Unified Video Action Model

Shuang Li **Yihuai Gao** **Dorsa Sadigh** **Shuran Song**
Stanford University

https://unified-video-action-model.github.io/

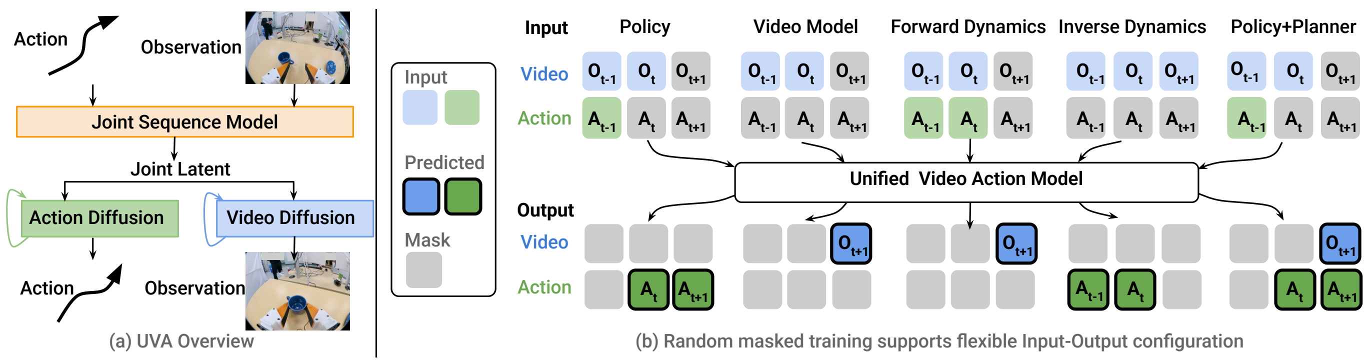


Fig. 1: **Unified Video Action Model.** (a) UVA features a *joint* video-action latent representation and *decoupled* video-action decoding. The joint latent representation effectively models the underlying relationships between video and action sequences, while the decoupled diffusion enables high-speed action inference by bypassing video generation. (b) By leveraging masked training, UVA support flexible input-output combinations for actions and videos. This versatility allows the model to function as a robot policy, video model, forward or inverse dynamics model, or even a combined policy and video planner—all within a unified framework.

***Abstract*—A unified video and action model holds significant promise for robotics, where videos provide rich scene information for action prediction, and actions provide dynamics information for video prediction. However, effectively combining video generation and action prediction remains challenging, and current video generation-based methods struggle to match the performance of direct policy learning in action accuracy and inference speed. To bridge this gap, we introduce the Unified Video Action model (UVA), which jointly optimizes video and action predictions to achieve both high accuracy and efficient action inference. The key lies in learning a joint video-action latent representation and decoupling video-action decoding. The joint latent representation bridges the visual and action domains, effectively modeling the relationship between video and action sequences. Meanwhile, the decoupled decoding, powered by two lightweight diffusion heads, enables high-speed action inference by bypassing video generation during inference. Such a unified framework further enables versatile functionality through masked input training. By selectively masking actions or videos, a single model can tackle diverse functions beyond policy learning, such as forward and inverse dynamics modeling and video generation. Via an extensive set of experiments, we demonstrate that UVA can serve as a general-purpose solution for a wide range of robotics tasks without compromising performance compared to methods tailored for specific applications. Results are best viewed on our website.**

I. Introduction

A unified video and action model that jointly learns an agent's actions and their effects on visual observations holds great promise for robotics – videos provide rich environmental context for predicting actions, while actions reveal how interactions drive visual changes, enabling more accurate modeling of real-world dynamics. However, despite its promise, previous approaches have often failed to fully realize this potential. A key challenge lies in the inherent mismatch between the requirements of action and video generation. Action modeling demands **high temporal speed** to capture dense, fine-grained motions, while video generation requires **high spatial resolution** to produce high-fidelity visual outputs, which often results in slower processing speeds.

Previous policy learning approaches have struggled to balance these conflicting requirements, often focusing on one aspect at the expense of the other. For instance, *action only* methods like [9, 22, 48] bypass video generation entirely. While such approaches reduce computational complexity, they overlook the benefits of video generation – adding observation supervision helps the model learn scene dynamics, which reduces overfitting to action history and enhances robustness to visual disturbances. On the other hand, *video generation* methods such as [12, 25] often first generate high-resolution videos and then predict actions based on the generated videos. While this hierarchical approach can utilize existing video models, it also introduces significant drawbacks, including slower processing speeds and the propagation of errors from the generated video into action prediction.

To address these limitations, we propose **UVA**, a **Unified Video and Action** Model designed to simultaneously model videos and actions – capturing the underlying interactions between visuals and actions to enhance task understanding, while maintaining high-speed action prediction during inference. We propose the following three design choices to achieve this:

1) Unified Latent Video-Action Representation: UVA introduces a unified latent representation that integrates both visual and action data. Unlike traditional *video generation* based policy methods which rely on a hierarchical video and action generation, UVA is trained simultaneously with supervision from both video and action data. This enables the model to capture the intricate dynamics shared between the visual and action domains with reduced computational overhead. Utilizing the rich scene information encoded in the latent representation unlocks UVA's superior performance in understanding complex environments and delivering precise action predictions.

2) Decoupled Video-Action Diffusion for Fast Inference: To further enhance efficiency and achieve inference speed comparable to *action-only* methods, UVA decouples video generation from action prediction. During training, the model employs two lightweight diffusion heads to decode video observations and actions from the unified latent space. At inference, this decoupling allows the system to bypass video generation entirely, directly utilizing the latent representation for fast action prediction. This design enables real-time policy deployment without sacrificing performance, as it still retains the rich representations learned during training from both visual motions and robot action trajectories.

