# Block Pushing: Diffusion Policy vs. Model-Based Approaches

**Yuanhang Zhang**
Robotics Institute
Carnegie Mellon University, United States
yuanhanz@andrew.cmu.edu

**Yifu Yuan**
Robotics Institute
Carnegie Mellon University, United States
yifuy@andrew.cmu.edu

**Zihan Qu**
Robotics Institute
Carnegie Mellon University, United States
zihanqu@andrew.cmu.edu

**Wentao Cao**
Robotics Institute
Carnegie Mellon University, United States
wentaoc@andrew.cmu.edu

***Abstract*— Pushing objects to desired locations is a fundamental manipulation skill in robotics, yet it remains a challenging task due to the complex contact dynamics and the need for precise motion planning. In this paper, we compare Diffusion Policy and Model-Based approaches for effective block pushing strategies. First, we implement a vanilla model-based planner which plans and executes a single push to the target; if unsuccessful, it resets and replans. Then, we train a vision-based Diffusion Policy, that learns an implicit control distribution from the demonstrations collected by the previous model-based planner with human intervention. Our experiments evaluate the success rate, and robustness of both methods under diverse task conditions. The results show that the Diffusion Policy excels in handling uncertainty and generalizing to unseen scenarios, while the model-based approach is more brittle and may require multiple retries. Finally, we validate our findings with real-world robotic experiments. Our real-world evaluation showed that Diffusion Policy achieved a 25% success rate compared to 15% for model-based planning, highlighting the advantages of diffusion policy in terms of both generalizability and efficiency.**

## I. INTRODUCTION

Reinforcement Learning (RL) and Imitation Learning (IL) have been widely explored for robotic manipulation tasks, particularly in block pushing, which serves as a fundamental testbed for dexterous motion planning and control. This has been demonstrated across a variety of tasks, including pick-and-place, grasping, and navigation [1], [2]. Learning-based approaches enable robots to acquire complex skills without explicit system identification, making them suitable for tasks with uncertain dynamics and contact-rich interactions.

Diffusion Policy offers a new paradigm in robot learning, leveraging the generative capabilities of diffusion models to learn visuomotor policies from demonstrations. Unlike conventional policy learning methods, Diffusion Policy generates actions by iteratively refining noise into control signals, capturing multimodal action distributions and improving adaptability to diverse scenarios.

In this work, we train a Diffusion Policy using demonstration data collected from a Model-Based planner with human intervention (i.e., adjusting pushing angle by human). It turns out that our trained Diffusion Policy can learn to dynamically adjust the pushing angle during execution, allowing it to adapt to object dynamics and environmental variations.

In this report, we aim to systematically compare the Diffusion Policy with Model-Based Methods in the specific context of block pushing using a Franka Emika Panda robot arm. Our main contributions are as follows:

- **A Vanilla Model-Based Planner**: We design a simple model-based planner that executes a single push per trial, serving as both a baseline and a data collection mechanism for Diffusion Policy.
- **Vision-Based Diffusion Policy**: We train a Diffusion Policy that learns an implicit control distribution from model-based demonstrations, enabling improved generalization in block pushing.
- **Evaluation in the Challenging Block Pushing Task**: We compare both methods in real-world experiments, demonstrating the superior generalizability and efficiency of the Diffusion Policy.

## II. RELATED WORK

Policy learning for robotic manipulation has been extensively studied in both model-based and model-free paradigms. Model-based methods rely on explicit system dynamics, where future trajectories are optimized based on a predefined dynamics model. These approaches have been widely used in robotic manipulation tasks due to their interpretability and ability to enforce physical constraints. However, they struggle in unstructured environments where accurate models are difficult to obtain.

On the other hand, model-free methods, particularly imitation learning (IL) and reinforcement learning (RL), have gained popularity for robotic control. Behavior Cloning (BC) [3] directly learns a mapping from observations to actions using supervised learning, but suffers from compounding errors due to its inability to correct mistake.

