Survey of Task Transfer Learning for Robotics

Joseph Kawiecki

Abstract

Reinforcement learning for the use of robotic motion training has been an exciting development in recent years. Many learning-based methods have performed on par or better than traditional robotic motion controllers. However, most of these methods often still require specific knowledge of the robot, such as the kinematic makeup, to be deployed. It would be beneficial for more general methods to transfer across a wider range of robots. This paper will delve into this problem deeper, explore a few recent methods, and compare the results of these methods both qualitatively and quantitatively with the D4RL dataset [5].

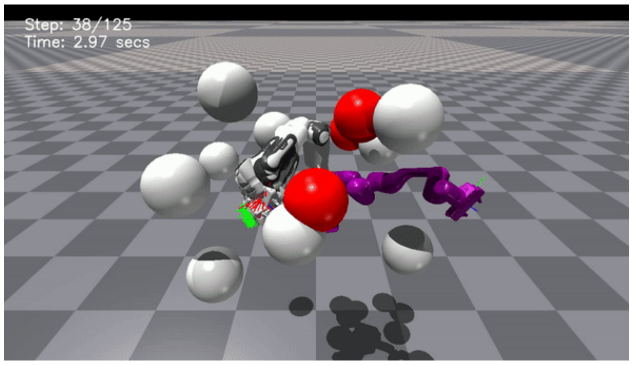


Figure 1. Kuka arm planning via a diffusion-based method.

1. Introduction

Over the past few years, robots have become increasingly common, now existing in nearly every facet of our lives. However, teaching a robot a specific behavior has traditionally been accomplished with motion controllers, a challenging task requiring extensive domain knowledge and effort. Furthermore, motion controller-based approaches are usually applicable only to a single or close set of movements for a particular robot. Thus it would be beneficial if there were more efficient means of training capable of generalizing to more than one specific robot agnostic of specific kinematics, degree of freedom, size, etc.

As such, many learning-based approaches to this problem have been introduced in recent years, with a large swathe coming from a reinforcement learning setup. Many of these early approaches have shown impressive results. While these approaches are more general than traditional controllers, they often still require unique or specific information about a certain robot or motion. As such, most have not shown great results when transferred to different robots or even to similar styles such as quadruped or arm-based. In all, learning-based approaches in recent years have not shown great success in a more generalized setting. This paper aims to explore, examine, and compare some methods to better generalize these imitation methods. A commonality among the following methods is a reward derived from real-world reference.

The first method discussed [11] extracts a ground truth movement from a real-world dog to teach a robot how to perform the same action. The authors translate the ground truth movement from the real animal to the robot to serve as a reward for a standard reinforcement learning problem. This translation makes use of the specific kinematics of the robot and does not perform well on other robot dogs with different kinematics. The next few papers discuss different methods to accomplish greater generalization and make this translation more effective. These methods range from simple randomization of hardware parameters to completely new architectures. Finally, the last methods examined involve using diffusion models tuned for motion trajectory. From a given start and end state, the idea is to use the denoising feature of a standard diffusion model to fit a distribution of trajectories as constrained by some penalty or reward. By using a generative model to fit these trajectories, the hope is that this will lead to a more generalized policy that can transfer well. This is expressed in Figure 1 where a Kuka robotic arm plans an optimal trajectory through a set of sphere obstacles.

This problem of designing a more universal controller has been the subject of my research. While still a work in progress, my current approach has been investigating diffusion models for motion trajectory and planning. The final steps will include the integration of the generalized trajectory model with a human reference motion and robot representation. The final model should hopefully be able to take a human ground truth motion and teach it to a set of robots from the same class such as robot arms.

2. Background

2.1. Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning in which an agent will learn to make decisions by interacting with an environment. As it relates to robotics, this can be especially useful in teaching complex behaviors. A reinforcement learning problem is comprised of a learning agent (robot) and the environment for which it interacts in. At a particular timestep, an agent will observe its environmental state, determine an action, and receive feedback based on that action in the form of an environmental reward. Over time, the agents may learn a policy that dictates a strategy mapping states to actions to achieve the optimal reward. Despite being widely used in practice within robotics, RL agents can take a long time and thus be resource-intensive to learn stable behaviors. Thus it would be ideal to learn more generalized approaches to new tasks and environments.

