Learned Dynamics Models for Robotic Fluid Manipulation

Michael Nath
Stanford University
Department of Computer Science
mnath@stanford.edu

Abstract

Many of the objects we interact with in the world are fluids — water, milk, air, etc. The world of robot learning has seen significant advances in fine-grained control of discrete solids, but it remains a challenge to endow robots with an intuitive understanding of fluid manipulation. As a novel attempt to impart robots with a learned physics model of a complex, multi-fluid system, this study aims to learn a dynamics model that takes in input RGB observations of the task scene and prescribes an action that progresses the robotic agent through the dynamics of the fluid environment, and to hopefully solve the manipulation task.

For this project, The cornerstone of our work is a goal-conditioned behavior cloning (GCBC) model, a goal-conditioned architecture that leverages both convolutional and multi-layer perceptron (MLP) layers. This model is designed to predict actions based on demonstrations generated by an exploration policy, enabling goal-oriented learning and adaptive behavior. A critical component in this setup is the agent's "myopia", or the forecast horizon for goal observation, which we can control using a hyperparameter. Through this element, we examine the implications of varying the temporal setting of the goal observation during training, allowing for a detailed analysis of behavior cloning in different temporal contexts. Our results show that agents conditioned on next-step goals are able to achieve a higher likelihood of cloning success, offering interesting insights into how the prediction horizon impacts learning and performance.

Accompanying the GCBC model is the creation of the first trajectory dataset for the `LatteArt-v0` *task. The data is generated through a dedicated simulator provided by FluidLab, a test-bed for fluid manipulation tasks featuring multi-fluid systems with a simulation engine that accurately mirrors real-world fluid dynamics. Using a combination of random and correlated noise policies, we investigate datasets that encompasses a broad spectrum of fluid particle arrangements. This extensive, diverse dataset acts as the bedrock of our models, providing a comprehensive foundation for training and evaluating our goal-conditioned agents. It also offers an open-ended resource for future studies in fluid manipulation and other physically complex tasks.*

To further augment the capabilities of our model, we investigate the incorporation of Variational Autoencoders (VAEs) within the GCBC framework, yielding a new model variant - the `GCBC-VAE`*. This agent learns to construct a more condensed and manageable representation of the fluid particles' arrangements, thereby enhancing the agent's ability to localize the goal trajectories. The* `GCBC-VAE` *agent shows a marked improvement in terms of trajectory localization compared to the base GCBC model, revealing the potential of integrating latent representations into goal-oriented learning.*

While these achievements advance our understanding of goal-oriented behavior in complex tasks, our research also uncovers areas requiring further exploration. One of these is the quality of the trajectory dataset, a factor that can significantly impact the performance of behavior cloning models. The existing dataset, albeit diverse due to the random trajectories, may lack the precise control required to create realistic artwork. Consequently, models trained on this data struggle to master the fine-grained manipulations necessary for creating intricate patterns. Future research could address this by focusing on creating expert trajectories that prioritize artistic patterns, thereby enabling more effective learning from high-quality demonstrations.

Moreover, there is an opportunity to refine the GCBC models to improve long-horizon predictions, which would enhance their ability to perform more complex tasks. Potential improvements could involve architectural modifications, such as adjusting the number of hidden units or layers, or altering activation functions. Another intriguing direction could involve designing models that operate directly in the latent space provided by the VAE. Such models could potentially yield more accurate long-horizon predictions, offering a promising area for future research.

1. Introduction

1.1. The Difficulty of Fluid Manipulation

Developing intuitive physical models for robotic fluid manipulation presents an intricate challenge due to the inherent complexity of fluid systems. These systems, characterized by hundreds of thousands of interacting particles, exhibit emergent properties such as fluid viscosity, which are difficult to interpret from a snapshot image. The transition from simulated to real-world learning (sim2real) introduces additional complexities, as fluids are more susceptible to natural world disturbances than solid objects. To address these issues, we need a robust learning model capable of 1) compactly representing numerous fluid particles, 2) accurately depicting complex multi-fluid dynamics, and 3) demonstrating adaptability to real-world variability. The necessity of overcoming these obstacles is underscored by the broad potential benefits, from increasing efficiency in industries like manufacturing to enhancing precision in medical procedures, thereby driving significant scientific and industrial advancements.

