A Comparison of Imitation Learning Algorithms for Bimanual Manipulation

Michael Drolet[1,2], **Simon Stepputtis**[3], **Siva Kailas**[3], **Ajinkya Jain**[4]
Jan Peters[2], **Stefan Schaal**[4], **and Heni Ben Amor**[1]
[1]School of Computing and Augmented Intelligence, Arizona State University, United States
[2]Department of Computer Science, Technical University of Darmstadt, Germany
[3]The Robotics Institute, Carnegie Mellon University, United States
[4]Intrinsic AI, An Alphabet Company, United States

Abstract: Amidst the wide popularity of imitation learning algorithms in robotics, their properties regarding hyperparameter sensitivity, ease of training, data efficiency, and performance have not been well-studied in high-precision industry-inspired environments. In this work, we explore the limitations and advantages of prominent imitation learning algorithms and evaluate them on a complex bimanual manipulation task involving multiple contacts. We show that while imitation learning is effective for such tasks, not all algorithms are equal in handling environmental and hyperparameter perturbations, training demands, and usability. Our study uses a carefully designed experimental procedure to assess these key characteristics.

Keywords: Bimanual Manipulation, Imitation Learning

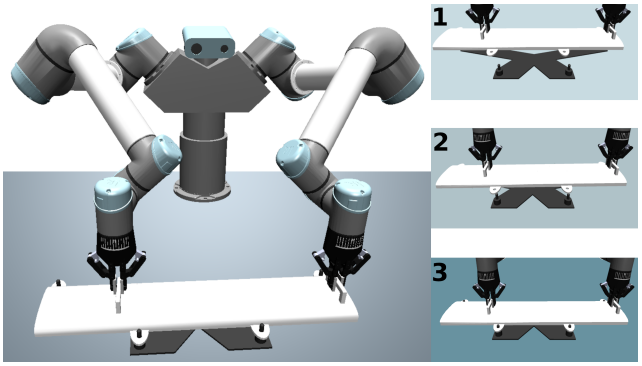

Figure 1: The robot completes the final stage of the high-precision four-peg insertion task.

1 Introduction

Given the desirable properties of imitation learning (IL) and the importance of bimanual manipulation in achieving more sophisticated human-like robots, a natural question is how to approach the intersection of these two fields. We address this question directly by combining several foundational algorithms in IL with a benchmark MuJoCo [2] environment that seeks to fairly and extensively compare algorithms in terms of sample efficiency, noise robustness, compute time, and performance. In doing so, we provide an extensive discussion related to the various advantages and disadvantages of these algorithms as well as the engineering approaches that allow for learning in such an environment. More specifically, we focus on Generative Adversarial Imitation Learning (GAIL) [3], Implicit Behavioral Cloning (IBC) [4], Dataset Aggregation (DAgger) [5], Behavioral Cloning (BC) [6], Acting Chunking Transformer (ACT) [7], and the Diffusion Policy [8]. In the environment, the robot learns to transfer an adapter with four holes and insert it into a stationary adapter with four pegs. The difficulty in the task lies within the low tolerance of the adapters, such that success only occurs when the robot is precise; i.e., the holes have a diameter of 11mm and the rounded pins have a base diameter of 10mm, leaving approximately 1mm of tolerance.

CoRL 2024 Workshop on Whole-Body Control and Bimanual Manipulation (CoRL 2024 WCBM).

2 Related Work

A few prominent learning-based approaches have emerged in the domain of bimanual manipulation. Reinforcement learning is one class of approaches that have been applied. For example, [9] presents a set of bimanual manipulation tasks and associated reward structures that were empirically found to work well with deep reinforcement learning. One study also utilized a marker-based vision system, sim-to-real, and reinforcement learning for connecting two blocks with magnetic connection points with two robotic arms [10].

A method for mastering contact-rich manipulation in a similar setup has been proposed, using motor primitives to train the robot for an insertion task [1]. This approach involves utilizing force feedback and environment feedback as stimuli during the filtering process outlined in Bayesian Interaction Primitives [11]. Although this method displays efficacy, our objective is to eliminate the inductive bias associated with motor primitives and explore neural network-based approaches instead. In doing so, we adopt a more general class of function approximation.

