On the Fragility of State-Space Augmentation in Diffusion-Based Robot Imitation Learning

Thomas Prosser, Angelos Plastropoulos, Adolfo Perrusquía

***Abstract*— Diffusion-based policies have shown strong performance in robotic imitation learning but typically require large demonstration datasets. This work examines whether simple state-space data augmentation can improve diffusion-based imitation learning in low-data regimes. We evaluate Gaussian noise injection and feature dropout applied to state inputs during training of state-only diffusion policies across four Meta-World manipulation tasks. Contrary to expectations, both augmentation strategies consistently reduce task success compared to baselines trained on unmodified demonstrations, indicating that naïve state perturbations may disrupt task-relevant information rather than promote generalization. We further present an experimental framework for systematic evaluation of augmentation methods in offline diffusion-based imitation learning, providing a foundation for future studies on more effective data-efficient training strategies.**

I. Introduction

Robotic manipulation policies often require large amounts of high-quality demonstration data to achieve reliable generalisation across tasks [1], [2]. This heavy data dependence creates a major bottleneck for deploying learning-based controllers outside controlled laboratory environments, where collecting demonstrations is costly, time-intensive, and sometimes infeasible. Reducing the data requirements of imitation learning methods is therefore a critical step toward practical real-world robotic deployment.

Recent advances in diffusion models have introduced a powerful paradigm for action generation in robotic control. Rather than mapping observations directly to single-step actions, diffusion policies learn to iteratively denoise action sequences sampled from a noise distribution conditioned on sensory input [3]. This formulation enables expressive trajectory modeling and has demonstrated state-of-the-art performance across diverse robotic manipulation benchmarks. Despite these successes, diffusion-based policies remain highly sensitive to dataset size and quality, often requiring substantial demonstration data to achieve stable training and strong generalisation.

A natural approach to improving data efficiency is data augmentation [4], which aims to increase effective dataset diversity without additional data collection. Augmentation strategies such as noise injection and feature dropout have proven effective in other machine learning domains, yet their impact on diffusion-based imitation learning remains underexplored, particularly for low-dimensional state-based inputs. In this work, we investigate the data efficiency of diffusion-based robotic control and examine whether state-space data augmentation can mitigate high data demands. We implement a conditional denoising diffusion policy within the Meta-World benchmark and systematically vary dataset sizes and augmentation strategies. Through controlled experiments, we analyse how performance scales with available demonstrations and evaluate whether simple augmentations improve or degrade task success in low-data regimes.

Thomas Prosser, Angelos Plastropoulos, and Adolfo Perrusquía are with the Faculty of Engineering and Applied Sciences, Cranfield University, MK43 0AL Bedford, UK. email: adolfo.perrusquia-guzman@cranfield.ac.uk

A. Related Work

Imitation learning (IL) enables robots to learn from expert demonstrations, providing an efficient alternative to reinforcement learning when reward functions are difficult to specify [5]. Behaviour Cloning (BC), a straightforward IL approach, learns policies via supervised mapping from observations to actions [6]. Although BC is sample-efficient compared to other IL methods, it suffers from covariate shift, leading to compounding errors and poor generalisation in states unseen during training [7]. Recent advances model multi-step action sequences with recurrent or transformer architectures to mitigate these issues [3].

Alternative algorithms such as Inverse Reinforcement Learning (IRL) aims to recover the underlying reward function explaining expert behaviour, which can then be used to train policies via reinforcement learning [8], [9]. While IRL offers interpretability and generalisation beyond observed demonstrations, its requirement to solve an RL problem at each iteration makes it computationally expensive and sample inefficient, limiting practical use in robotics [10], [11]. Adversarial Imitation Learning (AIL), such as Generative Adversarial Imitation Learning (GAIL), formulates imitation as a minimax game between a discriminator and a policy, bypassing explicit reward inference [12]. Although effective in simulation, adversarial training can be unstable, and the need for extensive policy rollouts reduces sample efficiency, posing challenges for real-world robot learning.

Diffusion models have recently been adapted to imitation learning, resulting in diffusion policies that generate entire action trajectories by iteratively denoising noisy sequences conditioned on observations. These policies capture multi-modal behaviours and improve temporal consistency compared to AIL [13], [14]. However, diffusion policies typically require large datasets for effective training and are sensitive to data quality, limiting their use in low-data robotic settings.