3) Mask Training for Flexibility: The ability to predict both videos and actions through unified representations further unlocks the potential to perform a diverse set of functions using masked training. UVA can handle versatile functions that go beyond traditional policy learning by masking inputs and outputs as needed, as illustrated in Figure 1. This versatility enables the model to tackle complex scenarios, such as operating as a forward or inverse dynamics model, learning effectively from video-only datasets where action labels are unavailable, or simultaneously performing both low-level control and high-level planning.

We evaluate UVA on seven publicly available benchmarks to assess its diverse capabilities. UVA outperforms or matches state-of-the-art baselines, demonstrating particularly strong performance in multi-task settings. For instance, UVA outperforms the best baseline by 13% in success rate on Libero10 [27] and by 20% on PushT Multitask [9, 14]. The experiments show that UVA can serve as a general-purpose framework for different robotics tasks without compromising performance compared to methods tailored for specific applications. In sum, the unified video action model is:

- **Capable:** UVA matches the state-of-the-art approaches that are tailored for robot policy learning [9, 22] or planning [12], especially for multi-task learning.
- **Practical:** The use of decoupled diffusion heads eliminates the need for video generation during policy inference, and the use of lightweight diffusion heads reduces the costs of the denoising process. As a result, UVA achieves a similar speed as Diffusion Policy [9], making it practical for robot applications.
- **Versatile:** Beyond policy learning, UVA can also serve as a forward dynamics model for planning, as an inverse dynamics model to generate actions, a video generation model, or a combined policy and video planner.

II. Related Work

There is a rich literature on video generation where the dominant approach includes diffusion-based methods [2, 3, 16, 18–20, 34–37] or autoregressive-based methods [11, 15, 42–44, 47]. Our model utilizes the ideas from both of these techniques, as we will discuss in section III. Another line of related work involves the use of masked training in robot learning. In this section, we will review both video generation and masked training approaches used in robotics.

Video Generation for Policy Learning: Video models aid policy learning by simulating task dynamics and predicting future states. Models like [12, 25] leverage video generation techniques to produce high-quality videos, which are then used for action prediction. The work by [46] leverages video models to generate object flow as an intermediate representation, which captures physical interactions and is used to predict actions for skill transfer across different robotic embodiments and environments. In [21], a video diffusion model is fine-tuned on robotics tasks, with the latent representations from the predicted videos serving as inputs to a policy network for action prediction. Existing approaches that utilize video generation for policy learning often suffer from slow inference speeds, making them impractical for real-world applications. Additionally, these methods often require auxiliary components, such as low-level policies [12] or image-tracking techniques [46], to extract actions from the generated videos. As a result, the final action accuracy suffers from compounded errors in video generation and action prediction.

Video Generation as Dynamics Models: Video models can serve as dynamics models by predicting future states conditioned on current observations and actions, enabling robots to simulate and plan tasks. GameGen-X [8] introduces a diffusion transformer model for generating and controlling open-world game videos, enabling interactive simulations. Genie [4] utilizes a foundation world model to transform static images into interactive 3D environments, providing rich simulations for embodied agents. Additionally, [40] demonstrates how diffusion models can act as real-time game engines, generating dynamic and interactive scenarios to facilitate decision making. These advancements highlight the versatility of video generation models in robotic applications. In this work, we propose a unified video and action model, showcasing its ability to address both policy learning and dynamics modeling within a single framework.

Masked Training: Recent works in robotics have explored masked training techniques [28, 31, 45]. For example, Liu et al. [28] and Wu et al. [45] randomly mask observations and actions and reconstruct the missing portions. Their results show that masked training improves generalization to downstream tasks and enables the model to be used for various applications. However, these methods primarily rely on low-dimensional state observations rather than videos, which are more natural but harder to predict. Radosavovic et al. [31] first pretrain a model to predict actions or observations using masked training and then finetune the model or use a linear

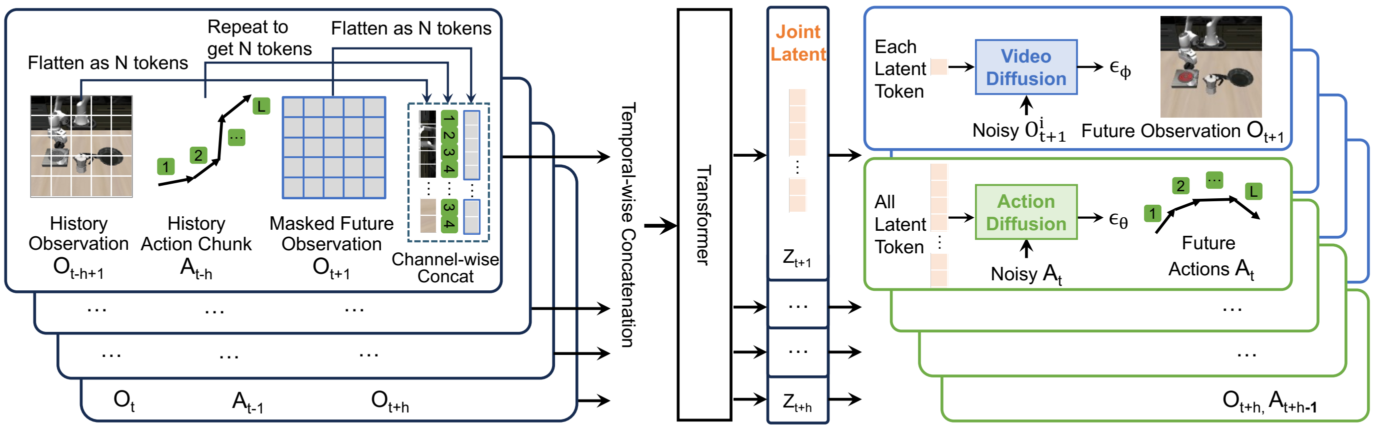


Fig. 2: **Network Architecture.** Given historical observations $\{O_{t-h+1}, \dots, O_t\}$ and corresponding action chunks $\{A_{t-h}, \dots, A_{t-1}\}$, the model predicts future observations $\{O_{t+1}, \dots, O_{t+h}\}$ and actions $\{A_t, \dots, A_{t+h-1}\}$. Each image observation is represented as a sequence of N tokens, and each image corresponds to an action chunk with L actions sampled at a higher frequency. During training, future observations are used with randomly masked tokens, while at inference time, the model starts from an empty image to predict the complete image. History observations, actions, and masked future observations are combined, passed through a Transformer, and decoded into actions and videos using diffusion heads.

probe for downstream tasks. Their work focuses solely on action prediction results. In contrast, our method does not require finetuning for downstream tasks and can be directly applied to various functions beyond policy learning, such as video generation, forward dynamics, and inverse dynamics.