ALOHA is a recent approach to learning several fine-grained bimanual manipulation tasks with everyday objects [7]. This approach is notable due to its high degree of success on many tasks that – to our knowledge – were previously only achievable by human demonstrators. ALOHA presents an action-chunking transformer (ACT), which we implement for comparison in this work. Although there are related works in the area of bimanual manipulation, such as HDR-IL [12] and SIMPLe [13], we seek to investigate algorithms that have been widely used over the last several years in robotics and require a minimal number of demonstrations.

A separate study discusses the essential components of training adversarial imitation learning algorithms [14]. This study is extensive in terms of evaluating different hyperparameters, discriminator configurations, and training-related metrics. In this investigation, we are interested in not only adversarial methods but also how they compare to other methods for bimanual manipulation. Other studies evaluate various imitation learning algorithms and their hyperparameters [15, 16]. However, key algorithms such as DAgger and IBC have been excluded from the scope of these studies. Moreover, the gym environments employed in these studies fall short in directly capturing the nuanced dynamics and fine-grained behaviors inherent in our bimanual robot setup.

3 Algorithm Selection

The selection of algorithms is a critical point of consideration; consequently, the chosen algorithms can be seen as an orthogonal set of approaches to imitation learning in general. On one hand, the offline and supervised learning (SL) aspect of IL is captured by both an expressive energy-based policy implementation (IBC) and the widely familiar Gaussian neural network-based policy (BC). On the other hand, methods that interact with the environment in the form of (a) an oracle (DAgger) to minimize covariate shift, and (b) a reinforcement learning policy (GAIL) help capture the class of approaches reliant on sampling states online. We additionally adopt some of the most recent and successful methods in IL for robotics, namely ACT and Diffusion Policy. We believe that while there are many derivatives of these methods, their longstanding impact on the field of imitation learning helps justify the need to compare them.

BC is one of the most well-known and widely used imitation learning algorithms, largely due to its simplicity and effectiveness on large datasets. BC is typically implemented using the following objective: $\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}} \, \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\boldsymbol{\tau}_E} \left[\log(\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}))\right]$, where $\boldsymbol{\tau}_E$ represents the state-action trajectories from the expert, $\pi_{\boldsymbol{\theta}}$ is the current policy, $\mathbf{a}$ is a continuous action, and $\mathbf{s}$ is the observed continuous state. We additionally include a tunable L_1 and L_2 penalty on the parameters of the policy (known as elastic net regularization) to help prevent overfitting. In the context of this paper, we refer to BC as solely training a Gaussian policy whose mean is given by a feed-forward neural network, $\pi(\mathbf{a}|\mathbf{s})$. Recent successes in behavioral cloning for robot manipulation include the RT-X [17] model and its precursors, all trained on very large datasets (each consisting of $130,000$ training demonstrations or more). Our work, however, studies BC in the small data regime, using a maximum of 200 expert demonstrations.

Algorithm	Env. Interaction	Policy Class	Train π
BC, ACT	✗	Gaussian, Deterministic	SL
IBC, Diffusion	✗	EBM, Langevin	SL
DAgger	✓	Gaussian	SL
GAIL	✓	Gaussian	RL

Table 1: Comparison of Algorithms.

ACT performs behavioral cloning using a conditional variational autoencoder (CVAE) implemented as a multi-headed attention transformer [18]. While the objective for training ACT is largely the same as vanilla BC (Section 3), we will briefly describe its differences from the standard formulation. For one, ACT is trained (as in the original work) using the L_1 loss, which can be interpreted as being proportional to the square root of the Mahalanobis distance used in the objective of vanilla BC, assuming a fixed variance. This deterministic policy then predicts a sequence of actions instead of a single action and- in our formulation- uses a history of observations as input to the transformer instead of image observations.

IBC is a supervised learning approach using energy-based models, trained using the Negative Counter Example (NCE) loss function [19]. Energies are assigned to the state-action pairs, and the policy takes the action that minimizes the energy landscape. As the minimum over the actions is taken, IBC has the advantage of handling discontinuities that can arise in the typical regression setting, where behavioral cloning may simply interpolate. This is a desirable feature of implicit models, and it is one of the presented advantages in the IBC work that makes it unique compared to other imitation learning algorithms. In short, the IBC policy can be summarized as: $\hat{\mathbf{a}} = \text{argmin}_{\mathbf{a}} E_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$, where $E_{\boldsymbol{\theta}}$ is the energy function and $\hat{\mathbf{a}}$ is the optimal action. However, many works have found that the IBC objective is numerically unstable and does not consistently yield high-quality policies [20].

