

Demystifying Diffusion Policies: Action Memorization and Simple Lookup Table Alternatives

Project Page: <https://stanfordmsl.github.io/alt/>

Code: <https://github.com/StanfordMSL/alt>

Chengyang He^{1,2,*} Xu Liu^{1,*} Gadiel Sznaiier Camps¹ Joseph Bruno^{1,3}
Guillaume Sartoretti² Mac Schwager¹

¹Stanford University ²National University of Singapore

³Temple University *Equal Contribution

hecy@stanford.edu liuxujsw@stanford.edu gsznaiier@stanford.edu
brunoj6@temple.edu schwager@stanford.edu guillaume.sartoretti@nus.edu.sg

Abstract: Diffusion policies have demonstrated remarkable dexterity and robustness in intricate, high-dimensional robot manipulation tasks, while training from a small number of demonstrations. However, the reason for this performance remains a mystery. In this paper, we offer a surprising hypothesis: diffusion policies essentially memorize an action lookup table—and *this is beneficial*. We posit that, at runtime, diffusion policies find the closest training image to the test image in a latent space and recall the associated training action sequence, offering reactivity without the need for action generalization. This is effective in the sparse data regime, where there is not enough data density for the model to learn action generalization. We support this claim with systematic empirical evidence. Even when conditioned on wildly out of distribution (OOD) images, the Diffusion Policy still outputs an action sequence from the training data. With this insight, we propose a simple policy, the Action Lookup Table (ALT), as a lightweight alternative to the Diffusion Policy. Our ALT policy uses a contrastive image encoder as a hash function to index the closest corresponding training action sequence, explicitly performing the computation that the Diffusion Policy implicitly learns. We show empirically that for relatively small datasets, ALT matches the performance of a diffusion model, while requiring only 0.34% of the inference time and 0.85% of the memory footprint, allowing for much faster closed-loop inference with resource constrained robots. We also train our ALT policy to give an explicit OOD flag when the distance between the runtime image is too far in the latent space from the training images, giving a simple but effective runtime monitor.

Keywords: Robot Manipulation, Diffusion Policy, Action Memorization

1 Introduction

Imitation learning for robot manipulation requires training a policy to map from image inputs to action sequence outputs given a relatively small number of demonstrations. Recently, the Diffusion Policy [1] has emerged as a powerful and novel approach to this problem by modeling the robot’s visuomotor policy as a denoising diffusion probabilistic model [2]. Diffusion models are generative models that train a filter to iteratively remove noise from noise-corrupted training data. At inference time a random output is sampled, and progressively denoised (conditioned on the input) to produce an inference. The diffusion model was originally introduced for image generation and remains the dominant architecture in that domain [3, 4]. The primary advantage of the Diffusion Policy for robot manipulation lies in its ability to model multi-modal action distributions, scale to high-dimensional

outputs, and produce long-horizon action sequences. Indeed, recent studies have shown that diffusion policies outperform many existing methods on challenging manipulation benchmarks [5, 6].

The performance of the Diffusion Policy is unquestionable, however the explanation for this performance remains elusive. In particular, typical diffusion policies are trained on 50–200 task demonstrations (a small amount of data), while maintaining the same number of parameters (typically over 100 million) as image generation models trained on billions of images [3, 4]. Furthermore, the common practice is to train Diffusion Policies until the training loss is low, but the test loss is high—the classic signal for over-fitting in machine learning. In typical machine learning, overfitting is associated with poor test-time performance and poor generalization. Yet it is observed that this overfitting is actually necessary for strong test time performance of the Diffusion Policy. The question arises:

Why do diffusion policies trained to overfit small data sets appear to give strong test-time performance in robot manipulation?

In this paper we show that, indeed, diffusion policies severely overfit the training data, such that they memorize the training action sequences. At inference, diffusion policies recall the training action sequences with nearly no generalization. They essentially perform a lookup table that maps runtime images to training action sequences. Combined with online closed-loop execution with runtime images, this action memorization appears to be a winning recipe for strong manipulation policies obtained from small amounts of demonstration data. The stochastic multi-modality of the policy results from slight randomness in which training action sequence is indexed, not from generalization over action sequences. However, one key drawback of the Diffusion Policy is slow inference time, which leads to slow robot execution punctuated by pauses as the model recomputes the next inference at the end of each action sequence. This motivates our second research question:

Can the same action memorization behavior be accomplished with a simpler, faster model architecture to yield faster runtime performance?

To answer this question, we propose a simple lookup table policy with a trained image encoder to map from images to actions, which we call the Action Lookup Table (ALT) policy, illustrated in Fig. 1. The ALT policy performs similarly to the Diffusion Policy, while being 300 times faster at inference, and requiring less than 1/100th the memory footprint.

Concretely, we design a lightweight image-joint pose encoder that maps each provided observation into a low-dimensional feature representation (LFR), which we then use to create a memory bank of LFRs from the training expert demonstrations. During inference, we then encode new observations, find the nearest code from the training set in the latent space, and output the memorized training action sequence associated with that code. The encoder is trained using a contrastive learning objective [7], encouraging positive sample pairs to be embedded closer together while pushing negative pairs apart in the latent space. ALT avoids the costly iterative denoising steps, enabling faster (single forward-pass) inference.

Our **contributions** are as follows: (1) We hypothesize the Diffusion Policy implicitly memorizes an action lookup table. We provide conceptual intuition for this hypothesis and support it through extensive empirical validation. (2) We present a new ALT policy (as shown in Fig. 1), a contrastive learning-based alternative to diffusion policies, explicitly implementing an action sequence lookup table by encoding image-joint pose inputs into a latent space to index training action sequences. By setting a threshold distance in the latent space, we obtain a simple OOD detector to flag potential policy failures at runtime.

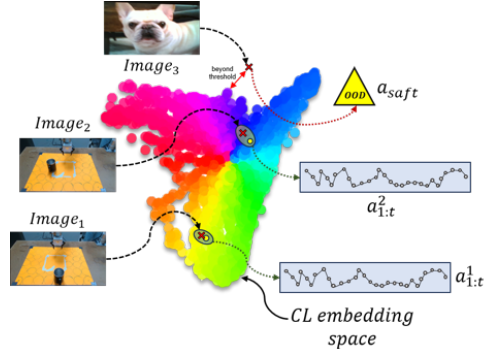


Figure 1: Resulting latent space of our contrastive learning (CL) based ALT on our training data, illustrating the distribution of training examples and example in- (1,2) and out-of-distribution (3) test images.

2 Prior Work

Diffusion models, trained by gradually adding Gaussian noise to data during training [2, 8, 9], were originally developed for high-dimensional data generation tasks such as image, video, or audio synthesis [10, 11]. These models can produce seemingly novel high-quality images and videos in a variety of different styles through simple text [8, 12, 13, 3] and image [14, 15, 16] prompt conditioning. In order to capture the complex multimodal distributions inherent in visual and auditory data, these models are often large, containing from hundreds of millions to billions of parameters [3, 4], and are trained over large datasets with hundreds of millions to billions of examples [11].

Leveraging the strong performance of diffusion models, the Diffusion Policy [1] achieves state-of-the-art performance in visuomotor control for single skill imitation learning. Trained with a limited number of expert demonstrations, the model learns to predict a sequence of robot actions [1, 17, 18, 19] conditioned on a given observation. This observation can be images [1], point clouds [5], semantic labels [20, 21] or potential fields [22]. Due to its apparent robustness to perturbations, diffusion policies have been deployed for a wide range of robotics tasks, including manipulation [23, 24, 1], multi-skill learning [25, 26], and motion planning [27, 28, 29]. Diffusion models have also been used in robotics for data augmentation [30, 31] to aid in the training of other models.

The phenomenon of memorization in diffusion models has been well-studied in image generation, but not in robotics, to our knowledge. While unsurprising for smaller datasets [32], as much as 2% reconstructive memorization in diffusion models has also been documented on larger datasets [33, 34]. Although, this can be mitigated with synthetic data augmentation [35], it has been shown that the traditional denoising process biases the model to learn an attraction basin for each training sample [32, 36, 37]. Similarly, both [38] and [39] note that, although less prevalent than conditioned models, memorization still occurs in unconditioned models and [40] finds that memorized data are often associated with individual neurons.

Modifying the loss function [41], gradients [41], conditioning approach [36], or keyword prompts [37, 38, 42] during training and inference are all typical methods for reducing model memorization. Although considered undesirable in image generation, we posit that memorization is actually beneficial for robotics. When diffusion models are used in domains with rich input space (e.g., images) but limited output space (e.g., robot actions), the gap between model capacity and output dimensionality, combined with imitation learning (which lacks task-level supervision), makes memorization via overfitting a plausible explanation for its strong performance in in-distribution settings. Along these lines, previous approaches have explored reusing past demonstrations in conjunction with parametric generalizations. They obtain the next action using frame-level nearest-neighbor action retrieval [43], action prototypes obtained based on a bounded residual [44], or by utilizing a learned context embedding composed of multiple trajectory fragments [45]. In comparison, our proposed ALT method directly obtains the closest action sequence from a lookup table via a contrastive image encoder, bypassing any parametric policy entirely, thereby allowing it to be lightweight and fast unlike more compute-intensive hybrid alternatives.

3 Diffusion Policy Analysis

3.1 Preliminaries

The output, $\mathbf{x}^0$, of a diffusion model, ε_θ , is obtained by iteratively removing noise (i.e. denoising) from a starting value, $\mathbf{x}^k$, sampled from a Normal Distribution, $\mathcal{N}(0, \sigma^2 I)$. The denoising process evolves according to

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

to remove the noise in k steps based on a predetermined noise schedule that specifies the values of α , σ , and γ at each iteration. This procedure can be thought of as a single stochastic gradient descent step $x' = x - \gamma \nabla E(x)$, where the model ε_θ is used to predict the gradient field $\nabla E(x)$. A more detailed explanation of the denoising process can be found in [1] and [2].