Recent advances in Diffusion Models have introduced new capabilities in generative modeling, with applications extending to policy learning. Diffusion Policy [4] formulates robot control as a denoising diffusion probabilistic model (DDPM), enabling it to capture multimodal action distributions and exhibit stable training behavior. Unlike traditional IL methods, Diffusion Policy refines stochastic noise into actions using Stochastic Langevin Dynamics, allowing it to

generate smooth and consistent motion sequences. Empirical results show that Diffusion Policy outperforms traditional BC and RL methods in high-dimensional manipulation tasks [4].

Moreover, previous research highlights the importance of data quality in IL. Studies have shown that expert demonstrations significantly improve policy performance, while noisy or suboptimal data can degrade performance [5]. In the original Diffusion Policy work, human operators were required to practice tasks extensively before recording demonstrations. Our work builds on this by quantifying the impact of bad data on policy performance, and evaluating how unpracticed demonstrations compare with refined expert demonstrations.

This study contributes to the growing body of research in robotic policy learning by benchmarking Diffusion Policy against Model-Based Methods and investigating how data quality affects its effectiveness. Our results will provide insights into the trade-offs between data-driven and model-driven approaches in robotic manipulation.

## III. PRELIMINARIES

### A. Basic Framework of Diffusion Models

Diffusion model is a special case of Variational Autoencoder (VAE), whose core idea stems from thermodynamics: a distribution can be transformed into another through the continuous addition of noise. In the context of image generation tasks, this implies that images from the training set can be progressively perturbed by noise until they conform to a standard normal distribution. Specifically, it contains two process: forward process and backward process:

*1) Forward Process (Diffusion):* The forward process is a fixed Markov chain that incrementally adds Gaussian noise to the input data $\mathbf{x}_0$ over $T$ timesteps. At each step $t$, the noised sample $\mathbf{x}_t$ is generated as:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \tag{1}$$

where $\alpha_t$ is a noise schedule parameter, and $\varepsilon_t$ is isotropic Gaussian noise. The process transforms the data distribution $q(\mathbf{x}_0)$ into a tractable prior $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{0},\mathbf{I})$.

*2) Reverse Process (Denoising):* The reverse process learns a parameterized model $p_\theta$ to iteratively denoise samples. Starting from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, it approximates the true posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ by:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t,t), \Sigma_\theta(\mathbf{x}_t,t)) \tag{2}$$

Here, $\mu_\theta$ and $\Sigma_\theta$ are neural networks trained to predict the noise component and covariance at each step. The training objective minimizes the variational bound on the negative log-likelihood:

$$\mathscr{L} = \mathbb{E}_{t,\mathbf{x}_0,\varepsilon}\left[\|\varepsilon - \varepsilon_\theta(\mathbf{x}_t,t)\|^2\right] \tag{3}$$

where $\varepsilon_\theta$ is the prediction of the noise injected at step $t$.

In robot control scenarios, this process is redefined as a conditional generation problem: given an observation sequence $O_t$, an action sequence $A_t$ is generated through K denoising iterations, mathematically expressed as:

$$A_t^{k-1} = \frac{1}{\alpha_k}(A_t^k - \alpha_k \varepsilon_\theta(A_t^k, O_t, k)) + \sigma_k z \tag{4}$$

where $\varepsilon_\theta$ is the noise prediction network, $\alpha_k$ controls the denoising step size, and $\sigma_k$ determines the intensity of random perturbations. This iterative optimization mechanism enables the policy to explore multimodal distributions in the action space while maintaining action coherence across the time dimension.

## IV. METHODOLOGY

### A. Block Pushing Task Description

The block pushing task involves controlling a 7-DoF Franka Emika Panda robotic arm to push a cubic object to a designated target position on a planar workspace. The cube has uniform friction properties between its surfaces and the workspace, but initial object poses and target locations vary across trials. The robot observes the environment through an overhead RGB-D camera, which provides object pose estimates for policy execution. The task requires the robot to plan pushing trajectories that account for contact dynamics, including potential object sliding and rotation during manipulation.

### B. Vanilla Model-Based Planner with MoveIt!

In our study, we implement a Vanilla Model-Based Planner using MoveIt as a baseline for comparison with our diffusion-based approach. This planner follows a detect-plan-push strategy, where the robotic arm first detects the wooden block's initial position using traditional perception methods, such as vision-based object detection or predefined markers. The target position on the plane is predefined, and once both the start and goal locations are identified, the system plans a feasible pushing trajectory using MoveIt's motion planning framework. The planned motion is executed by the robotic arm, aiming to push the block toward its goal in a controlled manner.