1.2. FluidLab and the `LatteArt-v0` Task

To provide realistic multi-fluid systems to simulate complex fluid manipulation environments, Xian et al. released FluidLab [7]. Although FluidLab possesses a myriad of practical, yet interesting manipulation tasks, one in particular is chosen to be the environment of choice for this project: the latte-art making task, identified as the `LatteArt-v0` environment. The environment setting is as follows (visuals are provided in Figure 1): the scene is initialized with a coffee mug that is already filled with brown fluid representing coffee. Then, a white injector is positioned directly, albeit randomly, above the coffee mug. Wherever the injector is, it drops white particles representing foam directly below and onto the latte. The foam then mixes with the cofee particles, upon which the underlying fluid engine of FluidLab kicks in and simulates any interactions and emergent behavior of the coffee and the foam.

The task setting is as follows: at the beginning of each episode, a desired configuration of the foam particles on the coffee is given to the environment. We can think of this as the desired artwork that we wish our agent to draw out using the foam injector. Then, within the horizon H of the episode the agent is tasked to move the foam injector such that by the episode's termination, the configuration of the injected foam particles and the coffee particles match closely to the input desired configuration. In other words, the agent has made the desired latte art.

1.3. Goal-Conditioned Behavior Cloning (GCBC)

Goal-conditioned learning methodologies in deep reinforcement learning (DRL) open up promising possibilities

(a) $t = 300$ (b) $t = 315$ (c) $t = 330$

Figure 1: A pre-programmed optimal trajectory from FluidLab that produces a clef. We see the foam injector moved around across time to delicately draw out the artwork.

to address the intricate problems identified in fluid manipulation tasks. These methodologies condition an agent's policy on the present state of the environment as well as a specified goal state. This alignment with our `LatteArt-v0` task, which sets a unique target configuration of foam particles for each episode, empowers the model to adapt its policy dynamically in response to the changing objectives in every episode.

An insightful extension to goal-conditioned learning is proposed by Ding et al. in their work titled "Goal-conditioned Imitation Learning" [1]. They demonstrate that the integration of expert demonstrations into reinforcement learning can expedite the learning process and improve policy efficiency, especially in contexts with complex state spaces. These demonstrations act as a valuable guiding signal, steering the agent towards effective behaviors, thus effectively initializing the agent with a reasonably good policy. This mitigates the steep learning curve often observed in DRL, particularly beneficial in our fluid manipulation task, which is marked by inherent complexity. The method outlined by Ding et al. also shows efficacy even when expert demonstrations lack action sequences, leveraging third person or kinesthetic demonstrations.

1.4. Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) [3] have emerged as powerful generative models that can capture complex data distributions and learn latent representations of input data. In the context of computer vision, VAEs have been successfully applied to tasks such as image generation, data synthesis, and anomaly detection. VAEs consist of two main components: an encoder network and a decoder network. The encoder network takes an input data point and maps it to a latent space representation, typically modeled as a multivariate Gaussian distribution. This latent space representation is then used by the decoder network to reconstruct the original input data. The encoder and decoder networks are jointly trained using a combination of a reconstruction loss, which encourages the decoder to generate outputs that resemble the input data, and a regularization term, often

based on the Kullback-Leibler (KL) divergence, which encourages the latent space distribution to approximate a predefined prior distribution, typically a standard Gaussian.

We believe that a VAE can augmenet our goal-conditioned behavior cloning model and tackle the issues outlined in Section 1.1. Although there are thousands of particles (115,480 in the `LatteArt-v0` environment) in the scene, it is not necessary to distinguish each and every particle. This is because multi-fluid emergent behaviors are a consequence of indeed thousands of particles, not just a couple. In this task, the responsibility of the encoder component of a VAE would aim to compactly represent the scene in a latent space. This latent space may pick up on the most salient features of the scene, which may include intuitive observations such as the presence of the gray coffee mug, the position of the foam injector, and distinct clusters of particles.

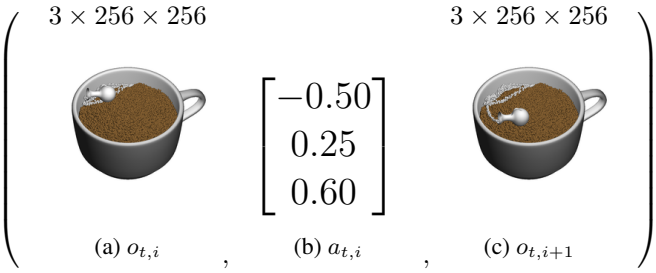
2. Related Work

Solid object manipulation has been a core focus of robot learning, leading to advances such as the works of Zhu et al. [8] on dexterous manipulation and Mahler et al.'s Dex-Net project [4]. However, due to the intricate nature of fluid interactions, direct application of these methods to fluid systems has proven challenging. Recent work by Finn et al. [2] on deep spatial autoencoders showcase the power of learning compact representations for control tasks. These works provide the basis for our approach, which focuses on utilizing Variational Autoencoders (VAEs) [3] for learning fluid dynamics. To the best of our knowledge, this represents a novel exploration of using VAEs in the context of fluid manipulation tasks.