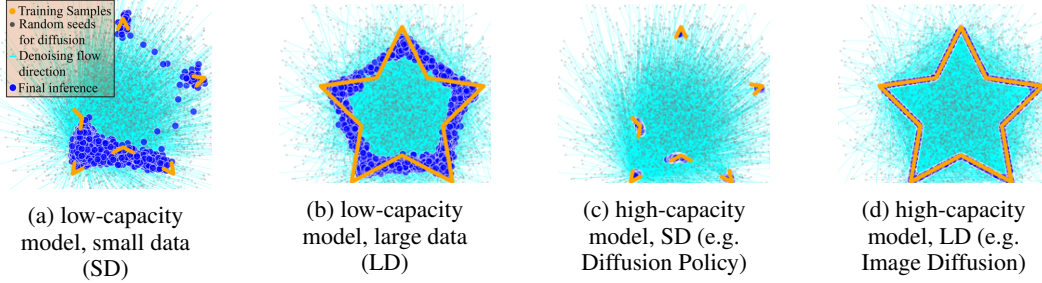


Figure 2: Generative model trained on 2D points uniformly distributed on a star-shaped 1D manifold. Each subplot shows a different training regime: (a) Low-capacity model (~ 400 parameters) trained on a small dataset (3k samples) gives erratic inferences. (b) Low-capacity model trained on a large dataset (100k samples) generalizes to the wrong manifold. (c) High-capacity model (~ 9.5 million parameters) trained on a small dataset (the Diffusion Policy regime) approximately memorizes the dataset, but does not generalize. All the inference (blue) overlay the training data (orange) points, essentially implementing a lookup table. (d) High-capacity model trained on a large dataset shows strong generalization to the correct data manifold (regime of large scale image diffusion models). Similar observations are demonstrated for other 2D manifolds in Appendix C.

3.2 Diffusion Model Generalization Regimes

The degree that a model memorizes or generalizes lies on a spectrum, but can be approximately grouped into four regimes: low-capacity model with little data; low-capacity model with ample data; high-capacity model with little data; and high-capacity model with ample data. We qualitatively demonstrate these regimes with a simple Multi-Layer Perceptron (MLP) diffusion denoising model trained to learn a 2D distribution consisting of points uniformly sampled on a 1D star shaped manifold (Fig. 2).

As expected, when a low-capacity model is trained on a small dataset (Fig. 2a), it fails to fit the data adequately. Similarly, due to its limited capacity, when such a model is given sufficient data (Fig. 2b), it is only able to learn an approximation that oversimplifies the data manifold (here, approximating a star shape as a hexagon). In comparison, when a high-capacity model is trained on a small dataset (Fig. 2c), as proposed by the original Diffusion Policy paper, it tends to memorize the individual training samples rather than generalizing or interpolating between them. This memorization allows accurate fitting of the limited training points (good for robot manipulation tasks), but results in the model failing to capture the broader underlying data manifold, a behavior that is consistent with our findings for diffusion policies for robot manipulation. This phenomenon is related to manifold overfitting [46]: when a powerful generative model is trained on data lying on a narrow sub-manifold, it might fit the data too closely while struggling outside that sub-manifold. When the model is provided sufficient data (Fig. 2d), it is now able to effectively fit both the data and the true underlying distribution, representing the regime common in large scale image generation models.¹ However, acquiring large-scale expert demonstrations for robot manipulation that evenly and densely cover the action sequence space remains a significant practical challenge. As Diffusion Policies are trained on larger and larger datasets, they may move toward the large data regime (Fig. 2d) with true generalization on the action manifold, but this seems to be beyond the current state of the art.

3.3 Hypothesis and Experiments

To evaluate the hypothesis that the Diffusion Policy implicitly implements an action sequence lookup table, we analyzed two different simulated manipulation benchmarks performed by Diffusion Policy (results shown in Appendix D) as well as designed a series of real-world cup grasping experiments. In the real-world experiments, we trained a diffusion policy to grasp cups using the standard codebase from [1], trained with 30 demonstrations of cup locations evenly spaced throughout the workspace, with a held-out square in the middle, as indicated

¹More results can be found in Appendix C

by the green circles and blue tape in Fig. 3. The robot has a third-person view fixed camera and a wrist mounted camera, both used to condition the policy. For each position, we performed one demonstration (to remove the confounding effect of multi-modal action generation). We then validated the learned policy on the 30 in-distribution cases, confirming its ability to reproduce the training demonstrations. To further investigate the action generalization behavior of the policy, we also systematically introduced a variety of interpolation and extrapolation inputs, ranging from in-distribution (InD) to out-of-distribution (OOD) and analyzed the resulting behavior. Specifically, we designed four scenarios: (1:InD-Interpolate) Placing the cup at evenly spaced test positions located between the original training positions (Fig. 3 green border); (2:OOD-Interpolate) Slowly moving the cup from one in-distribution position through an OOD region (blue tape square) to another in-distribution position (Fig. 3 blue border); (3:OOD-Extrapolate) Gradually moving the cup from an in-distribution position to an OOD location outside the fixed camera’s field of view (Fig. 3 purple border). (4:OOD-Distractors) Introducing OOD visual distractors of varying difficulty levels (Fig. 3 yellow border), including differing lightning conditions and wildly OOD images of a cat and a dog; These settings allowed us to explore and analyze the generalization behavior and potential memory-driven characteristics of the Diffusion Policy. If the Diffusion Policy were performing action generalization, one would expect the following in each scenario: (1:InD-Interpolate) interpolation in the action space; (2:OOD-Interpolate) some action interpolation with degraded performance in the middle, where it is far from the training examples; (3:OOD-Extrapolate) progressively degraded action performance as the object moves farther from the training set; and (4:OOD-Distractors) degraded action performance as the number and severity of distractors grows, with dog and cat inducing erratic action sequences.

In fact, all of these behavioral expectations are incorrect. In every case, the Diffusion Policy almost exactly reproduces one of the training action sequences as explained below. This is consistent with our action lookup table hypothesis.

3.4 Results

In this subsection, we introduce a custom metric, the similarity score, designed to quantify how closely an inference action sequence resembles sequences from the training set. It is defined as: $\mathcal{S} = 1 - \frac{s(\tau^{(r)}, \tau^{(1)})}{s(\tau^{(1)}, \tau^{(2)})}$, where $s(\tau^{(r)}, \tau^{(1)})$ denotes the average Euclidean distance between the matched points on the current action sequence and its closest training sequence, and $s(\tau^{(1)}, \tau^{(2)})$ denotes the distance between the second-closest and the closest training action sequence.

If an action sequence closely follows a specific training sequence while maintaining a clear separation from other nearby sequences, this provides strong evidence of memory-based retrieval rather than action generalization. Note that this similarity metric does not measure action *quality*, just action recall. For example, the robot may take a highly ineffective action sequence, but if it closely matches one of the training sequences, the similarity score will be high. We make no claims on the effectiveness of the Diffusion Policy actions. Just that they are recalled from a memorization of the training actions. We first validate our hypothesis in an in-distribution setting, where cups are placed exactly as they were seen during training. In this setting, we observe that in each action sequence almost perfectly overlaps with the corresponding training sequence (see Fig. 11 in Appendix D). This indicates that the model is essentially replaying an action sequence memorized during training when presented with familiar inputs.

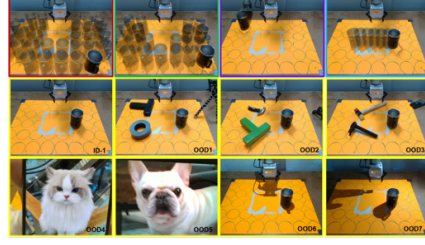


Figure 3: The first panel (red outline) shows in-distribution (InD) tests. The second panel (green outline) shows an InD interpolation test, with cup evenly placed between training positions. Panels with purple and blue outlines illustrate cases where the cup is gradually moved from an in-distribution location to new OOD position. The remaining panels (yellow outline) introduce OOD image distractors to assess the model’s robustness.

We find that this behavior persists even under OOD scenarios (as introduced to the environment in Fig. 3 in the yellow borders). In Fig. 4 (bottom), regardless of whether distractors are present, the distance to the closest training trajectory remains very low, and the distance to the second closest remains high. As a result, the similarity score of the nearest neighbor remains close to 1.0, while the second nearest similarity approaches 0.0. Again, we do not find that the action sequence is the “right one” in the face of distractors, but rather that the model chooses one of the training sequences to re-execute, even when the input image is OOD. Furthermore, Fig. 4 (top) provides a global view of similarity scores across all training trajectories. In the presence of distractors, almost all high-similarity matches are sharply concentrated on a single training trajectory, indicating a surprising OOD default behavior. The diffusion model seems to revert to one or two fallback action sequences when presented with OOD images. Even when the input is entirely unrelated to the task, for example, an image of a cat or a dog, the diffusion model still produces an action sequence that closely resembles one from the training set. We believe those results show that the Diffusion Policy’s decision-making is largely governed by memory retrieval, rather than by generalized reasoning over the action space.

In both clean and distractor-laden scenarios, the model demonstrates consistent action replay behavior, supporting our hypothesis that its decision-making is fundamentally memory-driven. Additional analyses and results also support this observation (for example, in the InD-Interpolate cases, the Diffusion Policy outputs a trajectory that closely matches one of the four corresponding nearest-neighbor training trajectories). See Appendix D for more OOD scenarios.

