# Imitation Learning for Generalizable Household Manipulation

**Bohan Yang, Nicole Li**
by93@cornell.edu
yl3558@cornell.edu

## 1 Introduction

We present SkilledRobot, a simulated robot capable of performing household manipulation tasks, by learning from demonstrations from expert data. Our study investigates three imitation learning approaches: Behavior Cloning (BC) with a multilayer perceptron (MLP), DAgger, and a Transformer-based BC model. The objective is to evaluate how well each method generalizes across variations in initial states and visual observations under data constraint settings. We begin by training a standard `Multimodal-MLP-BC` using expert trajectories from the ManiSkill2 dataset [1]. To address the issue of compounding errors, we then incorporate DAgger into our pipeline. Finally, we introduce a `Transformer-BC` policy to leverage its capacity for modeling long-range dependencies and fusing multimodal inputs through self attention and cross attention mechanisms. We benchmark these methods on the `PegInsertionSide-v0` and compare against its `Baseline-BC`. We hypothesize that the Transformer-based BC model will outperform the baselines due to its ability to model long-range dependencies in sequential decision-making, better fuse multimodal inputs, and generalize to out-of-distribution states.

## 2 Problem Statement

We consider the standard imitation learning setting, where the dataset consists of expert demonstrations in the form of state-action trajectories:

$$\mathcal{D} = \{\tau_i\}_{i=1}^{N}, \quad \text{where each trajectory } \tau_i = ((s_1, a_1^*), (s_2, a_2^*), \ldots, (s_T, a_T^*))$$

Here, $s_t$ is the state at time step $t$, and $a_t^*$ is the corresponding expert action.

We aim to learn a deterministic policy $\pi_\theta$, parameterized by $\theta$, that maps states to actions:

$$a = \pi_\theta(s), \quad \text{where} \quad s = (\texttt{qpos}, \texttt{qvel}, \texttt{tcp}, \texttt{hand\_rgbd}, \texttt{base\_rgbd}), \quad a \in \mathbb{R}^8$$

The training objective is to minimize the expected loss over all trajectories in the dataset:

$$\theta^* = \arg\min_\theta \mathbb{E}_{\tau \sim \mathcal{D}} \left[ \sum_{t=1}^{T} \mathcal{L}(\pi_\theta(s_t), a_t^*) \right]$$

where $\mathcal{L}$ is typically chosen to be the mean squared error (MSE):

$$\mathcal{L}(\pi_\theta(s_t), a_t^*) = \|\pi_\theta(s_t) - a_t^*\|_2^2$$

The state $s$ is a multimodal observation vector composed of proprioceptive data and two RGB-D images. The action is an 8-dimensional vector.

Table 1: State Observation Components

| Component | Description | Shape |
|---|---|---|
| qpos | Robot joint positions, including arm and gripper joints | [9] |
| qvel | Joint velocities for all 9 degrees of freedom | [9] |
| tcp | Tool Center Point pose, including 3D position and quaternion orientation | [7] |
| hand_rgbd | RGB-D image from the wrist-mounted hand camera, capturing close-up views | [4, 128, 128] |
| base_rgbd | RGB-D image from a static base-mounted camera, capturing global context | [4, 128, 128] |

## 3 Approach

### 3.1 Simulation and Dataset

We use the ManiSkill2 simulation environment and select the `PegInsertionSide-v0` task as our primary benchmark. Expert demonstration data is sourced from ManiSkill2's HuggingFace datasets. For all experiments below, we use 100 trajectories from their dataset and split the training and validation data in a ratio of 8:2.

### 3.2 ManiSkill Baseline Behavior Cloning

As a baseline for evaluating cross-task generalization, we conducted an experiment in which a `Baseline-BC` model designed for a relatively simple `PushCube-v0` task was trained using the data from `PegInsertionSide-v0`.

We took the base image RGBD and proprioceptive states, and was trained for 2000 iterations using a batch size of 32. The architecture followed a standard two-tower setup, consisting of a CNN-based visual encoder and an MLP-based state encoder, with latent fusion of features.

