Extended Abstract

Motivation Gathering a large number of high-quality, realistic demonstrations for robot policy training remains a challenge, especially for high-risk or less common tasks. We explore whether base and/or fine-tuned video generative models can generate robot simulation videos that are beneficial to an in-simulation robot training policy for a single manipulation task.

Method We focus on the PushT task Li et al. (2025); Chi et al. (2024); Florence et al. (2021) in a simulation environment; this task involves an agent (blue dot) pushing a gray T to align with the green T. We train a behavior cloning policy that takes an image as an input and outputs an action. First, we evaluate a CNN vs. a finetuned ResNet18 model on Columbia Diffusion Policy's PushT Dataset Chi et al. (2024) to choose the best model architecture. Secondly, we adapt and evaluate Unified Video Action (UVA)'s inverse dynamics model for one-step action extraction. This is the first publicly reported evaluation of an inverse dynamics model on the PushT task. Third, we generate videos of the PushT task using NVIDIA's Cosmos World Foundation Model NVIDIA et al. (2025a) and finetuned LTX-Video HaCohen et al. (2024). Finally, we compare the performance of BC policies trained on AI-generated videos vs. PushT dataset (with both original actions and inverse dynamics model actions).

Implementation We describe the implementation details for the behavior cloning policy training, video generation, inverse dynamics action extraction, and a comparison between models using these different components. We compare a CNN vs. a finetuned ResNet18 model both trained on the PushT dataset by using Mean Squared Error (MSE) and R^2. Then, we compare mean-reduction and first-action methods of reducing a sequence of 8 actions to 1 action for UVA's inverse dynamics model, since its default is to return multi-step actions. Videos were generated on 8 RTXs with 400GB VRAM, with an inference time of 7 minutes per video. The intended setup was A100s/H100s, so extra efforts were made to get ML attention packages working on the RTXs. Finally, we evaluate the BC policies in a PushT environment across 100 rollouts.

Results We found that: (1) The ResNet18 architecture proves to perform better than the CNN architecture (106 vs 419 Test MSE), so we use ResNet18 to train our future behavior cloning policies. (2) We choose mean-reduction for generating actions moving forward. (3) We perform qualitative and quantitative evaluation on the Cosmos-generated videos, finding that many times the shapes morph into final configurations rather than moving across the screen. Additionally, the original PushT actions are more direct than the inverse dynamics PushT actions, but the Cosmos extracted actions are more direct. (4) Finally, we train behavior cloning policies on the Inverse Dynamics PushT actions and Cosmos actions, finding that both perform similarly well (Test MSE 339 vs 333), but worse than the ResNet18 baseline (Test MSE 99). (5) For the PushT simulation rollouts, the Cosmos Policy performs the best out of all of the BC policies with respect to average reward per episode step, although the overall performance of all policies is quite low.

Discussion We encountered many challenges while running and training the video generation models, deciding how to extract actions, and accessing resources. We overcame them by modifying package architectures, communicating with owner research teams, and switching to lighterweight models. We concluded that generated videos are not helpful for training unless fine-tuned on the particular task and environment, in agreement with others in the field. However, we still believe that video generation holds a lot of promise in the context of robot learning. Also, we were able to do more to evaluate the actions themselves, but had we had higher-quality or a larger number of generations, we may have considered evaluating actions directly from the video with a VLM.

Conclusion By focusing on the abstract PushT task in a simplified two-dimensional environment, we are able to highlight and analyze the many failure modes that can come up through a multi-step pipeline involving diffusion models and inverse dynamics models. We find that finetuning the LTX Video model does make big difference in the quality of generated videos. Following work would emphasize more fine-tuning and experiments with different configurations for this. We could evaluate the generalizability of fine-tuning on a single task (like PushT) on unseen manipulation tasks, like opening/closing drawers. As these Physical AI models are built for training policies, it would be interesting to analyze the action-extractability of the generated content.

PushT Start: Exploring Video Generation for Robot Learning

Ashna Khetan
Department of Computer Science
Stanford University
ashnak@stanford.edu

Daphne Liu
Department of Computer Science
Stanford University
daphnehl@stanford.edu

Poonam Sahoo
Department of Computer Science
Stanford University
pnsahoo@stanford.edu

Abstract

We explore whether base or fine-tuned video generative models can generate robot simulation videos that are beneficial to an in-simulation robot training policy for a single manipulation task. We focus on the PushT task Li et al. (2025); Chi et al. (2024); Florence et al. (2021) in a simulation environment. This is the first publicly reported evaluation of an inverse dynamics model on the PushT task. First, we train a behavior cloning policy that takes an image as an input and outputs an action. Secondly, we adapt and evaluate Unified Video Action (UVA)'s inverse dynamics model for one-step action extraction. Third, we generate videos of the PushT task using NVIDIA's Cosmos Text2World model NVIDIA et al. (2025a) and finetuned LTX-Video HaCohen et al. (2024). Finally, we compare the performance of BC policies trained on AI-generated videos vs. PushT dataset through test metrics and simulations in a PushT environment across 100 rollouts. We find that the Cosmos Policy performs the best out of all of the BC policies, although the overall performance of all policies is quite low.We concluded that generated videos are not helpful for training unless fine-tuned on the particular task and environment. However, we still believe that video generation holds a lot of promise in the context of robot learning. We are able to highlight and analyze the many failure modes that can come up through a multi-step pipeline involving diffusion models and inverse dynamics models. While finetuning the LTX video model was promising, following work would emphasize fine-tuning for longer periods of time and experiment with different configurations for this.