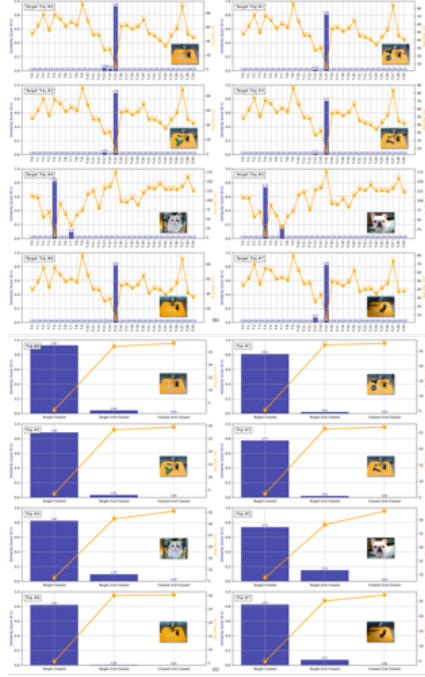


Figure 4: Similarity and distance statistics between inference and training trajectories. Each subplot shows the similarity scores (blue bars) and average distances (orange lines) between the Diffusion Policy inference and training trajectories. The large gap between the closest and second-closest neighbors indicates strong alignment with specific training examples.

4 ALT: the Action Lookup Table Policy

Building upon these results, we design a lightweight alternative method while still achieving comparable functionality to show the bound of memorization mechanism. Our policy, functioning similarly to a lookup table, uses an image encoder trained with contrastive learning as a hash function to retrieve demonstration trajectories (as shown in Fig. 1). If our hypothesis holds, this method should deliver performance on par with the Diffusion Policy, while also offering more predictable fallback behaviors in the presence of out-of-distribution (OOD) inputs, therefore improving safety and robustness.

4.1 Training Phase

At each timestep, our data consists of three parts: a first-person view from the robot arm end-effector, a third-person view, and the end-effector pose denoted as $D = \{(I_i^h, I_i^t, p_i)\}_{i=1}^N$, where I_i^h and I_i^t are the hand camera image and third-view image respectively, and p_i represents the position and orientation of the end-effector. We designed a fusion encoder to integrate these inputs into a unified embedding for contrastive learning. The architecture of this model is illustrated in Fig. 5, and can be formulated as:

$$z_i = f_{fusion}(f_{img}(I_i^h), f_{img}(I_i^t), f_{pose}(p_i)). \quad (2)$$

For visual encoding f_{img} , we employ a pre-trained ResNet-18 [47] as the image encoder backbone. Trained on large-scale datasets such as ImageNet, this pretrained network is capable of extracting general-purpose visual features. Leveraging such pre-trained features is beneficial in our setting, where the available dataset is relatively small.

To perform contrastive learning, we adopt the NT-Xent loss, which requires the fused embeddings to be L2-normalized $z_i \leftarrow \frac{z_i}{\|z_i\|_2}$. This ensures that the similarity computation in the embedding space is stable and consistent.

As shown in Fig. 5, we adopt a contrastive learning framework to extract robust and discriminative representations from each frame using alignment across multiple modalities. We generate two different data augmentation views for each sample $d_i = (I_i^h, I_i^t, p_i)$. Specifically, we apply a composed image augmentation pipeline $\mathcal{A}_1$ and $\mathcal{A}_2$ that transforms each input d_i into augmented version views $v_i^{(1)}$ and $v_i^{(2)}$. We feed $v_i^{(1)}$ and $v_i^{(2)}$ into the fusion encoder to obtain their embeddings $z_i^{(1)}$ and $z_i^{(2)}$. These two embeddings form a positive pair, and we train the network using the normalized temperature-scaled cross-entropy (NT-Xent) loss [7] as our contrastive loss function:

$$\mathcal{L}_c = -\frac{1}{2B} \sum_{i=1}^B \left[\log \frac{\exp(\text{sim}(z_i^{(1)}, z_i^{(2)})/\tau)}{\sum_{k \neq i}^{2B} \exp(\text{sim}(z_i^{(1)}, z_k)/\tau)} + (1 \leftrightarrow 2) \right], \quad (3)$$

where $\text{sim}(\cdot, \cdot)$ denotes the cosine similarity, and both inputs are L2-normalized prior to computation.

The parameter τ is the temperature (set as 0.4 in practice), which controls the sharpness of the similarity distribution, effectively scaling the logits to adjust the contrastive loss sensitivity.

4.2 Inference Phase

After training the model, we build a low-dimensional feature representation latent space to support trajectory matching and prediction during real-time execution. An overview of the full process is shown in Fig. 5. Specifically, we need to encode each frame in the raw trajectory dataset using our fusion encoder, and the resulting embedding tensors are stored as entries in the database. Another important component of the low-dimensional feature representation database is the trajectory ID and local frame index corresponding to each input. During real-time inference, the incoming observation, consisting of the current third-person view, first-person view, and end-effector pose, is encoded into an embedding using the same fusion encoder. We then perform a search for best cosine similarity against the latent contrastive learning space to find the most similar stored embedding. If the maximum similarity falls below a predefined threshold γ , the input is considered OOD, and the robot executes a safe fallback behavior when the input deviates significantly from the training distribution. Otherwise, the system retrieves the matched trajectory ID and frame index, enabling real-time trajectory prediction and policy execution based on the stored demonstrations.

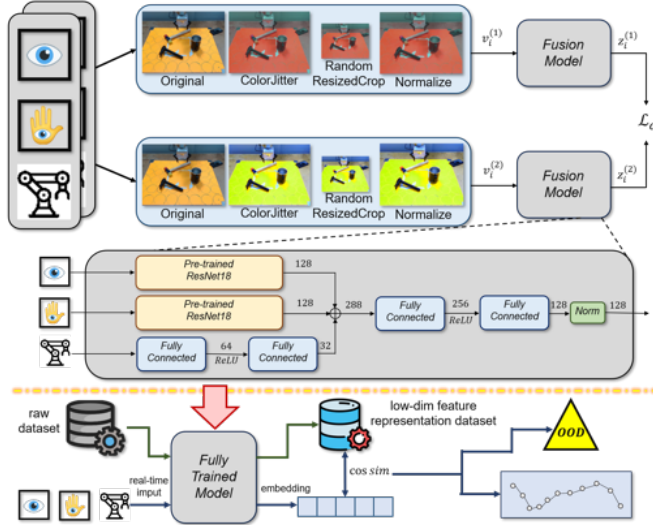


Figure 5: Contrastive training (top, above the yellow dashed line) and inference (bottom) phases of our ALT policy. The inference process has two stages: the green arrows indicate building the ALT latent space with the trained model, while blue arrows represent real-time inference.

Table 1: Experimental Results. ID and OOD refer to in-distribution and out-of-distribution cases, respectively. The first column reports the success rate of the policy in retrieving the correct training trajectory given the observations. The second column shows the success rate of the policy in completing the task during a real-robot rollout. The remaining column headers refer to scenarios shown in Fig. 3. The MIT column denotes the **Model Inference Time** required for each method in seconds.

Methods	Recall	InDs	ID-1	OOD1	OOD2	OOD3	OOD4	OOD5	OOD6	OOD7	MIT
K-D Tree	100%	63.3%	✓	✓	✓	✓	✓	×	✓	✓	~0.09
Diffusion Policy	100%	100%	✓	✓	✓	✓	×	×	✓	✓	~2.65
Ours w/ p, $\gamma = 0.9$	100%	-	✓	✓	OOD	OOD	OOD	OOD	OOD	✓	~0.009
Ours w/o p, $\gamma = 0.9$	100%	-	✓	OOD	OOD	OOD	OOD	OOD	OOD	OOD	~0.009
Ours w/o p, $\gamma = 0.75$	100%	100%	✓	✓	✓	✓	✓	✓	OOD	OOD	~0.009

4.3 Results

In this section, we conduct two experiments to demonstrate the effectiveness of our proposed ALT policy and its action memorization mechanism. First, we test the model to assess its ability to enable successful task execution on a real robot under in-distribution conditions. Next, we assess the model’s performance under OOD conditions by introducing various distractors into the environment. As shown in Fig. 3, we create OOD scenarios by placing additional unseen objects (e.g., tape, hammer), altering lighting conditions to produce varying shadows, or replacing the third-person viewpoint with entirely task-irrelevant images (such as pictures of a cat and a dog), while maintaining the cup in its original training position. We compared our method with both a KD-Tree-based nearest neighbor retrieval and the Diffusion Policy to validate the feasibility of our explicit action memorization mechanism and demonstrate the advantages of our ALT policy. The results are summarized in Table 1, where a checkmark (✓) indicates a successful match with the correct training action sequence, while a cross (×) indicates an incorrect match. Because our method explicitly detects OOD cases, a green OOD denotes cases where the input is correctly identified as OOD yet the trajectory match remains accurate, demonstrating robustness. The red OOD indicates that an OOD input has been detected, and using such an input would result in an incorrect trajectory output.

We observe that in the data-limited training regime, all methods achieve perfect trajectory matching performance (100%) in all in-distribution cases, demonstrating the effectiveness of the underlying memorization mechanism. In OOD scenarios, the KD-Tree method is able to correctly retrieve trajectories in most cases due to its exhaustive nearest neighbor search. However, because it performs pixel-wise comparisons between the input and the raw dataset, its worst-case computational complexity scales as $\mathcal{O}(N \cdot d)$, where N is the number of stored trajectories and d is the data dimensionality. This leads to high computational cost and inference time. In comparison, the Diffusion Policy performs well in most scenarios but fails under more severe distribution shifts, such as in OOD4 and OOD5. Furthermore, due to the architectural complexity of diffusion models, its model inference time (MIT) is relatively long. In contrast, our proposed ALT policy achieves comparable performance with a significantly smaller model size (45.5 MB vs. 5.3 GB for the Diffusion Policy), and a significantly lower inference time. Although our approach has a complexity of $\mathcal{O}(N \cdot d)$, the embedding dimensionality d is much smaller, leading to a drastic reduction in computation time.