In terms of learning dynamics models for control, the work of Watter et al. [6] on Embed to Control (E2C) presented a similar approach of learning a latent representation and a dynamics model simultaneously. Our work diverges in the complexity of the system being modeled — a fluid system versus solid object manipulation. Perhaps most similar is the work of Xian et al. on their development of FluidLab. Despite the shared environment, their focus did not extend to the specific `LatteArt-v0` task nor the integration of VAEs to learn fluid representations. Thus, our work uniquely combines the idea of learning latent representations using VAEs with a learned dynamics model for controlling a robotic manipulator in fluid environments. In doing so, we add to the growing body of knowledge on data-driven approaches for complex physical system manipulation, specifically in multi-fluid systems.

3. Data

To the best of our knowledge, there **does not exist** any dataset of episode trajectories for any of the tasks present in FluidLab. The process of constructing this dataset from scratch, primarily from random trajectories in the `LatteArt-v0` environment, formed a substantial part of this project's groundwork. Formally, we have initially generated a dataset $\mathcal{D}$ of $N = 2500$ trajectories where each trajectory t is a collection of $H = 330$ tuples $(o_{t,i}, a_{t,i}, o_{t,i+1})$ as visualized in Figure 2. Here, $o_{t,i}$ has dimension $(3 \times 960 \times 960)$ and represents the RGB observation of the scene at the ith timestep in trajectory t. $a_{t,i}$ has dimension (3) and represents control values inputted to the foam injector at the ith timestep in trajectory t to produce $o_{t,i+1}$. We choose to generating random trajectories so our models can be trained on a significantly encompassing distribution of scene observations. Intuitively, if N and H are sufficiently high, then the random injector actions should generate a diverse distribution of particle states, which allow our autoencoder to learn a robust representation of the scene. For clarity, we can partition $\mathcal{D}$ into $\mathcal{D}_o$ and $\mathcal{D}_a$, where $\mathcal{D}_o$ only contains image observations and likewise $\mathcal{D}_a$ only contains actions.



(a) $o_{t,i}$, (b) $a_{t,i}$, (c) $o_{t,i+1}$

Figure 2: An example entry in $\mathcal{D}$.

In total, our dataset $\mathcal{D}_o$ contains $330 \times 2500 = 825000$ image observations of dimension $3 \times 960 \times 960$. This presented a significant strain on the storage resources provided by the computing environment hosting this project, which called for the following preprocessing of the data. First, image observations were down-sampled to be of dimension $3 \times 256 \times 256$. Next, for the sake of easier model learning the pixel values of all the images were normalized to be in the range $[0, 1]$. Finally, to fit as many trajectories in our computing environment's disk space as possible, the precision of the image observations were reduced from being 64 bit precision to instead being 32 bit precision.

4. Methods

4.1. Leveraging GCBC

Our approach is motivated by the success of Ding et al.'s methodology, extending these principles to a goal-conditioned setting for fluid manipulation tasks. Traditional imitation learning techniques focus on replicating expert actions without explicitly considering the changing goal

states, limiting their ability to generalize beyond the demonstrations. Our approach aims to overcome these limitations by conditioning on both current and goal states, which equips the agent to construct meaningful representations of fluid particles and their dynamic interactions.

We posit that our goal-conditioned imitation learning approach can capture the three capabilities highlighted earlier: compact representation of fluid particles, accurate depiction of complex multi-fluid dynamics, and adaptability to real-world variability. Furthermore, our agent, trained from a diverse range of expert demonstrations, is expected to be robust against real-world variability, as these demonstrations provide a rich array of scenarios potentially encountered in actual deployment.

4.2. Trajectory Generation

Section 3 discusses the contents of $\mathcal{D}$, any required sanitization steps, and a overview of the generation mechanism yielding $\mathcal{D}$. In this section we detail and formalize the trajectory generation process.

