Diffusion Policy: Visuomotor Policy Learning via Action Diffusion
Article Study

Lounès Meddahi
ENS Rennes
MVA, ENS Paris-Saclay
lounes.meddahi@ens-paris-saclay.fr

Tom Ravaud
ENPC
MVA, ENS Paris-Saclay
tom.ravaud@enpc.fr

Abstract

Diffusion policy is a visuomotor policy that leverages the latest advances in diffusion models to perform complex manipulation tasks from human demonstrations. More specifically, it involves denoising a sequence of actions to be performed, conditioned on the visual observation of the scene. The use of diffusion provides a real advantage to the approach, enabling it to surpass previous methods. In this study, we aim to understand what motivated the development of this approach, reproduce the reported results, and explore its limitations.

1. Introduction

Robotics has traditionally been state-based, meaning we attempt to estimate the state of a dynamic system from our sensor measurements and construct the action to perform based on this state. Today, mainly due to technical advances in deep learning, it has become common to bypass this intermediate representation and directly construct the action from the observation, using sensor-based approaches. A visuomotor policy refers to a policy that generates actions based on visual observations of the scene. An early example of such a policy can be obtained by image-based visual servoing, where the goal is to move the robot so that its observation matches a target observation (for example, by overlaying local correspondences between the two images) [2]. However, we are quickly limited to simple positioning tasks. Imitation learning is a popular framework for designing such a policy in the context of complex robotic manipulation, as it enables learning from human demonstrations in a supervised way [4, 11].

Multiple challenges arise in this setting:

- **Action multimodality**. How to handle situations where different actions are equivalent?
- **Temporal consistency** of the generated actions. How to ensure a smooth trajectory?
- **High-dimensional action space**. How to scale up?

Several approaches have addressed the problem of action multimodality. The idea is always the same: to sample an action from the (potentially multimodal) distribution of possible actions at the current moment to perform the task. This distribution can be explicitly predicted by discretizing the action space [8] or using a Gaussian mixture [6]. Today, generative models allow for better capturing of multimodality by implicitly learning the underlying distribution. In the literature, approaches include energy-based models [4], conditional variational autoencoders (VAEs) [11], or diffusion models [3]. Regarding temporal consistency, receding-horizon-like strategies have proven to be effective but require the model to be able to predict a sequence of actions, which can be of high dimension.

DiffusionPolicy (DP) [3] is a candidate solution that addresses theses challenges: It leverages conditioned diffusion models to sample from an unknown complex and multimodal distribution of action sequences, and a receding-horizon control strategy.

2. Method

There are different diffusion frameworks, including DDPM [5] and DDIM [9], which involve a denoising network ε_θ whose architecture is generally independent of the diffusion parameters. We will focus more on the network architecture than on the diffusion process itself, as no contribution is made to the latter. Note that the authors of the paper used the DDIM framework so as to be able to use a different number of denoising iterations in training and inference, which allows for a faster sampling process at test time.

At timestep t, let A_t be the action sequence and O_t the observation. The observation may contain features from several previous images of the scene, extracted by a trainable ResNet encoder, and the end-effector pose. This observation vector is seen as a conditioning input to guide the action sequence generation. Once the action sequence is

generated, the agent executes only a subset of it, in a receding horizon fashion. Two architectures are proposed for the denoising network (Figure 1). The first model is a 1D U-Net composed of strided and transposed convolutions, with conditioning incorporated through FiLM layers. The second model includes attention layers and uses cross-attention for conditioning. The integration of the network within the diffusion process is shown in Figure 2.

Method	max / avg
LSTM-GMM	0.69/0.54
IBC	0.75/0.64
DP-CNN	**0.91/0.84**
DP-Transformer	0.78/0.66

Table 1. Success rate in the PushT environment. The goal of the agent is to push a T-shaped block to a fixed target. The CNN version of DP outperforms its competitors.

The approach is evaluated on the PushT environment (Table 1). DP has proven to be capable of modeling multimodal distributions (Figure 3).

3. Experiments

In this section, we present our test-time experimental setup used to evaluate the DP framework under varying conditions and introduce some points of discussion we found relevant and decided to explore further.