5 Conclusion

In this paper, we propose and validate a counter-intuitive hypothesis explaining the impressive performance of diffusion policies. Rather than generalizing actions, diffusion policies essentially memorize training actions through severe overfitting, effectively acting as implicit action lookup tables. Our systematic experiments reveal that, even when confronted with unseen InD or completely OOD inputs, diffusion policies reliably reproduce memorized action sequences rather than generalizing beyond their training demonstrations. This trend was consistent across simpler MLP architectures and the more complex image encoder + U-Net backbone used in Diffusion Policy. Guided by this insight, we introduce a lightweight alternative, the ALT policy, explicitly utilizing a contrastive image encoder as a hash function to retrieve stored trajectories and action indices from memory. When using a relatively small dataset, our ALT policy achieves performance comparable to the the Diffusion Policy while requiring only a fraction (0.34%) of its inference time and substantially reduced memory usage (0.85%), making it particularly suitable for deployment on resource-constrained robots.

6 Limitations

We acknowledge several limitations in our hypothesis and method. All current experiments and validations have been conducted on small-scale dataset, and it remains uncertain whether the performance can be maintained at the same level when applied to a larger dataset. For example, while KD-Tree exhibits strong matching performance in small datasets, its computational efficiency may degrade significantly as the number of entries increases. Similarly, although our ALT method shows clear advantages in terms of computational complexity for small and moderately sized datasets, its scalability is also dependent on the scale of training data, unlike the Diffusion Policy (whose model size is fixed). Therefore, it is likely that our approach will not scale well to large datasets. This behavior still needs to be fully explored in our future work.

Lastly, another limitation of our approach is its sensitivity to hyperparameters, such as the number of training epochs, the temperature parameter in the contrastive loss, and the OOD detection threshold. We observed that careful tuning was required to achieve optimal performance.

Acknowledgments

This project was supported in part by ONR grant N00014-23-1-2354 and the ARPA-H HEART project. This research was also partly supported by the Singapore Ministry of Education (MOE), as well as by NUS Overseas Research Immersion Award. We would also like to thank the reviewers and workshop organizers for their thoughtful comments.

References

- [1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.
- [2] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [3] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35: 36479–36494, 2022.
- [4] Wikipedia contributors. Stable diffusion. https://en.wikipedia.org/wiki/Stable_Diffusion, 2024. Accessed: 2025-04-07.
- [5] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.
- [6] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt. Equivariant diffusion policy. *arXiv preprint arXiv:2407.01812*, 2024.
- [7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. Pmlr, 2020.
- [8] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.
- [9] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

- [10] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.
- [11] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [12] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.
- [13] Y. Balaji, S. Nah, X. Huang, A. Vahdat, J. Song, Q. Zhang, K. Kreis, M. Aittala, T. Aila, S. Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.
- [14] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.
- [15] N. Tumanyan, M. Geyer, S. Bagon, and T. Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1921–1930, 2023.
- [16] D. Ceylan, C.-H. P. Huang, and N. J. Mitra. Pix2video: Video editing using image diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23206–23217, 2023.
- [17] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.
- [18] G. Lu, Z. Gao, T. Chen, W. Dai, Z. Wang, W. Ding, and Y. Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.
- [19] S.-W. Lee and Y.-L. Kuo. Diff-dagger: Uncertainty estimation with diffusion policy for robotic manipulation. *arXiv preprint arXiv:2410.14868*, 2024.
- [20] Y. Wang, G. Yin, B. Huang, T. Kelestemur, J. Wang, and Y. Li. Gendp: 3d semantic fields for category-level generalizable diffusion policy. In *8th Annual Conference on Robot Learning*, volume 2, 2024.
- [21] H. Li, Q. Feng, Z. Zheng, J. Feng, and A. Knoll. Language-guided object-centric diffusion policy for collision-aware robotic manipulation. *arXiv preprint arXiv:2407.00451*, 2024.
- [22] K. Mizuta and K. Leung. Cobl-diffusion: Diffusion-based conditional robot planning in dynamic environments using control barrier and lyapunov functions. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13801–13808. IEEE, 2024.
- [23] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.
- [24] G. Kim, T. Kwon, and J. C. Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2426–2435, 2022.
- [25] L. Chen, S. Bahl, and D. Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *Conference on Robot Learning*, pages 2012–2029. PMLR, 2023.

- [26] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song. Xskill: Cross embodiment skill discovery. In *Conference on robot learning*, pages 3536–3555. PMLR, 2023.
- [27] Y. Shaoul, I. Mishani, S. Vats, J. Li, and M. Likhachev. Multi-robot motion planning with diffusion models. *arXiv preprint arXiv:2410.03072*, 2024.
- [28] A. Serifi, R. Grandia, E. Knoop, M. Gross, and M. Bächer. Robot motion diffusion model: Motion generation for robotic characters. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–9, 2024.
- [29] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.
- [30] I. Kapelyukh, V. Vosylius, and E. Johns. Dall-e-bot: Introducing web-scale diffusion models to robotics. *IEEE Robotics and Automation Letters*, 8(7):3956–3963, 2023.
- [31] S.-F. Chen, H.-C. Wang, M.-H. Hsu, C.-M. Lai, and S.-H. Sun. Diffusion model-augmented behavioral cloning. *arXiv preprint arXiv:2302.13335*, 2023.
- [32] X. Gu, C. Du, T. Pang, C. Li, M. Lin, and Y. Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023.
- [33] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6048–6058, 2023.
- [34] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Schwag, F. Tramèr, B. Balle, D. Ippolito, and E. Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270, 2023.
- [35] Z. Xue, S. Deng, Z. Chen, Y. Wang, Z. Yuan, and H. Xu. Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning. *arXiv preprint arXiv:2502.16932*, 2025.
- [36] A. Jain, Y. Kobayashi, T. Shibuya, Y. Takida, N. Memon, J. Togelius, and Y. Mitsufuji. Classifier-free guidance inside the attraction basin may cause memorization. *arXiv preprint arXiv:2411.16738*, 2024.
- [37] Y. Wen, Y. Liu, C. Chen, and L. Lyu. Detecting, explaining, and mitigating memorization in diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [38] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein. Understanding and mitigating copying in diffusion models. *Advances in Neural Information Processing Systems*, 36: 47783–47803, 2023.
- [39] Y. Chen, X. Ma, D. Zou, and Y.-G. Jiang. Extracting training data from unconditional diffusion models. *arXiv preprint arXiv:2406.12752*, 2024.
- [40] D. Hintersdorf, L. Struppek, K. Kersting, A. Dziedzic, and F. Boenisch. Finding nemo: Localizing neurons responsible for memorization in diffusion models. *Advances in Neural Information Processing Systems*, 37:88236–88278, 2024.
- [41] C. Chen, D. Liu, and C. Xu. Towards memorization-free diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8425–8434, 2024.
- [42] J. Ren, Y. Li, S. Zeng, H. Xu, L. Lyu, Y. Xing, and J. Tang. Unveiling and mitigating memorization in text-to-image diffusion models through cross attention. In *European Conference on Computer Vision*, pages 340–356. Springer, 2024.

- [43] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.
- [44] K. Sridhar, S. Dutta, D. Jayaraman, J. Weimer, and I. Lee. Memory-consistent neural networks for imitation learning. *arXiv preprint arXiv:2310.06171*, 2023.
- [45] K. Sridhar, S. Dutta, D. Jayaraman, and I. Lee. Regent: A retrieval-augmented generalist agent that can act in-context in new environments. *arXiv preprint arXiv:2412.04759*, 2024.
- [46] G. Loaiza-Ganem, B. L. Ross, J. C. Cresswell, and A. L. Caterini. Diagnosing and fixing manifold overfitting in deep generative models. *arXiv preprint arXiv:2204.07172*, 2022.
- [47] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [48] 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.
- [49] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024.
- [50] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

A Data Collection Pipeline

A.1 Robot Arm Data Collection

Choosing an efficient, cheap, and safe method for data collection, crucial for robot imitation learning, remains an open problem. One of the most common solutions is to use remote controllers, such as VR, 3D space mouse, or smartphones. However, due to high latency and indirect operation, the data collected in this way is often messy and low-quality, making it difficult to accurately capture human skills. Fully synchronized systems with human operators, such as ALOHA [48] and GELLO [49], can solve this problem by allowing humans to teleoperate the robot in a more intuitive way while tracking the actions of this system in real time. But, these methods require an additional specialized puppeting system, which incurs an additional cost. In comparison, UMI-gripper [50] is a cheap, intuitive, and robot-agnostic solution for data collection. Yet, it cannot be used in our work as it is incompatible with situations where third-person perspectives are necessary, and limits the robot to a single manipulator that requires an expensive hardware interface. Thus, to collect the necessary data, we utilized a motion capture data collection method, MoDA, to capture high-quality action sequences with low latency.