To begin, we construct an exploration policy $\pi_{\text{exp}}(\cdot|o_{t,i})$ that will serve as the main mechanism for choosing $a_{t,i}$ given $o_{t,i}$. Since for this project we have decided to create random trajectories, we thus leverage a random Gaussian policy $\pi_{\text{Gauss}}(\cdot)$ invariant to $o_{t,i}$. Concretely, at timestep i for trajectory t, we sample $a_i \sim \mathcal{N}(\mu, \Sigma)$ where $\mu \in \mathbb{R}^3$ is a vector of zeros and $\Sigma \in \mathbb{R}^{3\times3}$ is an identity covariance matrix. Although μ and Σ are technically hyperparameters, we affix them with the above traditional values for all experiments conducted with $\mathcal{D}_a$.

Although π_{Gauss} is effective for rapidly generating trajectories encompassing a wide distribution of particle arrangements, the resulting $\mathcal{D}$ is not realistic for many desired artworks. Figure 3 visualizes a sequence of image observations induced by actions sampled from π_{Gauss}. What we observe is that the motion of the foam injector is jittery, and there's no correlation between where the injector was at timestep i and where it was at timestep $i+1$. Since the performance of behavior cloning is significantly dependent on the dataset it is learning from, and since desired artworks generally have tightly correlated action sequences (e.g. Figure 1), it is in our best interest to augment π_{Gauss} with correlatedness. Thus, we yield π_{Corr}, a correlated noise policy described by Nagabandi et al. [5].

π_{Corr} operates by maintaining an exponential moving average of $a_{t,i}$ where each term to be averaged is noise sampled from a Gaussian having zero mean and unit covariance. In other words:

$$n_{t,i} \sim \mathcal{N}(\mathbf{0}, I_3)$$
$$a_{t,i} = \beta \cdot a_{t,i-1} + (1-\beta) \cdot n_{t,i}$$

(a) t = 30 (b) t = 40 (c) t = 50

Figure 3: Sample trajectory from a rollout of π_{Gauss}. Actions are taken independent of the previous ones, resulting in purely noisy arrangements of fluid particles.

(a) t = 30 (b) t = 40 (c) t = 50

Figure 4: Sample trajectory from a rollout of π_{Corr} under $\beta = 0.95$. Actions are related to the previous ones via an exponentially moving average. This manifests in the agent "following" a path, and producing a low intra-trajectory action variance.

We can think of $\beta \in [0, 1]$ as the correlation coefficient, a hyperparameter that controls how tightly correlated we want our action sequences to be for a trajectory in $\mathcal{D}_a$. A setting of $\beta = 1$ qualitatively holds the foam injector still since every action is just the initial action, and a setting of $\beta = 0$ recovers π_{Gauss}. Figure 4 visualizes another sequence of image observations where this time the underlying actions are sampled from π_{Corr}. We observe that the actions are quite correlated across time, and thus with this π_{Corr} as π_{exp} we carry on with creating $\mathcal{D}$ as described in Section 3.

4.3. GCBC Model Behavior

There are two key components of the GCBC architecture: the **Conv** layers and the **MLP** layers. The architecture is visualized in 5. Adhering to the principles of goal-conditioned reinforcement learning, the model is first fed the current observation $o_{t,i}$ as well as some goal observation $g_{t,i}$. Since $\mathcal{D}_o$ contains observations of random arrangements of fluid particles, it is not immediately what the goal observation is. For this project, we choose *another* image observation $o_{t,j}$ $(j > i)$ in the trajectory to be the goal observation. We pretend that this observation in our trajectory is the goal we have been trying to reach all along. The intuition is that if our GCBC model can learn the actions needed

to take the scene from having one arbitrary arrangement of fluid particles to another somewhat arbitrary arrangement of particles, then we cover a vast distribution of "goals". When an image corresponding to a desired artwork is fed into the model as an actual goal, we can hope that aspects of that goal has been seen during the behavior cloning process and so the model can pull from that knowledge to take actions.

The output of the GCBC model is a vector of independent means and standard deviations which parameterize a Gaussian for each action component. The model is trained to find such parameters that maximize the likelihood of the expert actions. During a policy rollout, the model will choose an action whose components are sampled from these independent Gaussians.