In Section 3.1, we begin by describing the pre-trained LeRobot model we use for the Push-T environment. We then detail how we managed to train our own models from scratch in Section 3.2. In Section 3.3, we briefly discuss the trade-off between the number of denoising steps and the model's performance. Next, in Section 3.4, we investigate the agent's behavior in multimodal scenarios, where multiple valid action trajectories are possible (e.g. bypass the T by the left or right). Finally, in Section 3.5, we detail the evaluation protocols and experimental setup used to introduce and measure the impact of visual perturbations on the policy's performance.

3.1. LeRobot's Pre-Trained Action Diffusion Model

While the authors of the original DP article have made their code available, a recent and more accessible implementation has been provided by Hugging Face through their LeRobot framework [1].

Our first experiments were based on a pre-trained diffusion policy model provided by the LeRobot team and applied to the Push-T environment [1]. The goal of this task is to push a T-shaped block to a fixed target location in a 2D space. There are two ways of assessing the agent's performance: the success rate and the average maximum overlap between the object and the target. To be considered successful, the object must overlap the target by at least 95%.

While the original DP authors proposed two implementations, a Transformer-based and a CNN-based architecture, only the latter is available in the LeRobot library. Visual observations are processed using a ResNet-18 encoder trained from scratch (i.e., without ImageNet pretraining), transforming raw RGB inputs into compact feature vectors. These observation features condition the diffusion process that sequentially predicts actions, which are end-effector positions. This diffusion-based policy has been trained from 206 human demonstrations [2], for 7 hours on an Nvidia RTX 3090. More precisely, it consists of 200k steps of training with a batch size of 32, using various data augmentation techniques (image brightness, contrast, saturation, hue and sharpness).

Figure 4 illustrates three possible outcomes with different object-to-target coverage levels. The evaluation results[3] of the LeRobot's implementation compared to a CNN-based architecture similar to the original DP article are summarized in Table 2. We observe a gap of nearly 20% in success rate compared to the results presented in the paper. We are not able to explain this difference, as the original training used less training samples and the exact same architecture.

Metric	HF	DP
Average max. overlap (%)	95.9	95.7
Success rate (%)	63.8	64.2

Table 2. Evaluation results of the pre-trained diffusion policy model ('DP') compared to the LeRobot implementation ('HF'). Metrics are computed over 500 evaluation episodes.

3.2. Training our own models

For a more in-depth understanding of the DP framework, we wanted to train a diffusion policy model from scratch. However, due to the high computational cost of training such models, we were unable to complete this task with our limited resources. We decided to reduce the number of training steps and benefit from Kaggle's free GPU resources (30h hours per week per user on a Nvidia Tesla P100 GPU) to train a model on the Push-T environment for few hours. We observed that at least 5k steps are required to obtain a policy not too fuzzy. In this case, the success rate is not an appropriate metric anymore, as it is hard for the policy to reach a

[1] Available at https://github.com/huggingface/gym-pusht

[2] Collected via https://github.com/real-stanford/diffusion_policy

[3] Available at https://huggingface.co/lerobot/diffusion_pusht

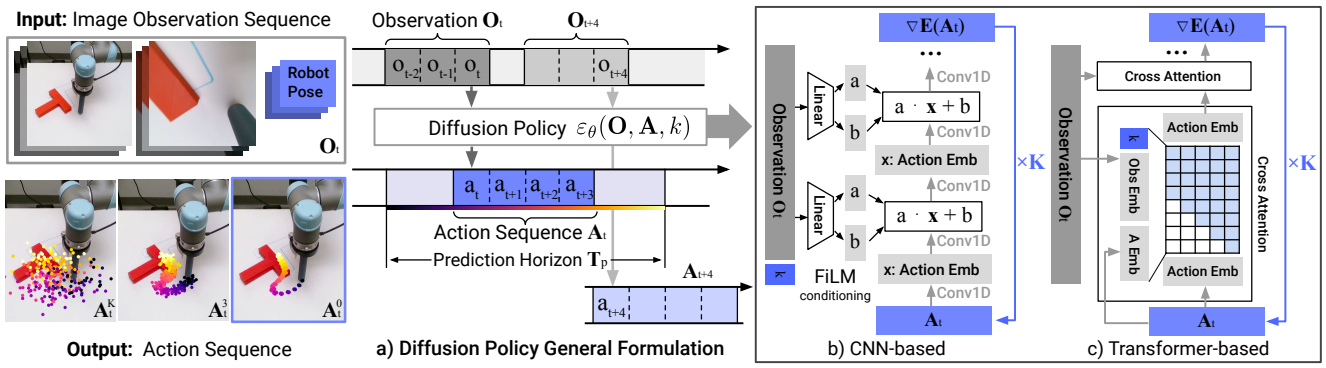


Figure 1. a) Denoising network overview, b) CNN-based and c) Transformer-based architectures.