1 Introduction

Gathering a large number of high-quality, realistic demonstrations for robot policy training remains a challenge, especially for high-risk or less common tasks. To generate realistic data at scale, roboticists have explored generating videos. However, since these generative models are predictive and only as relevant as the data they have been trained on, raising questions about realism and usefulness. Recent works have 1) used generated videos of humans performing tasks and/or 2) defined high-compute pipelines to train generalizable polices on fine-tuned video generation model data. We explore whether base and/or fine-tuned video generative models can generate robot simulation videos that are beneficial to an in-simulation robot training policy for a single manipulation task.

Our research questions are as follows:

- Can we use base and/or fine-tuned video generation models to create large-scale datasets that are optimal for action extraction?
- Can policies trained from AI-generated videos outperform those trained on real demonstrations?

2 Related Work

2.1 AI-generated demonstration as policy-training data

Bottlenecked by the curation cost of real data, recent works have explored the role of synthetic data in reinforcement learning. Kim, et. al (2022) first introduced using syntehtically-genered video clips for action recognition tasks and found that the simulation-to-real gap (where models excel in simulation but fail in a real setting) is minor for datasets with low object and scene bias, and found these to outperform its real data counterparts whan Kim et al. (2023)

Most recently, both Stanford and NVIDIA released works with generated videos. Bharadhwaj et. al used generated videos of human demonstrations of manipulation tasks to train and test a policy and showed the impressive generalizability of their zero-shot human video generation plus single policy pipeline. They did not need to fine-tune their video model but noted that they chose not to use robot videos because that *would* require fine-tuning Bharadhwaj et al. (2024). NVIDIA very recently (a few days after we chose the scope for our project) launched a 4-stage pipeline to generate synthetic robot data generation from video world models: 1) fine-tune the world model, 2) rollout videos from it, 3) label pseudo actions with latent action or inverse dynamics models, and 4) use these *neural trajectories* for downstream visuomotor policy learning. They were the first to enable zero-shot behavior and environment generalization, with their humanoid robot performing 22 new behaviors in 10 new envrionemts Jang et al. (2025). Google's UniPi uses a video generator acting as a trajectory planner to generate videos with snippets of what the agent's trajectory should look like to achieve its goal, and feeds this into an inverse dynamics model that extracts low-level control actions Du et al. (2023).

2.2 Video generation models built for this purpose

Text-to-video models rose in the 2020s and fall into several categories. Autoregressive models such as VideoGPT generate one frame at a time, conditioned on previous ones Yan et al. (2021). Diffusion models denoise random noise into video frames via a learned reverse process Blattmann et al. (2023). GANs use a generator-discriminator setup to produce videos optimized for reality Tulyakov et al. (2017).

As of 2024, high-resource organizations have begun creating World Foundation Models for the purpose of predicting world environments and generating consistent videos to align with these predictions. These models are explicitly encoded to follow physical laws, spatial relationships, and causal consistency. The caveat being that they are significantly harder to train, given their size NVIDIA et al. (2025a). For specific known use-cases, companies release specific foundation models to lower the barrier to simulate and thus train robot policies. Just this year, NVIDIA released GR00T N1, a foundation model for humanoid robots NVIDIA et al. (2025b). This movement of Physical AI has promising applications in training robots that can learn from and adapt to their complex, physics-hard environments.

2.3 Extracting Actions from Video

As video is a high-dimensional representation of actions that describes *what* happened but not precisely *how* it happened, states and actions need to be extracted from them. Used by DreamGen are Latent Action Models and Inverse Dynamics Models. Latent Action Pretraining for general Action models (LAPA) trains an action model with a VAE-based objective to learn discrete latene actions between image frames, then pretrains a latent VLA model to predict these actions from an observation and task description. When fine-tuned on small-scale robot manipulation data, this can map the latent actions to robot actions Ye et al. (2025). OpenAI introduced Video PreTraining (VPT) trained on small datasets of contractors performing an action along with the keypresses and mouse movements needed to make them. On this they trained an inverse dynamics model that can use past