The motion planning process is based on inverse kinematics and sampling-based algorithms, such as RRT-Connect or PRM, to generate a collision-free trajectory for the end-effector. The robot moves to a designated starting position, aligns with the block, and applies a forward motion to initiate the push. However, since the planner does not explicitly model frictional interactions or dynamic effects, there is a possibility that the push may not succeed in moving the block precisely to the target. If the block does not reach the desired position due to slippage, misalignment, or execution inaccuracies, the system detects the failure and resets. The robotic arm returns to its home position, re-evaluates the block's new pose, and re-plans the pushing trajectory. This process is repeated iteratively until the block is successfully moved to the goal.

While this model-based approach provides a structured and interpretable solution for block pushing, it faces several limitations. The primary drawback is its limited adaptability to unmodeled contact dynamics, as the system does not

account for variations in friction, surface properties, or unexpected disturbances.

### C. Diffusion Policy Training

Diffusion Policy is trained as a Denoising Diffusion Probabilistic Model (DDPM), where the objective is to learn a mapping from noisy actions to clean actions conditioned on visual and state observations. This formulation allows the model to express multi-modal action distributions and leverage the generative modeling capabilities of diffusion processes for policy learning.

Diffusion models define a forward process where noise is gradually added to the actions over $K$ timesteps, transforming them into a Gaussian distribution. The reverse process, parameterized by a neural network, learns to iteratively denoise the corrupted actions and recover the original action distribution:

$$q(A_t^k | A_t^{k-1}) = \mathcal{N}(A_t^k; \sqrt{\alpha_k} A_t^{k-1}, (1-\alpha_k) I) \tag{5}$$

where $\alpha_k$ is a noise schedule controlling the level of corruption. The policy is trained to approximate the denoising function by minimizing the expected squared error between the predicted noise and the true noise:

$$L_{\text{diff}} = \mathbb{E}_{A_t, O_t, k} \left[ |\varepsilon_k - \varepsilon\theta(O_t, A_t + \varepsilon_k, k)|^2 \right] \tag{6}$$

*1) Observation Representation:* The observation space consists of both visual and proprioceptive inputs. Both visual and state embeddings are concatenated and used as conditioning inputs to the denoising network:

- **Visual Input**: RGB-D images from an Intel RealSense D415 camera, processed using a ResNet-34 backbone with frozen parameters. The extracted features are further encoded with temporal convolution layers to capture spatiotemporal dependencies.
- **State Input**: The robot's proprioceptive state $s_t$, including joint positions, velocities, and end-effector forces. This is encoded into a 128-dimensional latent representation using a multi-layer perceptron (MLP) and further refined using an LSTM for sequential modeling.

*2) Action Representation:* The action space for the block pushing task is formulated as a continuous control problem, where each action consists of:

$$A_t = (x_t, y_t, z_t) \tag{7}$$

where:

- $x_t, y_t, z_t$: Cartesian position of the end-effector,

Actions are iteratively refined through the diffusion denoising process, ensuring smooth and feasible motion trajectories for the block-pushing task.

## V. Experiment

### A. Simulation Experiment

We develop Mujoco [6] simulation environment to validate our training pipeline before real-world training and deployment. We follow the open-sourced code [4] and reproduce the *Block Pushing* task with diffusion policy trained on a Linux machine of Ubuntu 22.04 with Nvidia 4090 GPU. We validate both state-based and vision-based diffusion policy.

### B. Real-world Experiment

*1) Hardware Setup:* In our real-world experiments, as shown in Fig. 1, we use two Intel RealSense cameras to provide RGB-D inputs for the learning-based policy. One camera is mounted at the front of the workspace, while the other is attached to the Franka robot arm's end-effector as a wrist camera. The Franka arm is teleoperated using end-effector pose control to perform block-pushing motions and collect demonstration data for training the Diffusion Policy. During data collection, we iteratively adjust the pushing angle of the end-effector until the block is successfully placed within a black-taped target area. This setup enables a thorough evaluation of policy performance in both simulated and real-world environments.

