Extended Abstract

Motivation Standard offline imitation learning pipelines are limited in their ability to generalize to long-horizon robotic tasks. Most rely on short observation windows and struggle when tasked with reasoning over extended temporal dependencies. Increasing the observation window naively often leads to overfitting and degraded test-time performance due to spurious correlations in high-dimensional input space. These issues are particularly pronounced in offline settings, where there is no corrective feedback. The CS224R introductory assignment involved an offline imitation learning policy where we applied DAgger. Inspired by the assignment and motivated by recent work on long-horizon pipelines, our goal was to explore whether observation embeddings could improve standard offline imitation learning.

Method We propose a framework composed of two networks: (1) a latent observation prediction model that forecasts the next latent embedding z_{t+1} from the current embedding z_t, and (2) an inverse dynamics model that infers the action a_t that caused this transition. We evaluate this design against two alternative model variants: a frozen-latent encoder model, and a delta conversion approach that attempts to extract actions directly from observation differences. We train our final method using a composite loss comprising:

1. **Action prediction loss**, measured via mean squared error between the predicted action $\hat{a}_t$ (inferred from the predicted future latent $\hat{z}_{t+1}$) and the ground-truth expert action a_t.
2. **Latent prediction loss**, computed as mean squared error between the predicted future latent $\hat{z}_{t+1}$ and the encoder-derived future embedding z_{t+1}.
3. **Observation reconstruction loss**, which ensures the latent encoding retains enough fidelity to reconstruct the original raw observation o_t, also supervised with mean squared error.

Our method is novel in that, unlike most existing long-horizon pipelines which rely on action chunking or pair observation embeddings with action inputs at test time, we instead focus on reducing input complexity by relying solely on observation embeddings. This design simplifies the input space while enhancing the model's ability to reason over extended temporal contexts.

Implementation We evaluate our method on three robotic manipulation domains: Franka Kitchen (long-horizon, multi-stage), Push-T (short-horizon), and RoboMimic Square (medium-horizon). All experiments are conducted using consistent hyperparameters and implemented in PyTorch Lightning with training on an A100 GPU. Our evaluation compares model variants across observation window sizes (1, 8, 32, 64) using test-time action MSE. We also conduct a qualitative evaluation on simulated rollouts for the datasets in their respective environments.

Results Our method outperforms behavioral cloning (BC) baselines on Square and Franka Kitchen, especially at longer context windows, reducing test loss by over 60% at window size 32 in Franka Kitchen. In Push-T, performance is strong at short horizons but declines with longer windows, likely due to noise sensitivity. Qualitatively, predicted actions align closely with expert behavior (Figure 5), and latent states form smooth trajectories (Figure 6). Best-case rollouts (Figures 7–8) show successful task execution, though most runs diverge, likely due to compounding errors and data limitations.

Discussion Our experiments show that an offline, behavior-cloned policy can learn accurate mappings from observations to actions and latent embeddings to observations without relying on large-scale compute. However, the method depends on expert-labeled actions and struggles with covariate shift in underrepresented states, leading to rollout failures. Future work could address these limitations through self-supervised learning techniques and hybrid offline-online training to improve robustness and generalization.

Conclusion Our framework avoids the memory overhead of transformer-based sequence models and generalizes across variable task complexity. Our results suggest that latent inverse dynamics modeling is a promising direction for future offline control systems.

Latent Observation Forecasting for Long-Horizon Imitation Learning

Annmaria Antony
Department of Computer Science
Stanford University
annmaria@stanford.edu

Rebecca Joseph
Department of Mathematics
Stanford University
rjoseph5@stanford.edu

Mikul Rai
Department of Computer Science, Department of Electrical Engineering
Stanford University
mikulrai@stanford.edu

Abstract

We present a framework for long-horizon imitation learning that compresses sequences of raw observations into structured latent embeddings, enabling policies to reason over extended temporal dependencies. Our method uses a two-stage architecture: a latent forward module that predicts the next future observation embedding from the current one encapsulating a window of recent observations, and an inverse module that infers the action that caused the transition. This structure is trained with a composite loss over action prediction, latent forecasting, and observation reconstruction, allowing the policy to leverage long-term context while protecting against overfitting. We evaluated our model on three robotic manipulation tasks (Franka Kitchen, Push-T, and RoboMimic Square) and found that our model outperforms behavioral cloning baselines at longer window sizes, particularly in temporally complex settings like Franka Kitchen. Qualitative analysis of rollouts in simulated environments shows that, in best-case scenarios, the learned policy produces smooth, task-consistent behaviors that closely align with expert demonstrations, suggesting strong alignment between latent structure and control intent. However, this consistency is not guaranteed across all rollouts, reflecting limitations inherent to the narrow distribution of training data used due to compute constraints, which likely limited the policy's generalization capacity.