Data efficiency and generalisation remain key challenges in robotic imitation learning due to the high cost of collecting expert demonstrations [15]. Data augmentation is a common strategy to improve robustness and sample efficiency by

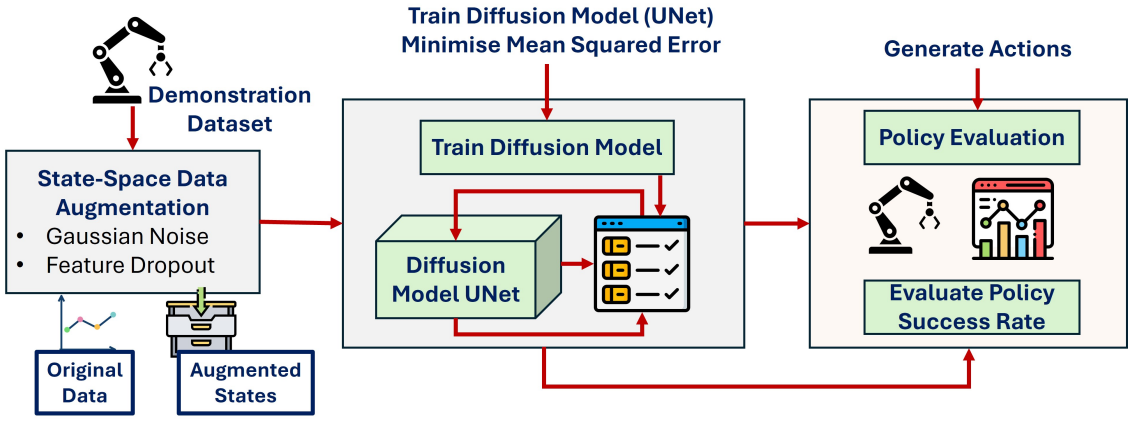


Fig. 1. High-level architecture of the proposed Methodology

artificially increasing dataset diversity without new data collection. While image-based augmentations have shown success in visual imitation tasks [16], state-space augmentations such as noise injection and trajectory perturbations have also been applied to improve robustness in robotics [17], [18]. However, the effectiveness of such augmentations for diffusion-based policies in low-data regimes is not well understood.

II. Methodology

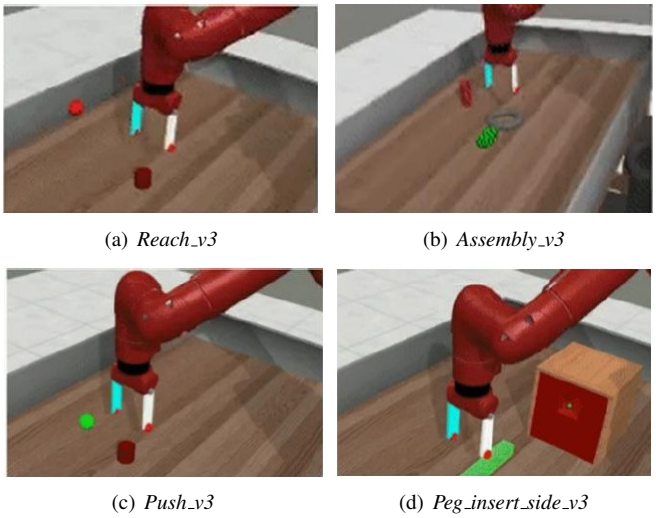
The proposed methodology is depicted in Fig. 1. The experimental pipeline consists of demonstration collection, off-line state-space augmentation, diffusion policy training, and policy evaluation. For each Meta-World manipulation task, we first collect expert demonstrations using pre-trained reinforcement learning controllers. Only successful episodes are retained and truncated at the first success signal to remove post-task behaviour, yielding concise state–action trajectories. From the full demonstration stream, we construct nested dataset subsets of increasing size to enable controlled evaluation under limited data. For each subset, we optionally apply off-line augmentation to the privileged state sequences, generating additional synthetic trajectories while keeping the corresponding action sequences and episode boundaries unchanged. This preserves temporal alignment between states and actions and allows direct comparison between baseline and augmented training sets at identical demonstration counts.