Diffusion Policy, like IBC, performs an iterative procedure to generate actions. Diffusion models have achieved significant success in areas like image generation [21]. Their observed stability, compared to energy-based models, makes them a promising method for the robotics domain. Using a series of denoising steps, this method presents a way to refine noise into actions through a learned gradient field. The Diffusion Policy in this work is implemented using a U-Net architecture, which conditions on an observation history and generates an action sequence similar to ACT. Algorithms such as IBC and Diffusion Policy (based on Langevin dynamics) provide viable alternatives to the standard behavioral cloning formulation, which may lack the expressiveness these models provide.

GAIL formulates imitation learning as an inverse reinforcement learning (IRL) problem, wherein the reward function is learned based on the discriminator's scores [3] using a Generative Adversarial Network (GAN). The parameters $\mathbf{w}$ for the discriminator D are updated using the following objective: $\hat{\mathbb{E}}_{\boldsymbol{\tau}_i}\left[\nabla_{\mathbf{w}} \log\left(D_{\mathbf{w}}(\mathbf{s}, \mathbf{a})\right)\right] + \hat{\mathbb{E}}_{\boldsymbol{\tau}_E}\left[\nabla_{\mathbf{w}} \log\left(1 - D_{\mathbf{w}}(\mathbf{s}, \mathbf{a})\right)\right]$, where $\boldsymbol{\tau}_i$ represents state-action trajectories from the most recent policy (at iteration i). Our policy is updated using Trust Region Policy Optimization (TRPO) [22]. Derivitavies of GAIL, such as VAIL [23], seek to address the issue of generator/discriminator imbalance by using a variational bottleneck to constrain the gradient updates of the networks.

DAgger addresses the covariate shift problem, where the distribution of observations the policy encounters differs from those in the expert dataset [5]. To tackle this challenge, a data aggregation scheme is used wherein the policy is re-trained on the history of expert-labeled states encountered over time. The need to specify an optimal action at all possible states without using a human can make implementing DAgger challenging. DAgger theoretically bounds the training loss of the best policy under its distribution of sampled trajectories, where the tightness of this bound depends on, e.g., the number of iterations, samples per iteration, and mixing coefficient. However, its practical efficacy is constrained by factors such as the quality of the oracle and the difficulty of the environment.

4 Methodology

In the following section, we describe the methods for creating the bimanual manipulation insertion expert, as well as the design considerations necessary for learning with such a

system. The implementation contains a two-stage expert, as outlined in Algorithm 1. A dynamics model is proposed that allows for implicit control of the torso, such that the state formulation (described in Section 4.2) can be predominantly characterized by the end-effectors. The robot consists of two UR5 arms, each mounted to a rotating torso and equipped with Robotiq 2F-85 grippers.

4.1 Bimanual Manipulation Expert Controller

We are first given n original demonstrations, where the i'th demonstration contains positions $\boldsymbol{\rho}_*^{i,j}(t) \in \mathbb{R}^3$ and quaternions $\boldsymbol{\phi}_*^{i,j}(t) \in \mathbf{so}(3)$ for all timesteps $t \in [1, 2, ..., T_i]$ for DoF j. Every original demonstration is then converted to a sequence of expert state-action pairs using the GENERATEEXPERTDEMO procedure. In doing so, we obtain state-action pairs from the same environment and robot used during training. Instead of using quaternions to represent the end effector rotations, we use the six dimensional rotation representation (6DRR) [28] when training the policies. In Algorithm 1, $g(.)$ represents the conversion to 6DRR and $f(.)$ represents the conversion back to quaternions.

Algorithm 1 Bimanual Insertion Expert

procedure PATHFOLLOWEXPERT(i, t)
  $\boldsymbol{\rho}, \boldsymbol{\phi}$ = GETROBOTSTATE() ; t' = MIN(t, T_i)
  for j in $[1, ..., J]$ **do**
    $\Delta\boldsymbol{\rho}^j$ = FEEDBACKCTRL($\boldsymbol{\rho}^j - \boldsymbol{\rho}_*^{i,j}(t')$)
    $\Delta\boldsymbol{\phi}^j$ = CLIP(DIFF($\boldsymbol{\phi}^j, \boldsymbol{\phi}_*^{i,j}(t')$))
  return $\left([\Delta\boldsymbol{\rho}^1, ..., \Delta\boldsymbol{\rho}^J], g([\Delta\boldsymbol{\phi}^1, ..., \Delta\boldsymbol{\phi}^J])\right)$