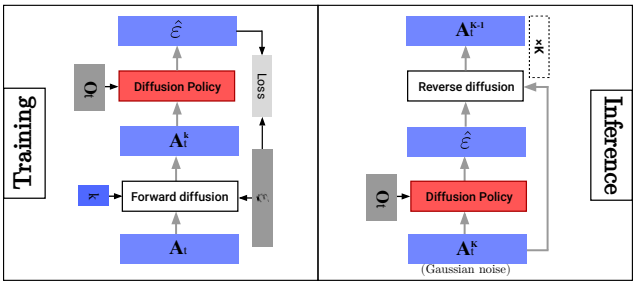


Figure 2. Diffusion involves training the network to denoise noisy action sequences, in order to be able to sample sequences from pure noise during inference.

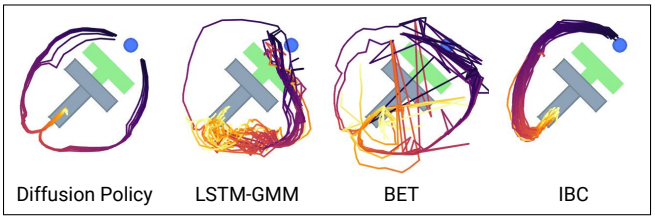
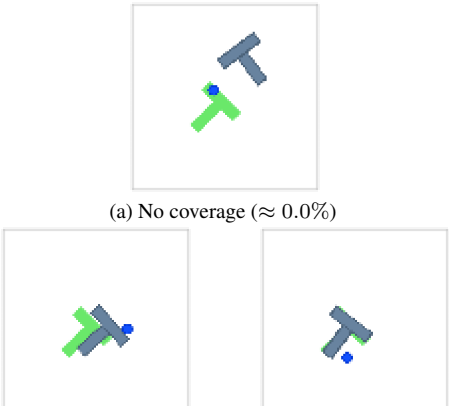


Figure 3. Ability of different approaches to handle multimodality. The agent (blue) must push the object (grey) to the target (green). In this specific case, the agent should be able to bypass the T-shaped obstacle by the left or right.

95% overlap. We then decided to use the average maximum overlap as the main evaluation metric.

3.3. Number of denoising steps: a first trade-off

When dealing with diffusion models, the number of denoising steps is a crucial hyperparameter that can impact the model's performance. Ideally, one would set this parameter as high as possible to ensure the best possible denoising. However, this comes at the cost of increased computational complexity, which can be prohibitive at test time for real-time applications. As mentioned previously, the DDIM framework allows us to use a much lower number of denoising steps at inference than during training, but we may be able to do better. We conducted a simple qualitative experiment by increasing the number of denoising steps at inference time and analysed the influence on a batch of generated sequences. With a single denoising step, the generated sequences are still very noisy (Figure 5b). By increasing the number of denoising steps, the sequences become more coherent and gather around a common trajectory. In order to quantify this observation, we computed the standard deviation of the batch of predicted actions at each time step for different numbers of denoising steps (Figure 5a). From 2 steps, the standard deviation is already significantly reduced, but we chose to use at least 10 steps during our experiments for a less noisy policy. Whatsmore, note that it is quite rare to have an important spread of the predicted actions at certain time steps, which suggests that the policy is quite confident in its predictions. We would expect a large standard deviation when multiple actions are possible to reach the goal (multimodality) or when we are out of the



(a) No coverage (≈ 0.0%)

(b) Low coverage (≈ 16%) (c) High coverage (≈ 89%)

Figure 4. Illustrations of different object-to-target coverage levels obtained by the LeRobot pre-trained diffusion policy on the Push-T task.

training distribution.

3.4. Focus on action multimodality

Action multimodality is a common issue in visuomotor policy learning. It refers to the presence of multiple valid action trajectories that can lead to the same goal. In a scenario involving multimodality, we want the agent to execute one of the possible actions, but ideally, we would like it to know the different options and choose the best one.