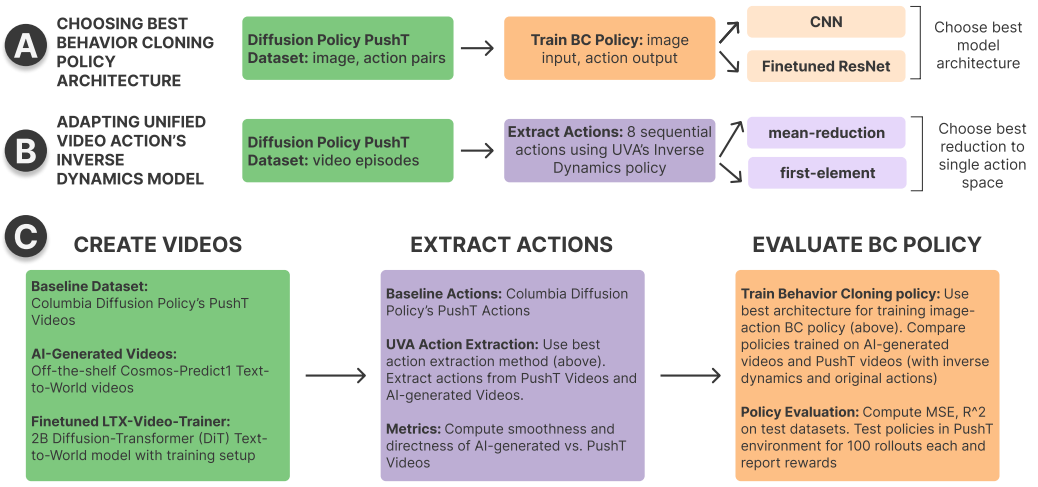

Figure 1: Through this project we first (A) choose the best behavior cloning policy architecture by comparing a CNN to finetuned ResNet-18, then (B) adapt the Unified Video Action model's inverse dynamics policy for a single-action output, then (C) generate simulation videos of the PushT task using diffusion models. With the AI-generated videos, we use the inverse dynamics checkpoint adapted during (B) and the architecture from (A) to train a behavior cloning policy and evaluate it in the PushT simulation environment.

and future information to guess the action at each step Baker et al. (2022). Stanford's United Video Action Model (UVA) was the first to integrate video generation with real-time policy inference. The system learns a unified latent space for both the video and action generation, thereby skipping video generation and reducing overfitting Li et al. (2025).

3 Method

For the purposes of this project, we focus our finetuning and training efforts on the PushT task Li et al. (2025); Chi et al. (2024); Florence et al. (2021) in a simulation environment. This task involves an agent (blue dot) pushing a gray T to align with the green T.

In our project, we have the following steps (Figure 1) to evaluate AI-generated videos in a simulated environment such that the project could be completed within the scope and compute constraints of CS 224R's resources and duration:

1. Evaluating BC Policy on original PushT Dataset to choose the best model architecture
2. Adapting and evaluating UVA's inverse dynamics model for one-step action extraction
3. Finetuning and generating videos of PushT task using diffusion models
4. Comparing the performance of BC policies trained on AI-generated videos vs. PushT dataset (with both original actions and inverse dynamics model actions)

After completing video generation/finetuning and action extraction, we use policy evaluation as a benchmark to help us with evaluating those generated videos and extracted actions. Finally, we evaluate our policy in a simulation environment. More experimental details will be described in Section 4.

3.1 Initial Dataset

We build off of Columbia's Diffusion Policy PushT dataset Chi et al. (2024) for video generation and baseline policy training. This dataset features 206 episodes in a simulated PushT environment for a total of 25650 frames. See Figure 2 for examples frames from the dataset.

3.2 Choosing Behavior Cloning Policy

We train a behavior cloning policy that predicts actions (x,y) coordinates from a given image. We use Columbia's Diffusion Policy PushT dataset. We preprocess images by normalizing each channel. We split our data into a training, validation, and test set with $80\%, 10\%, 10\%$ of the data, respectively.

We first build a baseline which takes actions and images from the PushT dataset to train a vanilla CNN policy and finetune a ResNet18 model (He et al., 2015)We compare all performances on the PushT Dataset, and select the strongest model to determine what architecture to use for our future experiments. To evaluate, we refer to the training MSE loss, but also the MSE and R^2 value on an unseen test set.

3.3 Creating Image-Action Pairs

We use Unified Video Action's Li et al. (2025) inverse dynamics model finetuned on PushT videos to extract actions from the AI-generated videos. The Unified Video Action model works as a inverse dynamics model by predicting action A_t as a function of $f(O_t, O_{t+1})$, where the observations are images. This model outputs 8 step trajectories and requires a minimum of 16 frames to be able to generate an action sequence so that there is temporal context before outputting a sequence of actions.

Because we aim to train a behavior cloning policy that outputs a one-step action for every image input, we experiment with three methods of combining the 8 action steps $a'_1, a'_2, \ldots a'_8$ (see Figure 1 B): we use mean-reduction $\frac{\sum_{t=1}^{8} a'_t}{8}$, and first-action a'_1. We refer to the final chosen action as a_i for each frame i. Because the PushT dataset has image-action pairs, we evaluate the best method of extracting actions by measuring the L_2 norm between the original actions and the extracted actions.

Once we have chosen the best method of action extraction, we use this method to surface actions from the AI-generated videos from the previous section. We aim to evaluate the videos via smoothness and directness metrics. Since each action a_t takes on the form (x_t, y_t), and each action is generated off of an additional subsequent frame, we first calculate velocity as $v_t = a_{t+1} - a_t$, acceleration as the second-order difference $a_t^{(2)} = v_{t+1} - v_t$, and jerk as the third-order difference $j_t = a_{t+1}^{(2)} - a_t^{(2)}$. For each episode, we then calculate the average velocity, standard deviation of velocity, average acceleration, standard deviation of acceleration, and average jerk. The standard deviation of velocity, average acceleration, and average jerk in particular are important measures of smoothness which we want to be small if possible to show steadier and smoother movement. Then, across episodes, we average all of these metrics to report final averages. For directness, we calculate the per-episode path length (assuming there are pairs numbered 0 through $T-1$) actions divided by displacement:

$$\text{directness} = \frac{\sum_{t=0}^{T-2} \|a_{t+1} - a_t\|}{\|a_{T-1} - a_0\|}.$$

Then we report the average and standard deviation of this directness metric. Similar to before, a ratio that is closer to 1 is more desirable.