2.1.1. Proximal Policy Optimization (PPO)

Many of the methods to be discussed utilize a strategy to learn and update a policy called Proximal Policy Optimization (PPO). The idea is to estimate an advantage score which quantifies how much better a particular action is relative to a certain action suggested by the current policy. This advantage score can then be combined with a ratio of old to new policies to determine the update. However, the main insight of PPO relative to previous policy update algorithms and why it is relatively stable is due to the clipping of this gradient update. Clipping the advantage and policy ratio multiplication ensures each particular update is not too drastic. PPO is widely used in practice offering a good balance between efficiently achieving an optimal policy with stability due to clipping. Algorithm 1 demonstrates PPO with a clipped gradient update.

2.1.2. Deep Deterministic Policy Gradient (DDPG)

Another optimal policy search algorithm is the Deep Deterministic Policy Gradient (DDPG). DDPGs run with an actor-critic architecture in which an actor will suggest an action and the critic will evaluate the quality. Here the actor network is the policy and the critic is the value function, which estimates the cumulative reward for being in a particular state or choosing an action. DDPG is ideal for problems that contain a continuous action space. DDPG also utilizes experience replay to leverage past knowledge in a buffer to enhance data efficiency and stability.

Algorithm 1 PPO with Clipped Gradient

```
**Input:** initial policy parameters $\theta_0$, clipping threshold $\epsilon$
**for** $k = 0, 1, 2, ...$ **do**
    1. Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
    2. Estimate advantages $\hat{A}_t^{\pi\theta_k}$ using any advantage estimation algorithm
    3. Compute policy update
        $$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{\tau\sim\pi_k}\left[\mathcal{L}_{\theta_k}^{CLIP}(\theta)\right]$$
    by taking K steps of (minibatch) SGD
    4. $\mathcal{L}_{\theta_k}^{CLIP}(\theta)$ is
        $$= \min\left(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k}\right)$$
        $$r_t(\theta) = \frac{\pi_\theta}{\pi_{\theta_{\text{old}}}}$$
**end for**
```

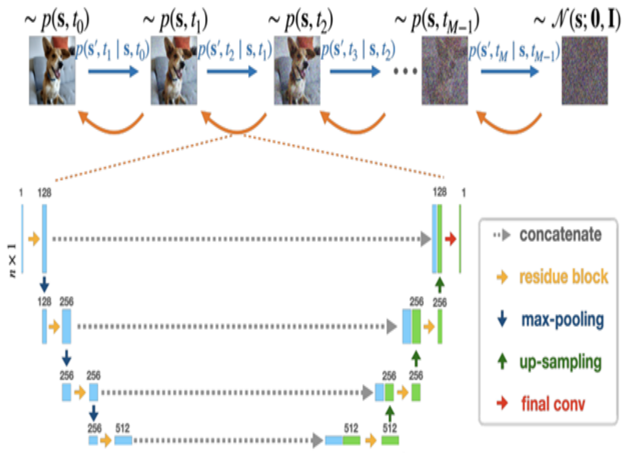
2.2. Denoising Diffusion Probabilistic Model (DDPM)

The later papers involve a relatively new generative architecture called a diffusion model [6]. More formally, denoising diffusion probabilistic models (DDPMs) were introduced in 2020 and have since been implemented in many practical settings with promising results in image generation, audio synthesis, inverse problem solving, and more. DDPMs work with two passes: a forward and a reverse. The forward, or diffusion pass, involves repeatedly adding noise following a scheduler to iteratively destroy the input training data. The reverse process involves transforming the noisy training data back to match the distribution of the original input data. A key feature of the DDPM is the reverse pass in which denoising occurs. This denoising feature has been applied to several papers to inversely solve trajectory optimization given a start and end state. Figure 2 further articulates the architecture of a DDPM. Most architectures employ a UNet for the forward and reverse passes.