We replicated such a scenario in the Push-T environment by aligning the agent with the object and the target (Figure 6). We predicted multiple action sequences from different initial noisy sequences and progressively shifted the agent's position to the right, parallel to the T bar. We observe that DP is able to deal with multimodality-related decision ambiguities but is suboptimal: the agent favors the left side to bypass the T, while the right side is also a valid option. We need to shift the agent sufficiently to the right to make it choose the right side. As the human providing the demonstrations is not an expert, suboptimality is an inherent issue. The demonstrations could be biased and/or not provide enough data to cover all possible scenarios. Another explanation, that we believe is less likely, is that the model is not expressive enough to capture the complexity of the underlying multimodal distribution it tries to sample from.

3.5. Visual perturbations on the input images

To evaluate the visual robustness of the pre-trained diffusion policy, we apply controlled perturbations to the input image observations at test time. These perturbations simulate realistic challenges such as camera noise, or environmental distractions (e.g. waving hand in front of the camera). Beyond robustness to visual perturbations, which largely depends on the data and augmentations used during training, we wanted to more broadly observe the behavior of DP when it encounters situations outside the training distribution. Detecting such situations and adapting the method is a key challenge for real-world applications. Illustrated in Figure 7, the following scenarios are considered:

(1) No Perturbation: Figure 7a shows this baseline condition, where the input image stream remains unaltered. This scenario provides a reference point for the model's expected performance under ideal visual conditions and helps us quantify any degradation resulting from additional disturbances.

(2) Gaussian Noise: To simulate sensor-level noise and low-quality vision input, we corrupt the input images by adding Gaussian noise with zero mean and a standard deviation of 0.1. Assuming the input pixels are normalized to the [0, 1] range, this corresponds to a 10% intensity-level noise degradation. Figure 7b illustrates this condition.

(3) Black Patch Occlusion: Inspired by the original DP article [3], we insert a fixed-size black patch, 10×10 pixels, over random regions of the input image as shown by Figure 7c. This occlusion simulates dynamic occlusions that might occur, for example, when a human hand passes in front of the camera. Such partial visibility tests the model's ability to infer the scene's underlying spatial structure and maintain temporall consistency even when visual information is partially obstructed.

By comparing policy performance across these conditions, we aim to draw meaningful conclusions about the LeRobot provided action diffusion model's visual robustness and more generally, the model's ability to generalize to unseen scenarios.

4. DP capabilities under visual perturbations

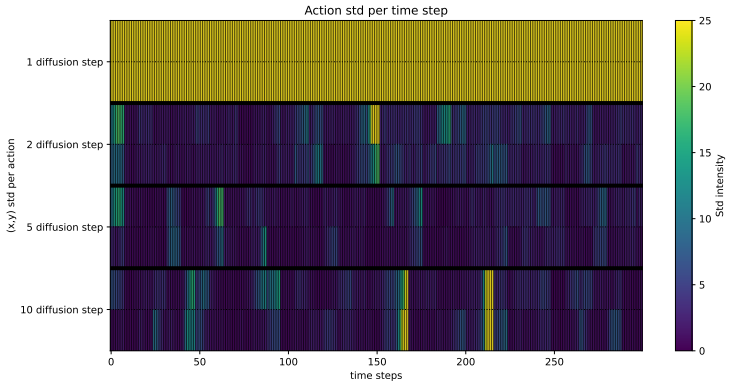
We first evaluate in Section 4.1 the performance of the LeRobot DP under the visual perturbations presented in Section 3. Then, in Section 4.2, we present and evaluate different strategies aimed at addressing the model's lack of robustness to occlusions.

4.1. Impact of visual perturbations on DP capabilities

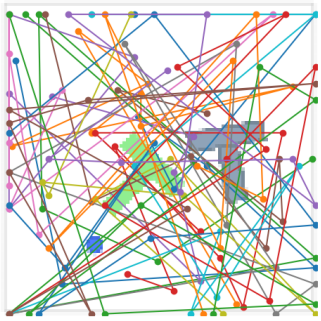
To evaluate the capabilities of LeRobot's DP under different visual perturbations, we conducted experiments on 20 random seeds of the PushT environment. The success rate is measured across varying action horizons under the input conditions described in Section 3. The results are summarized in Table 3.

