Diffusion Policy on D4RL

Danila Goncharenko[1] and Yanxing Chen[2]

Abstract— **This report discusses the project work in the course "Advanced Deep Learning in Robotics" (CIT433027) at TUM. In this project, a diffusion policy is employed on Franka kitchen and Push-T environments, remaking the reference paper's [3] simulation, as well as employed on the Antmaze dataset from D4RL dataset library. The performance and implementations are compared.**

I. INTRODUCTION

Recent advances in reinforcement learning have highlighted the promise of diffusion policies as a robust method for tackling complex control tasks.

Diffusion models, originally developed for generative modeling in computer vision, have recently demonstrated state-of-the-art performance in tasks such as image synthesis by progressively denoising data that has been gradually corrupted with noise [1]. In these models, a forward process incrementally adds noise to a data sample until it becomes nearly pure noise, and a reverse process—parameterized by a neural network—is then trained to gradually remove the noise to recover the original sample. This reverse denoising process is typically optimized using a mean-squared error loss between the predicted noise and the actual noise, encouraging the model to learn the underlying data distribution.

Building on these ideas, diffusion policies represent a novel approach for generating robot behavior by casting a robot's visuomotor policy as a conditional denoising diffusion process. Here, the model is conditioned on the current state or visual input of the robot, and it learns to generate an action by reversing a diffusion process applied to expert demonstrations or offline data. Unlike conventional reinforcement learning approaches that rely directly on reward signals for policy improvement, diffusion policies learn from offline data by modeling the distribution of expert actions in a robust, iterative manner.

D4RL [2] is an open-source benchmark for offline reinforcement learning. It provides standardized environments and datasets for training and benchmarking algorithms.

This project builds on the approach provided in assigned paper [3] by first replicating some of the original results and then integrating the diffusion policy framework into a new environment.

The contribution team has made is testing the performance of a diffusion policy on Push-T, Franka Kitchen and Antmaze environments. These experiments are conducted using a Virtual Machine on Google Cloud. The scripts were uploaded to GitHub repository

[1]Informatics, Technical University of Munich, Oulu, Finland, matriculation number 03801140, go52qaq@mytum.de

[1]Mathematics, Technical University of Munich, Munich, Germany, matriculation number 03781891, yanxing.chen@tum.de

II. DIFFUSION POLICY

A. Pipeline

To shortly summrize the paper's pipeline: Authors implement Denoising Diffusion Probabilistic Models (DDPMs) for robot behaviours. Policy learns the gradient of the action-distribution score function, and iteratively optimizes actions using stochastic Langevin dynamics steps. Diffusion process is conditioned to incorporate visual observations instead of putting them into joint distribution with actions. The model's parameters from the training process are stored at the checkpoints. Method was evaluated on 15 tasks, and it outperformed state-of-the-art baselines such as IBC, LSTM-GMM, and BET. Authors also conducted real-world experiments and ablation studies.

In more detail: The main idea of the diffusion policy is to start from Gaussian noise and iteratively refine it to produce an optimal action. One of the novel contributions of the referenced paper is that diffusion policy outputs action sequence instead of a single action, thus enabling receding horizon control. The authors have made two implementations for predicting the noise to subtract from the action sequence: CNN and Transformer.

In this project, only CNN implementation was utilized, since it is less computationally complex and the disadvantage of it is less smooth actions, which is not the focus of this study. For noise prediction, the gradient term pulls the action guess toward high-probability regions (low "energy"). After the noise predicted by the CNN is subtracted, small Gaussian perturbations are added at each diffusion step, as implemented in Stochastic Langevin Dynamics. This allows the model to explore more trajectories, in comparison to other methods, as shown in Fig. 1.

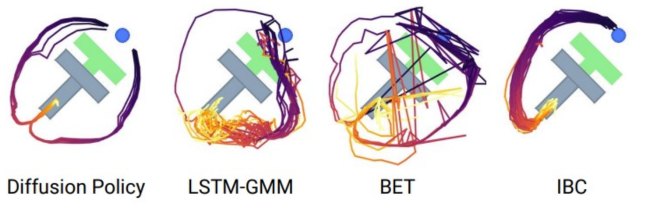

Fig. 1. Multimodal behavior. From Cheng et al. [3]

The visual encoder, based on ResNet-18, transforms camera observations into low-dimensional feature representations. These features are fed into the CNN, which employs

Feature-wise Linear Modulation (FiLM) [4] to apply affine transformations at each layer. This mechanism ensures that the network continuously incorporates the most recent observations into the denoising process. The complete pipeline is illustrated in Fig. 2.