procedure GETEXPERTACTION(i, t)
  if DOPATHFOLLOW **then**
    $(\Delta\boldsymbol{\rho}, \Delta\boldsymbol{\phi})$ = PATHFOLLOWEXPERT(i, t)
  else if DOINSERTION **then**
    $(\Delta\boldsymbol{\rho}, \Delta\boldsymbol{\phi})$ = INSERTEXPERT()
  return $(\Delta\boldsymbol{\rho}, \Delta\boldsymbol{\phi})$

procedure GENERATEEXPERTDEMO(i)
  $\mathcal{D}_i \leftarrow \emptyset$; $t = 1$; $\Delta\boldsymbol{\phi} = \mathbf{0}$; $\Delta\boldsymbol{\rho} = \mathbf{0}$
  while not DONEINSERT **do**
    $\boldsymbol{\rho}, \boldsymbol{\phi}$ = GETROBOTSTATE()
    $\mathbf{s}$ = APPLYOSC($\boldsymbol{\rho} + \Delta\boldsymbol{\rho}, \boldsymbol{\phi} + f(\Delta\boldsymbol{\phi})$)
    $\mathbf{a}$ = GETEXPERTACTION(i, t)
    $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup (\mathbf{s}, \mathbf{a})$; $t = t + 1$
  return $\mathcal{D}_i$

Algorithm 1 describes the process for collecting the expert demonstrations. The process consists of a PATHFOLLOWEXPERT used to transfer the dynamic object above the stationary object, similar to the original demonstrator. The INSERTEXPERT is then used to precisely align the holes and pegs of the adapters to complete the task, using the features of the objects to create a feedback signal. The INSERTEXPERT is omitted for brevity in Algorithm 1, but it can be used independently of the first stage (hence its independence of time t and demonstration index i).

The operational space controller (OSC) [26] is used in this work to facilitate learning in the task space [27]. We adopt the Jacobian pseudo-inverse method, using the dynamically consistent generalized inverse. Because we control two UR5 arms and one base joint implicitly, $\mathbf{q}$ is 13-dimensional. Consequently, we have the Jacobian $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{12\times13}$, inertia matrix $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{13\times13}$, and forces due to gravity $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^{13}$. The force vector used to control the robot is then calculated as: $\mathbf{u} = \mathbf{J}(\mathbf{q})^\top \mathbf{M}_x(\mathbf{q})\tilde{\mathbf{x}} + \mathbf{M}(\mathbf{q})\tilde{\dot{\mathbf{q}}} + \mathbf{g}(\mathbf{q})$, where $\mathbf{M}_x(\mathbf{q}) = (\mathbf{J}(\mathbf{q})\mathbf{M}(\mathbf{q})^{-1}\mathbf{J}(\mathbf{q})^\top)^\dagger$. Finally, we apply a nullspace filter to the force output: $\mathbf{u} = \mathbf{u} + \left(\mathbf{I} - \mathbf{J}(\mathbf{q})^\top \bar{\mathbf{J}}(\mathbf{q})^\top\right)\mathbf{u}_{\text{null}}$, where $\bar{\mathbf{J}}(\mathbf{q}) = \mathbf{M}^{-1}(\mathbf{q})\mathbf{J}(\mathbf{q})^\top \mathbf{M}_x(\mathbf{q})$ and $\mathbf{u}_{\text{null}} = \mathbf{K}_n\mathbf{M}(\mathbf{q})(\dot{\mathbf{q}}_* - \dot{\mathbf{q}})$. We take $\dot{\mathbf{q}}_*$ to be $\mathbf{0}$ (as in the velocity controller), and $\mathbf{K}_n$ is a parameterized diagonal matrix.

4.2 Environments

The base environment consists of an 18-dimensional action space and a 36-dimensional observation space. The action space, as previously described in Section 4.1, consists of a delta-position ($\Delta\boldsymbol{\rho}$) and delta-rotation ($\Delta\boldsymbol{\phi}$) command for both end-effectors. The observation space is characterized by (1) the difference in the "expert's pose at the hover position above the stationary adapter" and the "current end-effector pose"; (2) the cube-root of the distance between the end-effector and respective near-side pin; and (3) the forces and torques acting

upon the gripper sensors. This observation space is "duplicated" for both arms, so it can be viewed as having an 18-dimensional observation per arm.

