Learned Latent Representations for Robotic Fluid Manipulation

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

Stephen Tian (Non-CS231N Contributor)
Stanford University
Department of Computer Science
stian@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 good intermediate representations of fluids off RGB observations of the particular fluid task.

For this project, we investigate training a Variational Autoencoders (VAEs) from scratch for a fluid manipulation task, specifically the `LatteArt-v0` *latte-art making task. The VAE is trained to learn compact latent representations of fluid scenes and is evaluated in terms of reconstruction quality and its integration with a goal-conditioned behavior cloning model (*`GCBC-VAE`*).*

*We analyze reconstructions provided by the VAE compared to a baseline mean image of our custom trajectory dataset. We show that the reconstructions are practically indistinguishable from the mean image, indicating that the VAE has learned the invariant properties of each image but not much more. We exponentially back-off the influence of the KL-divergence loss term, and show the reconstructions gradually deviate from the mean image, demonstrating the impact of latent regularization on the reconstruction similarity. Additionally, varying the latent dimensionality (**dim**(z)) shows negligible differences in reconstruction error, but higher dimensions allow for more nuanced reconstructions and deviations from the mean image.*

Using the VAE in the `GCBC-VAE` *model shows promise in localizing latte artwork. However, generating realistic and intricate designs remains challenging due to our trajectory generation scheme. We highlight a tradeoff between compiling unrealistic trajectories covering a wide distribution of fluid arrangements and focusing on realistic but cumbersome trajectories as a next-step investigation to be conducted.*

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 [5]. 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

(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.

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. Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) [2] 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.

2. Related Work

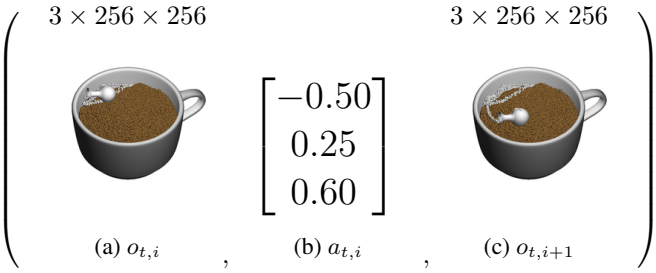
Solid object manipulation has been a core focus of robot learning, leading to advances such as the works of Zhu et al. [6] on dexterous manipulation and Mahler et al.'s DexNet project [3]. However, due to the intricate nature of fluid interactions, direct application of these methods to fluid systems has proven challenging. Recent work by Finn et al. [1] 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) [2] 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. [4] 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 VAE 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. Since training a VAE does not require knowledge of the action control values, we can partition $\mathcal{D}$ into $\mathcal{D}_o$ and $\mathcal{D}_a$ and focus only on $\mathcal{D}_o$, where $\mathcal{D}_o$ only contains image observations and likewise $\mathcal{D}_a$ only contains actions.

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



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

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

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 a VAE

We believe that a VAE is a good model choice to tackle the issues outlined in (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.

4.2. VAE Architecture

Figure 3 details the VAE architecture used for this project. There are 3 components of interest: the encoder layers, the latent layers, and the decoder layers.

4.2.1 Encoder

First, a RGB observation o is inputted to the VAE, where it is immediately processed by the encoder. The encoder is a stack of four nearly identical modules, differing only in the parameterization of the internal layers. A given encoder module first applies a convolutional layer on the input, with the kernel size depending on where the module is situated in the broader encoder. Smaller kernel sizes were used for the earlier modules in order to capture local spatial filters, and since the receptive field becomes larger as more convolutions are applied, the kernels in the later modules were given greater size to more effectively pick up on more global features. The convolutional layer is followed by a batch normalization layer, which is generally known to be good practice for more stable optimization. Finally a ReLU activation is included in all modules except the last one, so as to provide non-linearity in the encoder.