Effect of Action Horizon Table 3 results indicate that increasing the action horizon generally enhances the policy's success rate under clear input conditions. Specifically, the success rate improves from 20% at an action horizon of 1 to 70% at a horizon of 16. This trend suggests that the diffusion policy benefits from longer prediction horizons, possibly due to better planning and anticipation of future states. Furthemore, we find similar results as the ones promised by LeRobot in Table 2. However, it is important to keep in mind that in practice, within potentially dynamic environments, a horizon that is too large results in a loss of reactivity for the agent. A trade-off must be found to ensure the best performance based on the frequency of changes in the environment.

Robustness to Gaussian Noise Under Gaussian noise conditions, the policy maintains a performance level comparable to, and in some cases slightly exceeding, that with clear inputs. Notably, at an action horizon of 8, the success rate peaks at 70%, matching the best performance un-

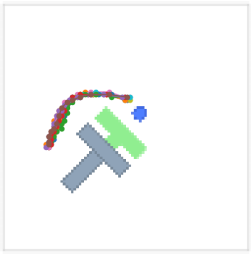


(a) Standard deviation of the predicted actions (thresholded at 25 pixels).

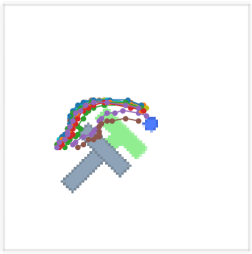


(b) Generated action sequences with a single denoising step in a 96×96 action space.

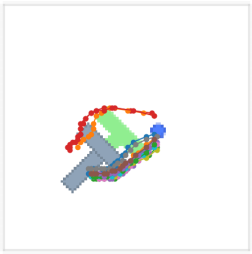
Figure 5. Influence of the number of denoising steps on the generated action sequences.



(a) Centered



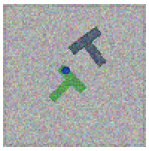
(b) Shifted to the right



(c) Shifted further to the right

Figure 6. Generated actions while shifting the agent's position parallel to the T bar.

(a) No perturbation



(b) Gaussian noise

(c) Black patch

Figure 7. Example input images for the three considered visual conditions.

der clear conditions. This resilience indicates that the LeRobot diffusion policy training effectively handles sensor-level noise, likely owing to its data augmentation as presented in Section 3.1. The fact that the optimal sequence length is shorter aligns with our observation on reactivity: the agent quickly replans its trajectories and performs well on average.

Sensitivity to Occlusions In contrast, the policy's performance drops to 0% success across all action horizons when faced with black patch occlusions. The random placement of the occluding patch may block critical visual information necessary for task completion. This sharp decline highlights the policy's vulnerability to occlusions and suggests that it heavily relies on uncorrupted visual inputs to make accurate predictions. Since the success rate is no longer a suitable metric for evaluating the agent's performance, we relied on the mean maximum coverage. Although the coverage is very minimal, we observed that the best results were achieved with a horizon of length 1: the agent performs better by averaging multiple predictions rather than following a single uncertain generation. Furthermore, we observe degraded performance even when the black patch does not affect the visual information used, i.e., the agent, the object, and the target. This case illustrates poor handling of situations outside the training distribution: the agent's movements are fuzzy, or even almost random.

Table 3. Success rates for different action horizons (steps) under various input conditions.

Action Horizon	Clear	Gaussian	Patch
1	0.2	0.2	0.0
2	0.3	0.45	0.0
5	0.35	0.4	0.0
8	0.65	**0.7**	0.0
16	**0.7**	0.5	0.0

The experiments reveal that while the diffusion policy is robust to certain types of visual perturbations like Gaussian

noise, it is significantly affected by occlusions, a case that is not present during training time. The inability to handle occluded inputs indicates a limitation in the model's capacity to infer missing information or focus on unoccluded regions. These findings underscore the need for strategies to enhance the policy's robustness to occlusions, such as incorporating data augmentation with occlusions during training, or leveraging additional sensory modalities.

In the following section, we explore potential methods to overcome the challenges posed by occlusions and improve the diffusion policy's performance in the presence of such visual disturbances.

4.2. Overcoming the Patch Problem

In this section, we present our strategies to mitigate the performance degradation caused by visual occlusions, focusing specifically on addressing the challenges posed by the black patch.