Each dataset variant is then used to train a state conditioned diffusion policy that models a distribution over action sequences. The policy is implemented as a one-dimensional UNet operating along the temporal axis and is conditioned on a short window of recent privileged states. During training, Gaussian noise is progressively added to action sequences according to a fixed diffusion schedule, and the network is optimised to predict the injected noise using a denoising diffusion objective. At test time, the trained model generates action sequences through iterative denoising with a DDIM sampler in a receding-horizon control loop. We evaluate task success rates across dataset sizes and augmentation strategies to assess whether state-space perturbations improve the sample efficiency of diffusion-based imitation learning.

A. Simulation Environments

The experiments are conducted in Meta-World [19], a benchmark suite of robotic manipulation environments based on a simulated Sawyer robot arm. In this paper we focus on four main environments (as depicted in Fig. 2) which are described as follows,

- *Push_v3*: Push the puck to a goal. Randomize puck and goal positions.
- *Assembly_v3*: Pick up a nut and place it onto a peg. Randomize nut and peg positions.
- *Reach_v3*: Reach a goal position. Randomize the goal positions.
- *Peg_insert_side_v3*: Insert a peg sideways. Randomize peg and goal positions.



(a) *Reach_v3* (b) *Assembly_v3*

(c) *Push_v3* (d) *Peg_insert_side_v3*

Fig. 2. Meta-World Environments

These environments provide a range of manipulation task ranging from more simple movement to single object manipulation and contact rich manipulation.

B. State and action spaces

A four dimensional continuous action space is used in Meta-World given by

$$\boldsymbol{a}_t = [\Delta x, \Delta y, \Delta z, g] \in \mathbb{R}^{1\times 4}, \tag{1}$$

where $\Delta x, \Delta y, \Delta z \in [-1, 1]$ represent the end-effector displacements in the Cartesian space and g denotes the gripper command (open/close) at time step t.

The state space includes low-dimensional observations in the form of a 39-dimensional observation vector at time step t, representing the robot position, related kinematic variables, position, spatial orientation and rotation of objects and goal position, denoted as $\boldsymbol{s}_t \in \mathbb{R}^{1\times 39}$.

C. Data Collection

The demonstration datasets, consisting of state-action pairs, were collected across four Meta-World environments using expert reinforcement learning policies. Only successful trajectories were retained to ensure high-quality training data. To promote concise demonstrations free from extraneous motion, episodes were terminated immediately upon receiving the first success signal. This procedure resulted in consistent and task-relevant data suitable for imitation learning.

Rather than aggregating all collected trajectories into a single dataset, we created structured subsets containing fixed numbers of successful episodes: 10, 20, 30, 40, 50, 100, 150, 200, 250, and 300 per task. Each subset is a strict prefix of the full success stream; for example, the 100-episode subset includes exactly the first 100 successful demonstrations and is fully contained within the 150-episode subset. This hierarchical organization enables controlled comparisons across dataset sizes, facilitating systematic analysis of sample efficiency and data scaling effects under consistent conditions. In total, 1200 successful demonstration episodes were collected across the four environments, comprising approximately 100,000 time steps.

D. Data Augmentation

To evaluate whether offline perturbations of the state improve the sample efficiency of diffusion policies, we generate augmented copies of each dataset prior to training. Augmentation is applied exclusively to the state vectors $\boldsymbol{s}_t$, while the corresponding actions $\boldsymbol{a}_t$ and episode boundaries remain unchanged. For each original timestep, we create M synthetic variants that are concatenated with the original data to form an expanded training set.