III. Unified Video Action Model

In robotics, we are interested in learning generalizable policies that map observations to actions. However, this objective often tends to overfit the training data, thereby limiting the ability of learned policies to adapt to new scenarios. In contrast, video generation [3, 33] demonstrates strong generalization to novel scenes and supports training on datasets without actions. However, effectively leveraging video data for policy learning presents challenges such as the ability to match the high temporal speed required for outputting dense, fine-grained motions. In this section, we discuss our approach to leveraging video-generation methods for robotics tasks.

Problem Statement: Given a sequence of image observations $\{\mathbf{O}_{t-h+1}, \dots, \mathbf{O}_t\}$ and action chunks $\{\mathbf{A}_{t-h}, \dots, \mathbf{A}_{t-1}\}$, where h is the history horizon, our goal is to predict the future actions $\{\mathbf{A}_t, \dots, \mathbf{A}_{t+h'-1}\}$ and observations $\{\mathbf{O}_{t+1}, \dots, \mathbf{O}_{t+h'}\}$, where h' is the future horizon. Each action chunk, e.g., $\mathbf{A_t} \in \mathbb{R}^{L\times m}$ consists of L actions, and each action has m dimensions. We set $h = h'$ in the experiments. For simplicity, we refer to both as h in the following sections.

We first introduce the model with complete video and action inputs and outputs (§III-A-§III-C). We then discuss how masked training can flexibly learn from any combination of video and action data (§III-D), enabling UVA to perform various functions, including policy learning, video generation, forward and inverse dynamics, and integrated policy and planning.

As shown in Figure 2, our method encodes the history of observations and actions (§III-A), along with masked future observations (§III-B), and passes them to the Transformer [41]. For the masked observations, we randomly mask tokens within the future observation frames during training and train the model to reconstruct them. During inference, the model generates the full set of tokens, starting from an empty sequence. In §III-C, we then discuss the choice of decoupling video-action diffusion for fast inference addressing the high temporal speed demand of robot policies.

A. Encode History

We first process the historical image observations through a pre-trained VAE encoder (kl-f16) [32] to obtain their latent representations. Each image is encoded into a latent map of dimensions $\mathbb{R}^{w\times h\times c}$, where w and h represent the width and height, and c is the latent dimension. The map is then flattened and processed by a fully-connected (FC) layer, projecting each element into a d-dimensional latent vector. Thus, each image is represented as a sequence of N visual tokens, each with d-dimensional features.

For history actions, we use a higher sampling frequency compared to observations, as observations typically exhibit redundancy and minimal changes over short time intervals. Each image observation (e.g., $\mathbf{O}_{t-h+1}$) corresponds to L actions within an action chunk (e.g., $\mathbf{A}_{t-h}$). we repeat the action chunk M times to match the number of visual tokens as shown in Figure 2. The repeated sequence is then passed through an FC layer, and converted into a sequence of N action tokens, each with a d-dimensional latent representation. These history visual and action tokens serve as conditions for predicting future observations and actions.

B. Masked Autoencoder for Observation Prediction

Our work is closely related to [7, 24]. Their method focuses on image generation conditioned on class labels. It begins by generating a subset of visual tokens for the image and then sequentially predicts additional tokens based on the previously generated ones, following an autoregressive process to complete the image. This step-by-step autoregressive approach has been shown to outperform the single-step generation of all visual tokens simultaneously. To facilitate step-by-step prediction, they employ a masked autoencoder [17] framework. During training, some visual tokens are randomly masked, and the model is trained to reconstruct these masked tokens.

We follow this setting for video prediction. Future observation frames $\{\mathbf{O}_{t+1}, \ldots, \mathbf{O}_{t+h}\}$ are processed similarly to historical observations: they are passed through a VAE encoder to extract latent representations, followed by an FC layer, resulting in a sequence of N tokens per frame, each with a d-dimensional latent vector. Some tokens are randomly masked out during training. These visual tokens are concatenated channel-wise with historical visual tokens and action tokens, as shown in Figure 2, to form a new sequence of latent features. Latents from h different time steps are then temporally concatenated with latent representations from other time steps to produce a $N \times h$ latent sequence. The resulting sequence is passed through a Transformer to fuse the video and action information, resulting in a set of joint video-action latent representations, $\{\mathbf{Z}_{t+1}, \ldots, \mathbf{Z}_{t+h}\}$, where each latent (e.g., $\mathbf{Z}_{t+1}$) contain N latent tokens. These joint video-action latent tokens are then used to reconstruct the future observations and corresponding action chunks.

To minimize information leakage across different frames, we consistently mask the same positions across all video frames. At inference time, the model generates complete videos by predicting all tokens starting from an empty sequence. At each autoregressive generation step, visual tokens at the same position across all video frames are generated simultaneously as shown in Supplementary §X-A. Unlike image generation conditioned on class labels or text, historical observations provide rich contextual information about the environment. We found that a single-step generation is sufficient to generate high-quality observations, while using additional steps can further enhance the quality.