1 Introduction

We propose a method to improve policy learning for long-horizon robotic manipulation tasks by operating in a structured latent space. Standard imitation learning pipelines condition policies on a short history of past observations, often limited to 1-3 frames; however, many real-world tasks often demand reasoning over longer temporal horizons. Unfortunately, naively increasing the observation context length often results in degraded performance, likely due to the model overfitting to spurious correlations in high-dimensional observation space and instability during optimization. To mitigate these challenges, we train a model that first encodes a window of raw observations into compact latent embeddings and then predicts the next embedding z_{t+1} from the current embedding z_t. From this predicted future embedding, the model infers the preceding action a_t, which is supervised against expert-labeled demonstrations. We hypothesize that this two-stage architecture (latent forward prediction followed by backward action preference) facilitates long-horizon reasoning while maintaining constant memory requirements and improving training stability. We evaluate

our method on a diverse set of robotic manipulation tasks from the Franka Kitchen, Push-T, and RoboMimic Square datasets.

In this work, our objective is to investigate whether modeling observation trajectories in a predictive latent space enables more effective policy learning for long-horizon robotic manipulation tasks. Specifically, we ask: *Can latent observation forecasting support temporally extended imitation learning without requiring explicit action history or reward feedback?* Through a two-stage architecture that predicts future latent embeddings and infers actions from them, we aim to explore whether such abstraction improves generalization, especially under limited supervision and offline constraints.

2 Related Work

Motivated by the limitations of current imitation learning systems, particularly in handling long-horizon tasks, we draw inspiration from prior work in latent dynamics modeling and sequence transformers to guide our approach. These challenges are especially pronounced in offline reinforcement learning, where supervision is limited and long-term dependencies must be maintained efficiently. Several lines of prior work have aimed to address different aspects of this space. Our approach builds primarily upon insights from four foundational papers.

2.1 World Models

The work by Ha and Schmidhuber on World Models Ha and Schmidhuber (2018) introduced a paradigm of using latent space representations learned through pixel-level reconstruction. These models were effective at reducing high-dimensional observations like images into a low-dimensional latent space using a variational autoencoder. However, they prioritized visual reconstruction rather than capturing features relevant for decision-making in control tasks. As a result, the learned latent space may preserve appearance details while overlooking task-relevant semantics such as object positions and causal relationships between actions and outcomes.

2.2 Transformer-based Sequence Models

Models such as Decision Transformer Chen et al. (2021) and Trajectory Transformer achieve strong performance by leveraging attention mechanisms to model long-term dependencies. Despite their success, these architectures suffer from a common limitation in transformer models: high memory complexity that scales quadratically with sequence length. Although memory-augmented variants have been proposed to mitigate these issues, the challenge of scalability and efficiency remains largely unresolved.

2.3 Latent Action Space Methods

Latent action space methods Ozair et al. (2021) discretize continuous action spaces using vector quantization to reduce the complexity of learning. We found the spirit of these methods similar to World Models Ha and Schmidhuber (2018), though the emphasis shifts from pixel-level reconstruction to encoding control-relevant features like object positions and velocities. Both approaches introduce bottlenecks at different stages of the decision-making process—either in perceiving the environment or executing precise actions. In latent action space methods, this compression often sacrifices expressiveness. For high-dimensional or smooth control tasks, such as those found in robotics, discretization can fail to preserve the nuanced structure of continuous actions, leading to poor generalization.

2.4 Imitation-from-Observation Methods

Imitation-from-observation approaches Liu et al. (2017) eliminate the need for explicit action labels by inferring reward signals directly from raw video demonstrations. We found the essence of these methods to be similar to transformer-based sequence models Chen et al. (2021), in that both attempt to learn from high-dimensional sequential data without dense supervision. However, while transformers struggle with memory efficiency and sequence scaling, imitation-from-observation shifts the burden to reward inference. These methods often rely on complex generative models and indirect supervision,

introducing their own bottlenecks—particularly in reward modeling and task grounding. As a result, they frequently suffer from training instability and limited robustness to real-world variation.