At the start of every episode, the stationary adapter remains at a fixed location and the dynamic adapter is placed at a randomly chosen starting location based on the 200 original expert demonstrations. During the original expert demonstrations, this adapter is placed at a randomly generated position on the right side of the robot's workspace. An environment reward is available at every timestep, although it is not used by the learning algorithms. The environment reward, which we use to help measure an algorithm's success (Section 6), is defined as follows:

$$R(\mathbf{s},t) = \sum_{j=1}^{J} \left[e^{\gamma(\mathbf{x}^j(t) - \mathbf{x}^j_*(t))^2} \oplus e^{\lambda(d(\boldsymbol{\phi}^j(t), \boldsymbol{\phi}^j_*(t)))^2} \right] - \eta + \omega \tag{1}$$

where η is a time penalty; ω is a positive reward for successfully inserting the adapter; and $\oplus$ is used to concatenate the six terms in the summand with the previous iteration, and then take mean of this result after iteration J (for lack of better notation). There are three environments in total, namely, the Zero Noise, Low Noise, and High Noise environments (see Section 5.2 for more detail). In all environments, $\gamma = -10$, $\lambda = -10$, $\eta = 1$, and $\omega = 100$, resulting in an average expert reward of 64.03 ± 0.45 (1 SE) over 600 demonstrations.

5 Experimental Setup

Our experiment is divided into Analysis of Action and Observation Noise (5.2), Hyperparameter Search (5.1), and Analysis of Hyperparameter Sensitivity (5.3). The results of each phase are used to provide an interpretation of key metrics presented in Section 6.

5.1 Hyperparameter Search

A hyperparameter search is conducted to obtain the best parameters for all algorithms, using the Zero Noise environment with 200 expert demonstrations. For a given set of hyperparameters, 10 evenly spaced evaluations during training are conducted, and the evaluation resulting in the highest average reward (out of 10 rollouts) is returned to the optimizer. Each algorithm's best (i.e., reward maximizing) hyperparameters have an asterisk in Table 2. We note that for behavioral cloning-based algorithms, alternative hyperparameter configurations achieved performance similar to that of the selected configuration. We found that the default U-Net architecture for the Diffusion Policy (having channel sizes $[256, 512, 1024]$) worked well without the need for additional hyperparameter tuning. Additionally, using an action, observation, and prediction horizon of 4, 4, and 8, respectively worked well for both ACT and Diffusion without the need for tuning.

DAgger HP	HP Search Points
Learn Rate	[5e-5*, 1e-4, 5e-4]
π Layers	[2, 3*]
π Units	[256, 512*]
π Activation	[Relu, Tanh*]
Normalize	[Expert, None*]
Epochs	[64*, 128]
Decay β	[0.9*, 0.95]
BC L1 λ	[0.0, 1e-6*, 1e-4]
BC L2 λ	[0.0, 1e-6*, 1e-4]
BC Batch Sz.	[128, 256*]

GAIL HP	HP Search Points
π Layers	[2*, 3]
π Units	[256*, 512]
π Activation	[Relu, Tanh*]
π Max K.L	[1e-2*, 3e-2]
π C.G Damping	[0.1, 0.3*]
π Ent. Reg	[0.0, 1e-3*, 1e-2]
Normalize	[Expert*, None]
$\mathcal{R}$ Learn Rate	[1e-5, 5e-5*, 1e-4]
$\mathcal{R}$ Layers	[1, 2*]
$\mathcal{R}$ Units	[128, 256*]
$\mathcal{R}$ Activation	[Relu*, Tanh]
$\mathcal{R}$ Ent. Reg.	[0.0, 1e-3*, 1e-2]
Discount λ	[0.97, 0.99*]
$\mathcal{V}$ Layers	[1, 2*]
$\mathcal{V}$ Units	[128*, 256]
$\mathcal{V}$ Activation	[Relu*, Tanh]
$\mathcal{V}$ Max K.L	[1e-2*, 3e-2]
$\mathcal{V}$ C.G Damping	[0.1*, 0.3]

Diffusion HP	HP Search Points
Learn Rate	[5e-5, 1e-4*, 5e-4]
Adam Decay	[1e-6*, 1e-3]
Batch Sz.	[128, 256, 512*]
Diff. Steps	[50, 100*]
L.R. Warmup	[500*, 1000]