C. Decoupled Video and Action Diffusions

Previous video generation-based policy learning methods rely on hierarchically generating videos first and then predicting actions, leading to slow speed and accumulated errors. To address this, we propose decoupling video and action prediction while training them jointly. During training, video generation helps the latent representations $\mathbf{Z}$ capture more detailed scene information, which benefits action prediction. During policy inference, where speed is crucial, the decoupled design allows us to skip video generation and decode only the actions. Similarly, for video generation, where quality is the priority, we can perform multi-step autoregressive video generation while bypassing action decoding.

We introduce two lightweight diffusion decoders for action and video prediction (see Figure 2). Instead of performing the denoising over the entire model [9], our approach restricts the denoising process to the lightweight decoders, delivering more efficient performance. This design preserves the generative strengths of diffusion models while significantly reducing inference time.

The joint latent $\mathbf{Z}$ serves as the conditioning input for the diffusion decoders. The video diffusion decoder processes each latent token $z_i \in \mathbf{Z}_{t+1} = \{z_1, \ldots, z_N\}$ to predict individual patches in the video frame, which are then reshaped and sent to the VAE decoder to reconstruct the full frame $\mathbf{O}_{t+1}$. For the action diffusion decoder, all latent tokens in $\mathbf{Z}_{t+1}$ are aggregated using a convolutional layer, followed by an MLP layer, to produce an action latent. This latent encodes both visual and action-related information for the current step and serves as the condition for the action diffusion model to generate the action chunk $\mathbf{A}_t$. We use the diffusion head (base size) from [24] for both action and video prediction.

During training, the decoders learn to predict the noise added to noisy action chunks or video patches. The action diffusion loss [18, 34, 36] is defined as:

$$\mathcal{L}_{\text{action}}(\mathbf{Z}, \mathbf{A}) = \mathbb{E}_{\epsilon,k}\left[\|\epsilon - \epsilon_\theta(\mathbf{A}^{(k)}|k, \mathbf{Z})\|^2\right],$$

where $\mathbf{A}^{(k)}$ represents the noisy actions, ϵ is the added noise, k is the diffusion timestep, $\mathbf{Z}$ is the joint video-action latent tokens, and $\epsilon_\theta(\mathbf{A}^{(k)}|k, \mathbf{Z})$ is the predicted noise.

Similarly, the video diffusion loss is defined as:

$$\mathcal{L}_{\text{video}}(\mathbf{Z}, \mathbf{O}) = \mathbb{E}_{\epsilon,k}\left[\frac{1}{N}\sum_{i=1}^{N}\|\epsilon_i - \epsilon_\phi(\mathbf{O}^{i,(k)}|k, z_i)\|^2\right],$$

where $\mathbf{O}^{i,(k)}$ represents the i-th noisy visual token in the video frame $\mathbf{O}^{(k)}$ at diffusion timestep k, N is the total number of visual tokens in a video frame, ϵ_i is the added noise to the i-th visual token, z_i is the latent token in $\mathbf{Z}$, and $\epsilon_\phi(\mathbf{O}^{i,(k)}|k, z_i)$ is the predicted noise for the i-th token.

The total loss at each time step is the combination of the action and video diffusion losses: $\mathcal{L} = \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{video}}$. The overall loss is calculated as the sum of these losses over the time horizon h. During policy inference or video generation, the decoders iteratively refine pure noise into actions or videos using the learned denoising process.

D. Masked Training with Flexible Objectives

Instead of training the model solely on the task of predicting future observations and actions based on historical data, we propose a masked training approach with multiple training objectives using a unified framework. As illustrated in Figure 1, the model is trained on five distinct tasks by varying input and output combinations. Unused components are masked and replaced with a learned mask token. The action loss and video loss are selectively applied to supervise the model depending on the specific task.

This training approach enables us to fully utilize the data in various combinations and supports the use of incomplete data, such as video data without corresponding actions. This masked training strategy enables the model to perform a diverse range of functions, including acting as a robot policy, video model, forward and inverse dynamics model, and a combined policy and planner. For instance, when given only image observations, the model can function as an inverse dynamics model to generate action labels from videos. Additionally, this strategy helps prevent overfitting to specific tasks, enhancing the model's overall versatility and robustness.

IV. Evaluation

In the following sections, we evaluate UVA's capacities as policy §V, a video generator §VI, a forward dynamics model §VII, and finally an inverse dynamics model §VIII. In each scenario, we compare UVA with methods that are specifically tailored for the corresponding application.

V. UVA as Policy

We first investigate the effectiveness of UVA on policy learning. As noted by Kim et al. [22], different policy designs excel in different settings. Their experiments show that while Diffusion Policy [9] performs better in single-task setups, it falls behind OpenVLA in multi-task scenarios. To comprehensively evaluate the performance of UVA as a policy, we conduct extensive evaluations in single-task and multi-task settings, and in simulated (Figure 3) and real environments (Figure 4). For simulation tasks, we used the same random seeds for different methods for a fair comparison. **For real-world tasks, to minimize evaluation bias, all evaluations use public benchmarks with released datasets**—no additional training data were collected.

A. Simulation Benchmarks

Single-Task Evaluation: We first evaluate single-task scenarios, where different policies are trained for different tasks. We compare UVA with the baselines on the PushT [9, 14] and Toolhang [29] tasks. We report the success rates of the best-performing checkpoint, averaging across 50 rollouts for PushT and Toolhang, respectively.

Multi-Task Evaluation: We train one policy for multiple task goals defined by image or text. We introduce a new task, PushT-M, which extends the PushT task to include varying target "T" positions. We evaluate the best-performing checkpoint over 50 rollouts and report its average reward. Libero10 [27] has 10 tasks. We evaluate each task in three different environments with varying random seeds and report the average rewards across all 10 tasks. See Supplementary §X-B for details.