4.2.2 Latent Layers

Once the encoder has processed the input o, we get some encoded representation z of the scene. However, it is not enough to send z to the decoder, because we are chiefly interested in having the decoder be expressive and handle a distribution of latent vectors related to the input o. Therefore, the role of the latent layers is to map z to a mean vector μ and covariance Σ that capture the latent distribution of z given the input o. At this point, we may produce $\bar{z}$ by sampling directly from this multivariate Gaussian parameterized by μ and Σ, which would indeed challenge the decoder to handle a distribution of z. However, direct sampling is difficult to optimize via backpropagation and so instead we follow the reparameterization trick described by [2] where first we sample some noise $n \sim \mathcal{N}(0, 1I)$ and then scale it up by Σ and add by μ to obtain an equivalent sample. This finally produces $\bar{z}$, a compact, perturbed representation of the scene, that is then passed to the decoder.

4.2.3 Decoder

Immediately after $\bar{z}$ is passed to the decoder, a linear layer projects the latent vector out to a flattened representation of some image. Then, this flattened representation is reshaped to be $3 \times d \times d$ image where d is a hyperparameter. Similar to the encoder, the decoder is made up of a stack of "decoder" modules. Each decoder module is made up of a tranposed convolution (which upsamples the image), a batch normalization layer, and a ReLU activation. We note that the number of decoder modules depends on the both the value of d and the upsampled resolution induced by each transpose convolution. For example, if $d = 32$ and we wish to have our image upsampled twice from each tranpose convolution to ultimately get back to our input image height/width of 256, then we can expect $\log_2(256/32) = 3$ decoder modules. Once these layers are applied, we get back a resulting image $\bar{o}$ which should be as similar (pixel-wise) to the o as possible.

4.3. Evaluating the VAE

4.3.1 Training/Validation

The VAE is trained to learn the minimize the following loss:

$$\arg\min_{\theta} \text{MSE}(o, \bar{o}) + D_{KL}(q_\theta(z|o)||p(z))$$

This is a loss comprised of a reconstruction error — a mean squared error between o and $\bar{o}$ — as well the KL divergence error between the latent distribution q induced by

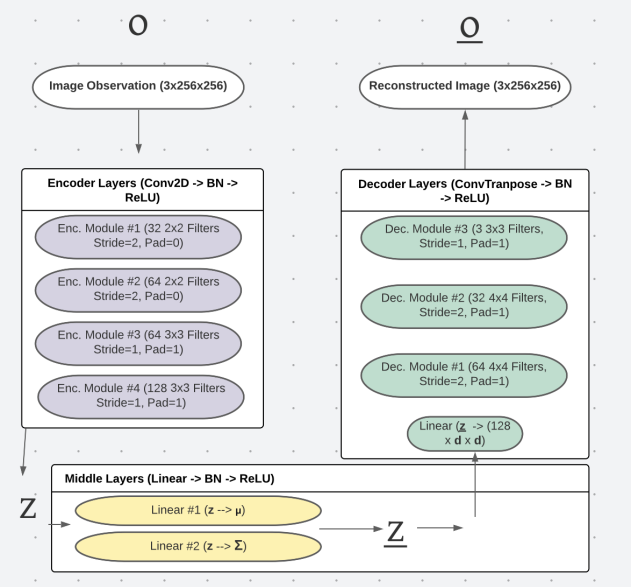


Figure 3: VAE architecture consisting of an encoder, latent layers, and a decoder.

the VAE and the prior distribution p, which for this study is a multivariate Gaussian.

4.3.2 Baselines

There are two baselines representation mechanisms that we have decided to use for this project. One very naive baseline is a representation created by deriving the mean image of all observations $o_{t,i}$. The intuition here is that the mean image should almost perfectly capture invariant features of all the observations, namely the coffee mug and the white background. However, since the mean image would be computed on a dataset of random trajectories, it would fall short of conveying anything meaningful about the foam particles interacting with the coffee particles. This is shown in Figure 4b, where we observe a blanket of foam particles in no peculiar arrangement. This baseline would help us determine if the VAE can pick up on **at least** the same features that the mean image picks up on.

The second baseline is an autoencoder similar to the VAE, except it is trained to minimize an objective consisting of only the reconstruction loss. In other words, it is not regularized. This baseline could be used as part of an ablation study to determine the importance of the Gaussian prior regularization. On a quantitative level, we can observe the difference in the mean squared error achieved by the VAE and the non-variational baseline. On a qualitative level, we can compare the reconstructions for a given image.