BC HP	HP Search Points
Learn Rate	[5e-5, 1e-4, 5e-4*]
π Layers	[2, 3*]
π Units	[256, 512*]
π Activation	[Relu, Tanh*]
Normalize	[Expert*, None]
BC L1 λ	[0, 1e-6, 1e-4*]
BC L2 λ	[0, 1e-6, 1e-4*]
Batch Sz.	[128*, 256, 512]

IBC HP	HP Search Points
Learn Rate	[5e-5, 1e-4*, 5e-4]
π Layers	[2, 4*]
π Units	[256*, 512]
π Activation	[Relu*, Tanh]
Dropout Rate	[0.0*, 0.1, 0.2]
Norm Batch Sz.	[50, 100*]
Norm Samples	[1e3, 5e3*]
Action Samples	[256, 512*, 1024]
Pct. Langevin	[0.8, 1.0*]
Langevin Iter.	[50, 100*]
Counter Ex.	[8, 16, 32*]
Batch Sz.	[256, 512*]
Replay Sz.	[1e3, 1e4*]

ACT HP	HP Search Points
Batch Sz.	[256*, 512]
Enc. Layers	[1*, 2, 3]
Dec. Layers	[1*, 2, 3]
Latent Dim	[8*, 16]
Attn. Heads	[4*, 8]
Learn Rate	[5e-5, 1e-4*, 5e-4]
Dropout Rate	[0.0, 0.1*, 0.2]
Hidden Dim	[128*, 256]
π Units	[256, 512*]
Activation	[Relu*, Gelu]
KL Weight	[1, 10, 100*]

Table 2: Hyperparameter search for all algorithms. Asterisks denote the value used for policy training.

5.2 Action and Observation Noise Analysis

We study the effects of both observation noise and action noise on the success of the learning algorithms. As such, a noise perturbation is applied to the actions ($\mathbf{a}'_i = m_{a_i}\mathbf{a}_i + b_{a_i}$) and observations ($\mathbf{o}'_i = m_{o_i}\mathbf{o}_i + b_{o_i}$) of the expert and environment, respectively. The bias terms, b_{a_i} and b_{o_i}, are randomly sampled from the uniform distributions Unif$[b_{a_i}^{\min}, b_{a_i}^{\max}]$ and Unif$[b_{o_i}^{\min}, b_{o_i}^{\max}]$, respectively. The scaling terms, m_{a_i} and m_{o_i}, are uniformly distributed about 1, having endpoints $[0.9, 1.1]$ in the Low Noise environment and $[0.7, 1.3]$ in the High Noise environment. The noisy actions are not only used to augment the dataset but they are also passed to the inverse dynamics model.

5.3 Hyperparameter Sensitivity Analysis

We conduct a sensitivity analysis on the hyperparameters of each algorithm to assess their local stability using the Zero Noise environment with 200 expert demonstrations. Local stability refers to an algorithm's robustness when hyperparameters deviate slightly from their optimal values. As the number of combinations grows exponentially with the number of hyperparameters, a grid search over all configurations is computationally infeasible. To address this, we use a NIST covering array [29], providing a computationally feasible subset of sampling configurations. We perturb each hyperparameter's best value by ± 15% and evaluate these combinations.

6 Results and Discussion

The results for the Action and Observation Noise Analysis (Section 5.2) are shown in Table 3, and those for the Hyperparameter Sensitivity Analysis (Section 5.3) are in Table 4. During training, 10 evenly-spaced evaluations are conducted, each with 10 rollouts. The evaluation with the highest average return determines the policy's performance. This average return is standardized across all tagged rollouts from different algorithms in the table.