1) Gaussian State Noise: We independently add dimension wise Gaussian noise [17], scaled relative to the empirical range of each state dimension in the source dataset. Let $x_{\min,d}$ and $x_{\max,d}$ denote the per-dimension minima and maxima (with a small ϵ added if $x_{\max,d} - x_{\min,d}$ is near zero to avoid degenerate scales). For a noise percentage $p_{\text{noise}} = 1$, the noise standard deviation for each dimension is computed as

$$\sigma_d = \frac{p_{\text{noise}}}{100}\left(x_{\max,d} - x_{\min,d}\right). \tag{2}$$

The augmented state at time t is

$$\tilde{s}_t = s_t + \epsilon_t, \quad \epsilon_{t,d} \sim \mathcal{N}(0, \sigma_d^2). \tag{3}$$

2) Element-wise Feature Dropout: We apply independent, element-wise dropout [20] to the state vector, defined as

$$m_{t,d} \sim \text{Bernoulli}(1 - p_{\text{drop}}), \quad \bar{s}_{t,d} = m_{t,d} s_{t,d}, \tag{4}$$

with a dropout probability of $p_{\text{drop}} = 0.05$. As with Gaussian noise, only the states are modified; actions and episode boundaries remain unchanged. This mechanism simulates partial observation failures while preserving temporal alignment and system dynamics. Both augmentation techniques were applied to dataset subsets containing 50 episodes per environment, resulting in 8 additional datasets each composed of 50 original and 50 augmented episodes.

E. Data Loader

We train on state-only demonstrations and therefore construct a data loader that delivers temporally coherent windows of normalised robot states and future actions. Let H_o denote the observation horizon and H_p the prediction horizon, then for each valid centre indec t within an episode, the loader returns a pair of the form

$$\begin{aligned} \boldsymbol{O}_t &= [\boldsymbol{s}_{t-H_o+1} \cdots \boldsymbol{s}_t] \in \mathbb{R}^{H_o\times 39}, \\ \boldsymbol{A}_t &= \{\boldsymbol{a}_t, \cdots, \boldsymbol{a}_{t+H_p-1}\} \in \mathbb{R}^{H_p\times 4}. \end{aligned} \tag{5}$$

1) Episode-safe windowing: Episode boundaries are encoded by a binary value *episode_ends*, where a value of 1 marks the final time step of an episode. From this vector, we recover the inclusive start and end indices (s, e) of each episode. To construct valid training samples, we extract fixed-length observation and action windows centred at time step t. However, to ensure that both the past observation horizon and the future action horizon lie entirely within the same episode, not all time steps are admissible as sample centres.

Given an observation horizon H_o and a prediction horizon H_p, the set of valid centre indices within an episode is defined as $t = s+(H_o-1), \ldots, e-(H_p-1)$. This guarantees that the observation window $[t-(H_o-1),\, t]$ and the action window $[t,\, t+(H_p-1)]$ are fully contained within a single episode. If an episode length is shorter than $H_o + H_p - 1$, it yields no valid training samples. In practice, with our collected dataset and chosen horizon values, all episodes satisfy this condition and contribute valid samples.

2) Normalisation: To stabilise training and ensure consistent scaling across tasks, we apply per-dimension min–max normalisation independently to states and actions, mapping each feature to the range $[-1, 1]$. However, due to the structure of the privileged state vector $\boldsymbol{s}_t$, certain state dimensions remain constant (often identically zero) across all demonstrations in some environments. Direct min–max normalisation in such cases leads to ill-defined divisions by zero or extremely small denominators.

To avoid this, we first sanitise the empirical feature ranges. Let $x_{\min,d}$ and $x_{\max,d}$ denote the per-dimension minima and maxima computed over the dataset defined as

$$\tilde{x}_{\min,d} = \text{safe}(x_{\min,d}), \qquad \tilde{x}_{\max,d} = \text{safe}(x_{\max,d}), \tag{6}$$