4.3.3 Deploying on Dynamics Models

A intermediate representation provided by an encoder such as a VAE is best understood by deploying a dynamics that operates in latent space — a latent dynamics model. For the last set of evaluations, we incorporate our trained VAE on a model that works towards solving the `LatteArt-v0` task: a goal-conditioned behavior cloning model `GCBC-VAE`. The architectural details of this two model is out of scope for this report, but a summary of the model's behavior is as follows:

`GCBC-VAE` takes as input the current observation $o_{t,i}$ **and** some "goal" image observation (perhaps an image of the desired artwork), and outputs the next action $a_{t,i}$. Ideally, $a_{t,i}$ should progress the dynamics of the environment and result in a new observation $o_{t,i+1}$ that is closer to that goal observation. Internally, `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 `LatteArt-v0`, 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.

4.4. Experiments

4.4.1 VAE and Mean Image

For the first set of experiments, we are interested in analyzing the reconstructions provided by our VAE, and comparing it to the mean image of the trajectory dataset. For this experiment, the VAE operates with latent vectors having dimension 32, $d = 32$, and the VAE is trained for 15 epochs with 250 iterations each and a batch size of 64. Mean images are computed by taking batches of 64 images across random timesteps and trajectories, and taking an average across the batch dimension.

Putting the reconstructions side by side with the mean image (Figure 4), we observe that the two are practically indistinguishable from one another. This is reassuring in that at least our VAE has learned the invariant properties of each image (as described in Section **4.3.2**), but so much so that it has effectively learned to reconstruct to the mean image.

(a) + KL-Term (b) Mean Images (c) - KL-Term

Figure 4: The various reconstructions provided by the two autoencoders, compared to the mean image. Incorporating a KL-divergence loss term regularizes the VAE towards the mean of the dataset, encouraging it to reconstruct to the mean image. No KL-term encourages the autoencoder to discriminate each image sample.

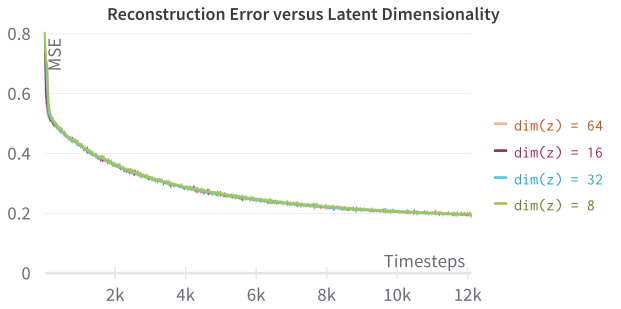
(a) MSE Loss

(b) Weight = 1 (c) Weight = 0.01 (d) Weight = 0.001 (e) Weight = 0

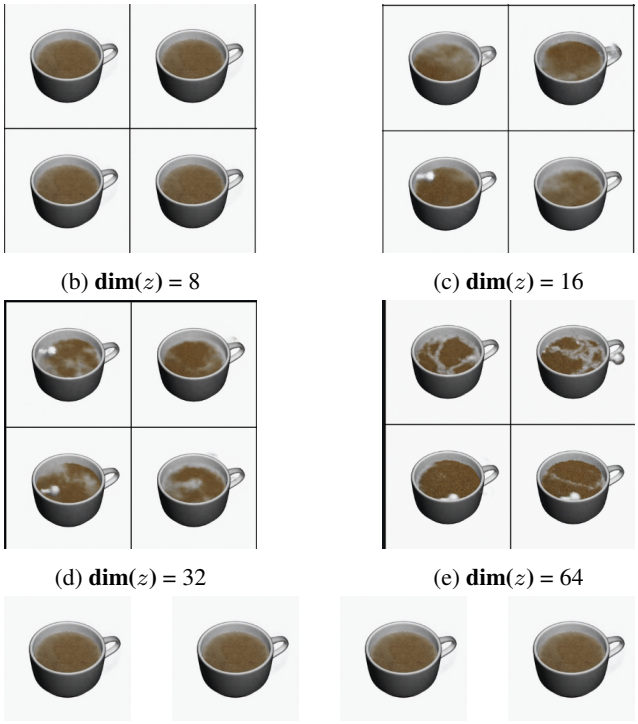
Figure 5: Reconstructions of some o, produced with varying latent regularization strengths. When the KL-divergence loss term is kept intact, we recover the mean image of $\mathcal{D}$. As we decrease regularization , we deviate from the mean image.