2.5 Our Approach

Drawing on insights from these bodies of work, we propose a two-stage architecture specifically designed for temporally extended imitation learning tasks. Our framework includes:

- **Latent Forward Prediction:** A forward dynamics module predicts future latent embeddings z_{t+1} based on the current state z_t, enabling temporal reasoning without requiring pixel-level reconstruction.
- **Backward Action Inference:** An inverse dynamics component recovers actions a_t from pairs (z_t, z_{t+1}), providing direct supervision for control without reconstructing entire trajectories.

This two-stage design maintains temporal context within a compact latent space while avoiding the quadratic memory cost typical of transformer architectures. By separating representation learning from control supervision, it supports more efficient training and improved generalization on long-horizon continuous control tasks.

3 Method

The central objective of our approach is to encode a sequence or window of raw observations into compact representations that preserve task-relevant structure and high-level intent across time. Our guiding hypothesis is that policy success is fundamentally rooted in the model's capacity to extract structure from observational data in robotic manipulation, especially in long-horizon tasks.

From this, we generated two guiding assumptions: (1) that actions are recoverable from transitions between observations, and (2) that repeating patterns in what the agent sees over time reflect consistent subtasks through intent. From these assumptions, we wanted to design a pipeline that would reduce the dimensionality by relying solely on observations during training. Before coming to our ultimate pipeline, we designed two model variants that tested the limits of our two assumptions above, before finalizing on observation embeddings. In this section, we present our design exploration process to illustrate the rationale behind adopting an observation embedding space and to demonstrate its advantages over alternate methods.

1. **Frozen z_t:** The first pipeline we hypothesized froze the latent representation z_t. This variant was composed of two transformer architectures. The first segment was used for our training loss and produced the ideal z_t^* by reading the whole set of data, segmented as windows, and incrementally freezing the window sizes of z_t. The second architecture formed the whole of the training and testing pipeline. While the second architecture also predicted z_t, before going on to predict the subsequent observations, during training, the loss of z_t would be minimized against the ideal z_t^*.

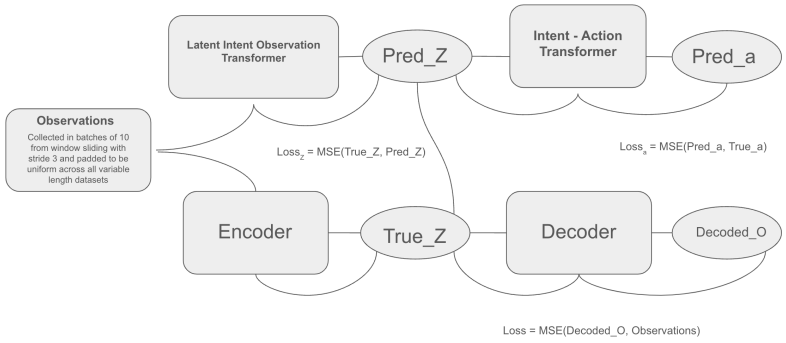


Figure 1: Frozen z_t

During early experimentation, we observed that there is no single canonical embedding z_t for a given observation window; multiple valid representations can emerge depending on the model's learned parameters. Additionally, directly optimizing for a fixed latent z_t is ill-posed. Instead, we concluded that accurate reconstruction in the observation space offers more interpretable training signal, prompting us to explore alternative architectures beyond the frozen encoder.

2. **Delta based action inference:** In our second pipeline, we aimed to predict actions at test time without requiring any additional training. This approach relied on converting observation-space coordinates into action-space coordinates. The latent encoder transforms each observation o_t into a 256-dimensional embedding z_t, and predicts the next observation o_{t+1}, thereby advancing to the next latent state z_{t+1}. We hypothesized that by computing the second-order difference, the discrete derivative of the delta between o_{t+1} and o_t we could recover the action that produced the transition:

$$\hat{a}_{t-1} = \lambda^{+} \Delta^2 o_t,$$

where $\Delta^2 o_t = o_t - 2o_{t-1} + o_{t-2}$, and λ^{+} denotes the pseudo-inverse (used in place of a true inverse when λ is non-square or not full-rank) of a scaling matrix $\lambda \in \mathbb{R}^{n \times m}$.