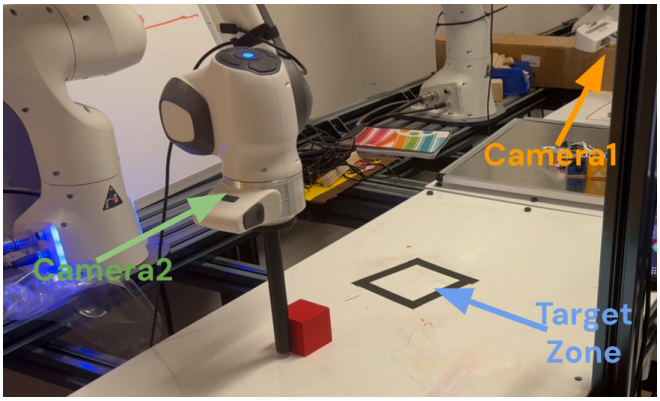



Fig. 1: Franka Panda robot arm with a stick-mounted end-effector for interaction with a wooden block and a front-facing RealSense camera for video recording. The black-taped area indicates the goal position for the block.

n

*2) Data Collection:* To collect demonstration data for training the Diffusion Policy, we teleoperate the Franka robot arm using Cartesian end-effector pose control. Each trajectory consists of time-stamped RGB-D observations from both the front-facing and wrist-mounted RealSense cameras, along with the robot's proprioceptive states and end-effector poses. The pushing motion is repeated under various initial block configurations and goal locations to introduce variability and improve policy generalization.

### C. Model-Based Approach Experiment

To benchmark our diffusion policy against a conventional planner, we evaluated the vanilla model-based method on the real robot.

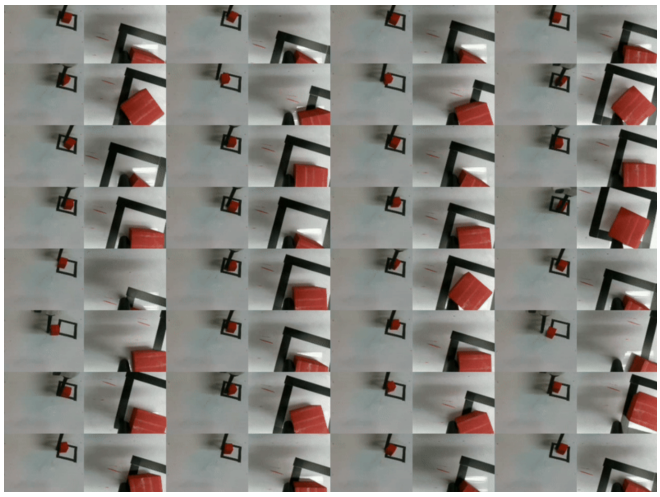


Fig. 2: Samples from the demonstration dataset collected for training the Diffusion Policy. Each frame pair shows synchronized RGB observations from the front-facing and wrist-mounted RealSense cameras during a pushing task.

Similar to the hardware setup mentioned above, we used Franka Emika Panda arm and a front-facing Intel RealSense D415 RGB-D camera, but removed the camera mounted on the end-effector. Workspace is a 60 cm×60 cm flat table and the object is a uniform wooden cube (5 cm edge length).

The procedure is as follows, we:

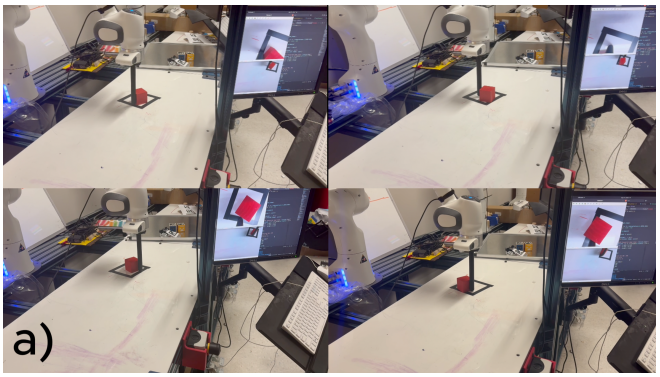
1) Captured the block's pose via opencv.
2) Compute a straight-line end-effector path: approach the block's center, then push at constant velocity for a distance to make the block move towards the goal position.
3) Repeat this procedure, stop until the block enter the goal position.
4) Marked the trial as a failure if the cube did not reach the goal after three pushes.