4.4.2 KL-Weighting and Reconstruction

Upon observing the reconstruction similarity between our VAE and the mean image, we conducted an experiment to exponentially back-off the influence of the KL-divergence loss term for each run until we have a run that is our baseline autoencoder trained to minimize just the reconstruction error. Quantitatively, we observe that there is negligible difference between the reconstruction error incurred by each setting of the KL-weighting (Figure 5a). Qualitatively, we observe that reducing the influence of the KL-divergence loss term has the effect of producing reconstructions that look less and less similar to the mean image (Figure 5e).



(a) MSE Loss



(b) **dim**(z) = 8 (c) **dim**(z) = 16 (d) **dim**(z) = 32 (e) **dim**(z) = 64

(f) **dim**(z) = 8 (g) **dim**(z) = 16 (h) **dim**(z) = 32 (i) **dim**(z) = 64

Figure 6: Reconstructions produced with varying latent dimensionality and KL-weighting. Figure (a) indicates that the VAEs achieve practically the same reconstruction penalty error. Figures (b) - (e) show reconstruction quality under a setting of no latent regularization, where it is observed that increasing **dim**(z) produces more nuanced reconstructions. Figures (f) - (i) show reconstruction quality with regularization penalty, where it is observed that the reconstructions are invariant of the latent dimensionality and collapse to the mean image.

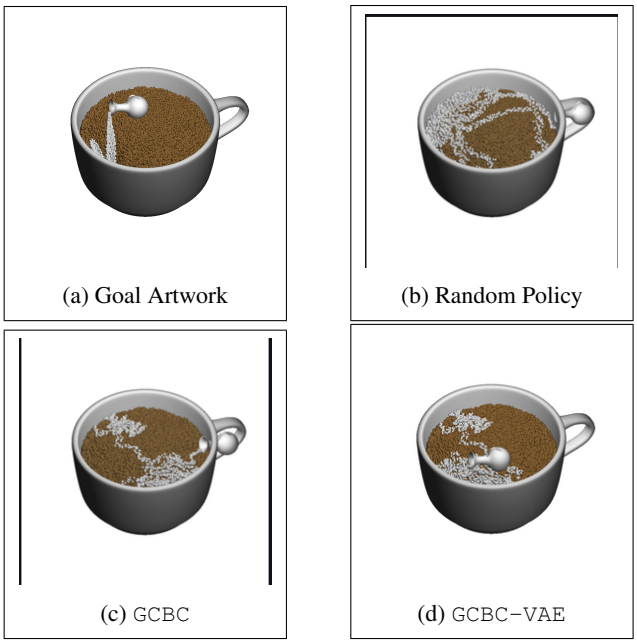
4.4.3 Latent Dimensionality and Reconstruction

The dimensionality of the latent vector z — **dim**(z) — can be interpreted as controlling the expressivity of the latent

distribution for our image observations. A more expressive latent distribution can better capture peculiar latte art designs, but is less likely to generalize. Thus, we conducted an experiment to analyze the difference in reconstruction quality as we vary **dim(z)**. For this experiment, KL-weighting is set to 1, $d = 32$, and all VAEs are trained for the same duration as for the experiment conducted in Section **4.4.1**. Quantitatively, we observe that the difference in reconstruction error achieved by the each setting of dim(z) is negligible. Furthermore, we observe a similar negligible difference in the reconstruction quality. When we re-run the experiments but have the latent regularization removed (KL-weight $= 0$), we still yield negligible differences in the reconstruction. However, the differences in **dim(z)** manifest in the reconstruction quality. Specifically, we observe that as we increase the latent dimensionality, the latent distribution is able to express more nuanced reconstructions and deviate from the mean image. This includes placing the foam injector somewhere, forming somewhat meaningful patterns, and having greater granularity of the fluid particles in the scene.