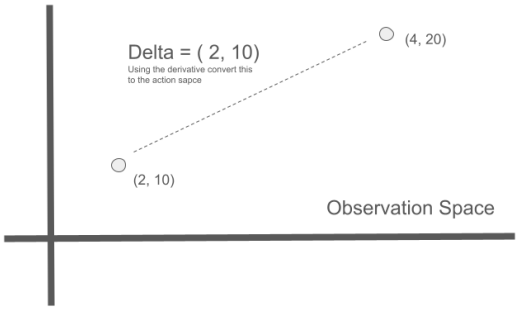


Figure 2: Delta initial sketch

However, we realized after that, although both action and observations appear to be coordinate-based, they are not inherently on the same scale. We can see this mathematically through this derivation.
Let $\Delta^2(o_t) = o_t - 2o_{t-1} + o_{t-2}$ denote the second-order finite difference (calculating the change in the rate of change) of the observation sequence.

Let $o_t \in \mathbb{R}^n$ be the observation at time t, $\quad \Delta o_t = o_t - o_{t-1}, \quad \Delta^2 o_t = o_t - 2o_{t-1} + o_{t-2}$.

We initially hypothesized:

$$\Delta^2 o_t \approx a_{t-1}, \quad \text{with } a_{t-1} \in \mathbb{R}^m.$$

But in practice, the relationship is closer to:

$$\Delta^2 o_t = \lambda\, a_{t-1} + \varepsilon_t,$$

where $\lambda \in \mathbb{R}^{n \times m}$ is an unknown scaling matrix, and ε_t accounts for noise and omitted dynamics.
We tried to invert this:

$$\hat{a}_{t-1} = \lambda^{+}\, \Delta^2 o_t,$$

but this only works under very restrictive conditions:

- λ must be known exactly,

- action needs to be in the format $a_t = (F_x, F_y)$,
- and the error term ε_t must be negligible.

Ultimately, using $\Delta^2 o_t$ to infer a_{t-1} is unreliable without knowing dynamics and scaling.

Unfortunately, even in a simulator, it is impossible to calculate the scalar and error factor. Additionally, the observation may omit important velocity force information that derivatives cannot recover.

3. **Observation Embedding with Action Predictor:** From our mathematical modeling and reasoning around the frozen z_t and delta conversion approaches, we deduced that reconstruction loss and action loss were instrumental in ensuring our policy was successful. So we formalized our final pipeline: observation embedding with action prediction. We kept the 2-stage architecture. In the first stage, a latent encoder transforms each observation into a 256-dimensional latent embedding z_t. In the second stage, the model predicts the next latent stage z_{t+1} from the current embedding z_t and decodes the corresponding action a_t responsible for this transition. This design models inverse dynamics in latent space, enabling the policy to reason over temporally abstracted representations without direct supervision on the full observation trajectories.

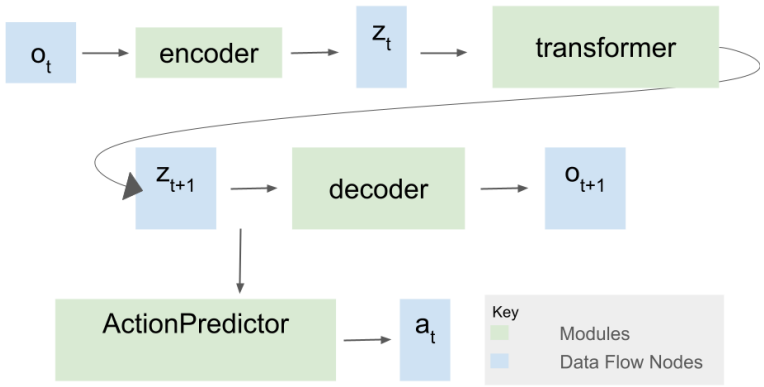


Figure 3: Observation Embedding Architecture

We optimize a composite loss comprising three components:

(a) *Action prediction loss*: Measured via mean squared error between the predicted action $\hat{a}_t$ (inferred from the predicted future latent $\hat{z}_{t+1}$) and the ground-truth expert action a_t.

(b) *Latent prediction loss*: Computed as mean squared error between the predicted future latent $\hat{z}_{t+1}$ and the encoder-derived future embedding z_{t+1}.

(c) *Observation reconstruction loss*: Computed as the mean squared error between the reconstructed observation $\hat{o}_t$, decoded from the latent representation, and the actual current observation o_t. Ensures the latent space retains sufficient information to faithfully reconstruct the input.