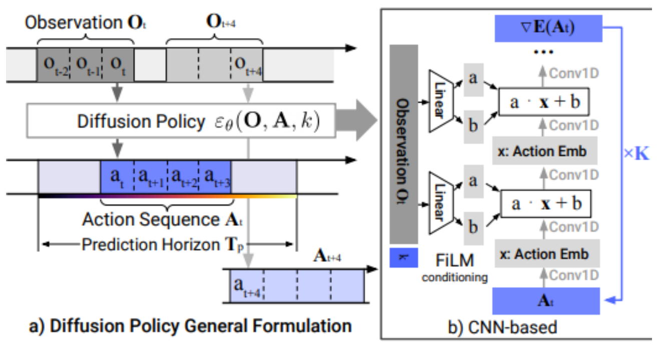
Observation $\mathbf{O}_t$
$\mathbf{O}_{t+4}$
o_{t-2} o_{t-1} o_t o_{t+4}
Diffusion Policy $\varepsilon_\theta(\mathbf{O}, \mathbf{A}, k)$
a_t a_{t+1} a_{t+2} a_{t+3}
Action Sequence $\mathbf{A}_t$
Prediction Horizon $\mathbf{T}_p$
$\mathbf{A}_{t+4}$
a_{t+4}
a) Diffusion Policy General Formulation
$\nabla E(\mathbf{A}_t)$
Observation $\mathbf{O}_t$
Linear
a
b
$a \cdot \mathbf{x} + b$
Conv1D
x: Action Emb
×K
FiLM
conditioning
$\mathbf{A}_t$
b) CNN-based

Fig. 2. System architecture. From Cheng et al. [3]

Another key aspect of the paper is saving parameters at checkpoints every 50 epochs and evaluating performance by both selecting the best checkpoint and averaging the last 10. This dual approach highlights Diffusion Policy's training stability. Deployment of the policy showed an average improvement of 46.9% in success rates.

In ablation studies, authors explored the trade-off between temporal consistency and responsiveness when selecting the action horizon. Shorter horizons allow for faster re-planning, while longer horizons enable dealing with multi-step tasks. Their experiments showed that optimal action horizon is around 8 steps for most tasks. They also discovered that position control significantly outperforms velocity control. Comparison of different encoders, such as ResNet and ViT-B/16, showed that Fine-tuning pre-trained visual encoders delivers the best results.

Finally, the authors demonstrated the performance of the method on real robotic systems, including tasks like Push-T, Mug Flipping, Sauce Pouring/Spreading, and various bimanual tasks (Egg Beater, Mat Unrolling, and Shirt Folding). Notably, the diffusion-based policies achieved near human-level performance and exhibited robustness to perturbations and latency.

B. Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models adopt a fundamentally different approach from traditional generative models [1]. Instead of directly learning a mapping from noise to data, these models simulate a two-step process: first, a forward process systematically corrupts a clean sample, and second, a learned reverse process gradually removes the noise to recover the original data.

In the forward process, a given sample x_0 is gradually transformed through a series of steps into a highly noisy version. Mathematically, at each step k, noise is added such that the sample evolves into x_k. The reverse process then attempts to recover x_{k-1} from x_k by removing the noise. One commonly used reverse update is formulated as:

$$x_{k-1} = \alpha\Big(x_k - \gamma\, \epsilon_\theta(x_k, k)\Big) + \mathcal{N}(0, \sigma^2 I),$$

where $\epsilon_\theta(x_k, k)$ is a neural network that estimates the noise present at step k, and α, γ, and σ are hyperparameters that control the update magnitude and the scale of the additive Gaussian noise.

