: Panda robot in Robosuite environment.

August

- Configured Robosuite for collecting simulations with randomized object placement.
- Collected Robosuite data for the “Stack” and “Lift” task.
- Wrote the necessary code to convert proprioception data to visual data to be fed to the model.
- Created a pipeline to pre-process data for the Diffusion Policy model.
- Trained the model.
- Created pipeline for processing results of the model at inference time.
    - Video captures of the robot on diffusion policy.

How to train diffusion policy?

Data Collection Dependencies

Mujoco: A physics engine for performing simulations.

Mujoco_py: Allows using Mujoco from Python3.

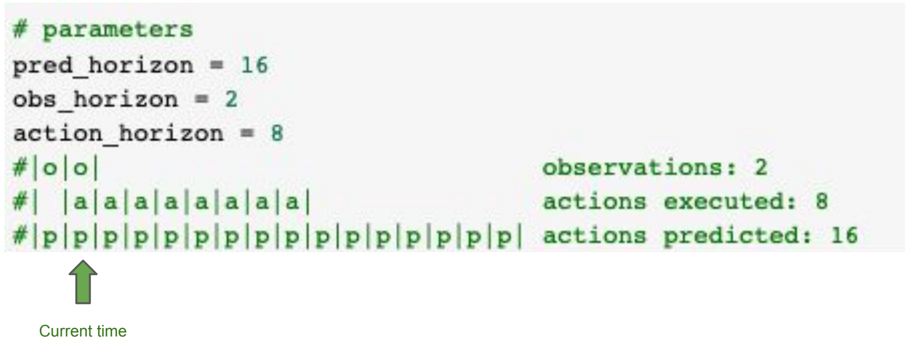
Robosuite: Used for performing simulations manually.

Robomimic: Used for retrieving robot camera captures from simulation data efficiently.

Data Preprocessing (For your reference :))

1. Collect data via Robosuite collect_human_demonstrations.
    a. Just run the file in python, a simulator will open where you can collect simulations.
2. Convert robosuite data to be processable by Robomimic convert_robosuite.
    a. Example script: $ python conversion/convert_robosuite.py --dataset /Users/sajjad/Desktop/robotics/robosuite/robosuite/models/assets/demonstrations/1692476434_952685/demo.hdf5
3. Get state images using Robomimic dataset_states_to_obs.
    a. Example script: $ /Users/sajjad/miniconda3/envs/RL_gym/bin/python dataset_states_to_obs.py --dataset /Users/sajjad/Desktop/robotics/robosuite/robosuite/models/assets/demonstrations/1692476434_952685/demo.hdf5 --output_name image.hdf5 --done_mode 2 --camera_names agentview robot0_eye_in_hand --camera_height 96 --camera_width 96
    b. Remember to keep --camera_height 96 --camera_width 96 because diffusion policy input should be of this size.
4. Put the image.hdf5 file in a folder in your google drive to be fed to the notebook that I made.
5. Put the path to the folder in the notebook.

The Training Data



```
# parameters
pred_horizon = 16
obs_horizon = 2
action_horizon = 8
#|o|o|                               observations: 2
#| |a|a|a|a|a|a|a|a|                 actions executed: 8
#|p|p|p|p|p|p|p|p|p|p|p|p|p|p|p|p|   actions predicted: 16
```

- State
  - Size: (batch_size, observation_horizon, color_channels, image_width, image_height)
  - Each sample in a batch contains
    - observation_horizon: how many states from the past are we taking into consideration.
    - Agent_view camera image
    - Robot_arm camera image
- Action
  - Size: (batch_size, action_horizon, num_robot_joints)
  - Action_horizon: how many time steps into the future are we predicting actions for.
  - num_robot_joints is the number of maneuverable robot actuators. This is the action vector.
    - Eg. the Panda robot of Robosuite, the number is 7.

The Model

CNN-based diffusion policy:

Conditional distribution → $p(A_t|O_t)$

- Condition action generation process on observation features with Feature-wise Linear Modulation (FiLM) & denoising iteration k
- Performs well except when action sequence changes quickly and sharply through time because CNNs have an over-smoothing effect.