The total loss is computed as:

$$\mathcal{L}_{\text{total}} = \text{MSE}(a_t, \hat{a}_t) + \lambda_{\text{obs}} \cdot \text{MSE}(z_{t+1}, \hat{z}_{t+1}) + \lambda_{\text{recon}} \cdot \text{MSE}(o_t, \hat{o}_t)$$

where λ_{obs} and λ_{recon} are tunable hyperparameters controlling the contribution of the latent dynamics and reconstruction terms, respectively.

Many latent-dynamics models rely on just two losses, typically a reconstruction term plus an action term, whereas joint pipelines often add a third action-prediction term to ground the model in expert behavior. Including all three terms ensures that our model learns a latent space that is both dynamic and richly informative. By balancing these complementary objectives, we obtain a representation that supports faithful imitation of expert behavior and stable long-horizon reasoning.

4 Experimental Setup

We evaluate our method across three robotic manipulation tasks designed to test temporal reasoning and generalization ability: Franka Kitchen, Push-T, and RoboMimic Square. These environments were selected for their diverse range of spatial layouts and task horizons.

- *Franka Kitchen*: A long-horizon manipulation task featuring the Franka robot, which must interact with multiple kitchen objects (e.g., microwave) in order to reach multi-goal end states.
- *Push-T*: A short-horizon manipulation task involving a T-shaped block. The robot is required to push the block to a designated region while maneuvering its end-effector to a specified endpoint.
- *RoboMimic Square*: A medium-horizon manipulation task where a robot must align and fit a square nut onto a square peg (also known as the Nut Assembly task).

To systematically study the effect of temporal history length on downstream policy performance, we evaluate our model using four observation history windows: 1, 8, 32, and 64. Each window represents the number of past timesteps $(o_{t-k}, \ldots, o_t)$ encoded before predicting the next observation $(\hat{o}_{t+1})$ and inferring the corresponding action $(\hat{a}_t)$. For each window size, the model encodes the stacked sequence into a fixed 256-dimensional latent representation using a multi-layer Transformer encoder.

4.1 Architecture and Training

Our model architecture is described in detail in Section 3. Briefly, the model consists of:

- An *observation encoder* with two-layer MLP and LayerNorm layers that maps raw observations to a compact latent space.
- A *temporal transformer* that models sequential dependencies over the encoded latent representations, with 4 layers and 4 attention heads.
- A *dual-head prediction setup* that infers both the next latent state and the corresponding action, with a decoder reconstructing observations from latent space which is used for loss.

All models are trained using PyTorch Lightning for 100 epochs with early stopping (patience = 20 epochs). We use the AdamW optimizer with a fixed learning rate of 1e-4 and weight decay of 1e-5. Learning rate scheduling is governed by a plateau-based scheduler monitored on validation loss. Batch size is set to 32 and gradient clipping is applied at 1.0 to ensure training stability.

4.2 Data Pipeline and Preprocessing

We build temporal datasets from offline trajectory data, which constructs sequences of the form $(o_{t-k}, \ldots, o_t) \rightarrow o_{t+1}, a_t$. All sequences are windowed in time and split into training, validation, and test subsets using an 80/10/10 split. The observations and actions are loaded from .npz files containing cleaned, aligned demonstration rollouts, and all tensors are cast to float32 and normalized if necessary.

Evaluation Metrics

We evaluate performance using the following reported metrics:

- *Action Prediction Loss:* MSE between inferred and ground-truth action a_t, capturing the model's effectiveness at behavioral imitation.
- *Rollout Success:* We deploy trained policies in the simulation environment and report qualitative success (i.e., how the agent attempted to complete the task) based on trajectory rollouts. This helps us see if our policy is able to complete the task successfully or not in the real environment.

Implementation Details

Each experiment is tagged and logged using the Weights & Biases platform for reproducability and analysis. Models are trained on a single A100 NVIDIA GPU. Dataloaders use persistent workers with pinned memory to optimize throughput, and performance is validated at the end of every epoch. During evaluation, we calculate action prediction error by comparing the model's predicted actions to expert actions on held-out test data. We also roll out the trained policy in the simulated environment to qualitatively assess attempt at task completion.

Baselines

To contextualize the performance of our transformer-based temporal modeling approach, we train a series of BC baselines using identical observation windows (1, 8, 32, 64) and compare their action prediction loss against our model. These baselines lack latent observation prediction and do not incorporate sequential modeling, allowing us to isolate the contribution of latent forecasting.