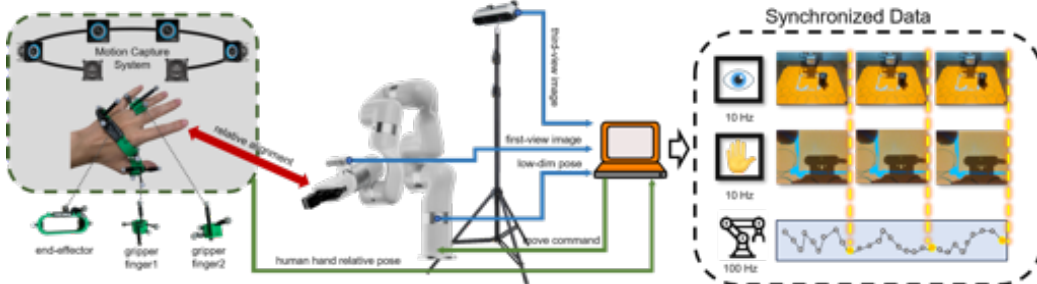


Figure 6: The data collection pipeline of MoDA (Motion-captured Demonstration for Arms). The green arrows indicate the process of aligning the relative positions of human’s hand and the robotic arm, and the blue arrows indicate the data of collecting the robotic arm.

A.2 Motion-captured Demonstration for Arms

Data collection plays a critical role in imitation learning, as the quality and generalizability of the learned policy depends heavily on the fidelity of the demonstrations. In this work, we introduce MoDA (Motion-captured Demonstration for Arms), a streamlined and cost-effective data collection pipeline built upon motion capture (MoCap) systems that are commonly available in robotics laboratories (see Fig. 6). This pipeline provides high-fidelity human demonstrations for the robot, where a human demonstrator performs the cup grasping motion while wearing specialized trackers, and the system translates these motions into corresponding joint targets for a 6-DoF robot arm. MoDA can be extended to any other robot arm system with almost negligible cost, because our data collection pipeline is both task-agnostic and robot-agnostic. To collect the necessary expert training demonstration data, we use an OptiTrack system to track the 6-DoF pose of the human palm in real time and map it directly to the end-effector of a robotic arm. Simultaneously, we estimate the inter-finger distance to control the opening and closing of the gripper, thereby allowing us to signal when to grasp the cup. We then synchronize these actions with the corresponding in-hand and 3rd person camera views. Compared to systems such as ALOHA, which rely on specialized and expensive teleoperation interfaces, our method does not need any active electronics or specialized wearables. Instead, the setup requires only a few 3D-printed brackets to attach passive IR reflective markers to the palm and fingers, making it an extremely low-cost and accessible solution when a MoCap system is already available in the lab. Furthermore, unlike UMI Gripper, which requires direct human interaction during data collection, our setup allows human demonstrators to operate out of frame, thereby ensuring clean third-person video demonstrations. Compared to systems such as ALOHA, which rely on external equipment like teleoperation interfaces or instrumented gloves,

our approach avoids the need for expensive or specialized hardware. In contrast to the UMI Gripper generated data, which often involves complex scenes with human demonstrators visibly present in the frame, our setup enables the collection of clean third-person video demonstrations where human demonstrators are minimally visible. This is particularly beneficial for training diffusion policies, as it minimizes noise and ambiguity in both the action and visual observation spaces, reducing the risk of learning failures due to poor-quality data. In summary, unlike alternative setups that rely on specialized grippers, force sensors, or teleoperation rigs, our system can be assembled in-house with minimal resources and negligible additional expense. Moreover, MoDA is not only task-agnostic, but also robot-agnostic, it does not rely on any specific type or model of robotic arm, making it highly adaptable across different hardware platforms and manipulation scenarios. This flexibility enables seamless integration into a wide range of experimental setups with minimal modification.

B Early Stopping Experiment

To further support our hypothesis, we conducted an early stopping experiment. Early stopping is a common technique used to prevent potential overfitting during training, with the goal of improving a model’s generalization ability. In this experiment, we reserved one-third of the dataset as a validation set and used the remaining two-thirds for training. During training, we recorded both the validation loss and the mean squared error (MSE) between the predicted actions and ground-truth actions on the training set. The first metric, validation loss, is used to determine when to stop training, thus preserving the model version with the best generalization. The second metric, actions MSE on training set, is used to monitor the model’s performance on the training set. As shown in Fig. 7, although the validation loss reaches its minimum at a certain point, the corresponding action MSE remains high, around 1800. This result indicates that overfitting a diffusion policy model to the training data is a necessary requirement for producing accurate trajectories, as choosing the best model (chosen based on the validation loss) results in a policy that cannot reproduce the correct in-distribution trajectories.

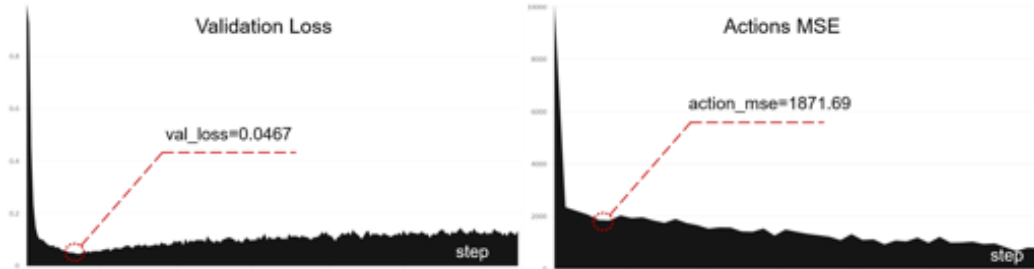


Figure 7: Early Stopping Experiment. The validation loss (left) reaches its minimum around step 650 before beginning to rise, indicating the onset of overfitting. However, at this point, the training action MSE (right) has not yet converged and remains as high as 1800. This suggests that more extensive training is necessary for the Diffusion Policy to output effective actions, even the input is in-distribution.

C Diffusion Mechanism Analysis

In this section, we present additional examples to further illustrate the behavior of diffusion models under varying model capacities and dataset scales, as discussed in Section 3.2. Specifically, we examine three additional 2D manifolds: an ellipse 8, a rectangle 9 and a heart shape 10.

Consistent with the observations from Section 3.2, low-capacity models trained on small datasets fail to accurately reconstruct manifolds, often producing noisy or collapsed outputs. Even when trained on large datasets, these models are limited by their representational capacity: simple shapes like ellipses can be approximated reasonably well (albeit still worse than with high-capacity models), more complex structures suffer significant distortion. For example, due to limited model expressive-

ness, low-capacity models approximate the heart shape as a crude triangle and smooth out the sharp corners of the rectangle.

Memorization behaviors are obvious when high-capacity models are trained on small datasets. Interestingly, although global structure reconstruction fails, local smoothness can still emerge. For instance, while the overall manifold may not be recovered, segments of an ellipse can still be accurately captured. This suggests that memorization in diffusion models is not absolute: when the training data is locally dense, models may still interpolate between nearby points, preserving some local structure. However, when larger gaps exist between segments, interpolation fails, and the model instead memorizes discrete samples without capturing the broader underlying manifold. When high-capacity models are instead trained on large datasets, all manifolds are accurately reconstructed, almost perfectly matching the true geometry. In practice, though, acquiring such large-scale expert demonstrations that sufficiently cover the state space remains a significant challenge in robotic manipulation tasks.

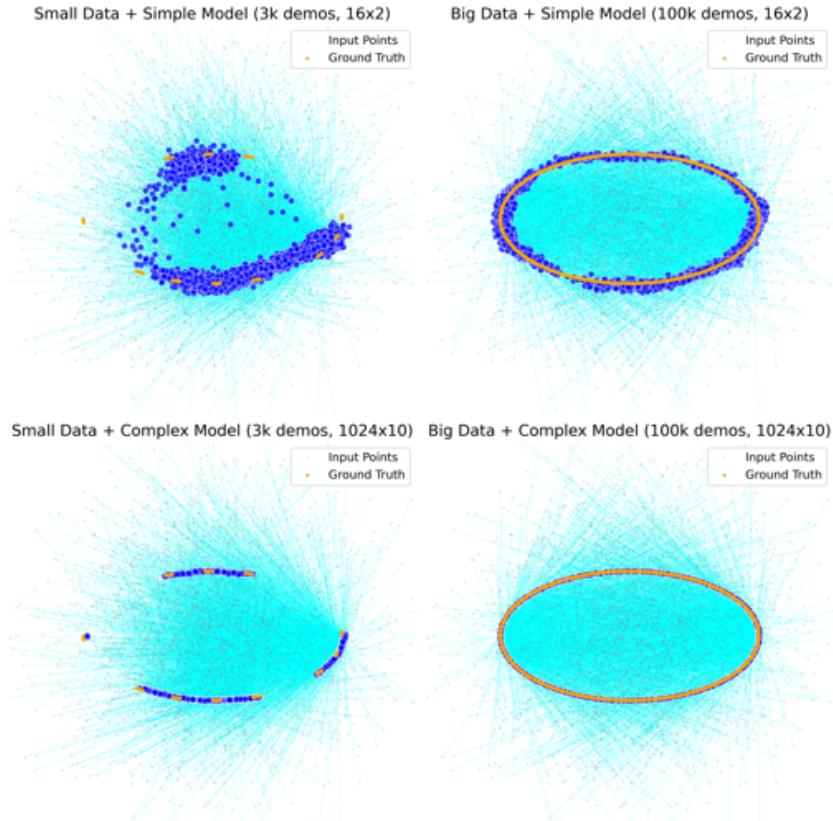


Figure 8: Training a generative model from 2D points on an ellipse-shaped 1D manifold. Orange points indicate training samples, gray points are noisy inputs, blue points are denoised outputs, and cyan lines shows the denoising directions.