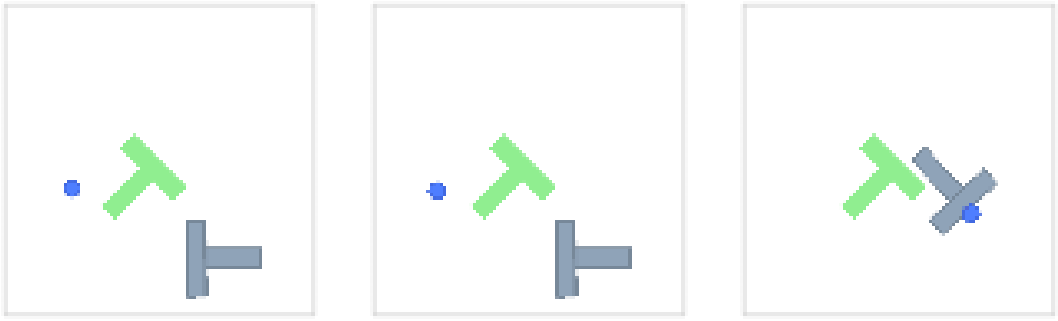

Figure 2: Example frames from Columbia's Diffusion Policy dataset.

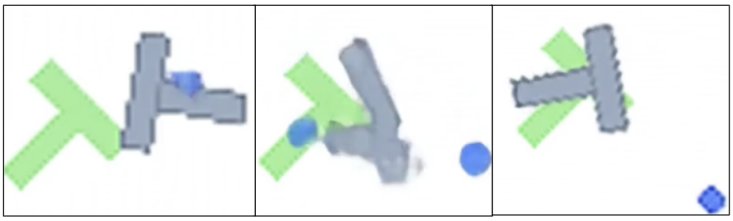

Figure 3: Frames from an AI-generated rollout of the PushT task exhibiting morphing. This video was generated using NVIDIA Cosmos Text2Video.

Model	Prompt
Cosmos	A small blue dot pushes a rigid gray T-shaped object toward a stationary green T-shaped object on a plain background. First, it pushes it from the right. Then it pushes it from the bottom. Then, it adjusts from the left. Follow physics: objects move normal to the motion that is applied to it. NEITHER T-shape changes size, color, or EVER DISAPPEARS. THE T CANNOT BREAK APART!!! The goal is for the gray T to fully cover the green T by the end.
LTX	A small blue dot moves around a canvas. As it does, if it touches a gray T, that gray T moves. The goal is for the gray T to be moved over a green T. The gray T does NOT morph or break or disappear.

Table 1: Descriptions of Cosmos and LTX systems.

3.4 Video Generation

We utilized three models for Video Generation. First, the NVIDIA Cosmos World Foundation Model Text2Video, a 7B diffusion predict model. This is a diffusion model trained with large-scale, diverse video datasets capturing different aspects of real-world physics. It is meant to be post-trained to fit a specific environment. As Cosmos is only fit to take in a start and not an end frame, we modified the model to also condition on an end-frame by doubling the condition images with a temporal shift.

We also fine-tuned a 2B-parameter latent diffusion model, LTX-Video HaCohen et al. (2024), on live demonstrations of the PushT task. We originally aimed to finetune Cosmos on the PushT dataset to increase trajectory-adherence and bring the simulation more in-distribution, but realized eventually that the 7B-parameter model was only compatible with training on 8 A100s/H100s and not 8 RTXs.

Our baseline are the simulation PushT videos from Columbia's dataset. To mimic them as a video, we took the frames and loaded them into the action-extraction model that same way we would a video.

To carry on to the next step, we generate 20 videos of the PushT task conditioned on a textual prompt, start frame, and end frame. The start and end frames are extracted from the Columbia PushT dataset. Prompts can be found in Table 1 and frames from a sample generation can be seen in Figure 3.

3.5 Comparing BC Policies trained on AI-generated videos vs. PushT Dataset

Once we have chosen the best behavior cloning policy architecture from Section 3.2, we can use that architecture to perform experiments to compare the baseline with our inverse dynamics models and the AI generated videos.

Specifically, we train an additional three policies to compare with each other and the baseline.

1. PushT images with inverse dynamics actions
2. AI generated videos with inverse dynamics actions

We evaluate these again by taking a look at their loss curves and then reporting MSE and R^2 on an unseen test set.

Then, for each policy, we conduct rollouts in a PushT gym environment. The rewards are computed based off of the position of the agent with respect to the T.

4 Experimental Setup

We describe the experimental setup for the behavior cloning policy training, video generation, inverse dynamics action extraction, and a comparison between models using these different components.

4.1 Behavior Cloning Architecture/Hyperparameter Search

When building our baseline, we experiment with different hyper parameters, model architectures, and epochs for training.

We train a behavior cloning policy on the simulation PushT dataset using a convolutional neural network (CNN) encoder followed by a fully connected prediction head. The encoder consists of two convolutional layers with ReLU activations, and the head maps the flattened features to the action space of the environment. The model is trained for 25 epochs using mean squared error loss.

We similarly finetune ResNet18 over 25 epochs with the same PushT image action data.

For each model, we saved the model at each epoch, and logged the training loss to visualize the training curve in Tensorboard. Our goal in this experiment is to determine which model architecture works best for us to use to train future policies.

4.2 Action Extraction from Videos With UVA

We extract actions from PushT videos as described in section 3.3 using Unified Video Action's model checkpoint that is finetuned on the PushT dataset. Our experiment chooses between extracting actions via averaging, sum, or choosing the first action outputted the model given the input frames to determine the best method for extracting actions. Because the model requires 16 frames to output actions, the original dataset of 25650 frames is reduced to 22414 frames.