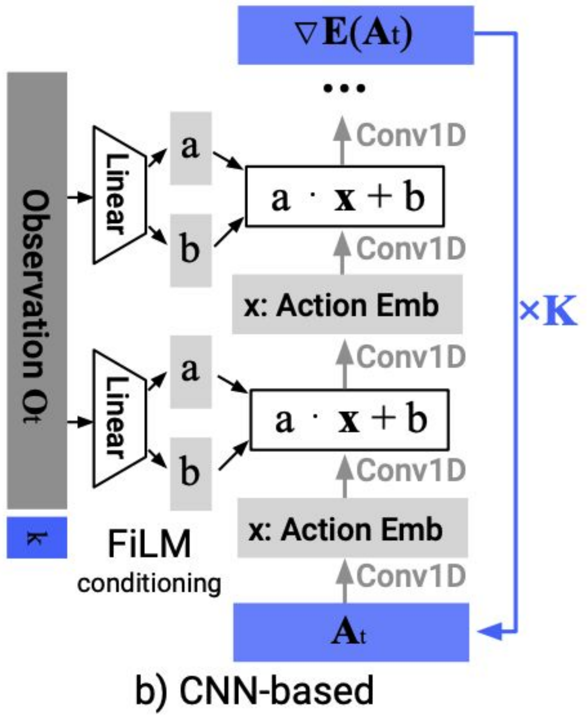
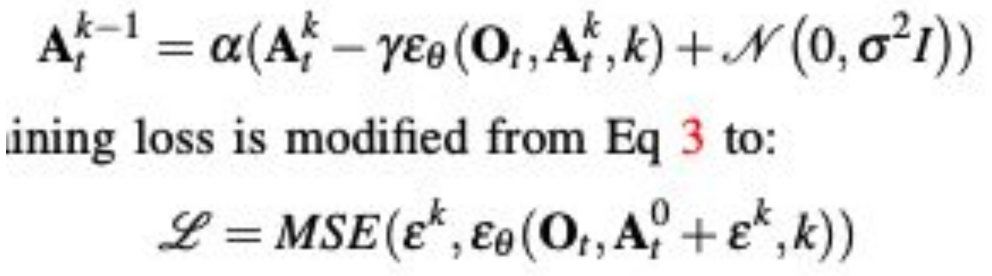


Figure 3: CNN-based diffusion policy [1].

Model Training



$$\mathbf{A}_t^{k-1} = \alpha(\mathbf{A}_t^k - \gamma\varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k) + \mathcal{N}(0, \sigma^2 I))$$

ining loss is modified from Eq 3 to:

$$\mathcal{L} = MSE(\varepsilon^k, \varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^0 + \varepsilon^k, k))$$

Figure 4: Diffusion policy loss and action prediction [1].

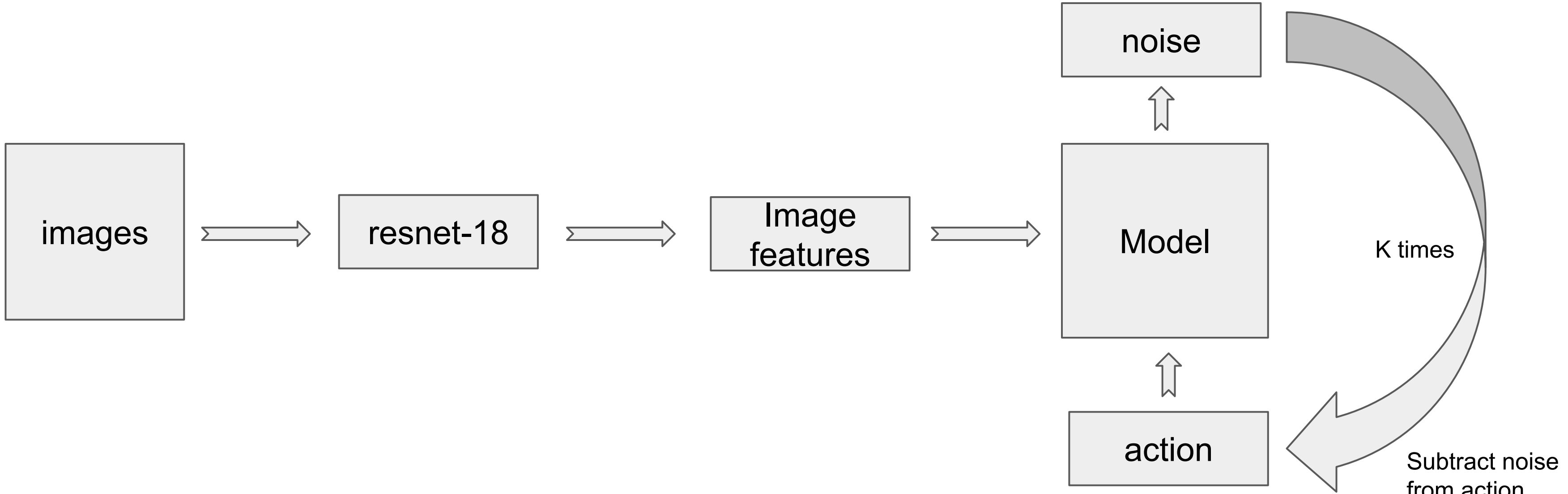


Figure 5: high-level process of diffusion policy.

Results

Model ready to be trained.

I trained one model on 6 demos but the resulting policy is not useful.

I attempted to train the model on a larger dataset, but due to memory limitations, I was unable to load the data.

- Need zarray library for loading compressed data into the memory.
    - Zarr is a file storage format for chunked, compressed, N-dimensional arrays based on an open-source specification.

Next Steps

- Load compressed data into the memory.
- Train the model on the larger dataset.

Sources

[1] Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., & Song, S. (2023, June 1). *Diffusion policy: Visuomotor policy learning via action diffusion*. arXiv.org. https://arxiv.org/abs/2303.04137