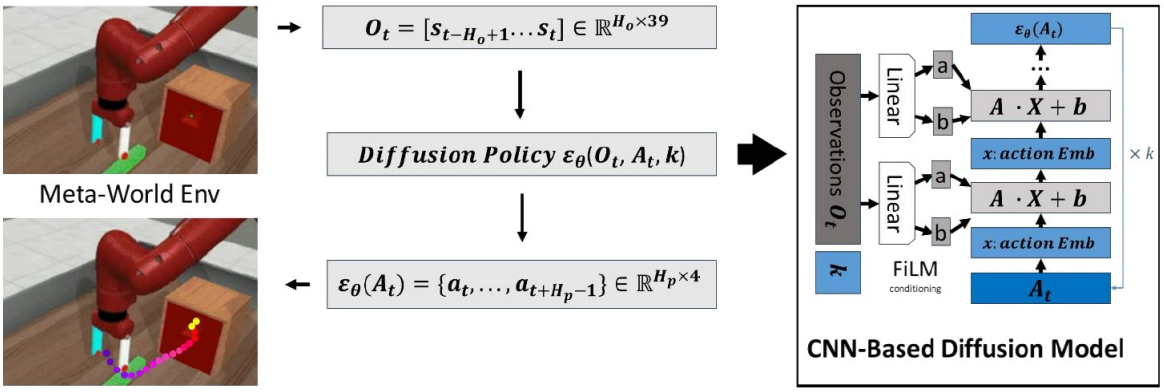


Fig. 3. Action Diffusion policy architecture

where safe$(\cdot)$ enforces a minimum span $\tilde{x}_{\max,d} - \tilde{x}_{\min,d} \geq \varepsilon$ for each dimension d, preventing degenerate normalisation scales, where $\varepsilon > 0$ is a small constant ensuring a non-zero normalisation span for near-constant features.

Normalisation is then performed element-wise as

$$\text{norm}(x_{t,d}) = 2\frac{x_{t,d} - \tilde{x}_{\min,d}}{\tilde{x}_{\max,d} - \tilde{x}_{\min,d}} - 1. \tag{7}$$

This guarantees numerically stable and well-defined values even for features with near-zero variance. Additionally, the data loader explicitly detects state dimensions that are constant zero in the raw data (i.e., $\tilde{x}_{\max,d} = \tilde{x}_{\min,d} = 0$ within a small tolerance) and forces their normalised value to exactly zero. This prevents the introduction of numerical artefacts in non-informative dimensions while preserving their presence for architectural consistency.

III. Action Diffusion Policy Architecture

We utilize a state-conditioned one-dimensional UNet architecture [3], illustrated in Fig. 3, to model action sequences along the temporal dimension. The network is trained to predict diffusion noise as part of the denoising diffusion probabilistic model for control. This conditional UNet captures action sequences over the prediction horizon through a symmetric encoder-decoder structure with skip connections. Notably, the network is non-causal, which is suitable for denoising across a fixed horizon, and it outputs noise predictions that are aligned with the input action sequences.

At each training step, the model receives as input: (i) a sequence of actions of length H_p, denoted as $\boldsymbol{A}_t \in \mathbb{R}^{B \times H_p \times 4}$; (ii) a diffusion timestep index k; and (iii) a flattened window of the most recent privileged states of length H_o, concatenated into a global conditioning vector $\boldsymbol{c}$. The conditioning vector $\boldsymbol{c} \in \mathbb{R}^{B \times H_o \times 16}$ is broadcast and injected into the UNet through feature-wise affine modulation at each denoising block. Given a noisy action sequence $\boldsymbol{A}_t^{(k)}$ at diffusion step k, the network outputs $\boldsymbol{\epsilon}_\theta(\boldsymbol{A}_t^{(k)}, k, \boldsymbol{c})$, an estimate of the Gaussian noise added during the forward diffusion process. Training follows the standard DDPM objective, minimising the mean-squared error between predicted and true noise. At inference time, action sequences are generated by iteratively denoising from Gaussian noise using a DDIM sampler, producing temporally coherent control trajectories conditioned on the current state history.

To isolate the effect of state-space data augmentation, the diffusion policy architecture and all training hyperparameters are kept fixed across all experiments. Only the demonstration datasets are modified through the augmentation strategies described in the sequel of this paper. This controlled design ensures that observed performance differences can be attributed directly to the impact of off-line state perturbations on diffusion-based imitation learning. Furthermore, we employ state-only conditioning to focus specifically on the effect of state-space transformations, avoiding confounding effects from visual or multimodal observation augmentations.