4.3 Video Generation

Cosmos videos were generated on 8 RTXs with 400GB VRAM, with an inference time of $\tilde{7}$ minutes per video. The intended setup was A100s/H100s, so extra efforts were made to get ML attention packages working on the RTXs. As the package is new and experimental, we remained in communication with the Cosmos team, informing them of bugs to document and asking for advice on compute constraints.

LTX was fine-tuned on 4 RTXs with 400GB VRAM for 2 epochs on 576 videos of live PushT demonstrations with a learning rate of 2e-4, 1500 steps per batch, a batch size of 1, gradient accumulation, and a first-frame conditioning of 0.1. Due to memory constraints, we also used bf16 mixed_precision, fp8 quantization, and loaded the encoder in 8bit.

Before training the policy, we wished to qualitatively evaluate videos generated from our three video models versus the baseline videos from the Columbia datasets.

We scored generations on:

1. *Adherence to task* (0-2 scale; task not completed, task completed not by robot/human, task completed by robot/human)
2. *Agent Responsibility* (0-1 scale; Were the actions directly caused by the agent?).
3. *Physics adherence* (1-5 scale; Does this follow physics properties?)

Once we have determined which model is the strongest in our Baseline Experimentation, and that our actions can be effectively extracted, we use that model to train a policy on our generated datasets.

4.4 All Model Comparisons

After we have performed our experiments on the video generation and action extraction models, we then want to train policies and compare their performance.

In comparing each of these models, we evaluate each policy across 100 test rollouts in the PushT gym environment. We use UVA's PushTImageRunner and set a maximum of 200 steps per episode

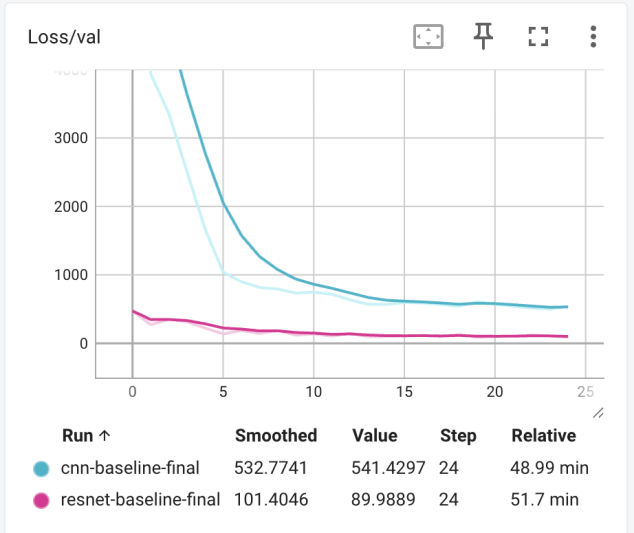

Figure 4: Training MSE loss curves for baseline PushT image-action models (Blue: CNN, Pink: ResNet18)

and number of observation steps to 1 (this accounts for the policy outputting one action). We report the average reward per step across episodes and qualitative observations.

5 Results

5.1 Behavior Cloning Policy Baseline

The training curves for the two first baseline models are shown in Figure 4. The validation MSE loss for the CNN baseline is about 540, whereas for the ResNet it is about 90. We then also test our model on an unseen test set, and achieve the results reported in Table 2.

Metric	PushT Dataset: CNN	PushT Dataset: ResNet18
Test MSE	419.02	106.1093
Test R^2	0.9568	0.9891

Table 2: Baseline Model Test Set Evaluation Metrics

We can tell that the ResNet model performs noticeably better, and so we use that architecture moving forwards.

5.2 Adapting and Evaluating Unified Video Action's Inverse Dynamics Model

Because UVA's inverse dynamics model returns sequences of 8 actions by default for a given set of frames, we first must determine the best method for reducing the 8 step action vector into 1 step. We compare the L_2 distance between the PushT dataset actions and the resulting actions from the inverse-dynamics model and report the average and standard deviation of these distances across frames. We find that both mean-reduction $(M = 122.56, SD = 67.67)$ and first-element $(M = 137.42, SD = 73.96)$ methods perform similarly. Because mean-reduction has both a closer distance to PushT dataset actions and also has a smaller standard deviation, we choose **mean-reduction** for generating actions moving forward.

However, it is still worth noting that an average L_2 distance of 122.56 between $2-$dimensional actions is somewhat substantial given that that x, y are constrained between 0 and 512 – it is about 16.8% of the maximum distance between actions that can exist. Because of this, we also compare mean-reduction PushT to original PushT with respect to training BC policies and simulation rollouts as a second baseline to temper our understanding of how our AI-generated videos perform.

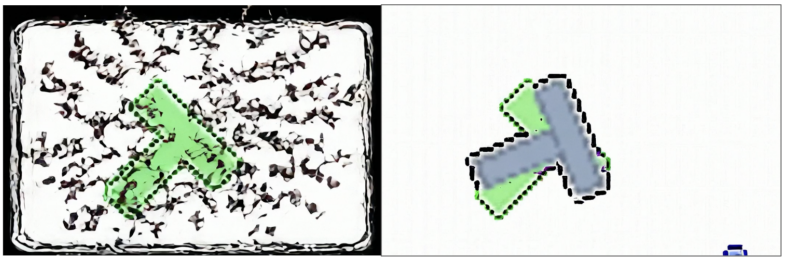

Figure 5: Frames sampled from the training process on LTX-Video at Iterations 500 and 2500.