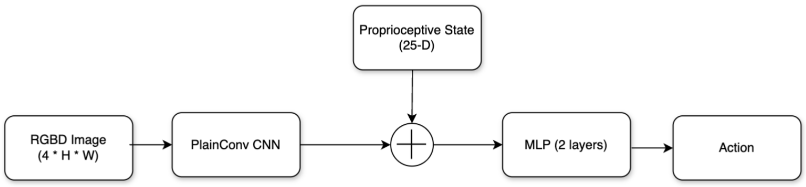

Figure 1: `Baseline-BC` architecture

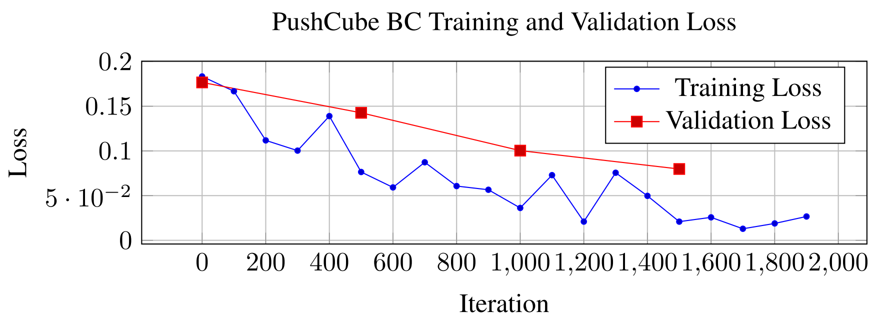

Figure 2: Loss curve of `Baseline-BC` tested on `PegInsertionSide-v0`.

### 3.3 Multimodal Behavior Cloning

We improve from the baseline. We built a `Multimodal-MLP-BC`. The policy learns from a multimodal observation space that includes both the hand image and the base image and proprioceptive state data (joint positions, joint velocities, and the end-effector pose). The goal is to directly map these observations to continuous action vectors provided by an expert demonstrator in the ManiSkill2 environment.

To prepare the data, the code defines a preprocessing function that first normalizes the RGB image to the [0, 1] range and ensures the depth image is in the correct shape. The RGB and depth images are concatenated along the channel dimension to form a single 4-channel image with shape (4, H, W).

The model architecture consists of two major components: a vision encoder and a state encoder, whose outputs are fused and decoded into an action prediction. The vision encoder uses a ResNet-18 model, pretrained on ImageNet. We modified it to accept 4-channel input to accommodate the RGBD data. Specifically, the original first convolutional layer is replaced to accept four input channels, copying the pretrained weights for the RGB channels and initializing the new depth channel weights to zero. The classification head is removed, and the encoder outputs a 512-dimensional feature vector after global average pooling.

In parallel, the proprioceptive state vector, composed of the concatenation of `qpos`, `qvel`, and `tcp_pose`, is passed through an MLP. This state encoder is composed of four fully connected layers with ReLU activation functions, reducing the dimensionality to a 64-dimensional latent representation. The two feature vectors, 512 from the vision encoder and 64 from the state encoder, are concatenated to form a 576-dimensional joint representation, which is then processed by another MLP that serves as the action decoder. This final head predicts an 8-dimensional continuous action vector.

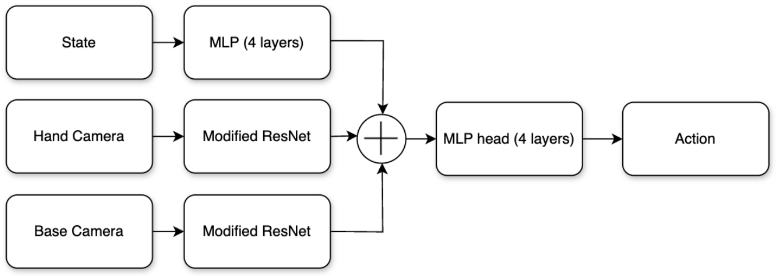

Figure 3: `Multimodal-MLP-BC` architecture

This architecture shows how vision and state information are encoded separately before being merged. It effectively leverages pre-trained visual features from large-scale datasets while adapting to robotic control via learned proprioceptive embeddings.