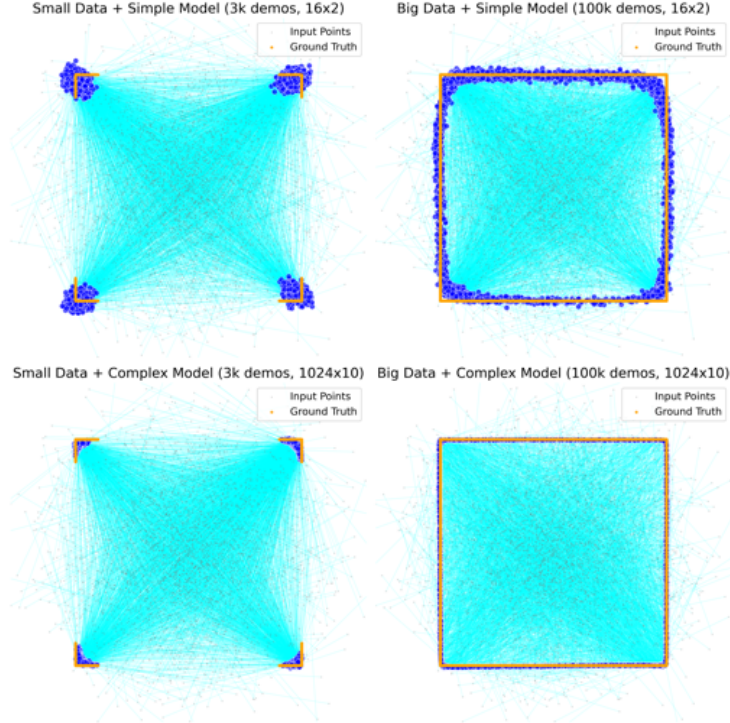


Figure 9: Training a generative model from 2D points on a rectangle-shaped 1D manifold.

D Additional Results and Analysis

D.1 Real-World

Fig. 11 Shows the trajectory similarity between the predicted trajectory and the nearest ground-truth trajectory for the real-world cup placement experiment. The distance to the nearest neighbor (yellow polyline) is near zero (implying high similarity), while the distance to the second nearest trajectory is substantially larger, resulting in a similarity score that is close to 1 (the blue bar).

Fig. 12 presents additional supplementary visualization of the trajectory matching results under in-distribution conditions. In the left panels, for each case, the first blue bar represents the similarity score of the most closely matched training trajectory, while the second blue bar the similarity score of second closest trajectory. The consistently high top-1 similarity scores, combined with significant gaps to the second closest match, indicate clear and confident retrieval from the training data. As shown in the right panels, the closest trajectory (blue) in the training dataset almost perfectly overlaps with the inference trajectory in all examples, showing that the Diffusion Policy can accurately retrieve the correct demonstration when the input remains within the training distribution. These results strongly support our hypothesis that the Diffusion Policy depends on a memory-based retrieval mechanism to achieve its compelling results. The sharp similarity peaks and trajectory overlaps provide strong evidence that the model is not merely approximating the behavior, but is explicitly recalling memorized training trajectories under in-distribution conditions.

We further analyzed several additional out-of-distribution (OOD) scenarios, including: placing the cup evenly between three or four in-distribution positions (as shown in Fig. 13), gradually moving the cup out of the field of view from the edge of an in-distribution position (Fig. 14), and slowly transitioning the cup between two distant in-distribution positions (Fig. 15). It is evident that under these OOD conditions, the Diffusion Policy continues to produce trajectories that closely resemble those seen during training. These experimental results provide further support for our core hypothesis regarding the memory-driven behavior of diffusion policies.

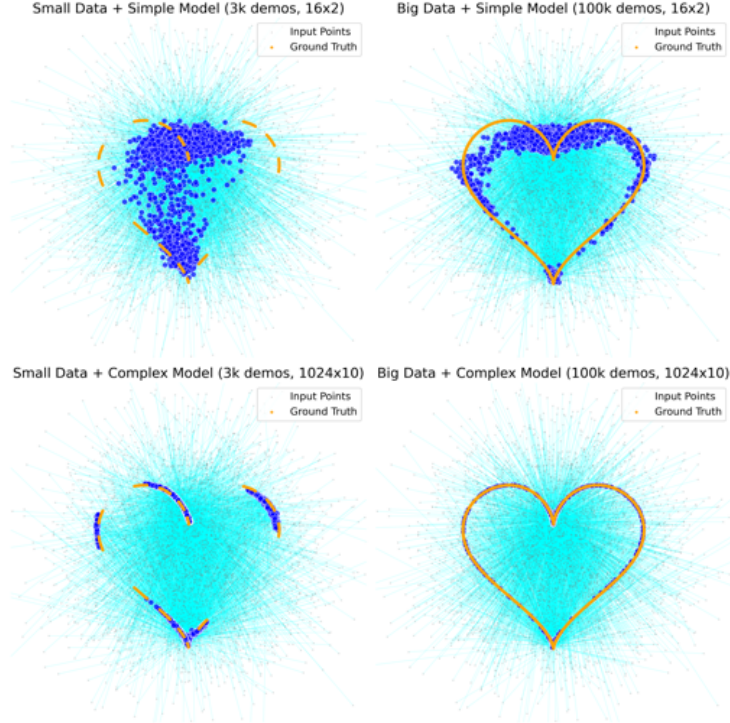


Figure 10: Training a generative model from 2D points on a heart-shaped 1D manifold.



Figure 11: Similarity and distance statistics between inference and training trajectories. Each subplot shows the similarity scores (blue bars) and average distances (orange lines) between the Diffusion Policy inference and training trajectories. The large gap between the closest and second-closest neighbors indicates strong alignment with specific training examples.

D.2 Simulations

We also see similar results for the two different simulated manipulation tasks: Can, and Square. Both tasks are drawn from the Robomimic simulation benchmark and contain 196 training trajectories and four validation trajectories. In the Can task, the model is taught to transfer a can from one table to another table while in the “Square” task, the model is trained to place a square ring on a square peg. We observed that these experiments also display action memorization (see Fig. 16 and Fig. 20), as each training observation was found to closely align with the corresponding ground-truth

demonstration. We observe similar memorization when examining the diffusion policy’s rollouts from observations taken from the validation set (see Figs. 17 through 19, and Figs. 21 through 23) for can and square experiments respectively. Overall, in the Can task, we observed an average similarity score of 0.88 and an average deviation of 2.0 mm. In comparison, in the “Square” task, we observe a similarity score of 0.82 with an average deviation of 4.4 mm.

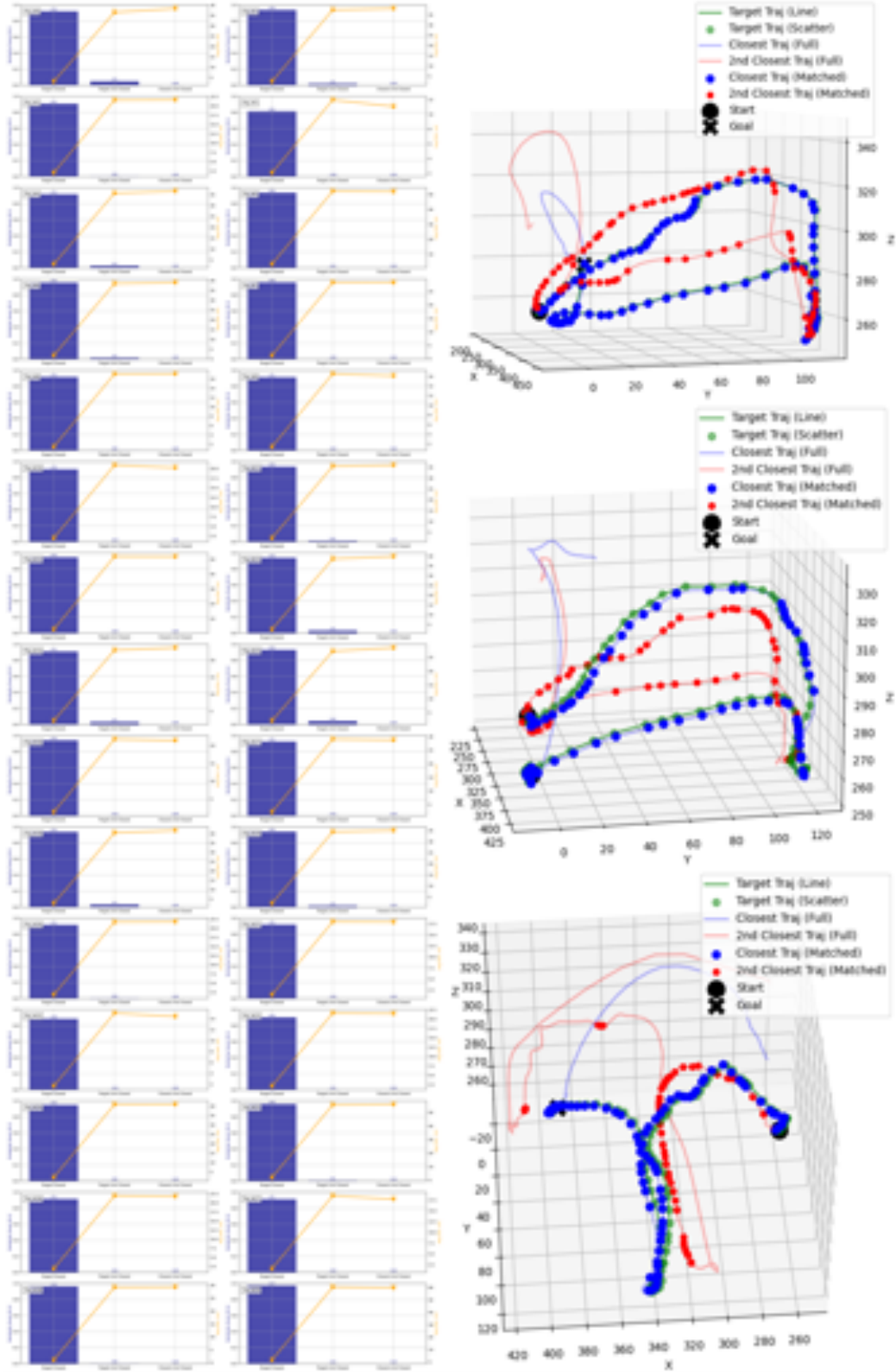


Figure 12: The Diffusion Policy inference result analysis under in-distribution conditions. On the left, each subplot shows the similarity distribution between the query inference trajectory and all stored trajectories in the database. On the right, the three figures provide 3D visualizations of representative matching cases. The green line represents the inference trajectory, while the blue and red dots show the closest and second-closest trajectories retrieved from the training set, respectively.