4.3.1 Conv Layers

Once the model is given the current observation and the goal, it then concatenates the two images along the channel dimension, producing an input of dimension $6 \times 256 \times 256$. This input is then processed by the **Conv** layers, which perform a series of 2D convolutions on the image with varying kernel sizes. As per good deep learning practice, each convolution is followed by a batch normalization layer before being processed by a ReLU non-linearity. The last convolution is not followed a ReLU activation. Once the **Conv** layers are done, we yield an output having dimension $64 \times 84 \times 84$.

4.3.2 MLP Layers

To transform this intermediate output into a vector of means, we have it processed by the **MLP** layers. This is a fairly standard stack of linear layers each followed by batch normalization and ReLU non-linearity. Like with the **Conv** layers, the last linear linear is not followed by a ReLU activation. Once the **MLP** layers are done, we finally have our vector of means.

4.4. Incorporating Latent Representations

Leaving this GCBC model aside, which we notate as `GCBC`, we also explore the influence of latent representations induced by a trained VAE on top of a learning GCBC model. We call this model `GCBC-VAE`. As opposed to `GCBC`, which works with the RGB observations of the scene, `GCBC-VAE` first encodes the observation and goal down into latent space according to the trained VAE, and the behavior cloning network works with the compact representation to predict the next action to take. By deploying the VAE in conjunction with this model, we can observe how well the VAE enables the dynamics models to progress the environment and achieve the desired goal. Ultimately, the success of the task at hand, such as creating latte art in the `LatteArt-v0` task, hinges on the ability of the VAE to provide informative and meaningful latent representations. Through these experiments, we will gain crucial insights into the efficacy of the VAE and its role in solving the task effectively. The architectural details of the VAE model is out of scope for this report.

4.5. Evaluating the GCBC Models

4.5.1 Validation Rollouts

Once our behavior cloning networks are trained, we validate the models on a collection of recorded trajectories for which we calculate the mean-squared error (averaged across the H timesteps and T_{val} trajectories) between the action vector predicted by our model, and the action vector actually taken under π_{exp}. Since we need to be observing the same scene during validation as our exploration policy did during trajectory generation for a proper evaluation, we need to have stored the low-level simulator state underlying each $o_{t,i}$. Thus, we have also constructed $\mathcal{D}_{\text{val}}$, a dataset of T_{val} trajectories where each entry is a quadruple $(o_{t,i}, a_{t,i}, o_{t,i+1}, s_{t,i})$. Here $s_{t,i}$ refers to the low-level simulator state of the FluidLab simulation engine, which can be retrieved by accessing the API of the environment. Another API call can be made that takes in $s_{t,i}$ and resets the state of the simulation to match the state articulated by $s_{t,i}$. For the `LatteArt-v0` task, $s_{t,i}$ holds state information for $115,480$ particles and due to the storage limitations of our computing environment, $|\mathcal{D}_{\text{val}}| = 50$.

4.5.2 Test Rollouts

Once the models are validated, we move on to rolling them out as policies to hopefully solve the `LatteArt-v0` task. During the environment initialization, we store the inputted desired artwork. When executing our GCBC policy, we feed the current image observation and our stored goal observation. We take the action prescribed by the policy and compile the image observations into a movie, which we view once the rollouts are over.

4.6. Experiments

4.6.1 Ease of Cloning and Agent Myopia

For the first set of experiments, we are interested in analyzing the difficulty of behavior cloning for different temporal settings of the goal observation $g_{t,i}$ during training. We call this placement of the goal observation **g** steps into future the "myopia" of our GCBC agent. The relationship between g and the goal observation is:

$$g_{t,i} = o_{t,i+\mathbf{g}}$$

For example, if we want to train our GCBC models to be trivially myopic $\mathbf{g} = 0$ then we simply set $g_{t,i} = o_{t,i}$. If we want our agent to be conditioned on preparing for the