B. Real-world Benchmarks

Training Data: We use two publicly available datasets introduced by [10] and [26] without collecting any additional training data. Both benchmarks collect data using the handheld UMI [10] device. We used three tasks, including Cup Arrangement, Towel Folding, and Mouse Arrangement, for training, and tested them on the ARX X5 arm.

Single-Task Evaluation: We train a single-task policy on the Cup task and directly compare it with the Diffusion Policy model provided by the author [10], both trained on the same data. We evaluate each method over 20 rollouts with varying initial configurations and report the average success rate.

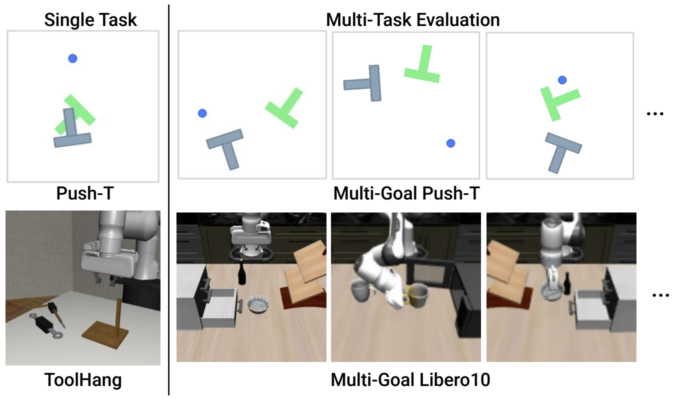


Fig. 3: **Simulation Environments.** We evaluate UVA and baselines in both single-task and multi-task settings. In the multi-task scenario, the goal can be defined through the image observations (PushT-M) or language descriptions (Libero10).

	Single-Task ↑		Multi-Task ↑		Speed ↓
	PushT	Tool	PushT-M	Libero10	
DP-C [9]	0.91	**0.95**	0.68	0.73	0.50s
DP-T [9]	0.78	0.76	0.63	0.80	0.36s
OpenVLA [22]	0.35	0.18	0.22	0.47	1.52s
UniPi [12]	0.42	0.00	0.19	0.00	24.07s
UVA-action	0.45	0.62	0.46	**0.93**	**0.22s**
UVA	**0.98**	0.88	**0.88**	**0.93**	0.23s

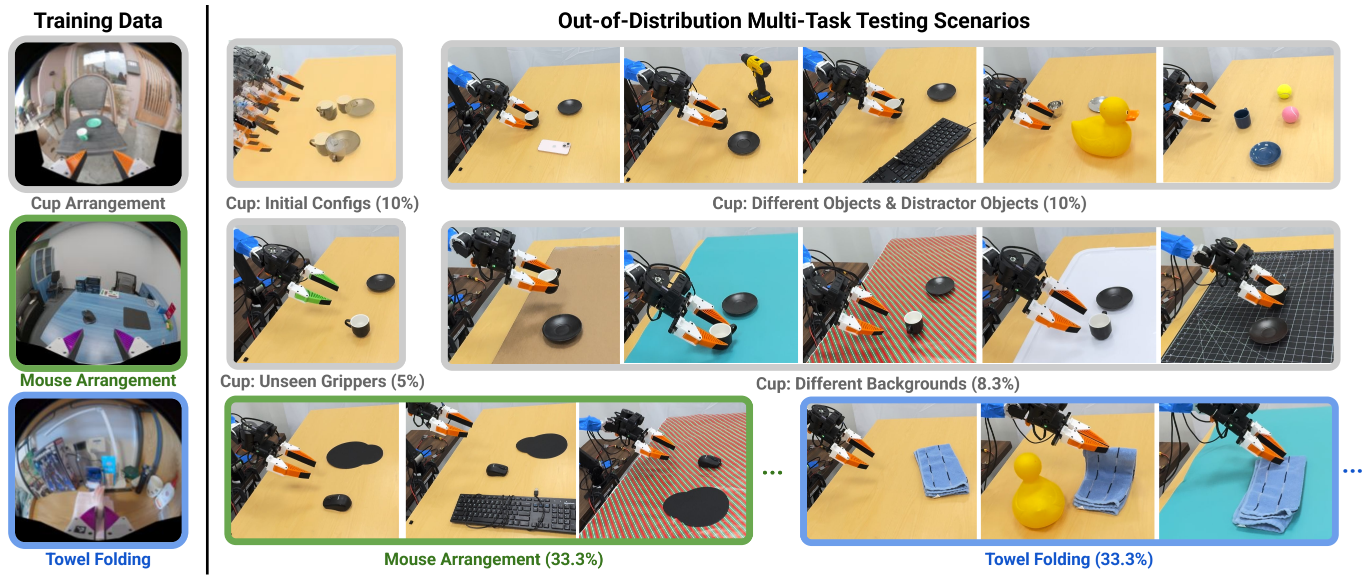
TABLE I: **Policy Learning Results in Simulation.** UVA has higher success rate than the baselines in most settings, with a strong performance in multi-task scenarios. Speed is measured by a single action trajectory inference. All methods, except OpenVLA, infer 16 action steps per trajectory with 8 executed steps. OpenVLA infers one action at a time, so it is run 8 times to match the inference time for 8 executed actions.

Multi-Task Evaluation: We train one model with all three tasks and then evaluate their performance on each task independently. We randomly selected 500 episodes from each dataset and combined them into a dataset to train both our model and Diffusion Policy [10]. Since the training data were collected independently in prior works, all evaluation cases are **Out-of-Distribution** (OOD), involving unseen environments, objects, and robots. To ensure a wide testing distribution, we include cases with varying initial configurations, object distractors, background textures, and an unseen gripper color (green), as shown in Figure 4. Each policy is evaluated over 60 rollouts, with 20 rollouts per task. See Supplementary §X-C for details.

C. Baselines

We compared with the following alternative methods, all methods are trained or fine-tuned on the same data as our model and tested using the same random seed and initial states.