As illustrated in Fig. 3, the forward diffusion process gradually corrupts the input sample by incrementally adding Gaussian noise, until the data becomes nearly indistinguishable from pure noise. In contrast, Fig. 4 demonstrates the reverse process, where the trained neural network iteratively removes noise to reconstruct the original sample. This dual-stage procedure—first adding noise and then learning to denoise—forms the core mechanism behind diffusion models and serves as the foundation for the diffusion policy approach applied in our work.

The training objective for the diffusion model is to learn to predict the noise that was added at each step. This is typically done by minimizing the mean squared error (MSE) between the actual noise ϵ_k and the network's estimate $\epsilon_\theta(x_0+\epsilon_k, k)$:

$$L = \mathrm{MSE}\Big(\epsilon_k,\ \epsilon_\theta\big(x_0 + \epsilon_k, k\big)\Big).$$

By optimizing this loss, the network progressively learns an effective denoising function, enabling it to generate samples that closely resemble the original data.

This denoising mechanism is central to the diffusion policy framework. In diffusion policies, the same idea is applied to robot control: rather than directly maximizing a reward signal, the policy is trained to generate actions that mimic expert behavior by reversing a noise process. In the next section, the adaptation of this conditional denoising process to generate control policies for complex tasks is described.

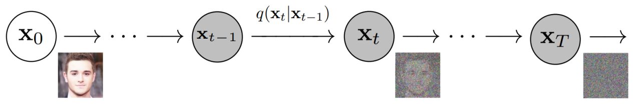

Fig. 3. Adding noise. From Ho et al. [1]

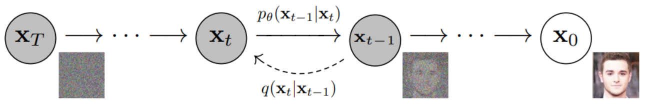
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$

Fig. 4. Denoising. From Ho et al. [1]

C. Diffusion for Policy Learning

In our approach, the reverse diffusion process for policy learning is defined by the following update equation:

$$A^t_{k-1} = \alpha\Big(A^t_k - \gamma\, \epsilon_\theta\big(O_t, A^t_k, k\big) + \mathcal{N}(0, \sigma^2 I)\Big)$$

Here, A^t_k represents the action sequence at diffusion step k for time t, O_t is the observation at time t, and $\epsilon_\theta(\cdot)$ is the neural network that predicts the noise. The parameters α and

γ control the scaling of the update, and $\mathcal{N}(0, \sigma^2 I)$ denotes Gaussian noise with covariance $\sigma^2 I$.

The training objective is to minimize the mean-squared error between the true noise ϵ_k and the network's prediction given the condition O_t and the noisy input $A_0^t + \epsilon_k$:

$$\mathcal{L} = \mathrm{MSE}\Big(\epsilon_k,\ \epsilon_\theta\big(O_t,\ A_0^t + \epsilon_k,\ k\big)\Big).$$

These two equations encapsulate the core mechanism of this diffusion policy: the reverse process progressively denoises the action sequences conditioned on the observations, while the training loss ensures that the model accurately learns to predict and remove the added noise.

III. ENVIRONMENTS

A. Push-T

Push-T image was the first environment to evaluate the algorithm in this project. The task is to push gray T-shaped block into green target area with blue circular end-effector. It was introduced in Implicit Behavioral Cloning [5] by Florence et al.

B. Franka Kitchen

Another environment that was evaluated in the project was Franka Kitchen, proposed in Relay Policy Learning [6] by Gupta et al. The initial study applied Relay Imitation Learning algorithm (RIL) and achieved a success rate of 21.7%. This environment modeled in MuJoCo was later added to D4RL library as one of the tasks. The task is to make 9 DoF position controlled Franka robot interact with a kitchen scene and complete 4 tasks in arbitrary order. The environment comes with 566 human demonstrations and 7 interactable objects: microwave, kettle, oven burners, light switch, two hinge cabinets, and a cabinet door slider.

The reward was defined to be sparse in the environment. It is equal to the number of completed tasks in the given Gymnasium step. Completion of the task means having the joint configuration within a 0.3 norm threshold of the goal configuration. Joint goal values are predetermined [7].