5.3 Video Quality

We discuss the quality of the AI generated videos with respect to our baseline PushT dataset via qualitative observations, human-rated quantitative results, and quantitative action analysis.

5.3.1 Qualitative Observations

With the base model, initial generations with the video generation models held strongly to the start and end frames provided. The trajectory to get from start to end did not exhibit the text-conditioned requests, though. As we are using a diffusion model, we often saw shapes *morphing* to fit the final configuration rather than moving across the screen. We believe this is because these models have not been trained on simulation data, and therefore shapes moving in an abstract space is an out-of-distribution task. Some prompting helped, such as mandating that no figure changes size or morphs, but then other behavior emerges, such as object splitting (the two parts of the T). Figure 3 features an example of such a rollout.

We tracked the quality of validation generations throughout the fine-tuning process on LTX-Video. Since LTX-2B is a smaller model and trained more-so for content creation, initial generations were chaotic. Figure 5 shows frames from inference generations over the training process. By iteration 2500, the chaos defused and the movement of the T-shape was more controlled, but it still moved independently of the blue dot (agent). Visually, we achieved similar results to base Cosmos with the short fine-tune process. See the videos here.

5.3.2 Human Quantitative Evaluation

Table 3 shows our human-evaluated scores on the dataset of Cosmos-generated videos across the three metrics.

Metric	Cosmos Score (/20)
Adherence to task	19/20
Responsibility	4/20
Physical adherence	6/20

Table 3: Human evaluation metrics for AI-generated videos

Overall, task adherence was high, with the agent being directly responsible for the movement in 4/20 cases. Physical adherence was off, with common occurrences of the agent splitting in two and morphing.

5.3.3 Quantitative Action Analysis

We report the smoothness and directness metrics described in 3.3 on the original PushT dataset, the inverse dyanmics PushT dataset, and the Cosmos AI-generated videos. For the latter two sets of frames, we extracted the actions using the UVA inverse dynamics model and mean-reduction method described above. Because we only generated 20 usable videos with NVIDIA Cosmos-Predict1, we augment the dataset with flips and rotations, resulting in a final image-action dataset of 15456

frames. This is used to train the policy. However, when analyzing the direct actions, we use the non-augmented dataset of 1104 frames.

First, we discuss the smoothness metrics; see Table 4 for the full results. We can clearly see that the original PushT actions are far more smooth ($SD_{\text{Velocity}} = 8.32$, $M_{\text{Acceleration}} = 5.36$, $M_{\text{Jerk}} = 7.61$) than the inverse dynamics actions ($SD_{\text{Velocity}} = 15.44$, $M_{\text{Acceleration}} = 51.02$, $M_{\text{Jerk}} = 93.03$) for the same dataset. The Cosmos Actions and the inverse dynamics PushT datasets perform similarly, but Cosmos is slightly worse.

Dataset	M Velocity	SD Velocity	M Acceleration	SD Acceleration	M Jerk
Cosmos Actions	30.60	16.13	52.88	27.71	96.83
Inverse Dynamics PushT Actions	29.56	15.44	51.02	26.60	93.03
Original PushT Actions	10.12	8.32	5.36	6.17	7.61

Table 4: We report the average velocity, acceleration, and jerk averaged across episodes and calculated between frames.

Next, we discuss the directness metric for the original PushT dataset, the inverse dynamics PushT dataset, and the Cosmos AI-generated videos. Clearly the original PushT actions are far more direct ($M = 9.37$) than the actions extracted via the inverse dynamics model on the same dataset ($M = 96.61$). Further, it is interesting that the Cosmos extracted actions ($M = 52.54$) are actually more direct than the PushT extracted actions.

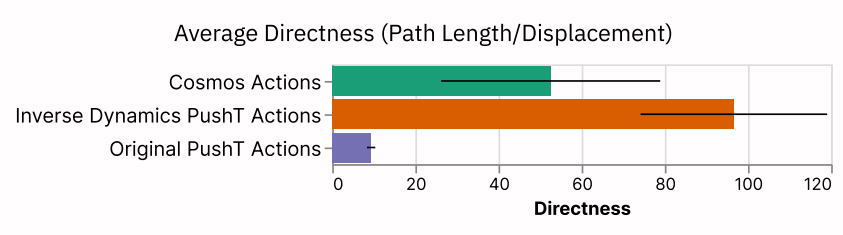

Figure 6: We report the average directness (path length/displacement) across episodes.

5.4 Inverse Dynamics Actions and Cosmos Generated Video Models

We use the selected model from Section 5.1 to train a ResNet18 model using actions extracted from the PushT images using our inverse dynamics model. We use this same inverse dynamics model to extract actions from our Cosmos videos and train a ResNet18 model. The loss curves of these two models compared to our baseline ResNet18 model are shown in Figure 7.

The validation loss for the inverse dynamics PushT policy is about 430 compared to 320 for the Cosmos actions.

In training, the ResNet baseline performs the best with validation MSE of 90, which is expected since the data is collected in real simulation. The inverse dynamics PushT model and the Cosmos actions model perform pretty similarly. The test performance, shown in Table 5, also confirms that the baseline performs better than the other models.