- **Diffusion Policy** [9] is a state-of-the-art visuomotor policy model. We use both CNN-based network [DP-C] and Transformer-based design [DP-T] from their original implementation for all simulation tasks. For real-world



Training Data
Out-of-Distribution Multi-Task Testing Scenarios
Cup Arrangement
Cup: Initial Configs (10%)
Cup: Different Objects & Distractor Objects (10%)
Mouse Arrangement
Cup: Unseen Grippers (5%)
Cup: Different Backgrounds (8.3%)
Towel Folding
Mouse Arrangement (33.3%)
Towel Folding (33.3%)

Fig. 4: **Real-World Out-of-Distribution Evaluation.** We use the training data provided by prior works [10, 43]. The test scenario is significantly out-of-distribution with unseen environments, objects, robots, and even gripper colors. The numbers in the parentheses show the percentage of such category of test cases in evaluation. Please refer to our website for all evaluation rollouts.

	Single-Task ↑	**OOD Multi-Task ↑**			**Speed ↓**
	Cup	Cup	Towel	Mouse	
DP-UMI [10]	**0.95**	0.50	**0.70**	0.40	**70ms**
UVA	0.85	**0.65**	**0.70**	**0.80**	95ms

TABLE II: **Success Rate on Real-World UMI [10] tasks.** We compare UVA with DP-UMI which is designed for UMI tasks. UVA performs worse than DP-UMI in the single-task setting but achieves better performance in the multi-task setting. Both UVA and DP-UMI use 16 denoising steps. Speed is measured by inferring a single action trajectory consisting of 16 actions.

tasks, we used an improved Diffusion Policy [10], which is optimized for UMI data. It leverages a CLIP-pretrained [30] ViT-B/16 [1] vision encoder, significantly improving visual understanding. We refer to it as [DP-UMI].

- **OpenVLA** [22] is a state-of-the art vision-language-action built on 7B Llama 2 [38] for multi-task setting. It is trained on a diverse dataset encompassing a wide range of robots, tasks, and environments. We finetune OpenVLA on each task to optimize its performance.
- **UniPi** [12] is a video-based policy model that generates videos first and then predicts actions based on the generated videos. Since the official implementation is not available, we used the code from [23]. This implementation relies on pixel-wise video generation, which results in slower video generation speed. For action inference, we train a model that processes two consecutive generated video frames using a pretrained ResNet-50 for the *PushT* and *PushT-M* tasks, and a ResNet-152 for the *Tool Hang*, *Libero10*, and *Cup Arrangement*.
- **UVA-action** is an ablation of UVA, where the video generation part is excluded, and the model is trained solely as a policy model. This baseline aims to evaluate the effectiveness of joint video and action training.

D. Policy Learning Results

We evaluate policy learning results with UVA compared to the baseline methods on a few different axes: 1) action prediction accuracy, 2) inference speed, 3) robustness to visual disturbances, 4) robustness to history length, and 5) the effect of joint video-action modeling.

Action Prediction Accuracy (Simulation Tasks): In Table I, we compare UVA with baseline methods in both single-task and multi-task settings. We use the same random seed for our method and baselines to ensure a fair comparison.

Simulation Single-Task: Our method is able to match the performance of the state-of-the-art model DP-C and significantly outperform other video-based methods such as UniPi and vision-language-action model OpenVLA.

Simulation Multi-Task: Our method is particularly strong in the multi-task setting. Specifically, UVA surpasses the best baseline by 20% on the *PushT-M* task and by 13% on the *Libero10* benchmark. This result demonstrates that UVA model is able to better learn and leverage the general dynamics that are shared across tasks and, therefore, improve overall performance in the multi-task setting.

Action Prediction Accuracy (Real-World Tasks): Table II shows the results of real-world tasks. We ensure a fair comparison by keeping the initial placement of objects and grippers identical across different methods for each test rollout.

Real-World Single-Task: First, we evaluate the policy performance in a single-task setting. This evaluation aims to compare our method with a strong baseline in prior works by replicating a similar evaluation setup. Overall, UVA performs comparable with DP-UMI, which is optimized with this particular training dataset. We noticed that the dataset contains extensive recovery data from the moments of failure to correct the policy. This data is particularly useful for models without history dependence, like DP-UMI, which can recover from the new observations included in the recovery data. In contrast, our model uses a longer history, which is advantageous for tasks requiring longer memory. However, in this case, the collected failure recovery data is less impactful for our model, as its longer memory window prioritizes learning from extended temporal patterns. While we could shorten our model's history window, we maintain a consistent design across all tasks rather than tailoring it to this specific task and training data.

Real-World Multi-Task: We train a single model using our method and evaluate it on three tasks individually. DP-UMI is trained and tested in the same manner for a fair comparison. For each task, we test the methods on 20 different cases, as shown in Figure 4. Our approach demonstrates superior performance in the multi-task setting, achieving a 15% higher success rate on the Cup task and a 40% higher success rate on the Mouse task compared to DP-UMI.

In general, our method can successfully complete the task with distractor objects or changing backgrounds but fails when the background color is the same as the objects. Its visual understanding could be enhanced by training on additional video data without action labels. DP-UMI performs well on the towel task but shows less stability when handling pick-and-place cups and grabbing the mouse. However, it performs well across different backgrounds due to its use of a pre-trained vision encoder. When the gripper color is changed to an unseen green, the performance of both methods slightly decreases compared to the orange gripper used for training. However, both UVA and DP-UMI show good generalization to the unseen gripper. Please refer to our website for details.

Inference Speed: Speed is evaluated based on a single action trajectory inference. All methods, except OpenVLA, infer 16 action steps per trajectory with 8 executed steps. OpenVLA infers one action at a time, requiring 8 runs to match the inference time for 8 executed actions. UniPi generates raw pixel videos, resulting in significantly slower inference.