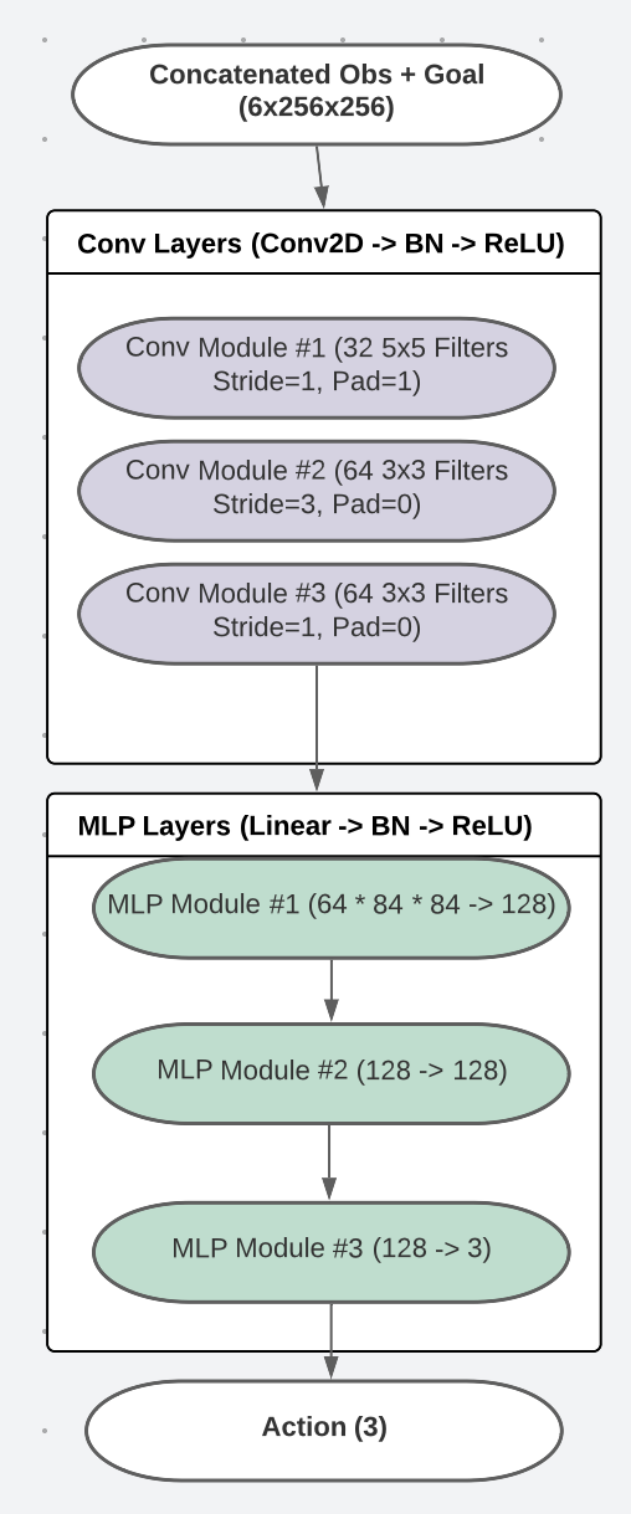


Figure 5: GCBC architecture consisting of **Conv** and **MLP** layers.

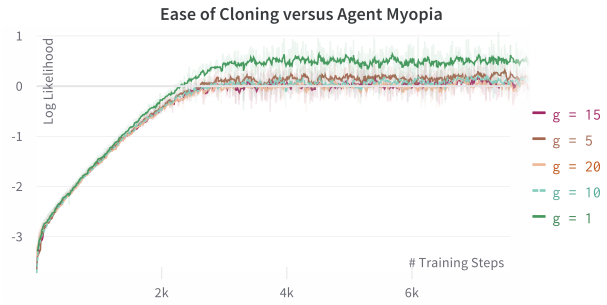


Figure 6: The difficulty of behavior cloning for various settings of **g**. We quantitatively observe the model having a greater ease of cloning when the model is conditioned on next-step goals. For further goals, it is not conclusive but it appears it is equally difficult.

next-step observation ($\mathbf{g} = 1$), we set $g_{t,i} = ot, i + 1$. So, in this fashion **g** becomes a hyperparameter of our model. "Ease of cloning" in this context refers to the likelihood the GCBC models can achieve when fitting their predicted distributions to the actions taken by π_{exp}. Figure 6 showcases this difficulty for various settings of **g** across training time. These experiments were done under a setting of $\beta = 0.15$ so that it is sufficiently challenging for the model to adapt to the dataset (see Section 4.6.2). We notice that for an agent predicting only considering the next image observation, the model seems to achieve greater likelihood. For further settings of the goal, we see a decrease in the likelihood, although the differences are not significant between the far-away settings.