A. FiLM conditioning with time step and state window

Conditioning enters every residual block via Feature-wise Linear Modulation (FiLM). The diffusion time step k is first mapped to a continuous embedding by a sinusoidal positional encoder of dimension $d_t = 256$, following by an MLP. This embedding is concatenated with the flattened state window $\boldsymbol{c}$ to form a global context

$$\boldsymbol{g} = [\text{MLP}(\text{Sinusoidal}(k)); \boldsymbol{c}] \in \mathbb{R}^{d_t + (H_o \times 16)}. \tag{8}$$

B. Training Procedure

Each policy is trained independently on the state-only subsets. This ensures the results could be directly compared across data-regimes and augmentation strategies. We adopt the denoising diffusion probabilistic model (DDPM) framework to learn action sequence generation. Given an expert trajectory of future actions $\boldsymbol{A}_t$ at time t, the forward diffusion process progressively corrupts the sequence with Gaussian noise over K timesteps. At an arbitrary diffusion timestep $k \in \{1, ..., K\}$, the noised action sequence is given by

$$q(\boldsymbol{A}^k | \boldsymbol{A}_t) = \sqrt{a_k}\boldsymbol{A}_t + \sqrt{1 - a_k}\boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{9}$$

where $a_k = \prod_{i=1}^{k}(1 - \beta_i)$ is the cumulative product of the variance schedule $\{\beta_i\}_{i=1}^{k}$.

The model $\boldsymbol{e}_\theta$ is trained to predict the added noise $\boldsymbol{\varepsilon}$ given the noisy action sequence $\boldsymbol{A}^k$, the conditioning state

observations $\boldsymbol{O}_t$, and the diffusion step k. The training objective is the mean squared error (MSE)

$$\mathcal{L} = MSE(\boldsymbol{\varepsilon} - \hat{\boldsymbol{\varepsilon}}_\theta(\boldsymbol{A}^k, \boldsymbol{O}_t, k)). \tag{10}$$

This is equivalent to learning the score function of the conditional action distribution. At inference, action trajectories are sampled by iteratively denoising from Gaussian noise using a deterministic DDIM update. Optimisation is performed using AdamW with weight decay and cosine learning rate scheduler with 500 warm-up steps. Each model was trained with a batch size of 256 for up to 1,000 epochs. To stabilise learning, an exponential moving average (EMA) of model parameters with decay rate 0.75 is maintained throughout training and these EMA weights are used for checkpointing and evaluation.

C. Inference procedure

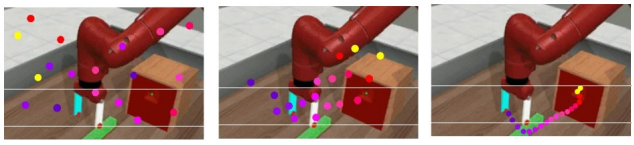
Policies are deployed on their corresponding Meta-World environments using the deterministic DDIM sampler. For each step, the most recent two states ($H_o = 2$) are concatenated, normalised with training statistics and flattened into a conditional vector. This is provided to the FiLM-conditioned UNet, from there we start with Gaussian noise and iteratively refine to which a sequence of $H_p = 16$ actions.

DDIM inference begins from Gaussian noise $\boldsymbol{A}^k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and applies iterative denoising steps indexed by $k = K, ..., 1$. At each step, the model predicts noise $\boldsymbol{\varepsilon}_\theta(\boldsymbol{A}^k, \boldsymbol{O}_t, k)$ and the trajectory is updated as

$$\boldsymbol{A}^{k-1} = \sqrt{a_{k-1}}\hat{\boldsymbol{A}}^0 + \sqrt{1 - a_{k-1}}\hat{\boldsymbol{\varepsilon}}_\theta, \tag{11}$$

$$\hat{\boldsymbol{A}}^0 = \frac{\boldsymbol{A}^k - \sqrt{1 - a_k}\hat{\boldsymbol{\varepsilon}}_\theta}{\sqrt{a_{k-1}}}. \tag{12}$$

This deterministic update reprojects the noisy trajectory back onto the data manifold, producing a normalised action plan $\tilde{\boldsymbol{A}}_t \in [-1, 1]^{H_p \times 4}$. The sequence is then unnormalised, clipped to the simulator's action bounds, and executed in open-loop chunks of length $H_a = 8$. Each episode terminated early upon success or at a maximum of 250 steps. Fig. 4 shows an example of the inference denoising process.