The performance drop observed in Table 3 when black patch occlusions are introduced highlights the vulnerability of the DP model to visual perturbations. To tackle this issue, we propose two solutions.

The first solution involves augmenting the training dataset with examples containing occlusions, thereby improving the policy's robustness to such challenges. By exposing the model to occlusions during training, it can learn to generalize better to scenarios with similar disruptions.

The other solutions explore working at test time only, using simple heuristics. This allows us to study whether satisfactory occlusion robustness can still be achieved with simple approaches.

Instead of evaluating performance based on the success rate as seen in Table 3, Table 4 focuses on the average max reward obtained during the rollout of the episodes. The average max reward is almost equivalent to the average max coverage. The reward is defined as:

$$\text{reward} = \text{clip}\left(\text{coverage}/\text{max coverage}, 0, 1\right)$$

where $\text{clip}(x, a, b)$ is a function that clips x to be between a and b, and max coverage is set to 0.95. This shift in evaluation metric is motivated by the high cost of training a diffusion policy from scratch for 200,000 steps, which is necessary to achieve a good success rate. However, the average reward serves as a reliable performance indicator even with fewer training steps (as few as 5,000). This makes it a practical alternative for studying the impact of our proposed strategies while minimizing training costs.

(a) Adding Random Erasing (RE) Data Augmentation The first approach introduces random erasing data augmentation [12] into the training process. This technique involves randomly erasing parts of the input image during training, effectively simulating occlusions and forcing the model to become more robust to such disruptions. An example of this augmentation is illustrated in Figure 8. The model was trained using this augmentation for 10,000 and 20,000 training steps, corresponding to approximately 2.45 and 4.9 hours of training time, respectively, on a Kaggle P100 GPU. As shown in Table 4 by the RE results, the model begins to recover good average rewards from 10,000 training steps. This demonstrates the effectiveness of random erasing in improving the model's performance while maintaining a reasonable computational cost.

Figure 8. Image with random erasing data augmentation.

(b) Combining Gaussian Noise (GN) with Occlusions This approach, illustrated by Figure 9, involves degrading the overall image quality by adding Gaussian noise (mean 0, standard deviation 0.1) to it, alongside the black patch occlusion. The motivation of this experiment is to desensitize the neural network to the black patch by reducing its relative impact on the input image. As simple visual data augmentation are already applied at training time, the noisy image forces the model to rely less on specific visual details within the input. As shown in Table 4 by the GN average max reward, this strategy results in no improvement in the model's ability to recover rewards compared to using the black patch alone.

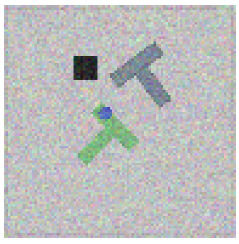


Figure 9. Image with Gaussian noise and a 10x10 occlusion patch.

(c) Dynamic Adjustment (DA) of Action Sequence Length This last approach targets the model's action generation process. By dynamically reducing the action se-

quence length during task execution, we hypothesize that the model becomes more responsive to changes in the visual input, allowing it to adapt better to challenging scenarios such as mooving black patches. We evaluate this strategy using both linear and exponential decay schedules for action sequence length, as illustrated in Figure 10. The results presented as DA in Table 4 show that reducing the action sequence length doesn't improve the policy's performance over time.

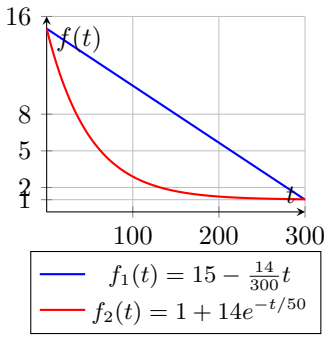


Figure 10. Adjustement functions used for the dynamic adjustment of action sequence length across time steps.

	Avg max reward
LeRobot	0.19
GN	0.23
DA (linear)	0.15
DA (exponential)	0.27
RE (10k)	0.52
RE (20k)	**0.59**

Table 4. Average max reward of the proposed stragies to overcome the black patch problem. "LeRobot" result is the average max reward of the Hugging Face LeRobot model.