An interesting observation is that the Cosmos actions policy actually performs better than the inverse dynamics PushT policy. This is surprising because according to our human qualitative evaluation (Section 5.3.2), many of the generated videos don't even follow physics properties. We wonder if the better performance is due to the fact that the Cosmos dataset is filled with many augmentations of

Metric	PushT Dataset Baseline	Inverse Dynamics PushT Model	Cosmos Actions Model
Test MSE	99.80	339.9058	333.5624
Test R^2	0.9897	0.5653	0.9570

Table 5: Test set evaluation metrics with ResNet18

Figure 7: Training loss curves for baseline ResNet (pink), inverse dynamics PushT actions model (orange), and cosmos actions model (green)

the same videos, so the model could just be fit very well to that set of videos. Although the test set is technically "unseen", the policy may have seen a different augmentation in the training process. Ultimately, the fact that the difference in training and testing outcomes occurs when comparing simulation actions to extracted actions, suggests that AI video generation holds a lot of promise.

5.5 PushT Environment Evaluation

We report the average reward per step via the histogram in Figure 8. Overall, the Cosmos Policy performed the best ($M = 0.0804$) and the PushT Policy and Inverse Dynamics PushT Policy performed similarly, but slightly worse ($M = 0.0619$ and $M = 0.0625$ respectively). A similar histogram for the maximum reward per episode can be seen in Appendix Section A.0.1.

We also perform a qualitative analysis by analyzing the policy's behavior in a simulated PushT environment. From 8 we notice there are some episodes with high rewards, so we take a closer look at those episodes.

Those episodes with high reward achieve high reward even when the T block covers the target block partially, even when not oriented well. In Figure 9b, the grey T's position was already already overlapping with the green T, and despite the agent not touching the block at all, there is positive reward. When more of the green T's surface area is covered, the reward is also greater.

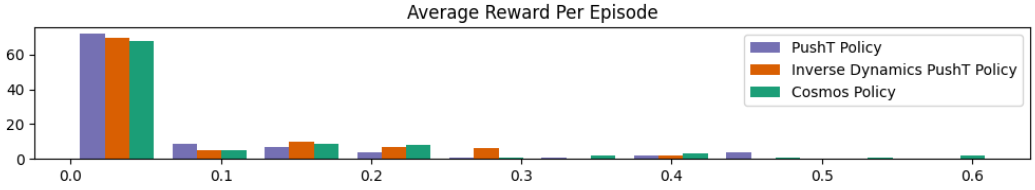

Figure 8: Histogram of Average Reward Per Step

(a) Final frame where agent successfully pushes Grey T onto Green T but not aligned (Reward: 0.61479, Model: Cosmos action policy)

(b) Final frame where Grey T starts already overlapping with Green T (reward: 0.40618, Model: Inverse Dynamics PushT action policy)

Figure 9: Example PushT simulation evaluation videos

We noticed that the end frame of the inverse dynamics PushT policy always shows the agent in the exact same spot. As shown in Figure 10, no matter where the T's are, the end frame is always in the bottom left of the frame. This is not the case with our baseline policy or the Cosmos action policy, so we wonder if this is since we did not augment our videos like we did for the Cosmos policy.

This might explain why our inverse dynamics action model performs worse than the Cosmos action model, despite the Cosmos videos being poorer quality.

Ultimately, many of the videos for all policies did not show the agent pushing the block into place.

6 Discussion

6.1 Challenges

The first issue we encountered was a missing crucial package (Apex) in the inference installation for Cosmos. We identified the correct package and informed the team, who had received notification of the same bug earlier that day from their team as well.
We previously underestimated the compute constraints of running inference and post-training a video generation model. Though we had access to a cluster with 8 GPUs, they were not a compatible kind, and we spent quite some time finding and trying our implementation on different borrowed A100s (AWS, RunPod, SF Compute, etc). Eventually, we got inference working on the RTX by manually

Figure 10: Inverse dynamics PushT policy unusual end frame with agent in the exact same spot

configuring attention parameters, but not post-training, as that requires several more Deep Learning modules.

6.2 Limitations

We concluded that generated videos are not helpful for training unless fine-tuned extensively on the particular task and environment, in agreement with others in the field.

While most foundation models condition on text and an image, we found it useful to condition on text and two (start and end) images. We hope to see more models with this feature.

Human evaluation and training a policy were the only ways we assessed the quality of videos. We were able to do more to evaluate the actions themselves, but had we had higher-quality or a larger number of generations, we may have considered evaluating actions directly from the video with a VLM.

After receiving feedback at the poster session, we decided to fine-tune a lighter-weight video diffusion model, hence choosing the 2B-parameter version of LTX-VideoTrainer. However, as all of these research repositories are relatively new and unmaintained, we once again had difficulties loading environments to properly configure training, leaving us minimal time for training.

7 Conclusion

While AI-generated videos present an exciting opportunity for introducing synthetic data to robot learning tasks, there is still a lot of progress to be had in order to generate usable simulation videos to train policies. By focusing on the abstract PushT task in a simplified two-dimensional environment, we are able to highlight and analyze the many failure modes that can come up through a multi-step pipeline involving diffusion models and inverse dynamics models. For example, with regards to the inverse dynamics model, future work might examine training a policy that generates 8 step trajectories, in line with UVA's setup, or using UniPi instead of UVA for an inverse dynamics model. Following work would emphasize fine-tuning and experiment with different configurations as well. For example, we could evaluate the generalizability of fine-tuning on a single task (like PushT) on unseen manipulation tasks, like opening/closing drawers. Something interesting we want to explore more is how video generation is impacted by expected trajectory length; that is, if a task needs several pushes from the robot arm or an extended number of actions, how does the video handle this? Does it rush through them, morphing moves together, does the task get cut off? As these Physical AI models are built for training policies, it would be interesting to analyze the action-extractability of the generated content. These are all high-resource experiments, of course.