C. Antmaze

Antmaze is a challenging offline reinforcement learning benchmark provided by D4RL. It involves a quadrupedal robot navigating a maze to reach a target area, a task that is particularly difficult due to its sparse reward structure and the complex, high-dimensional nature of the state space. Antmaze was selected as a testbed because its inherent difficulty forces algorithms to extract useful behaviors from limited expert demonstrations, making it an ideal environment for evaluating robust policy learning approaches.

Unique characteristics of Antmaze

Sparse Rewards: The environment provides minimal feedback, meaning that successful navigation is only rewarded upon reaching the target. This sparsity challenges conventional reinforcement learning methods and emphasizes the need for effective imitation and robust policy learning.

Complex Navigation: The maze structure demands precise and long-horizon planning. The robot must handle high-dimensional sensory inputs and generate smooth, coordinated actions to successfully traverse the maze.

Offline Data Richness: D4RL provides a substantial amount of offline data for Antmaze, which, despite the sparse rewards, contains valuable demonstrations of expert behavior. This makes Antmaze an excellent benchmark for offline policy learning techniques.

Modifications in the code

Configuration: The configuration settings were updated to capture the unique aspects of the Antmaze task. In project's configuration, the environment identifier is specified corresponding to the Antmaze task and the observation and action dimensions are adjusted to match the environment's characteristics. Task-specific hyperparameters, such as the rollout horizon, padding parameters, and the random seed are set. In addition, an `env_runner` component was defined in the configuration to enable video generation during evaluation.

Dataset: Drawing inspiration from the Push-T dataset, a new dataset class tailored for Antmaze is developed. This dataset class is responsible for loading the offline data (which includes only the `state` and `action` fields) and properly segmenting the data into episodes using the provided metadata. It also computes normalization statistics over the full dataset to ensure that the input to the policy is well-scaled. This adaptation was necessary to bridge the gap between the D4RL data format and our diffusion policy training pipeline.

Workload: The training pipeline was modified to integrate the new Antmaze dataset and project's diffusion policy model. The core training loop remains similar to the original framework, where the model learns to denoise corrupted expert actions via a mean-squared error loss—but with specific adjustments to handle the characteristics of the Antmaze task. For evaluation, a separate scipt was created to generate video from a checkpoint in the Antmaze environment.

Video Generation

While video recording was originally embedded in the environment runner for Push-T and other environments, the adjustments which were done for Antmaze required many changes to the runner, which is why team decided on this approach. A separate video generation module was developed to instantiate the Antmaze environment, perform rollouts with the trained policy, and record the RGB frames using the environment's rendering function. The frames are then compiled into a video (e.g., in MP4 format).

Summary

In summary, by tailoring the configuration, dataset, and workload components to the Antmaze environment, project's diffusion policy framework is able to learn from offline data in a challenging, sparse-reward setting. The additional video recording module further facilitates qualitative evaluation by producing rollout videos that visually demonstrate the robot's navigation behavior.

IV. RESULTS

Three environments with varying training durations and epochs were evaluated. The Push-T image environment was trained for 14h39min and 331 epochs, Franka Kitchen for 14h1min and 293 epochs, and Antmaze for 28h26min and only 222 epochs. The total number of training steps also varied significantly: 150k for Kitchen, 55k for Push-T image, and 7k for Antmaze. Performance metrics indicate that the Push-T image environment achieved a mean success score of 89.95%. In the Franka kitchen environment, the success scores for the four tasks were 100%, 100%, 100%, and 94%.

Interestingly, as shown in Fig. 5, even though the validation loss increased, the overall performance improved, a trend further corroborated by the produced videos. Additional graphs can be found in Weights & Biases report here.

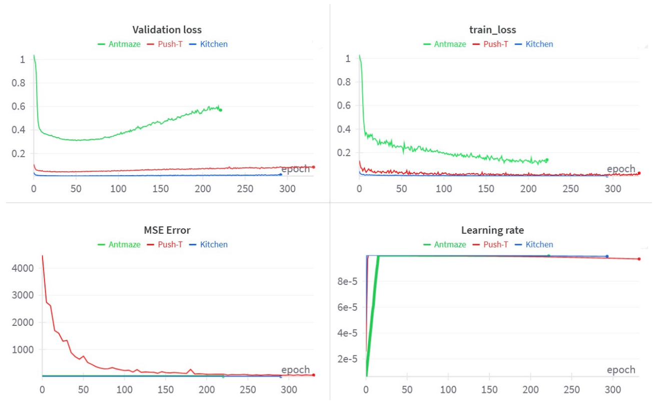

Fig. 5. Evaluation charts. From Weight & Biases.