3. Problem Statement

Robotics software has traditionally been designed for one robot to do one task through motion controllers. These controllers often require lots of time, effort, and specific domain knowledge to construct. As the practical applications of robotics become increasingly more realized and prevalent in our modern day, it would be ideal to design a more generalized, efficient approach. This problem has been increasingly common in research with a sharp uptick in papers published in recent years as seen in Figure 3.

Within robotics, constructing a universal controller able to



$\sim p(\mathbf{s}, t_0)$
$\sim p(\mathbf{s}, t_1)$
$\sim p(\mathbf{s}, t_2)$
$\sim p(\mathbf{s}, t_{M-1})$
$\sim \mathcal{N}(\mathbf{s}; \mathbf{0}, \mathbf{I})$
$p(\mathbf{s}', t_1 \mid \mathbf{s}, t_0)$
$p(\mathbf{s}', t_2 \mid \mathbf{s}, t_1)$
$p(\mathbf{s}', t_3 \mid \mathbf{s}, t_2)$
$p(\mathbf{s}', t_M \mid \mathbf{s}, t_{M-1})$
concatenate
residue block
max-pooling
up-sampling
final conv

Figure 2. Standard DDPM architecture with a UNet for the forward and reverse passes [13].

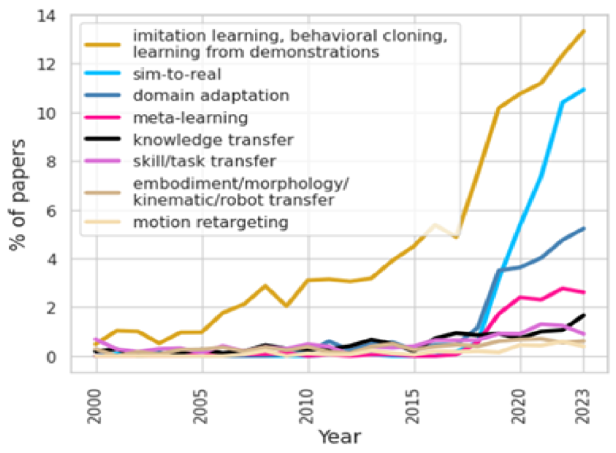


Figure 3. Recent increase in the number of papers relating to universal motion controllers.

transfer knowledge and experiences can be categorized as environment, task, or robot transfer. Environment transfer is the changing of the environmental complexity or conditions where the task is being executed. A large focus within this domain is the transfer from simulation to real world and vice versa. Task transfer is the idea of transferring learned behavior from one robot to another and is often referred to as generalization. Figure 4 visually summarizes the domains of transfer learning.

Task transfer for a universal controller is the primary area of my research and will be the focus of the remainder of this paper. With the area receiving much interest in recent years, there has been a wide diversity of methods and approaches. This paper will attempt to discuss and compare a few in greater detail.

4. Literature Review

4.1. Reinforcement Learning Baseline Approach

This method examines the use of reinforcement learning to teach a robot to imitate a real-world motion. Specifically, the authors split their approach into three components: motion retargeting, motion imitation, and domain adaptation. Figure 5 displays a summary of the procedure. Motion retargeting involves transferring real-world data to a simulation and finally to a robot. This is performed by pinpointing markers onto the joints of the real world and robot subjects. Next, these markers are adjusted, due to differences in kinematics and morphology, so that the real-world measurements are fit to the robot. Motion imitation involves using the ground truth reward, now transferred to the robot, to train the robot to learn the motion through Proximal Policy Optimization. Finally, the authors transfer the learned simulation policy back to the real world via Domain Adaption. This involves introducing perturbations and other randomization approaches into the policy. The result is an end-to-end method in which a real-world robot can learn from a real-world subject. The authors demonstrate their approach to a few common movements such as trotting and turning by a German Shephard to the Laikago quadruped robot. In all, the approach was successful in that their robot was able to learn these movements, but it was not very stable. Furthermore, the approach relied much on the specific kinematics of the Laikago robot and required extensive training for each particular skill.