Hyperparameters

We used a consistent set of training hyperparameters across all experiments, as summarized below:

Table 1: Model Training Hyperparameters (constant across experiments)

Hyperparameter	**Value**
Batch Size	32
Latent Dimension	256
Number of Attention Heads	4
Number of Transformer Layers	4
Learning Rate	1×10^{-4}
Weight Decay	1×10^{-5}
Maximum Training Epochs	100
Early Stopping Patience	20 epochs
Optimizer	AdamW
Hardware	A100 (NVIDIA)
Window Sizes Tested	{1, 8, 32, 64}
Datasets Evaluated	Franka Kitchen, Push-T, RoboMimic (Square)

5 Results

Our evaluation combines both quantitative metrics and qualitative visualizations to assess the effectiveness of our proposed model across the three manipulation tasks. The results highlight the strengths and limitations of latent observation space modeling in comparison to our baseline BC and explains why the proposed latent-observation policy succeeds on some benchmarks and struggles on others.

5.1 Quantitative Evaluation

We analyze the results in the table 2 for the proposed method compared to the BC baseline. Across all three manipulation tasks: RoboMimic Square, Push-T and Franka Kitchen, we can see that our policy consistently outperforms BC policy when using only the last observation (window = 1).

RoboMimic Square For the Square task, our model lowers error by **14%** at $w = 1$ and remains flat as the window grows to 64, whereas BC deteriorates by ~18 pp. The latent bottleneck removes frame-to-frame redundancy (e.g. minor gripper jitter) so that attention over longer sequences adds signal and avoids overfitting. In contrast, we can see that the BC degrades sharply for longer context lengths reflecting the inability to use long context without overfitting.

Push-T Push-T is a low-dimensional task characterized by noisy dynamics due to background contact. Our model is able to win at $w = 1$ but crosses the BC curve at $w = 8$ and keeps diverging. Extra context now introduces mostly noisy motion, and the latent encoder cannot fully understand it, shifting useful variance into uninformative directions. The takeaway is that long context helps only when that context is needed.

Franka Kitchen For the Franka Kitchen complex environment, we can see that our method outperforms the baseline significantly up to cutting the baseline's loss by 60 percent at window 32. This confirms that for tasks requiring multi-step dependency and intent dynamics, compact latent representation allow for more accurate action prediction. BC collapses on the other hand as longer context leads to actions—ambiguity it cannot resolve with per-frame inputs alone. Overall, we can see that latent observation space allows the policy to exploit longer context when its required. The mixed behavior on Push-T dataset points to future work for advanced attention mechanisms or context lengths for task complexity.

Table 2: MSE for predicted vs. expert actions on test set. Lower values indicate better performance.

Dataset	Window	**Our Policy**	**BC**
Square	1	0.139408	0.162013
	8	0.142312	0.193462
	32	0.139927	0.195074
	64	0.140046	0.191729
Push-T	1	0.011991	0.015168
	8	0.026771	0.025305
	32	0.041292	0.040454
	64	0.060148	0.055695
Franka Kitchen	1	0.020854	0.054805
	8	0.018367	0.043745
	32	0.012297	0.029578
	64	0.014829	0.025880

5.2 Qualitative Analysis

We complement the quantitative metrics with a visual inspection of the policy. First, we compare predicted actions to expert commands; next, we examine the geometry of the latent state space; finally, we review frame-by-frame roll-outs on two benchmark tasks.

5.2.1 Predicted vs. True Actions

Figure 4 compares the true and predicted expert actions for both output dimensions. The dense diagonal streak shows that the decoder learns a nearly linear latent state to action mapping. Outliers can be seen to cluster near high magnitude commands hinting at sparse data and coverage limits instead of representational flaws.

5.2.2 t-SNE of Latent States

Figure 5 shows a t-SNE projection of the latent state z_t. The smooth ring structure, rather than fragmented clusters, suggests that the encoder orders task states coherently, which is key for stable roll-outs. The tiny voids inside the ring coincide with rare visual occlusions (confirmed by manual replay) and explain small spikes in test loss; these latent gaps become spots that the decoder has never practiced steering through and is susceptible to crash in.