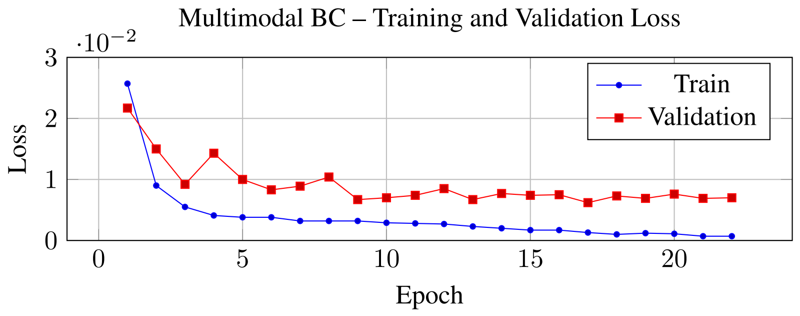

Figure 4: Loss curves for the `Multimodal-MLP-BC`. Early stopping triggered after epoch 22 when validation loss plateaued.

### 3.4 DAgger

To mitigate the distribution shift in Behavior Cloning, we explored the use of Dataset Aggregation (DAgger). Traditional DAgger involves querying an expert to relabel actions along the agent's own rollout trajectories, which requires expert annotation tools such as teleoperation. However, due to the lack of teleoperation equipment, we developed a modified DAgger approach. Instead of live expert queries, we approximate expert actions by performing a nearest neighbor (NN) search based on cosine similarity between the current observed state and states in a meta-dataset. This meta-dataset consists of the full ManiSkill dataset, whereas our original BC model was trained on only a subset.

The expert is simulated through NN search implemented in the `FullRgbdBCDataset` class, which loads all demonstration episodes and provides a `find_nearest_action` method. This method computes the cosine similarity between a query observation—constructed by flattening RGBD images and concatenating them with proprioceptive states (joint positions, velocities, and TCP pose)—and every entry in the dataset. The action associated with the most similar stored sample is used as the expert label.

A DAgger rollout is conducted by stepping through the environment using the current policy, and queries the simulated expert (find_nearest_action) at each timestep to label visited states. These new expert-labeled transitions are appended to the training set. We then retrain the model from scratch, applying supervised learning using mean squared error (MSE) loss over multiple epochs.

Despite its effectiveness in approximating expert behavior, this modified DAgger approach has limitations. The nearest neighbor search creates significant performance bottlenecks since cosine similarity calculation at every environment step is compute intensive. Even with optimizations such as batch similarity evaluation and matrix precomputation, the process remains too slow to finish within acceptable timeframes.

Listing 1: Lecture DAgger

```
Initialize random policy $\pi_1$
$\mathcal{D} \leftarrow \emptyset$ # empty buffer

for $i = 1, \dots, N$ do
    # Roll out current policy
    $\mathcal{D}_i = \{(s_0, a_0), (s_1, a_1), \dots\}$

    # Query expert $\pi^\star$ on learner
        states
    $\mathcal{D}_i = \{(s_0, \pi^\star(s_0)), (s_1, \pi^\star(s_1)), \dots\}$

    # Aggregate and retrain
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
    $\pi_{i+1} \leftarrow$ Train($\mathcal{D}$)

Select best $\pi_{1:N+1}$
```

Listing 2: Our DAgger implementation

```
Load BC checkpoint $\rightarrow$ $\pi_1$ (else random)

expertDB $\leftarrow$ FullRgbdBCDataset(full_demo.h5)
$\mathcal{D} \leftarrow$ RgbdBCDataset(train_split.h5)
Val $\leftarrow$ RgbdBCDataset(val_split.h5)

for $i = 1, \dots, 5$:
    $\mathcal{D}_i \leftarrow \emptyset$
    for episode $= 1, \dots, 5$:
        $s \leftarrow$ env.reset()
        while $\neg$ done:
            $a_{\texttt{pred}} \leftarrow \pi_i(s)$
            $a^\star \leftarrow$ expertDB.find_nearest_action($s$)
            $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{(s, a^\star)\}$
            $s$, done $\leftarrow$ env.step($a^\star$)

    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
    $\pi_{i+1} \leftarrow$ Train($\mathcal{D}$, Val, epochs $= 3$)
    save_if_best($\pi_{i+1}$, Val)

save($\pi_{\texttt{final}}$, "dagger_multimodal_policy.pt")
```

Figure 5: Lecture DAgger vs. our nearest-neighbor adaptation.