4.2. Hardware Parameter Approach

The first idea for improved task transfer involves feeding a wider range of hardware parameters for training. The authors of Genloco [4] simply construct a range for each specific parameter of quadruped morphology for a few of the most common robots. The base, foot, thigh, calf, and more of the quadruped are all given a range of minimum and maximum values. Additionally, a size factor is used to scale the overall size of the robot relative to others. From there, PPO is again used in a reinforcement learning setting with ground truth motions as the reward. A similar idea is used by the authors of [3]. But instead of using a direct range of parameters, the authors produce an encoding of the kinematics, dynamics, and other factors of the robot. These include the degree of freedom (DoF), structure, joint damping, friction, geometry, and more. The authors set up a reinforcement learning problem with the policy evaluated by either PPO or DDPG depending on the reward and the hardware encoding used to represent the robot kinematics. The rewards constitute of Euclidean distance between the

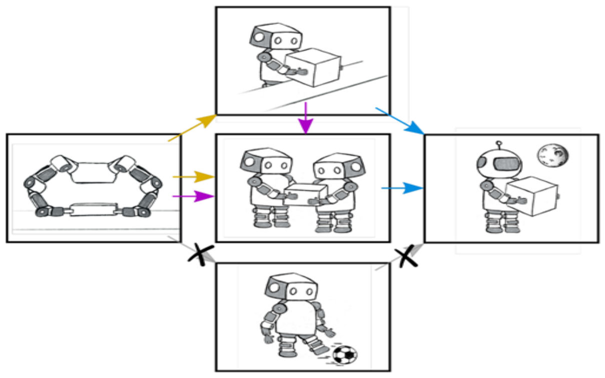


Figure 4. Illustration of transfer learning in robotics. A robot can store and leverage its experience performing a specific task in a particular environment to improve learning on a related task in a related environment. For example, the downward purple arrow portrays a robot using its gained knowledge from lifting a box to aid in teaching two robots to exchange a box with each other. This is known as task transfer as shown by the purple and yellow arrows. The blue arrows demonstrate environment transfer. The successful transfer usually at least requires some commonality between the source and target robot, task, and environments. The crossed-out arrows indicate a task is likely not transferable. A robot kicking a ball is likely not to have much useful experience in teaching a robot to carry a box in space.

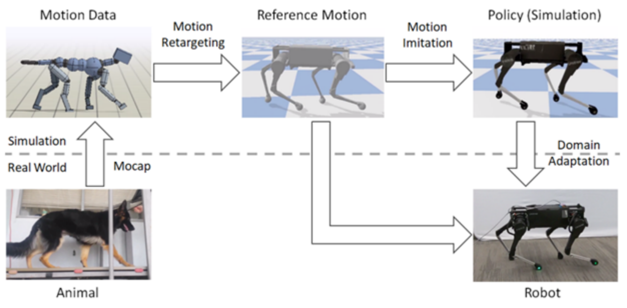


Figure 5. Overview of the process presented in Learning Agile Robotic Locomotion Skills by Imitating Animals

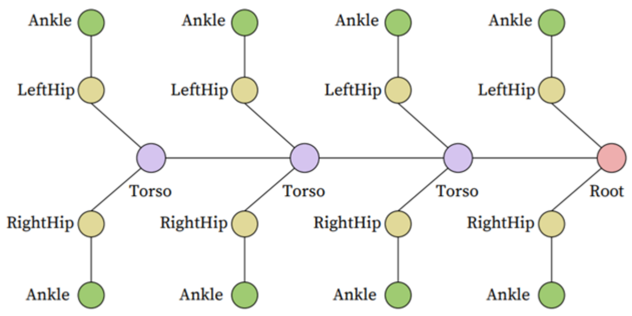


Figure 6. Sample insect robot represented as a graph.

robot's end position and the ground truth position. The approach is demonstrated on robot arms varying in DoF, length, and other kinematics. Both methods demonstrate limited success in zero-shot transfer to new robots of the same general morphological structure, such as quadrupeds to slightly different quadrupeds. However, neither method can generalize to robots out of the trained class distribution, but this was to be expected mostly. While these approaches demonstrate some success, they are still constrained and largely defined by the hardware considerations of each robot class.