4.6.2 Ease of Cloning and β Setting

[The other hyperparameter of interest in this project is the correlation coefficient β. As a natural step, we investigate the influence of intra-trajectory variance in the actions taken on the ease of cloning. Figure 7 showcases this influence for two extreme values of β: $\beta = 0.05$ and $\beta = 0.95$. Here, we observe a significant difference in ease of cloning by our model between the two settings. When the variance of the actions taken are low ($\beta = 0.95$), our GCBC models have a significantly easier time fitting the Gaussian distributions to the dataset. When the actions are highly de-correlated, our model achieves a smaller likelihood.

4.6.3 `GCBC` and `GCBC-VAE` Performance

The main experiment that we have run is evaluating the trajectories produced by our various models for a test desired artwork, compared to a baseline random policy. The artwork of interest is a checkmark centered at the left half of the coffee, and is designed to be produced in approximately

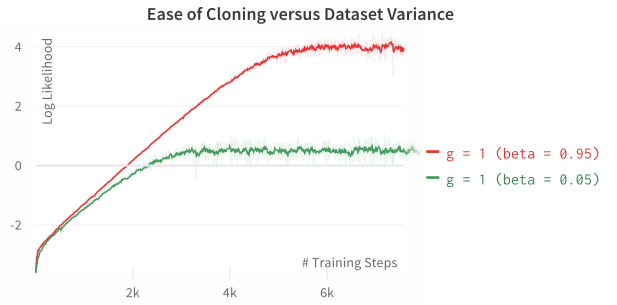


Figure 7: The difficulty of behavior cloning for two extreme settings of β. When the dataset consists of highly-correlated actions, we observe that our behavior cloning network has an easier time adapting to the dataset (as indicated by the relatively large likelihood values). When the dataset consists of loosely correlated actions, our network has more difficult fitting to the greater variance in the trajectory.

50 timesteps. Figure 8 visualizes the performance of all the agents, where all agents learned from a dataset created under a setting of $\beta = 0.95$. We critically note that none of the agents successfully solve the `LatteArt-v0` task and draw out the desired artwork. With that being said, there are a few observations we can make about the trajectories that these agents have made. For example, we observe that for all the agents employing a behavior cloning network, the agent appropriately follows the stroke direction implied by the checkmark artwork. All GCBC models start their foam injector at a reasonable, perhaps signaled by the location of the injector in the goal observation image. We notice that there is little difference in the trajectory produced between the myopic ($g = 1$) agent and the agent trained to predict considering what may happen 50 timesteps later. We do observe a difference between the VAE-incorporated `GCBC-VAE` agent and the non-VAE `GCBC` agent. Specifically, we notice that the robot shows a behavior of localizing the artwork within the coffee. This is deduced from the the placement of much of the foam particles clustering on the left half of the coffee, where the desired artwork is also concentrated in the goal observation image.

4.7. Analysis and Discussion

4.7.1 Quality of Trajectory Dataset

To reiterate, the performance of any behavior cloning model hinges on the quality of the dataset being used to train the model. Our experiments in Sections 4.6.1 and 4.6.2 illustrate this fact in terms of training difficulty, but this could also underscore the poor trajectories produced by these models during test time. We suspect that the intuition that has encouraged us to produce a dataset of random trajectories, although does cover a vast distribution of fluid arrangements, falls short of emphasizing the subset of fine-grained trajectories that would produce the kind of arrangements that would be typically considered "artwork". Thus one possible direction for future work is to compile trajectories involved in producing realistic, artwork. We hypothesize that this would bolster the performance of our GCBC