## VI. Evaluation

We evaluate the performance of the trained Diffusion Policy in a real-world block pushing task using the Franka Emika Panda robot. Our primary goal is to assess whether the policy can generalize beyond the limited training set and workspace configuration, and whether it can reliably push a wooden block into a designated target zone under variable conditions.

Each method was evaluated in a real-world block pushing task using the same tabletop setup. A total of 20 trials were conducted per method, with randomized initial block positions and a fixed target region marked on the workspace. The camera setup remained constant across all experiments. For fairness, all episodes started with the block placed within the same workspace bounds, and the robot was reset to a consistent initial configuration before each trial.

As shown in I, human demonstrations achieved perfect performance with a 100% success rate and an average episode duration of 15.6 seconds, serving as the performance




| Metric | Human | Target-Aligned Push | Diffusion Policy |
|---|---|---|---|
| IoU | 1.00 | 0.5 | 0.5 |
| Success Rate (%) | 1.00 | 0.15 | 0.25 |
| Duration (s) | 15.6 | 22.4 | 68.3 |

TABLE I: **Evaluation Summary.** a) Example of successful block pushing executions using the trained Diffusion Policy. Table: IoU measures the overlap between the final block pose and the target region. Success is defined by an IoU greater than 0.5. Duration refers to the average episode length in seconds.

upper bound. The model-based method, which executes a straight-line push aligned with the target, succeeded in 15% of trials. In comparison, the Diffusion Policy demonstrated a higher success rate of 25%, indicating some ability to adapt to variability in block position. However, its average episode duration was 68.3 seconds—over three times longer than human execution—suggesting that the generated trajectories were less efficient and potentially unstable.

While the Diffusion Policy showed a higher success rate than the model-based approach, it often exhibited slow, unstable, or curved motion paths. In successful cases, the policy was able to adapt to minor variations in block pose, but many failures involved imprecise pushes that rotated or nudged the block away from the goal area. These behaviors are likely attributed to the limited training data—only 40 demonstrations—and the small, constrained workspace imposed by Franka's safety limits. The policy appeared to lack confident strategies for edge cases and often produced hesitant movements, possibly due to poor generalization outside the observed training distribution.

## VII. Challenges

Despite the theoretical strengths of Diffusion Policy, our practical deployment revealed several critical challenges. We hypothesize that these limitations primarily stem from two key factors: the limited diversity in our training data (only 40 trajectories) and the constrained workspace imposed by the Franka robot's safety boundaries. Below, we outline the major challenges encountered

### A. No Obvious Feedback or Replanning Mechanism

The Diffusion Policy executes a precomputed sequence of actions based solely on the initial observation, without incorporating any real-time corrective feedback. In our constrained setup, where small pose shifts or execution errors frequently occurred, this open-loop behavior led to poor adaptability. Once the pushing motion diverged from expectation, the policy was unable to recover, often resulting in missed or ineffective pushes.

### B. Lack of Robustness to Pose Variations

The small dataset and limited workspace variation made it difficult for the policy to learn invariances to object pose changes. As a result, small perturbations in the block's initial position or extreme positions near the edge of the workspace often caused the policy to fail. This suggests that more diverse demonstrations and randomized initial conditions are necessary for robust generalization.

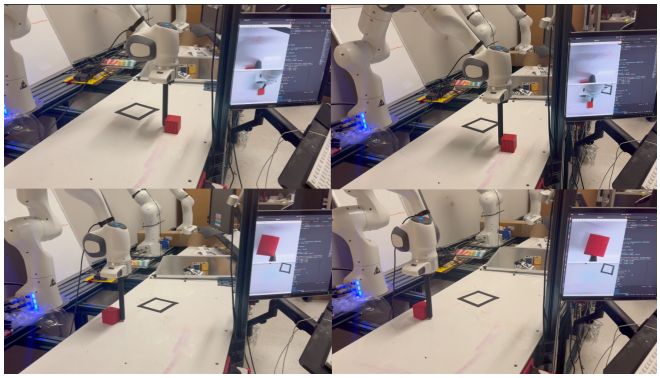


Fig. 3: **Failure Case of Diffusion Policy.** The robot fails to push the block into the target zone due to misalignment in the initial pose and limited adaptability of the trained policy.