4.3. Modular Approach

The following two papers present a different approach centered around modularization. The main insight of NerveNet [12] is that robot structure can be represented as a graph. This allows for training propagation beginning with the central structure of the agent before branching out to the specifics on the end nodes of the graph seen in Figure 6. The authors of REvolveR [10] build on this idea of modularity with robot decomposition. A policy can be trained on a set of modular, intermediary robots that can then be constructed into a target robot of different morphology and kinematics. Again using the reinforcement learning domain, each intermediary robot would be trained on a ground truth movement to learn a policy. Then, at the intermediary robots are fused to form a larger robot and their policies would also be combined to form a policy closer to the target. Rewards are scheduled to favor the larger robot.

In terms of task evaluation, REvolveR is performed on just one basic task of progressing forward while NerveNet includes a few tasks from the MuJoco Gym. One downside to graph-based methods is that graph neural network optimization is much more challenging than a traditional perceptron-based network. Thus as NerveNet outperforms most of the traditional perceptron-based reinforcement learning policy methods, it is usually less efficient.

5. Qualitative Comparison

5.1. Reinforcement Learning Baseline Approach

While the approach in [11] did produce good results and was successfully able to translate behavior from a real-world dog to a robot dog, it was still outclassed by manually designed motion controllers in terms of stability. In addition to the computational resources required to train a reinforcement learning model, the approach also requires extensive resources on the data collection side to source data from real-world subjects. However, datasets such as BEHAVE [1] do exist so this may not be as much of an issue. Last, as pre-

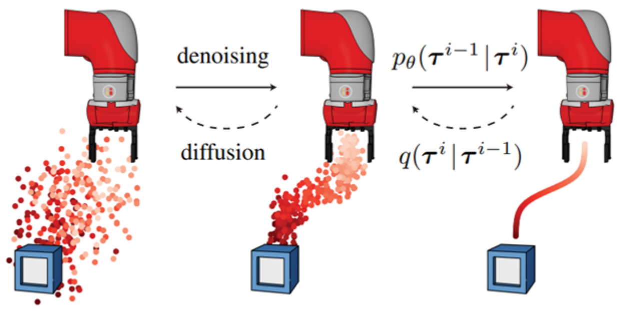

denoising
diffusion
$p_\theta(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^i)$
$q(\boldsymbol{\tau}^i \mid \boldsymbol{\tau}^{i-1})$

Figure 7. Iterative diffusion process demonstrating steps reliance on past and future iterations.

viously mentioned, this method requires specific parameter knowledge of the robot to be deployed, making it difficult to generalize behavior. In terms of application, at the time of publication, this method was likely an impressive step forward in teaching complex, agile locomotive motions to robotics. Particularly, the end-to-end solution beginning with data sourcing and ending with domain adaptation to the real world would have been especially valuable from a research perspective and within any setting teaching robots to automate human-based tasks such as industry.

5.2. Planning with Diffusion for Flexible Behavior Synthesis

In this paper [7], the authors present an approach called Diffuser to approximate trajectory optimization for various behaviors via the denoising component of a DDPM model. The high-level idea is to iteratively denoise a sample of random noise trajectories to fit the constraints existing usually as rewards for an environment. By giving just a start and end state, the Diffuser approach can fill in the blanks (inpainting). One interesting property of using a diffusion-based model is flexible conditioning. In particular, the denoising process can leverage a wide range of constraints such as gradient, ground truth, or reward to condition upon. Another key property difference relative to most RL-based approaches concerns the Markov Property, that the next step depends only on the previous. Many RL approaches are based on the Markov property in their progression. However, the diffusion approach blends trajectory sampling and planning together. This means that the next step of the diffusion model is dependent on both a future state and a prior state. While dynamics prediction follows the Markov property, decision-making and control in planning may not. Figure 7 expresses the diffusion process iterating back from the goal state. As such, the authors design their approach to predict all timesteps of a plan at once. Diffuser is then able to leverage concurrent sampling and planning processes for use on long-horizon trajectories.