Our method achieves faster inference speeds by performing diffusion iterations only on the lightweight action head, rather than the entire network as DP-C and DP-T. Thanks to the decoupled design, video generation can be skipped during policy inference, further improving efficiency. For simulation tasks, we use 100 denoise steps for action prediction. For DP-C and DP-T, we follow their original implementations and also perform denoising over 100 steps. With the same number of diffusion steps (Table I), our method achieves faster inference compared to DP-C and DP-T.

For real-world tasks (Table II), both UVA and DP-UMI use 16 denoising steps for real-time manipulation. We found that the UVA Attention module in the Transformer accounts for half of the inference time, making UVA slightly slower than DP-UMI. With future improvements, such as replacing the Attention with Flash Attention, our model could achieve faster speeds. We note that, although DP-UMI uses a pretrained ViT encoder, its model size (171.27M parameters) is smaller than DP-C (262.69M parameters). This explains why DP-C is slower than UVA in Table I and DP-UMI is faster in Table II. Overall, UVA achieves a good balance between speed and performance across diverse settings.

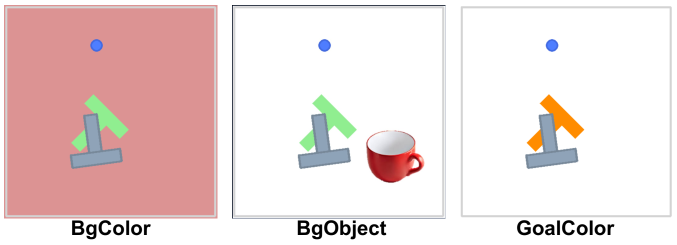


Fig. 5: **Visual Disturbances on PushT.** Tasks are performed under altered visual conditions, including changes in background color, distracting background objects, and goal color.

	Normal ↑	BgColor ↑	BgObject ↑	GoalColor ↑
DP-C [9]	0.91	0.12	0.21	0.17
DP-T [9]	0.78	0.22	0.17	0.28
OpenVLA [22]	0.35	0.17	0.13	0.32
UniPi [12]	0.42	0.31	**0.36**	0.40
UVA	**0.98**	**0.35**	0.31	**0.64**

TABLE III: **Visual Generalization Results on PushT with Visual Disturbances.** Our method and UniPi, both video generation models, have higher success rates compared to other policy learning approaches.

Robustness to Visual Disturbances: We have shown that UVA is robust to visual disturbances in the real-world multi-task setting in Figure 4. All tests are unseen during training, and even with more challenging distractor objects and backgrounds, UVA achieves higher success rates than DP-UMI.

To more rigorously evaluate this visual generalization capability, we perform a systematic evaluation in simulation by procedurally altering visual conditions in the *PushT* environment. The modifications include changes to the background color, the addition of object distractor, and variations in goal color, as shown in Figure 5. The evaluation was conducted on scenarios outside the training distribution, as the model was trained only in the standard environment in Figure 3. The results show that video generation methods, such as UniPi and UVA, exhibit superior performance in handling visual disturbances. For example, with changes in goal color, UniPi achieves a success rate of 40%, UVA achieves 64%, while OpenVLA only reaches 32%.

Robustness to History Length: Prior policy learning methods, such as DP-C, often experience performance degradation as the history length increases as shown in Figure 6 evaluated on the *PushT-M* task. In contrast, UVA can effectively adapt to longer history inputs. By jointly predicting video and action, our model sustains robust performance even as the history length grows. This highlights the better potential of UVA for tasks that require reasoning over extended temporal contexts.

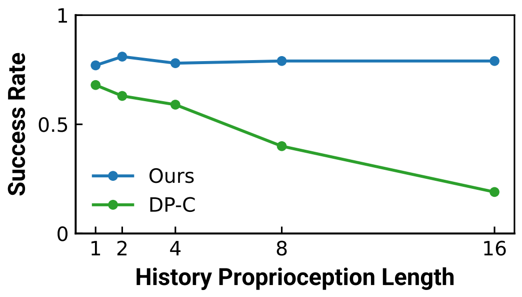


Fig. 6: **Robustness to History Length on PushT-M.** Typical policy learning frameworks such as DP-C [9] often experience performance drops with increased history length due to overfitting, while our model maintains robust performance.

	Libero10 ↓	CupArrange ↓
UniPi [12]	56.55	71.37
UVA(1 step)	89.36	51.34
UVA(8 steps)	**51.10**	**29.72**

TABLE IV: **Video Generation Results.** Our method outperforms UniPi in both simulated (Libero10) and real-world (Cup Arrangement) environments. The masked autoencoder training enables autoregressive video generation, with 8 steps performing better than a single step. FVD [39] results are reported.

Effect of Joint Video-Action Modeling: We evaluate this by comparing UVA with a baseline (UVA-action) that removes the video generation part. As shown in Table I, this modification led to reduced performance compared to the complete framework, highlighting the critical role of video generation in improving policy learning.

VI. UVA as a Video Generator

UVA can function as a video generation model by bypassing the action diffusion head during inference. We compare UVA with UniPi on video generation results across two datasets: *Libero10* and *Cup Arrangement*, in Table IV. Performance is evaluated by computing the *Fréchet Video Distance* (FVD) [39] for 500 videos generated by each method. FVD is a metric for assessing video quality by evaluating visual fidelity and temporal coherence. It compares statistical properties of feature representations from real and generated videos, using a pre-trained Inflated 3D ConvNet [6]. Lower FVD scores indicate greater similarity.

The masked autoencoder training in our method (§III-B) facilitates video generation in an autoregressive manner, where visual tokens are generated across all video frames in parallel during the first stage. In the subsequent stage, the next set of tokens is predicted sequentially, conditioned on the previously generated tokens. This iterative token prediction process continues until the entire video is generated. See Supplementary §X-A for details. In Table IV, we show that even with a 1-step process, our method outperforms UniPi on the more challenging Cup Arrangement task, while using 8 steps further improves performance.