Alg.	N.T.	Zero Noise	Low Noise	High Noise
BC	50	0.36 ± 0.01	−1.06 ± 0.13	−2.43 ± 0.13
	100	0.4 ± 0.0	−0.23 ± 0.05	−1.16 ± 0.08
	200	0.43 ± 0.0	−0.05 ± 0.03	−0.55 ± 0.04
IBC	50	0.39 ± 0.01	−1.58 ± 0.09	−3.14 ± 0.08
	100	0.4 ± 0.0	−1.33 ± 0.11	−2.81 ± 0.07
	200	0.4 ± 0.01	−1.35 ± 0.1	−2.46 ± 0.09
DAgger	—	0.42 ± 0.0	0.39 ± 0.01	0.39 ± 0.01
GAIL	50	0.43 ± 0.0	0.44 ± 0.0	0.43 ± 0.0
	100	0.44 ± 0.0	0.44 ± 0.0	0.43 ± 0.0
	200	0.44 ± 0.0	0.44 ± 0.0	0.43 ± 0.0
ACT	50	0.43 ± 0.0	0.43 ± 0.0	0.43 ± 0.0
	100	0.42 ± 0.0	0.42 ± 0.0	0.42 ± 0.0
	200	0.42 ± 0.0	0.42 ± 0.0	0.42 ± 0.0
Diffusion	50	0.43 ± 0.0	0.43 ± 0.0	0.42 ± 0.0
	100	0.44 ± 0.0	0.44 ± 0.0	0.43 ± 0.0
	200	0.45 ± 0.0	0.44 ± 0.0	0.43 ± 0.0
Expert	—	0.40 ± 0.0	0.40 ± 0.0	0.40 ± 0.0

Table 3: Action and Observation Noise Analysis-Environment Reward (Normalized ± 1 SE)

BC	IBC	DAgger	GAIL	ACT	Diffusion	Expert
0.35	0.25	0.30	−1.35	0.30	0.33	0.29
0.00	±0.01	0.00	±0.13	0.00	0.00	0.00

Table 4: Hyperparameter Sensitivity Analysis Environment Reward (Normalized ± 1 SE)

Figure 5 summarizes our findings from the experimental procedure (Section 5). For the *Hyperparameter Tolerance* metric (Section 5.1), we calculate the average percent success (where success indicates the adapter is correctly placed) across all hyperparameter configurations. The *Noise Tolerance* metric (Section 5.2) reflects the average percent success in Low and High Noise environments for all tagged rollouts. *Compute Efficiency* is the negative log of the average total time (in seconds) to reach the selected policy, linearly scaled to $[0, 100]$. We use the log scale due to the wide variance in runtime, exemplified by GAIL's $128,000$ training rollouts per policy (approx. 2 days). *Performance* indicates the average percent success in the Zero Noise environment across all numbers of expert demonstrations. *Training Stability* is determined by the frequency of evaluation intervals (during training) where the average return shifts from positive to negative, suggesting a decline in policy success. A higher value implies more instability; thus, we use the negative average, scaled to $[0, 100]$.

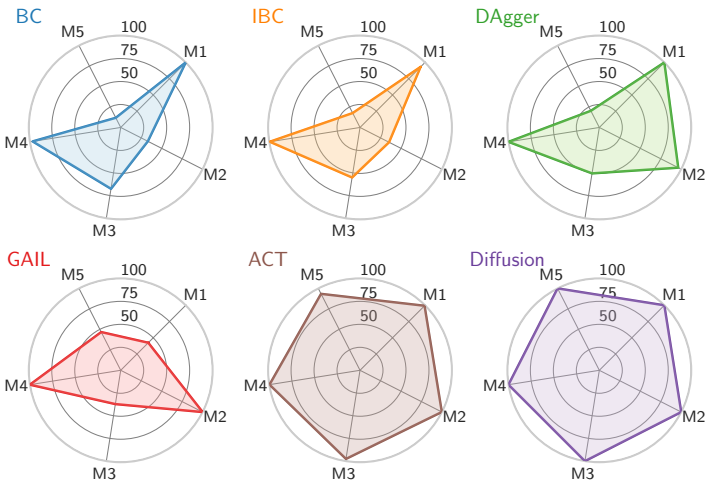

Figure 5: A high-level interpretation of the key metrics describing the algorithms in the bimanual insertion environments.

We observe that algorithms which either interact with the environment (i.e., GAIL and DAgger) or perform action/observation chunking (i.e., ACT and Diffusion) are more robust to noise perturbations. In the former case, this highlights the benefit of exploring during training and the effectiveness of having an oracle in the presence of increased noise, at the expense of more computing and training time. In the latter case, we observe that the action and observation horizons introduced by Diffusion and ACT help cope with potentially non-Markovian environments. However, this design choice increases the dimensionality of the observation and action space, making it less suitable for RL-based methods and IBC, which we observe to be more susceptible to the curse of dimensionality. As an ablation study, we found that observation and action chunking can improve the performance of BC in noisy environments. We also observe that the time required to generate an action during evaluation is longer for IBC and Diffusion; however, for Diffusion, this time is largely affected by the number of de-noising iterations and the length of chunking horizons.