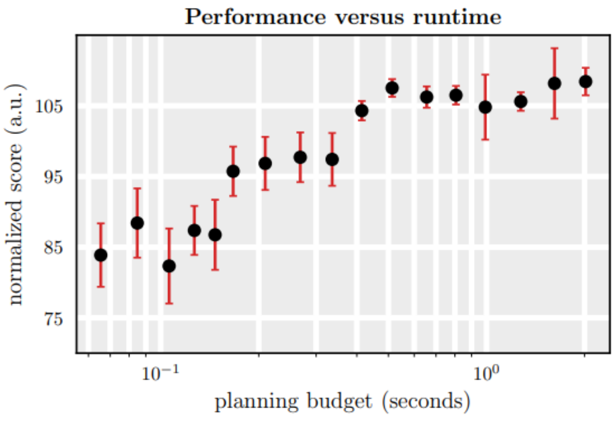


Figure 8. Results of Diffuser with experience warmup show slight improvements.

This property along with inpainting makes them better suited to sparse reward, goal-reaching settings where long horizons are common relative to traditional reinforcement learning algorithms. Despite these beneficial properties, diffusion-based models have a few downsides. Relative to [11] Diffuser requires substantially more time to compute an optimal solution due to the iterative approach. At each step of execution, a new plan must be generated. As a solution, the authors propose a warm-up process to reuse previously generated plans which does improve performance as seen in Figure 8. With diffusion models being a rather new architecture compared to many approaches in reinforcement learning, there is still much to go in terms of optimization. Caching previous experiences is a good approach that can likely be built upon substantially to optimize the diffusion process for the future.

5.3. Motion Planning Diffusion

The last paper concerns another diffusion-based approach called Motion Planning Diffusion (MPD) [2] specifically refined toward optimal motion planning. The authors leverage ground truth expert trajectories as priors to incorporate into a diffusion-based motion controller. Similar to Diffuser, the method starts with a set of randomized trajectories. Next, expert trajectories are used to train the diffusion model to iteratively plan random trajectories to be optimal. The expert trajectories are generated offline through conventional trajectory optimization methods, here the authors used RTT-Connect [9]. Finally, on inference, the denoising feature of the diffusion model is shown to be able to sample and bias a path to optimal regions of low cost. A visualization of the inference result is depicted in Figure 9.

The approach is very similar to Diffuser, except that it uses stronger reward conditioning in the form of expert trajecto-

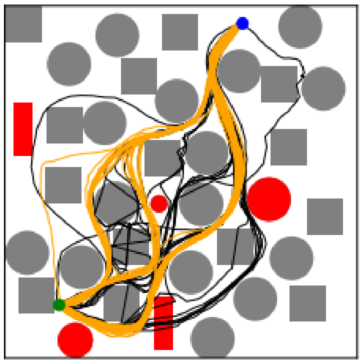


Figure 9. MPD inference visualization.

ries. While my initial goal was to perform a computational comparison of the two methods, I was unable to perform expert trajectory generation via RTT-Connect for the D4RL datasets. Theoretically, MPD should be quicker than Diffuser in solving the optimal path at runtime. But then again, it requires the generation of expert trajectories. However, the expert trajectories would only have to be computed once per environment. In comparison with [11], both Diffuser and MPD are significantly more computationally expensive requiring iteration of all trajectories to construct an optimal plan for each step. Furthermore, while there are benefits of relying on both prior and future as Diffuser and MPD do, it also requires knowledge about the end or goal states of the problem which may not always be available. This is in contrast to traditional reinforcement learning approaches that operate the Markov property of only the present and past rewards.

6. Numerical Comparison

A numerical comparison was performed between Diffuser and a more traditional reinforcement learning approach, Implicit Q-Learning (IQL) [8]. Regular Q-Learing is an RL algorithm that updates a behavior policy from actions originating from that policy or an unseen policy. Maintaining stability in sampling actions from the unseen policy while ensuring improvement is a key balance. IQL approximates the policy improvement step by estimating the best available action per given state via function approximation. This means IQL has been able to remove sampling from unseen policies while still achieving quality improvements. IQL has shown use in practice largely due to the simplicity of the model.