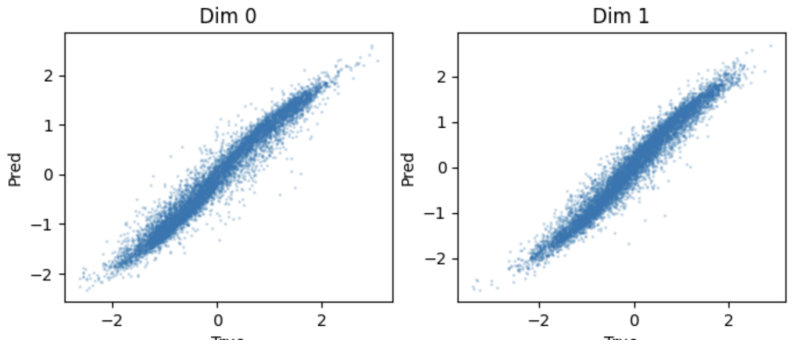


Figure 4: Scatter plot of true (y-axis) versus predicted (x-axis) actions.

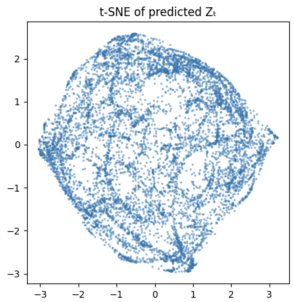


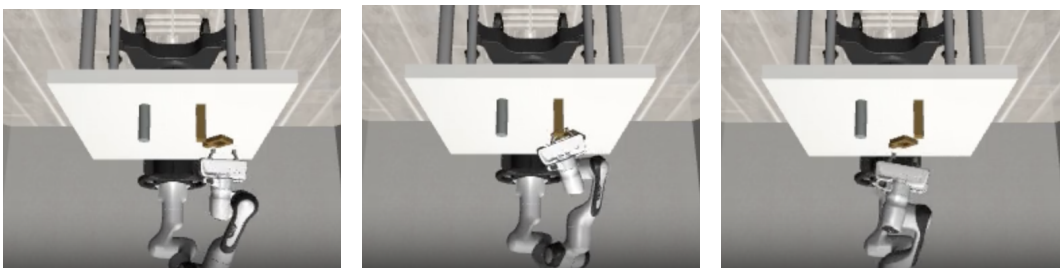
Figure 5: t-SNE embedding of latent states z_t.

5.2.3 Roll-out Snapshots

Figures 6 and 7 present frame-by-frame snapshots of the policy on the *Square* and *Push-T* tasks. In the best-case runs the agent (i) starts at the correct pose and moves toward the object, (ii) interacts purposefully, and (iii) completes the placement. Typical failures, however, occur after the first unseen observation—e.g. the block rotates 5° more than in any demonstration—leading to covariate-shift compounding. Because the policy never re-encounters the expert state, errors snowball. This aligns with the quantitative Push-T curve: even small deviations become unrecoverable when context is noisy.

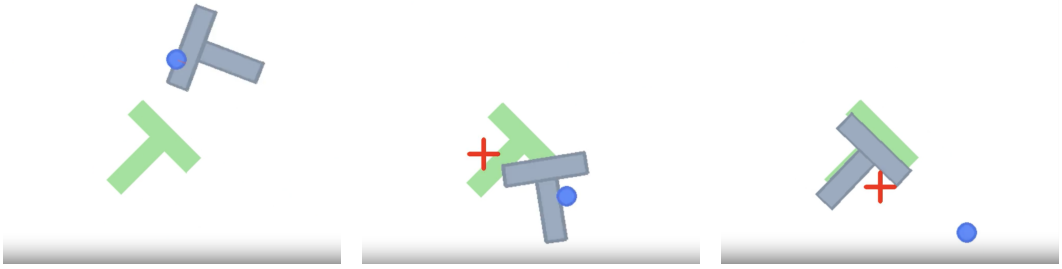
5.2.4 Qualitative conclusion

Overall, our visual diagnostics indicate that latent-observation modelling is representation-limited on inherently noisy tasks—where projecting high-variance sensor data into a low-dimensional latent can entangle signal and noise—and data-limited on tasks with sparse interaction events, where the demonstrations under-sample critical states. These findings motivate future work on stronger attention gating, targeted demonstration augmentation, or closed-loop corrections such as DAgger.



(a) Before interaction (b) Mid-task (c) After interaction

Figure 6: Qualitative sequence on the *Square* task.



(a) Before interaction (b) Mid-task (c) After completion

Figure 7: Qualitative sequence on the *Push-T* task.