Table 4 results demonstrate the effectiveness of data augmentation in significantly improving the model's robustness. Indeed, we obtain average max reward values similar to those achieved when training the model for 5k steps without augmentations and evaluating it on unperturbed images (around 0.6). Additionally, the analysis reveals that even with fewer training steps, the model retains an acceptable performance level, offering a computationally efficient alternative.

5. Recovering the multimodality

In Section 3.4, we highlighted the difficulty of DP in modeling the multimodal distribution of actions. More specifically, we concluded that the method's suboptimality was due to non-expert and potentially biased human demonstrations. To address this, as we do not have access to expert demonstrations, we decided to generate new trajectories to create a larger and more diverse dataset. We first practiced on the PushT task before creating 50 new demonstrations each to reduce biases. Once the new dataset was complete, we trained the model on these over 300 trajectories for 20k steps.

The effect of these new trajectories was immediate. In Figure 11, we reproduced the previous singular case and generated 16 different trajectories from different noise vectors. This time, we clearly see both trajectories emerging on either side of the T. To conclude on multimodality, DP is indeed capable of handling situations where multiple actions are possible, unlike methods that rely on an L2 loss. However, the key to effectively modeling a multimodal distribution lies in the training data. When a simulator is available, one option could be to generate new data from existing data by applying geometric transformations or leveraging an expert policy trained through RL.

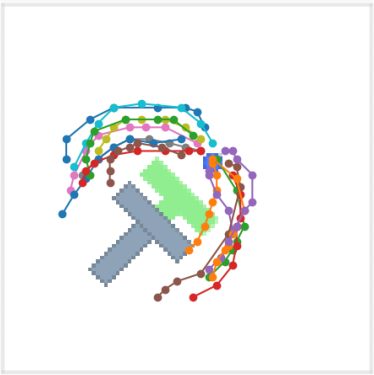


Figure 11. Illustration of the multimodal distribution of actions.

6. Conclusion

This work explores the capabilities and limitations of diffusion policies for visuomotor tasks, specifically within the PushT environment, leveraging the LeRobot framework. Our experiments confirm the potential of diffusion-based policies for sampling trajectories from a high-dimensional and multimodal distribution while ensuring temporal consistency. However, our analysis reveals significant vulnerabilities under out of distribution inputs, such as visual occlusion, and limitations in showing action multimodality behaviors.

Our results highlight that adding data augmentation leads to noticeable performance improvements, even with 10 times fewer training steps, by evaluating the policy using an alternative metric, (the coverage rate instead of the success rate). More generally, as it is challenging to cover all scenarios encountered in reality, it may be beneficial to detect when the agent faces a situation outside the train-

ing distribution. This could involve a module on top of the policy that identifies the agent's fuzzy movements, often linked to uncertainty in generation. In this context, we were interested in estimating prediction uncertainty from a batch of generated trajectories. Our intuition was that in a novel situation, DP would be unable to accurately generate a trajectory, and the uncertainty would be reflected in the spread of the batch predictions. Furthermore, to distinguish between multimodality and uncertainty, we considered extracting trends from batch predictions, for instance, through clustering. Unfortunately, we observed excessive confidence in trajectory predictions, even in novel situations.

Regarding action multimodality, we highlighted DP's ability to handle ambiguous situations as well as the sensitivity of training to the quantity and diversity of data in modeling the underlying multimodal distribution. More broadly, this issue is linked to the suboptimality of learning from human demonstrations. Several techniques could help address this problem, including data augmentation, generating expert trajectories in simulation, or fine-tuning the policy through RL [7, 10].

References

[1] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024. 2

[2] François Chaumette and S. Hutchinson. Visual servo control, Part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82, 2006. 1

[3] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023. 1, 4

[4] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit Behavioral Cloning, 2021. arXiv:2109.00137. 1

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, 2020. arXiv:2006.11239. 1

[6] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation, 2021. arXiv:2108.03298. 1

[7] Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion Policy Policy Optimization, 2024. arXiv:2409.00588. 8

[8] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior Transformers: Cloning k modes with one stone, 2022. arXiv:2206.11251. 1

[9] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models, 2022. arXiv:2010.02502. 1

[10] Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding Reinforcement Learning-Based Fine-Tuning of Diffusion Models: A Tutorial and Review, 2024. arXiv:2407.13734. 8

[11] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware, 2023. arXiv:2304.13705. 1

[12] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13001–13008, 2020. 6