### 3.5 Transformer-Based BC

We propose a `Transformer-BC` architecture that improves upon our original `Multimodal-MLP-BC` policy by introducing a more expressive design that leverages attention over both RGBD visual inputs and robot positional state. This model is designed for the ManiSkill2 manipulation environment and aims to enhance generalization and robustness by explicitly modeling temporal and multimodal dependencies.

The policy receives RGBD observations from two wrist-mounted cameras (a hand camera and a base camera). Each RGBD image is preprocessed into a 4-channel format (3 RGB + 1 normalized depth) and resized to $224 \times 224$. We adapt a pretrained Vision Transformer (ViT-B/16) to accept 4-channel input by replacing the initial 3-channel projection layer with a 4-channel convolutional layer, initializing the depth channel weights as the mean of the RGB weights. Each camera stream is encoded independently via this modified ViT, producing a compact visual embedding.

In parallel, the robot's proprioceptive state—composed of joint positions ($q_{\text{pos}}$), joint velocities ($q_{\text{vel}}$), and TCP pose—is embedded through a linear layer into a 64-dimensional vector. This vector passes through a learnable positional encoding MLP, followed by a 2-layer Transformer encoder with multi-head self-attention. This enables the model to capture internal dependencies among components of the robot's state.

The resulting visual features (from both cameras) and the encoded state embedding are concatenated and fused via a 4-layer MLP head that outputs an 8-dimensional continuous action vector. This multimodal fusion strategy allows the policy to condition its predictions jointly on visual scene context and dynamic physical state.

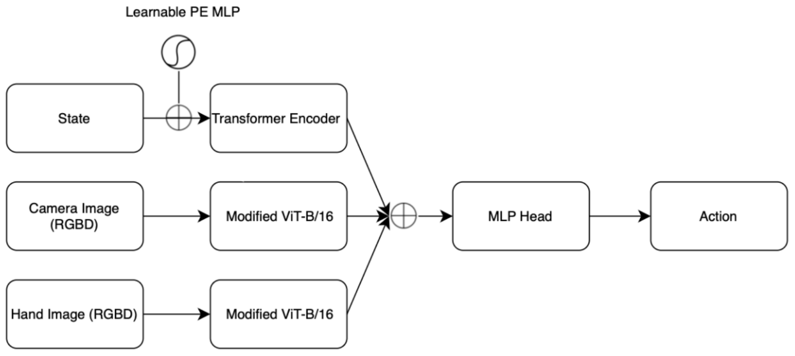

Figure 6: `Transformer-BC` architecture

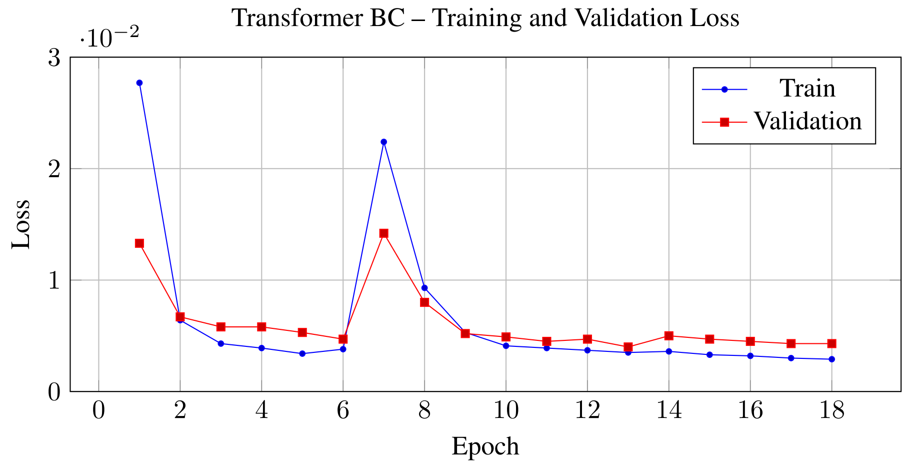

Figure 7: Loss curves for the `Transformer-BC`. Early stopping was triggered after epoch 18.

## 4 Experiments

When deployed in the `PegInsertionSide-v0` environment, as can be seen in the rendered roll-out, the `Baseline-BC` policy failed to perform meaningful behaviors. The arm movements were not directed toward the insertion target, and the peg rarely made contact with the hole. This poor performance highlights the task-specific nature of imitation learning and the brittleness of policies when exposed to unfamiliar distributions.