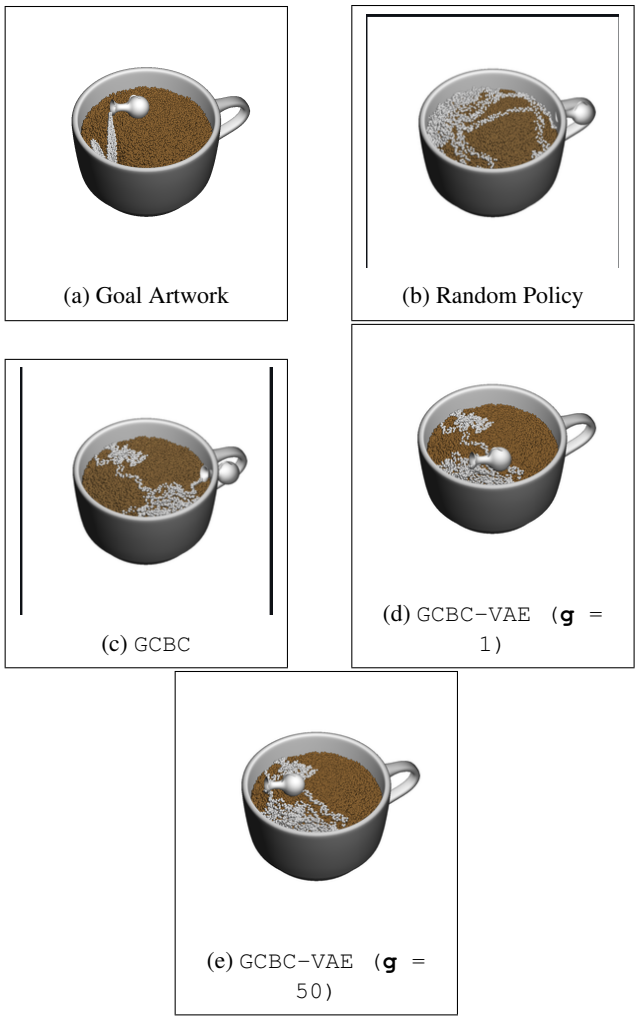


Figure 8: Resulting "checkmark" produced after 50 steps according to various control policies. The arrangement in (a) represents a checkmark and is fed as input into 3 policies: that of `GCBC-VAE`, that of `GCBC`, and a random policy. The random policy in (b) disregards the goal and produces a jittery trajectory that results in a random arrangement of particles. `GCBC` in (c) figures out a reasonable starting location for the foam injector, but escapes the region it should keep the foam injector. `GCBC-VAE` extends `GCBC` and somewhat localizes the foam particles to match the input checkmark location. `GCBC-VAE` in (d) and (e) show that training under long-horizon prediction manifests in little to no difference in the trajectories produced.

models since they would be learning from effectively expert trajectories.

4.7.2 GCBC Models and Realistic Representations

Our experiment with `GCBC-VAE` show that there is much room for improvement when it comes to deploying our VAE on a model to perform control. Although our VAE does aid in producing a more localized trajectory, it is nevertheless trained on a dataset of random trajectories. Thus, the advantages provided by a VAE are dampened by learning from these random trajectories that do not exhibit salient features. The performance of our `GCBC-VAE` model is then bottlenecked by both the quality of the trajectory dataset and the quality of the latent representations induced by the VAE, which itself is trained from the trajectory dataset. If it is not possible to obtain a more realistic trajectory dataset, another next step involves simply experimenting with the architecture of the GCBC models, modifying things such as the number of hidden units, layers, and activations used. It may be worthwhile to investigate using dynamics models that operate directly in latent space, as that might allow for more accurate long-horizon predictions (settings where $\mathbf{g}$ is sufficiently large). It would be an interesting investigation to compare the trajectories produced by such a model with the goal-conditioned behavior cloning networks that work with raw image data.

4.8. Conclusion

In this research, we propose a goal-conditioned deep reinforcement learning approach for fluid manipulation tasks. We contribute a new trajectory dataset whose generation combines random and correlated noise policies to produce diverse arrangements of fluid particles. Our GCBC models, utilizing convolutional and multi-layer perceptron layers, predicts actions based on expert demonstrations. Additionally, we explore the incorporation of a VAE within the GCBC framework. The `GCBC-VAE` model, operating in a compact latent space, exhibits improved localization of foam particles, indicating the potential of incorporating latent representations for enhancing performance. However, our findings also shed light on the challenges and limitations of our approach. We identify the quality of the trajectory dataset as a critical factor affecting model performance, emphasizing the need to gather trajectories specifically focused on realistic artwork. Furthermore, we suggest further investigation into architectural modifications and dynamics models operating directly in latent space to improve long-horizon predictions.

4.9. Acknowledgements

This project could not have been made possible without the high-level discussions I have had with Stephen Tian (`stian`) about interesting experiments to run. I greatly appreciate the Stanford Vision and Learning Lab for providing a computing environment to run the experiments.

References

[1] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel. Goal-conditioned imitation learning, 2020. 2

[2] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning, 2016. 3

[3] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2022. 2, 3

[4] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics, 2017. 3

[5] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation, 2019. 4

[6] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images, 2015. 3

[7] Z. Xian, B. Zhu, Z. Xu, H.-Y. Tung, A. Torralba, K. Fragkiadaki, and C. Gan. Fluidlab: A differentiable environment for benchmarking complex fluid manipulation. In *International Conference on Learning Representations*, 2023. 2

[8] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost, 2018. 3