6 Discussion

Our experiments show that an offline, behavior-cloned policy can learn accurate mappings from observations to actions and meaningful latent models to observation prediction even without access to specialized hardware or large-scale compute.

Nevertheless, the approach has limitations. Firstly, our model depends on expert-labeled action sequences, which restricts its applicability in real-world or low-supervision settings. Secondly, most rollouts ultimately fail—not because of poor modeling, but because minor prediction errors lead the agent into distributional outliers where the demonstrations provide little guidance. In these underrepresented states, errors compound, leading to covariate shift that the policy cannot recover from.

A final limitation of our approach stems from the narrow training distribution, which was constrained by compute resources. Because our model was trained offline on a fixed set of expert demonstrations, it was exposed to only a small fraction of the full state-action space. As a result, we found that it struggles when encountering unfamiliar or out-of-distribution observations during our rollout experiments in the simulated environments. This limitation is not unique to our work, but it is especially pronounced in long-horizon tasks where small errors accumulate and push the agent into sparsely represented regions of the observation space. In such settings, the model lacks the coverage needed to generalize or recover effectively. Addressing this issue will likely require either more diverse offline datasets spanning a broader range of environments and task variations or the integration of online data collection methods that allow the agent to adaptively explore and correct for its own blind spots. Without such mechanisms, even high-capacity models remain brittle when deployed.

7 Conclusion

Our results indicate that learning policies in an observation latent space offers a promising foundation for imitation learning, particularly in long-horizon settings. By encoding observation sequences into compact embeddings, the model captures subtask intent and temporal structure without relying on reward signals or online interaction. While reliance on action labels and sensitivity to distributional shift remain as limitations, this work provides initial evidence that observation embeddings alone can support the execution of complex, temporally extended tasks in purely offline regimes.

More broadly, our findings suggest that latent observation forecasting is not just a training convenience but actually provides a lightweight and scalable framework for structuring policy learning in environments where interaction is costly or unsafe. The approach sidesteps the memory overhead of sequence models while still capturing the benefits of temporal abstraction and long-term reasoning.

The failures highlighted in Section 6 propel the need for future work in two key areas. First, to reduce reliance on action supervision, self-supervised alternatives such as temporal contrastive learning (which train the model to distinguish between sequences of observations that occur in the correct temporal order and those that do not) could be used to train latent policies from raw sequences alone. Second, robustness to unseen observations might be improved through hybrid offline-online learning loops that allow for targeted correction in sparse or critical regions of the observation space. Together,

both of these directions aim to move beyond static behavioral cloning and toward more resilient, adaptable imitation learning systems capable of generalizing beyond their training distribution.

8 Team Contributions

- **Annmaria Antony:** Led dataset processing and cleaning, co-designed the model architecture and training objectives, and implemented the training pipeline, including data loaders, loss functions, and optimization routines.
- **Rebecca Joseph:** Co-designed model architectures, implemented training pipelines for BC, and authored the methods section, including detailed mathematical formulations of each approach.
- **Mikul Rai:** Led RL policy ideation and qualitative analysis, co-designed model architectures, loss functions and optimization parameters

Changes from Proposal In our proposal, we hypothesized that an observation-only latent representation could improve imitation learning policies by allowing access to long-horizon context without overfitting to spurious details. Our goal was to decouple historical reasoning from online inference by compressing long sequences of raw observations into a single latent vector z_t, which the policy would then use alongside a short context window of recent inputs. However, through experimentation, z_t could not accurately give the agent enough information to act using observation alone. Therefore, we decided to refine our method by jointly learning an observation embedding space and an action inference based policy that infers actions from predicted latent transitions. Our revised method preserves the spirit of our initial proposal—leveraging temporal abstraction from observations, while aligning more closely with stable training dynamics and real-world rollout performance.

References

Lili Chen, Kevin Lu, Aravind Srinivas, Pieter Abbeel, and Igor Mordatch. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *Advances in Neural Information Processing Systems*, Vol. 34. 15084–15097.

David Ha and Jürgen Schmidhuber. 2018. World Models. *arXiv preprint arXiv:1803.10122* (2018). arXiv:1803.10122

YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. 2017. Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation. *arXiv preprint arXiv:1707.03374* (2017). arXiv:1707.03374

Sherjil Ozair, Hidetoshi Furukawa, Yoshua Bengio, and Doina Precup. 2021. Latent Action Policies for Reinforcement Learning. In *NeurIPS Workshop on Deep Reinforcement Learning*.