Videos generated for the Push-T image environment show very smooth and precise movements, yet it is still evident that the algorithm may struggle with the task. In certain simulations, although the t-image nearly aligns with the target area, the overlap remains imperfect.

Videos generated for Franka Kitchen show that motion is much smoother for diffusion policy compared to the original method. Nevertheless, similar to the original RIL paper, the robot's movement becomes shaky toward the end of the simulation.

For Antmaze, the videos indicate that movement improves with the number of training epochs. However, it remains unclear whether the ant reaches its goal, as it occasionally stops moving altogether for several seconds. Latest run is shown in Fig. 6.

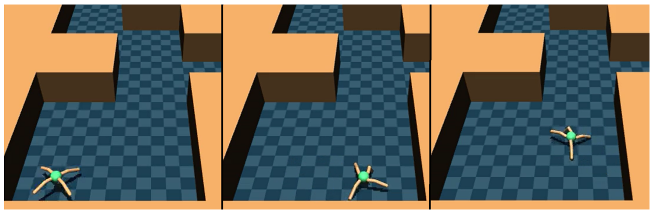

Fig. 6. Antmaze generated video

V. LIMITATIONS AND FUTURE WORK

Although the team has achieved the initial goal of implementing diffusion policy on the antmaze environment, there are many possible improvements that could be done to the project in the future work.

Firstly, the evaluation metrics, such as the success score, should be added to Antmaze.

Secondly, the video should also generate the target area. It already prints out goal position in the terminal console.

Thirdly, the environmental runner for Antmaze should be finalized. There was an attempt to modify Franka Kitchen runner to suit Antmaze's configuration, but as the main task was to create the video, a separate script was implemented instead.

Fourthly, diffusion policy training in the current framework does not directly incorporate reward signals; it focuses on imitating expert behavior through denoising. Future work could combine the denoising objective with reward-based fine-tuning.

VI. CONCLUSION

This project served as the team's first introduction to diffusion policies and reinforcement learning, providing invaluable hands-on experience in implementing policy learning for robotics. Through offline data processing, network training, and qualitative evaluation via video generation, the project laid a solid practical foundation for future research in robotic control.

The team explored the challenges and opportunities of connecting to, scaling, and using virtual machine instances to support high-GPU tasks. In addition, we gained valuable experience using GitHub for version control and collaboration, as well as leveraging the Google Cloud Console for managing cloud resources, which enabled us to work together efficiently and complete the project as a team.

Overall, the project expands the feasibility of using diffusion-based methods for complex robotic control tasks.

REFERENCES

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models.", 2020. Advances in neural information processing systems 33 : 6840-6851.

[2] Fu, Justin, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. "D4rl: Datasets for deep data-driven reinforcement learning.", 2020. arXiv preprint arXiv:2004.07219.

[3] Cheng Chi , Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, Shuran Song "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion", 2024. arXiv preprint arXiv:2303.04137

[4] Perez E, Strub F, De Vries H, Dumoulin V and Courville A. "Film: Visual reasoning with a general conditioning layer", 2018. In: Proceedings of the AAAI Conference on Artificial Intelligence.

[5] Florence P, Lynch C, Zeng A, Ramirez OA, Wahid A, Downs L, Wong A, Lee J, Mordatch I and Tompson J. "Implicit behavioral cloning", 2021. In: 5th Annual Conference on Robot Learning.

[6] Gupta A, Kumar V, Lynch C, Levine S and Hausman K. "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning", 2019. arXiv preprint arXiv:1910.11956

[7] Franka Kitchen - Gymnasium-Robotics Documentation. Information retrieved from 23/02/2025. https://robotics.farama.org/envs/franka_kitchen/franka_kitchen/