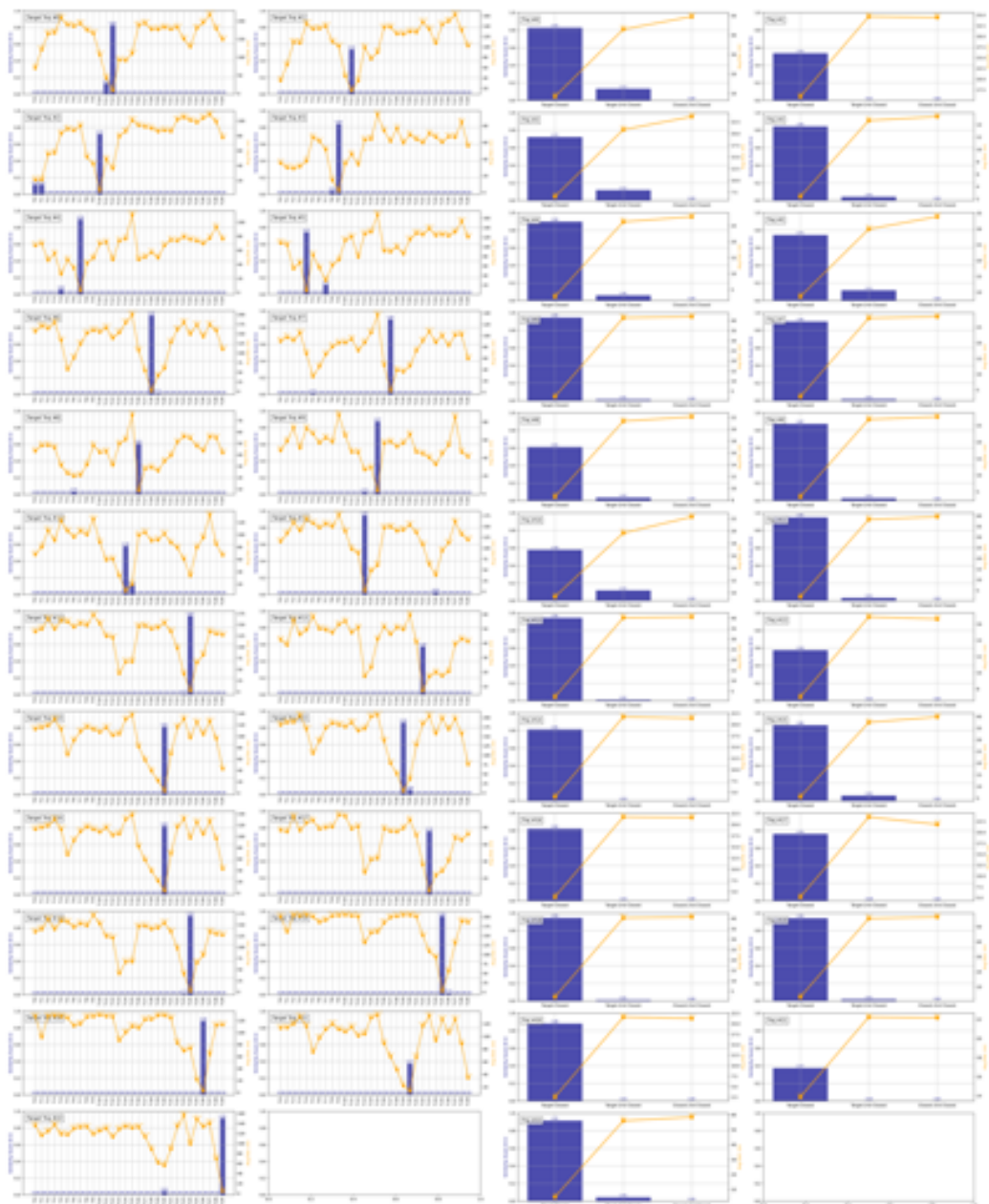


Figure 13: Inference trajectory analysis when the cup is placed between multiple In-distribution positions.

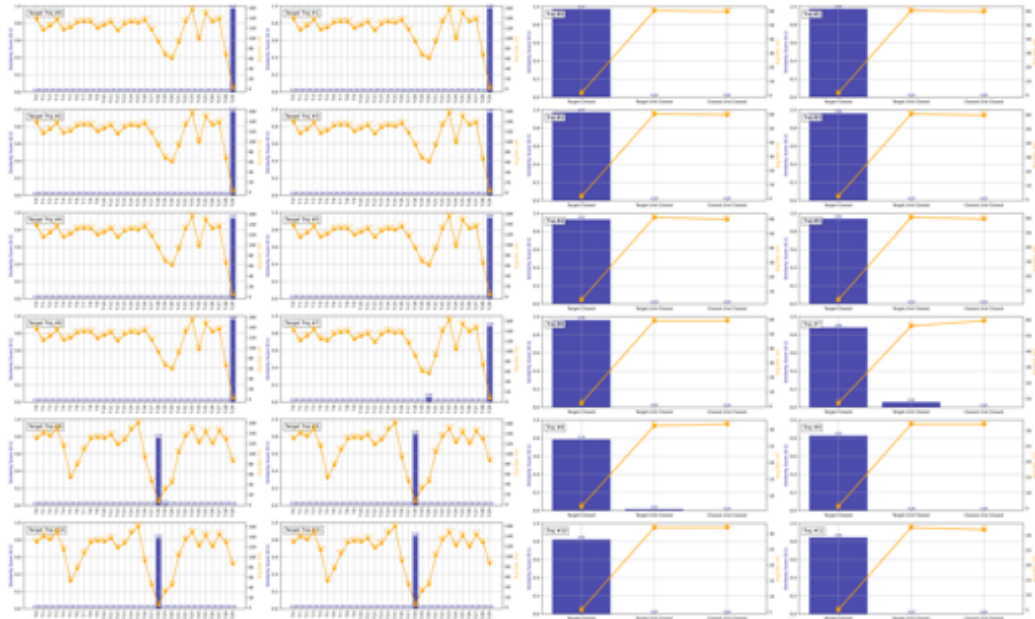


Figure 14: Inference trajectory analysis when the cup is gradually moved out of view from an in-distribution boundary.



Figure 15: Inference trajectory analysis when the cup is gradually moved between two distant in-distribution positions.



Figure 16: A subset of the total analysis done for the roll-outs of a Diffusion Policy model from the training set with respect to all ground-truth training trajectories. The model was trained on the “Can” pickup benchmark.

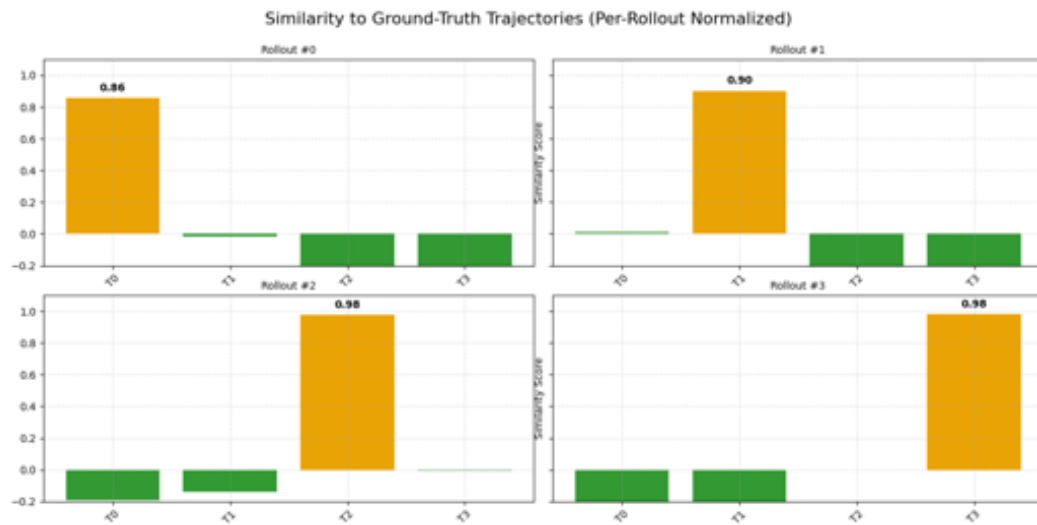


Figure 17: Analysis of a Diffusion Policy model roll-outs on the validation set with respect to all ground-truth validation trajectories. The model was trained on the “Can” pickup benchmark.