(a) $\boldsymbol{A}^k$ (b) $\boldsymbol{A}^{k-1}$ (c) $\tilde{\boldsymbol{A}}_t$

Fig. 4. Inference Denoising Process

IV. Simulation Studies

Fig. 5 shows the training loss curves, reporting the mean loss across epochs for the peg-insert-side-v3 task using both the original and augmented data subsets. For clarity, only one representative curve is presented, as all four variants exhibit nearly identical trends. In all cases, the loss decreases steadily with training and converges to a stable value after approximately 900 epochs. However, data augmentation adversely affected optimisation performance. Among the augmentation methods, feature dropout resulted in the smallest degradation, although its final loss remained higher than that of the corresponding 50-episode baseline. In contrast, Gaussian noise had the most detrimental effect, with its loss converging to a value worse than even the un-augmented 20-episode subset after 1000 epochs.

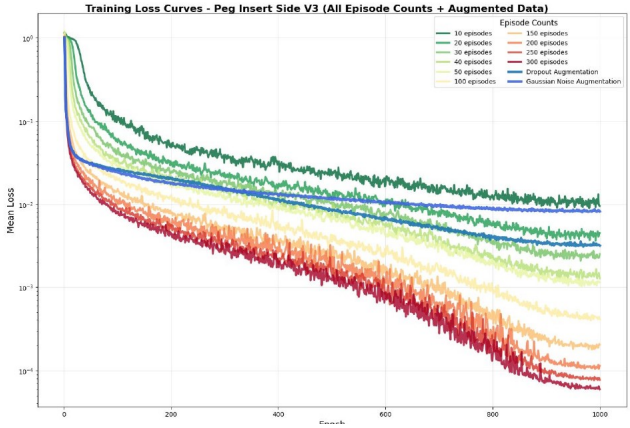
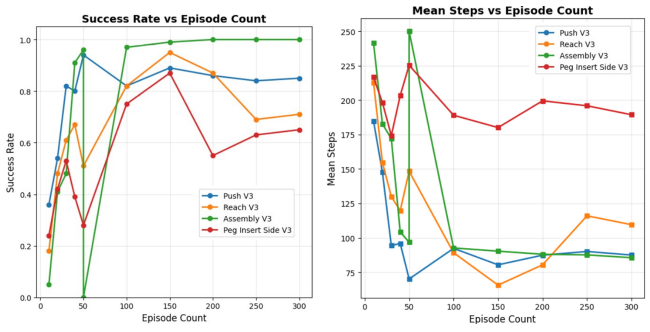


Fig. 5. Epochs vs Mean Loss for all Episodes and Augmented Data Subsets for Peg_Insert_Side_v3

Fig. 6(a) illustrates the relationship between episode count and success rate across all environments. In the low-data regime (up to approximately 50 episodes), the curves exhibit substantial variance, with success rates fluctuating before stabilising. Beyond this point, success rates generally increase across environments, reaching a peak at around 150 episodes. Notably, performance declines as episode counts increase further, with three of the four environments showing reduced success beyond this threshold. The only exception is *assembly-v3*, which maintains or slightly improves performance at higher data volumes.



(a) Episode Count vs Average Success Rate (b) Episode Count vs Average Mean Steps

Fig. 6. Results across all Environments

For reference, the expert policies used to generate demonstrations achieved near-perfect success across all environments, except for *peg-insert-side-v3*, where success was limited to 86.67%. This discrepancy underscores the greater difficulty of the peg-insert-side-v3 task and helps explain the

comparatively lower policy performance observed in this environment. Fig. 6(b) shows the relationship between episode count and mean number of steps across all environments. The trends closely mirror those observed in the success rate curves, with high variance in the low-data regime followed by performance improvements up to approximately 150 episodes. Beyond this point, degradation is evident across all tasks except *assembly-v3*, which again maintains comparatively stable performance. The curves also emphasise differences in task complexity; for instance, *peg-insert-side-v3* consistently requires a higher average number of steps, reflecting its lower success rates and greater difficulty relative to the other environments.