### C. Limited Generalization Across Environments

The policy struggled when deployed under slightly different lighting conditions or camera viewpoints compared to those seen during training. This is likely due to the policy overfitting to a narrow training distribution, which was collected under tightly controlled environmental conditions. Since both cameras and lighting remained fixed during collection, the model lacked the exposure needed to generalize across different visual appearances of the same task.

## VIII. CONCLUSION

This paper presented a comparative study between a vision-based Diffusion Policy and a vanilla model-based planner for the robotic block pushing task. We implemented a simple model-based approach using MoveIt, which served as both a baseline and a data source for training the Diffusion Policy via demonstrations collected with human intervention. Our real-world experiments using a Franka Emika Panda arm demonstrated that the Diffusion Policy achieved a higher success rate and generalized better than the model-based baseline, particularly in handling variations in initial block pose.

Despite its advantages in generalization, the deployed Diffusion Policy faced significant challenges. Its performance, while superior to the basic model-based approach, was notably less efficient and stable than human teleoperated demonstrations, often resulting in slow or hesitant motions. Key limitations identified include the policy's open-loop nature, lacking real-time feedback or replanning mechanisms to correct deviations during execution. Furthermore, its robustness was constrained by the limited size and diversity of the training dataset (40 trajectories) and the restricted workspace, leading to poor generalization to pose variations, particularly near workspace boundaries or under different environmental conditions.

In summary, while Diffusion Policy shows promise for learning complex manipulation skills like block pushing directly from demonstrations and handling uncertainty better than simple model-based methods, significant hurdles remain in achieving robust and efficient real-world performance. Addressing the challenges related to data limitations, adaptability, and generalization is crucial for advancing the practical applicability of such learning-based approaches in robotics.

## IX. FUTURE WORK

While our experiments demonstrate that the Diffusion Policy significantly outperforms a vanilla model-based planner in both simulated and real-world block-pushing tasks, its robustness to out-of-distribution conditions and its ability to generalize across novel scenarios remain limited. To address these shortcomings, we propose two complementary directions for future investigation:

First, building on insights from sim-to-real transfer, we plan to gradually introduce increasing levels of variability—both in simulation and on hardware—during data collection and policy training. By randomizing physical parameters (e.g., friction coefficients, block mass, lighting conditions) and visual observations (e.g., camera pose, background textures) in a curriculum fashion, the policy will learn invariances to these perturbations. We will systematically study how the schedule and magnitude of randomization affect robustness, enabling the policy to remain stable when faced with distribution shifts at test time.

Second, to further improve the Diffusion Policy's capacity to handle unseen geometries and push trajectories, we will expand our dataset to cover a much broader range of block positions, orientations, shapes (e.g., varying aspect ratios), and multi-step push sequences. Automated data augmentation techniques—such as procedural generation of target layouts and synthetic variations of recorded demonstrations—will significantly enlarge the support of the training distribution. We will measure how this richer data distribution reduces failure rates on held-out test scenarios and enables zero-shot generalization to entirely new block configurations.

In addition to these data-centric strategies, we will explore algorithmic enhancements—including uncertainty-

aware noise schedules, adversarial perturbations in action space, and hybrid model-based residual corrections—to further stabilize the iterative denoising process and guard against compounding errors. Together, these future efforts aim to deliver a Diffusion Policy that is not only effective in controlled settings but also reliably robust and generalizable in the face of real-world variability.

## REFERENCES


[1] D. Kalashnikov, A. Irpan, P. Pastor, and et al., "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, 2016.

[3] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," 2018.

[4] C. Chi, S. Feng, Y. Du, Z. Xu *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Conference on Robot Learning (CoRL)*, 2023.

[5] S. Belkhale, Y. Cui, and D. Sadigh, "Data quality in imitation learning," 2023.

[6] 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, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, 2012, pp. 5026–5033. [Online]. Available: https://doi.org/10.1109/IROS.2012.6386109