GAIL, Diffusion, and ACT obtained high rewards in all environments, suggesting they are viable options for bimanual manipulation. However, GAIL performs the worst in the hyperparameter sensitivity category, and while an improvement would be made here with an increased maximum time limit for training (approx. 2 days), other algorithms can achieve suitable results in a matter of hours or less. These results indicate that ACT and Diffusion are favorable options, whereas GAIL (and to some extent DAgger) are suitable options for tasks that benefit from extensive interaction with the environment during training. Furthermore, our findings regarding training stability and high performance in fine-grained manipulation environments align with those reported for Diffusion Policy and ACT, respectively.

7 Conclusion

We present a comprehensive assessment of various imitation learning algorithms in a bimanual manipulation environment. A carefully selected set of experiments is conducted to evaluate the key characteristics inherent to these algorithms, such as sample efficiency, sensitivity to perturbations in hyperparameter values, and robustness under observation and action noise. This investigation leads to new insights regarding the applicability of imitation learning for fine-grained industrial tasks and highlights the effectiveness of the chosen methodology. Furthermore, we discuss the implications of selecting these approaches, elucidating how they behave at an empirical and conceptual level. Further investigation into deploying these methods in physical environments will highlight the feasibility of using these systems in everyday situations.

References

[1] S. Stepputtis, M. Bandari, S. Schaal, and H. B. Amor, "A system for imitation learning of contact-rich bimanual manipulation policies," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[2] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012.

[3] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, 2016.

[4] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*. PMLR, 2022.

[5] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011.

[6] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, 1988.

[7] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[8] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[9] Y. Lin, A. Church, M. Yang, H. Li, J. Lloyd, D. Zhang, and N. F. Lepora, "Bi-touch: Bimanual tactile manipulation with sim-to-real deep reinforcement learning," *IEEE Robotics and Automation Letters*, 2023.

[10] S. Kataoka, S. K. S. Ghasemipour, D. Freeman, and I. Mordatch, "Bi-manual manipulation and attachment via sim-to-real reinforcement learning," *arXiv preprint arXiv:2203.08277*, 2022.

[11] J. Campbell and H. B. Amor, "Bayesian interaction primitives: A slam approach to human-robot interaction," in *Conference on Robot Learning*. PMLR, 2017.

[12] F. Xie, A. Chowdhury, M. De Paolis Kaluza, L. Zhao, L. Wong, and R. Yu, "Deep imitation learning for bimanual robotic manipulation," *Advances in neural information processing systems*, 2020.

[13] G. Franzese, L. d. S. Rosa, T. Verburg, L. Peternel, and J. Kober, "Interactive imitation learning of bimanual movement primitives," *IEEE/ASME Transactions on Mechatronics*, 2023.

[14] M. Orsini, A. Raichuk, L. Hussenot, D. Vincent, R. Dadashi, S. Girgin, M. Geist, O. Bachem, O. Pietquin, and M. Andrychowicz, "What matters for adversarial imitation learning?" *Advances in Neural Information Processing Systems*, 2021.

[15] K. Arulkumaran and D. O. Lillrank, "A pragmatic look at deep imitation learning," *arXiv preprint arXiv:2108.01867*, 2021.

[16] L. Hussenot, M. Andrychowicz, D. Vincent, R. Dadashi, A. Raichuk, S. Ramos, N. Momchev, S. Girgin, R. Marinier, L. Stafiniak, *et al.*, "Hyperparameter selection for imitation learning," in *International Conference on Machine Learning*. PMLR, 2021.

[17] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[19] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[20] S. Singh, S. Tu, and V. Sindhwani, "Revisiting energy based models as policies: Ranking noise contrastive estimation and interpolating energy models," *arXiv preprint arXiv:2309.05803*, 2023.

[21] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[22] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015.

[23] X. B. Peng, A. Kanazawa, S. Toyer, P. Abbeel, and S. Levine, "Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow," *arXiv preprint arXiv:1810.00821*, 2018.

[24] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[25] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," *arXiv preprint arXiv:1605.06450*, 2016.

[26] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, 1987.

[27] J. Peters and S. Schaal, "Learning to control in operational space," *The International Journal of Robotics Research*, 2008.

[28] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[29] M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Refining the in-parameter-order strategy for constructing covering arrays," *Journal of Research of the National Institute of Standards and Technology*, 2008.