8 Team Contributions

- **Ashna Khetan:** Ashna generated videos from Cosmos, tried fine-tuning on Cosmos, fine-tuned on LTX-Video, and helped with human evaluation of videos.
- **Daphne Liu:** Daphne performed all the behavior cloning experiments including infrastructure for Tensorboard logging, test set evaluation, and performing qualitative analysis of the policies on the PushT environment rollouts.
- **Poonam Sahoo:** Poonam extracted actions using UVA's PushT checkpoint inverse dynamics model, created all image-action datasets for policy training, set up boilerplate code for training a BC policy, ran rollouts for BC policies in a PushT environment (and modified UVA PushTRunner code to collect average and maximum rewards), and evaluated actions via smoothness and directness metrics.

Changes from Proposal Initially we were extremely ambitious with our scope and planned to compare videos with human movements vs. videos with robot movements and then extract actions, train a policy, and evaluate respective policies. However, we found that videos with human motions were extremely erratic for certain physics-aware tasks and struggled with the compute and time required to finetune a inverse dynamics model along with the video generation tasks from before. Because of this, we scoped down to a simpler task (PushT) and evaluated in simulation environments.

References

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. 2022. Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos. arXiv:2206.11795 [cs.LG] https://arxiv.org/abs/2206.11795

Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. 2024. Gen2Act: Human Video Generation in Novel Scenarios enables Generalizable Robot Manipulation. arXiv:2409.16283 [cs.RO] https://arxiv.org/abs/2409.16283

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. 2023. Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. arXiv:2304.08818 [cs.CV] https://arxiv.org/abs/2304.08818

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. 2024. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *The International Journal of Robotics Research* (2024).

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. 2023. Learning Universal Policies via Text-Guided Video Generation. arXiv:2302.00111 [cs.AI] https://arxiv.org/abs/2302.00111

Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. 2021. Implicit Behavioral Cloning. arXiv:2109.00137 [cs.RO] https://arxiv.org/abs/2109.00137

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. 2024. LTX-Video: Realtime Video Latent Diffusion. arXiv:2501.00103 [cs.CV] https://arxiv.org/abs/2501.00103

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV] https://arxiv.org/abs/1512.03385

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, Loic Magne, Ajay Mandlekar, Avnish Narayan, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Yinzhen Xu, Xiaohui Zeng, Kaiyuan Zheng, Ruijie Zheng, Ming-Yu Liu, Luke Zettlemoyer, Dieter Fox, Jan Kautz, Scott Reed, Yuke Zhu, and Linxi Fan. 2025. DreamGen: Unlocking Generalization in Robot Learning through Neural Trajectories. arXiv:2505.12705 [cs.RO] https://arxiv.org/abs/2505.12705

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. 2025. Unified Video Action Model. arXiv:2503.00200 [cs.RO] https://arxiv.org/abs/2503.00200

NVIDIA, :, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. 2025a. Cosmos World Foundation Model Platform for Physical AI. arXiv:2501.03575 [cs.CV] https://arxiv.org/abs/2501.03575

NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. 2025b. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. arXiv:2503.14734 [cs.RO] https://arxiv.org/abs/2503.14734

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. 2017. MoCoGAN: Decomposing Motion and Content for Video Generation. arXiv:1707.04993 [cs.CV] https://arxiv.org/abs/1707.04993

Yo whan Kim, Samarth Mishra, SouYoung Jin, Rameswar Panda, Hilde Kuehne, Leonid Karlinsky, Venkatesh Saligrama, Kate Saenko, Aude Oliva, and Rogerio Feris. 2023. How Transferable are Video Representations Based on Synthetic Data? https://openreview.net/forum?id=lRUCfzs5Hzg

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. 2021. VideoGPT: Video Generation using VQ-VAE and Transformers. arXiv:2104.10157 [cs.CV] https://arxiv.org/abs/2104.10157

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. 2025. Latent Action Pretraining from Videos. arXiv:2410.11758 [cs.RO] https://arxiv.org/abs/2410.11758

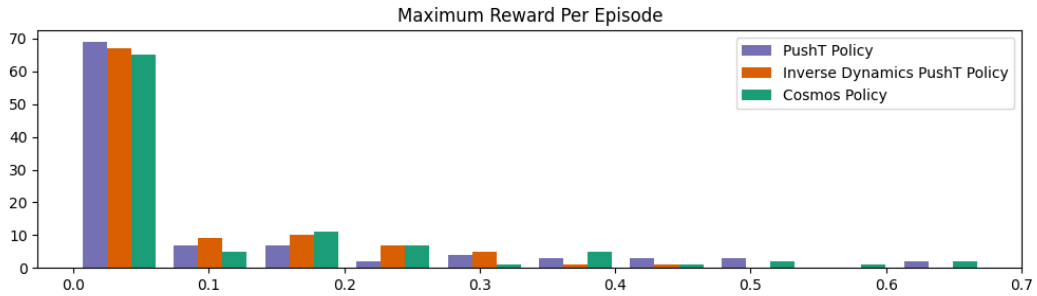

Figure 11: Histogram of Maximum Rewards Per Episode

A Additional Plots

A.0.1 Histogram of Maximum Rewards Per Episode

B Additional Experiments

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

C Implementation Details

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.