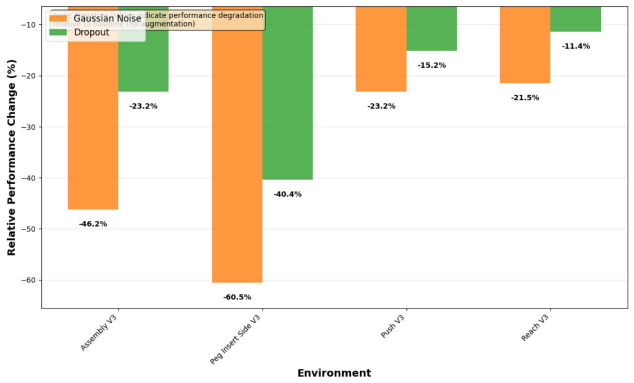


Fig. 7. Effect of Data Augmentation Strategies for a Given Environment

Fig. 7 compares the relative performance changes induced by Gaussian noise and feature dropout augmentations across the four environments. Both strategies consistently degraded performance relative to the baseline, with Gaussian noise producing the largest declines (up to -60.5% in *peg-insert-side-v3*). Feature dropout resulted in comparatively smaller, though still substantial, reductions (e.g., -40.4% in *peg-insert-side-v3*). Overall, these results indicate that, under the current experimental setup, the evaluated augmentation methods did not improve policy performance.

V. Conclusions

This study demonstrates that state-only diffusion policies can achieve reasonable success with modest amounts of data, particularly in tasks such as *assembly-v3* and *push-v3*, whereas more dexterous tasks like *peg-insert-side-v3* remain considerably more challenging. Performance generally improved up to approximately 150 demonstrations, beyond which success rates and efficiency declined in most environments, suggesting diminishing returns or potential overfitting at higher dataset sizes. Although additional demonstrations reduced episode length and improved behavioural consistency, simple augmentation strategies based on Gaussian noise and feature dropout proved ineffective, often leading to degraded optimisation and poorer performance compared to un-augmented baselines.

Future work should integrate a visual encoder with RGB observations alongside non-privileged robot state information (e.g., joint angles) would create a more realistic setting, helping to bridge the gap between simulation to real-world deployment.

References

[1] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," *IEEE Transactions on Cybernetics*, 2024.

[2] N. Lucotte, A. Perrusquía, A. Tsourdos, W. Guo, and H.-S. Shin, "Tactical planning interception enhancement using expert learning-twin delayed deep deterministic policy gradient," *IEEE Transactions on Intelligent Vehicles*, 2025.

[3] 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*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[4] A. Kumar, A. Perrusquía, S. Al-Rubaye, and W. Guo, "Wildfire and smoke early detection for drone applications: A light-weight deep learning approach," *Engineering Applications of Artificial Intelligence*, vol. 136, p. 108977, 2024.

[5] J. Ramírez, W. Yu, and A. Perrusquía, "Model-free reinforcement learning from expert demonstrations: a survey," *Artificial Intelligence Review*, vol. 55, no. 4, pp. 3213–3241, 2022.

[6] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

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

[8] L. H. Cooke, H. Klyne, E. Zhang, C. Laidlaw, M. Tambe, and F. Doshi-Velez, "Toward computationally efficient inverse reinforcement learning via reward shaping," *arXiv preprint arXiv:2312.09983*, 2023.

[9] A. Perrusquia and W. Guo, "Drone's objective inference using policy error inverse reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[10] A. J. Snoswell, S. P. Singh, and N. Ye, "Revisiting maximum entropy inverse reinforcement learning: New perspectives and algorithms," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 241–249.

[11] A. Perrusquía and W. Guo, "Uncovering reward goals in distributed drone swarms using physics-informed multiagent inverse reinforcement learning," *IEEE transactions on cybernetics*, 2024.

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

[13] B. Wang, G. Wu, T. Pang, Y. Zhang, and Y. Yin, "Diffail: Diffusion adversarial imitation learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 14, 2024, pp. 15 447–15 455.

[14] C.-M. Lai, H.-C. Wang, P.-C. Hsieh, F. Wang, M.-H. Chen, and S.-H. Sun, "Diffusion-reward adversarial imitation learning," *Advances in Neural Information Processing Systems*, vol. 37, 2024.

[15] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[16] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[17] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *Conference on robot learning*. PMLR, 2017, pp. 143–156.

[18] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[19] R. McLean, E. Chatzaroulas, L. McCutcheon, F. Röder, T. Yu, Z. He, K. Zentner, R. Julian, J. Terry, I. Woungang *et al.*, "Meta-world+: An improved, standardized, rl benchmark," *arXiv preprint arXiv:2505.11289*, 2025.

[20] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 2817–2826.