The experimental setup includes training and deploying Diffuser and IQL to infer an optimal path in four maze environments of varying difficulty from the D4RL dataset. Each maze contains a pointmass agent navigating from a

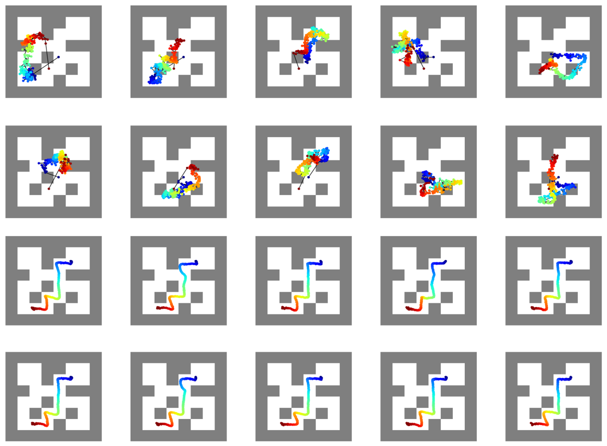


Figure 10. Sample training process of Diffuser. The top 2x5 grid represents trajectory results after only 1,000 iterations while the bottom grid is after 100,000. Diffuser optimizes randomly generated trajectories from a randomized start and end location into ideal paths.

preset start location to a target goal. Both IQL and Diffuser were run with hyperparameters specified by their respective papers for 250,000 iterations due to computation limits. Methods were compared in terms of normalized average reward for the environment on an Nvidia GeForce RTX 3090 GPU.

Despite the past success of IQL, in Table 1 it is clear that Diffuser outperforms IQL in all long-horizon planning tasks within the Maze2D environment. Yet the trade-off for this performance is a drastic increase in computation time. From training the models, IQL is significantly faster than Diffuser. But the results make sense. The iterative denoising approach for all trajectories at each step used in the Diffuser model achieves a significantly higher reward, yet is much slower than IQL. Additionally, the closer results between the two methods for the maze2d-open-v0 environment can be explained as that particular environment is simpler and does not require much long-horizon planning where Diffuser excels. A sample training image of Diffuser for the maze2d-medium-v1 environment is shown in Figure 10.

Table 1. Comparison of IQL and Diffuser.

Environment	Reward	
	IQL	Diffuser
Open	32.6	64.2
U-Maze	43.5	104.4
Medium	31.3	108.3
Large	52.3	109.8

In terms of conclusions and future directions, Diffuser, and other diffusion-based models, demonstrate promising results for motion trajectory planning, significantly outperforming IQL. While generalization was not evaluated in these experiments, I would estimate diffusion-based models also would perform reasonably well in new environments. As such, more experimentation needs to be done to determine if DDPM and other diffusion-based models are truly better than prior methods for generalized task transfer. Additionally, any useful, real-world application of diffusion-based planning models should surely consider the training time relative to simpler methods. More optimization and maturation of DDPM and other diffusion models is required.

References

[1] B. L. Bhatnagar, X. Xie, I. A. Petrov, C. Sminchisescu, C. Theobalt, and G. Pons-Moll. Behave: Dataset and method for tracking human object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15935–15946, 2022.

[2] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. *arXiv preprint arXiv:2308.01557*, 2023.

[3] T. Chen, A. Murali, and A. Gupta. Hardware conditioned policies for multi-robot transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[4] G. Feng, H. Zhang, Z. Li, X. B. Peng, B. Basireddy, L. Yue, Z. Song, L. Yang, Y. Liu, K. Sreenath, et al. Genloco: Generalized locomotion controllers for quadrupedal robots. In *Conference on Robot Learning*, pages 1893–1903. PMLR, 2023.

[5] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

[7] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

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

[9] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[10] X. Liu, D. Pathak, and K. M. Kitani. Revolver: Continuous evolutionary models for robot-to-robot policy transfer. *arXiv preprint arXiv:2202.05244*, 2022.

[11] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[12] T. Wang, R. Liao, J. Ba, and S. Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.

[13] Y. Wang, L. Herron, and P. Tiwary. From data to noise to data for mixing physics across temperatures with generative artificial intelligence. *Proceedings of the National Academy of Sciences*, 119(32):e2203656119, 2022.