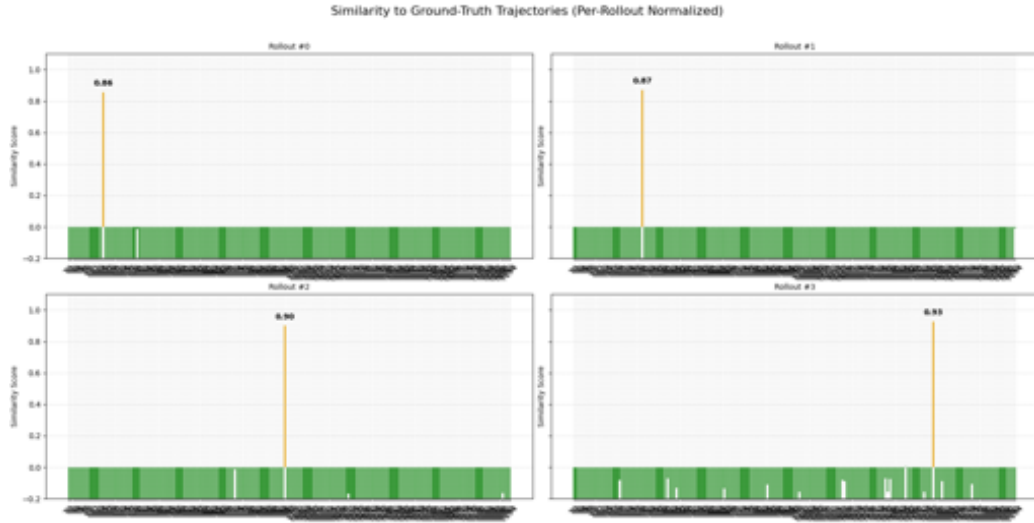


Figure 18: Analysis of a Diffusion Policy model roll-outs on the validation set with respect to both the ground-truth training and validation trajectories. The model was trained on the “Can” pickup benchmark.

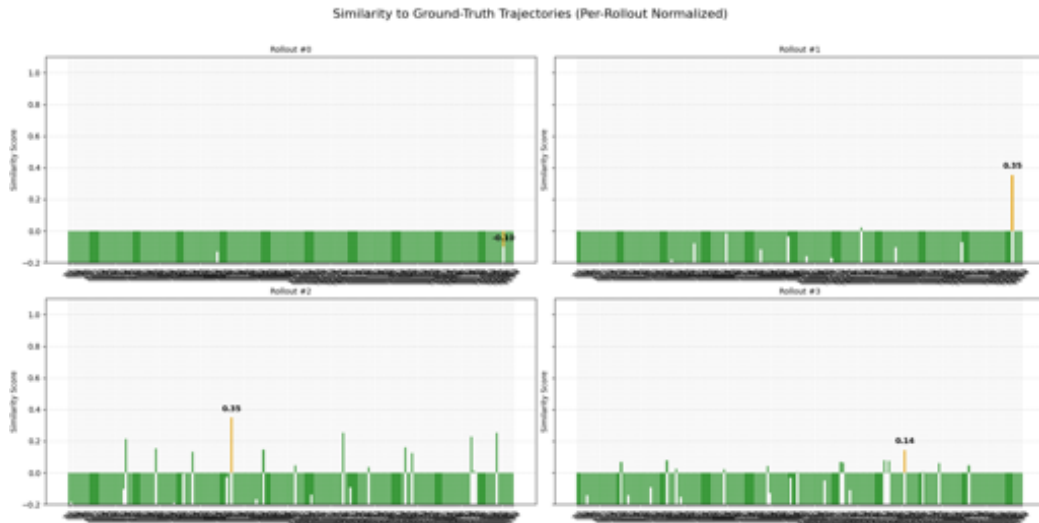


Figure 19: Analysis of a Diffusion Policy model roll-outs on the validation set with respect to the ground-truth training trajectories. The model was trained on the “Can” pickup benchmark.



Figure 20: A subset of the total analysis done for the roll-outs of a Diffusion Policy model from the training set with respect to all ground-truth training trajectories. The model was trained on the “Square” pickup benchmark.

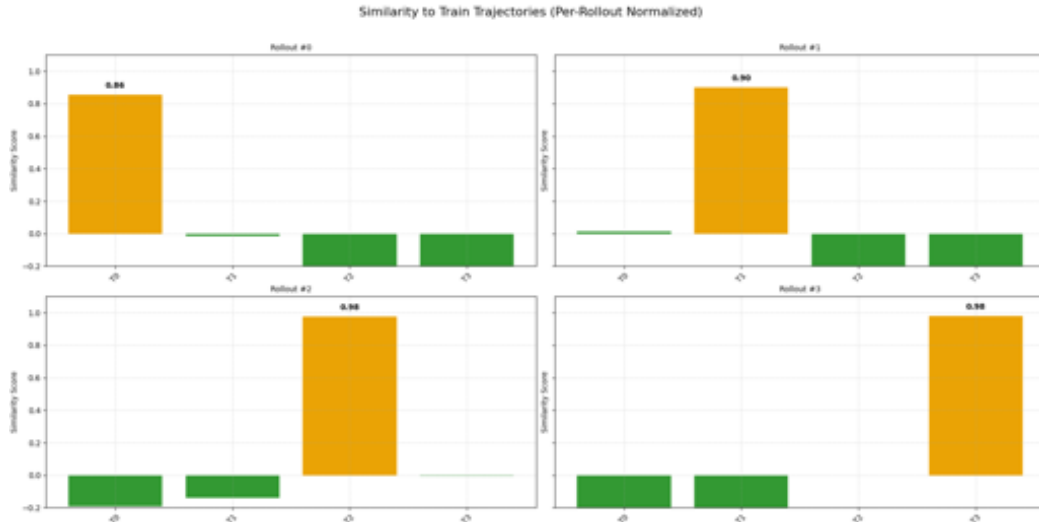


Figure 21: Analysis of a Diffusion Policy model roll-outs on validation set with respect to all ground-truth validation trajectories. The model was trained on the “Square” benchmark.

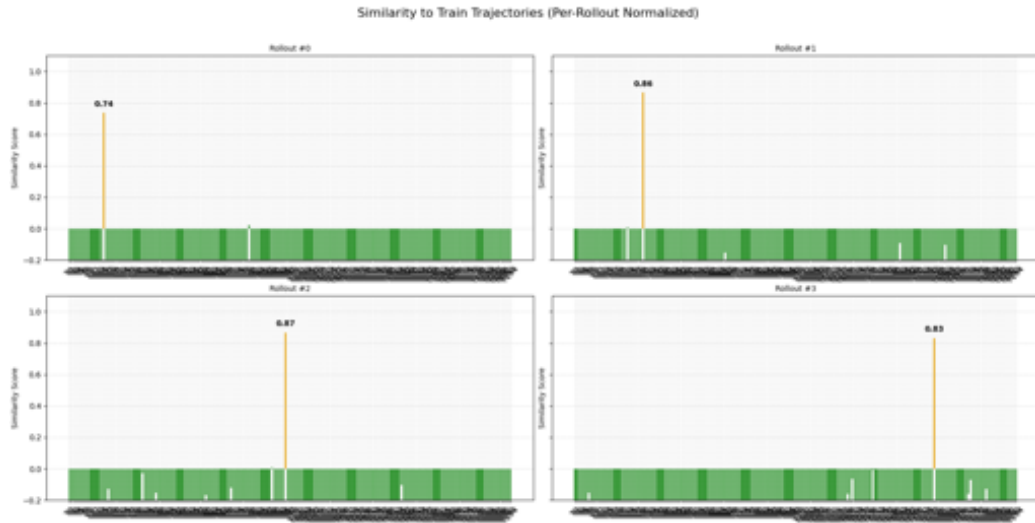


Figure 22: Analysis of a Diffusion Policy model roll-outs on validation set with respect to both the ground-truth training and validation trajectories. The model was trained on the “Square” pickup benchmark.

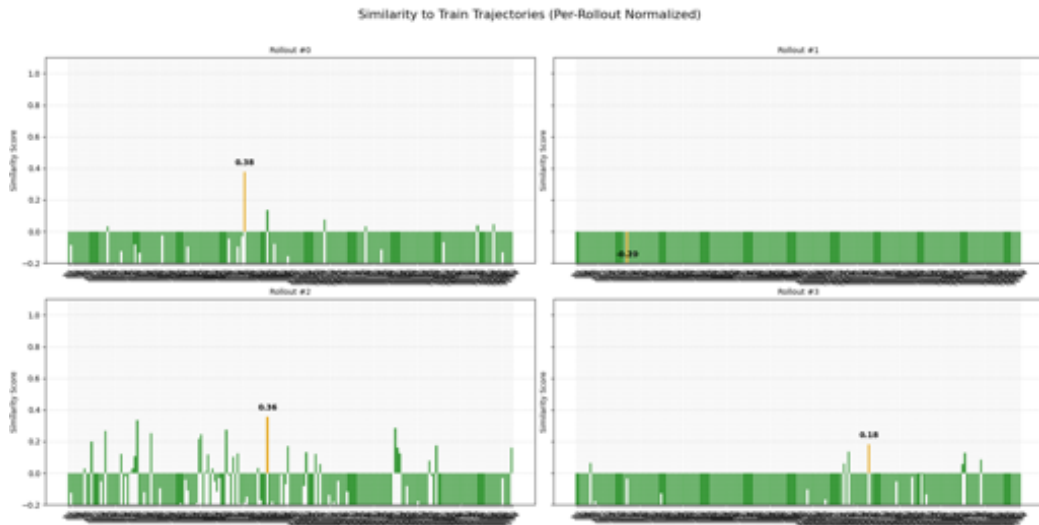


Figure 23: Analysis of a Diffusion Policy model roll-outs on training set with respect to all ground-truth training trajectories. The model was trained on the “Square” benchmark.