4.4.4 `GCBC-VAE` Performance

For the last set of experiments, we have rolled out `GCBC-VAE`, as well as a non-VAE `GCBC` baseline, to perform control for the `LatteArt-v0` task. In this experiment, the goal-conditioned behavior cloning network is trained to predict keeping in mind what may happen $g = 50$ timesteps later. Thus during test time, we input a desired artwork that could be produced in around 50 timesteps. As a baseline for the goal conditioned behavior cloning approach itself, we provide the performance of an agent following a random policy. We then run both `GCBC` and `GCBC-VAE` for 50 timesteps and qualitatively observe the resulting arrangement of foam and milk particles, which is presented in Figure 7. We discover that by adding in the latent representation from the VAE, the robot shows a behavior of localizing the artwork within the coffee.



(a) Goal Artwork (b) Random Policy (c) `GCBC` (d) `GCBC-VAE`

Figure 7: 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.

4.5. Analysis and Discussion

4.5.1 VAE and Mean Image

The above experiments lead us to interpret the latent regularization induced by the KL-divergence loss term as a mechanism to drive towards the mean image of the dataset. Since the VAE is regularized towards a Gaussian, the latent vector that we sample in the latent layers is likely to be the mean vector of some Gaussian. This could be advantageous for datasets where there are distinct clusters of related artwork (e.g. cosine and sine wave artworks are grouped together), so that the mean describes one of the clusters, and then any perturbations during sampling yield an artwork similar to what's in that cluster. However, because our dataset $\mathcal{D}_o$ is effectively a collection of random arrangements of particles, it is unlikely that there are distinct clusters. Thus, z may just treat all of $\mathcal{D}_o$ as just one cluster and samples of z may tend towards the mean of $\mathcal{D}_o$. One next step given this analysis is to take goal artworks that can be generated via a formula, and collate them into a new trajectory dataset. Since there would be distinct clusters present, our sampled z may be more meaningful and tend towards different means for different clusters depending on what the input o is.

4.5.2 dim(z) and Reconstruction Quality

In the absence of regularization, there appears to be a positive correlation between **dim(z)** and the expressiveness of the reconstructions. If the kind of trajectory dataset described at the end of Section **4.5.1** becomes available, then it may be an interesting investigation to compare the reconstruction quality for increasing **dim(z)** under a setting of

latent regularization. Figuring out a dimension for z that is compact yet sufficiently expressive can lead to interesting downstream applications of VAEs in models that perform dynamics entirely in latent space and which progress towards solving the `LatteArt-v0` task.

4.5.3 `GCBC-VAE` and Realistic Representations

Our experiments 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 that are unlike the fine-grained ones that would produce the intricate artwork that is usually desired. The performance of `GCBC-VAE` sheds some light on a tradeoff between compiling trajectories that are unrealistic, but objectively cover a vast distribution of arrangements of fluid particles, and focusing on realistic trajectories but are cumbersome to produce.

4.6. Conclusion

In this study, we explored the application of Variational Autoencoders (VAEs) in fluid manipulation tasks, focusing on the `LatteArt-v0` task. The VAE was trained to learn compact latent representations of fluid scenes and evaluated in terms of reconstruction quality and its integration with a goal-conditioned behavior cloning model (`GCBC-VAE`). We found that the VAE tended to converge towards the mean image of the dataset, indicating a need for more diverse feature capture. Varying the latent dimensionality and regularization scheme had negligible impact on reconstruction error but influenced the expressiveness and granularity of the reconstructions. The integration of the VAE within the `GCBC-VAE` model showed promise in localizing artwork within the coffee, but generating more realistic and intricate designs remains a challenge. Future work could focus on generating diverse and realistic training trajectories to improve the VAE's performance. Overall, our study demonstrates the potential of VAEs in learning representations of fluid scenes and their role in control models for multi-fluid manipulation tasks.

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

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

[3] 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. 2

[4] 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. 2

[5] 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. 1

[6] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost, 2018. 2