This result underscores the difficulty of transferring learned behaviors across tasks with different spatial and temporal requirements. It motivates the need for architectures that can better encode generalizable skills—such as those incorporating Transformer-based attention over diverse observations—and supports our goal of developing a more robust imitation learning pipeline for complex household tasks.

As shown in the loss curve (Figure 4), the `Multimodal-MLP-BC` model achieves significantly lower training and validation loss compared to the baseline BC implementation, indicating more effective learning. While the rendered policy execution reveals that the robot arm is now able to localize and reach toward the peg, it still struggles with lifting it and fails to align the peg with the target hole for successful insertion.

It can be seen from the loss plot (Figure 7) and rendered rollouts that the Transformer-BC model achieves the best performance among all evaluated architectures. The final evaluation loss is the lowest comparison among the three. During the experimental rollout, the robot arm successfully picks up the peg and aims toward the hole.

The results from the evaluation and the experiments supported our hypothesis. The `Transformer-BC` achieves the best result among the three: `Baseline-BC`, `Multimodal-MLP-BC`, and `Transformer-BC`. The superior result likely is due to the Transformer's attention mechanisms. The use of Vision Transformer provides power visual encoders than plain CNN or RestNet. Furthermore, the ability to effectively weigh and combine information from diverse sensor modalities enables the policy to build a more robust representation to success in the `PegInsertionSide-v0` task.

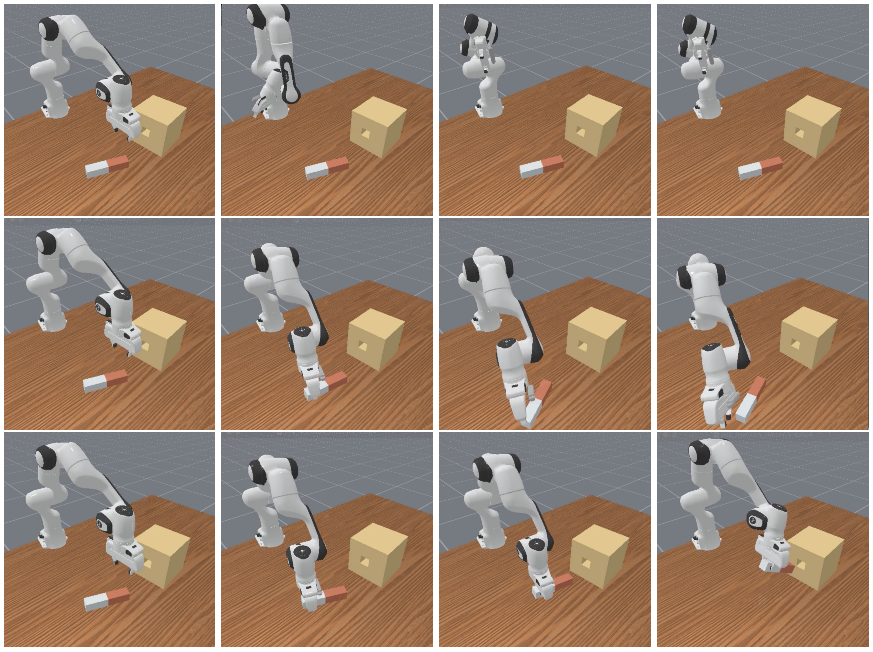

Figure 8: Snapshots from three imitation learning policies on the `PegInsertionSide-v0` task. Top row: `Baseline-BC` shows failure to grasp or insert. Middle row: `Multimodal-MLP-BC` improves stability and control. Bottom row: `Transformer-BC` demonstrates precise and robust insertion behavior.

## 5 Training

All model training are conducted on Google Colab system using either a T4 or A100 GPU. All models are conducted on the same training and validation dataset. The batch size is 32. We used Adam optimizer. Learning rate starts initially at 1e-4 and used a learning rate scheduler to decrease the learning rate by half if the validation loss do not decrease by two epochs. We have an early stopping mechanism to stop the training if the validation loss do not decrease in 5 epochs. The lowest validation checkpoint is saved and used for experiments.

## References

[1] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.