Figure 8 shows the generated videos. Conditioned on the history observations, UVA can predict future observations that closely match the ground truth. Even with a single autoregressive step, it can generate realistic video frames, and with additional steps, the details become more refined. In contrast, UniPi occasionally produces blurry images, such as the Cup Arrangement, or mismatched images, as seen in the Towel Folding task and the Libero10 task (left). Additionally, UniPi may fail to generate some objects entirely, as demonstrated in the Libero10 task (the second moka pot in the right figure). We believe that with more computational resources and larger video generation models, both UVA and UniPi could achieve further improvements. However, given similar computational resources, UVA consistently outperforms UniPi.

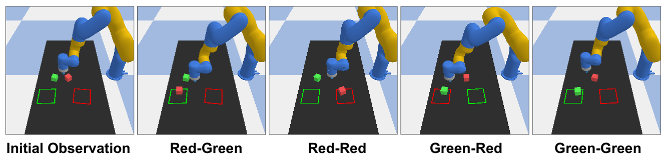


Fig. 7: **Forward Dynamics Model on Block Pushing Task.** During training, the robot pushes two blocks randomly to any target. During testing, the generated future image from UVA is used to select the proper action that moves a specific object to a specific target.

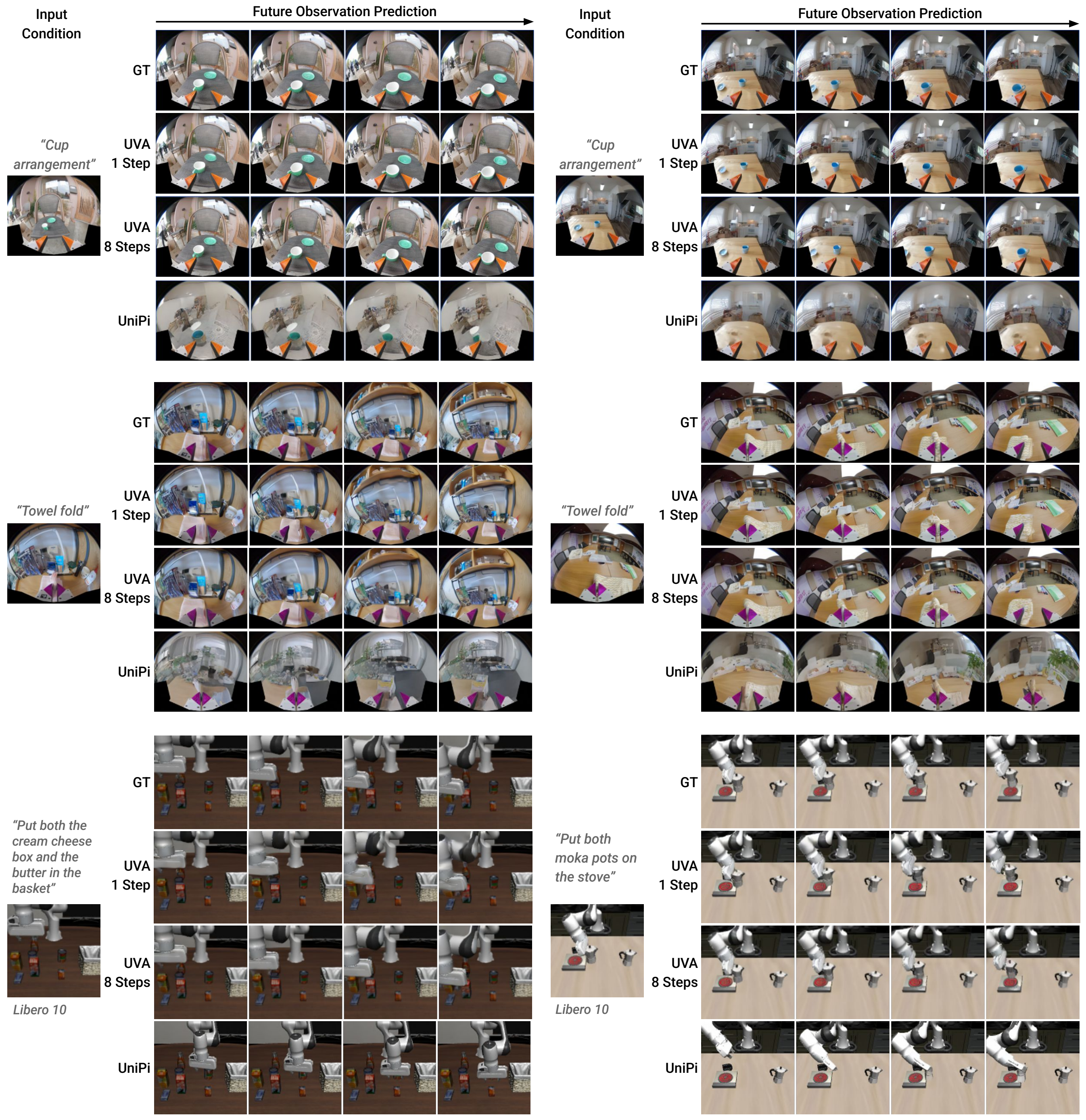
	R-R ↑	R-G ↑	G-R ↑	G-G ↑	Avg. ↑
DP-C	0.20	0.50	0.60	0.20	0.38
UVA	**0.80**	**0.70**	**0.50**	**0.40**	**0.60**
GT-Dynamics	0.80	0.80	0.70	0.70	0.75

TABLE V: **Success Rate on Block Pushing.** Our model functions as a forward dynamics model to guide the behavior of pretrained policy models, such as the DP-C [9]. DP-C alone achieves an 38% success rate, while incorporating our model to generate future observations for trajectory selection increases the success rate to 60%. Using a ground-truth simulator provides an upper bound success rate of 75%.

VII. UVA as a Forward Dynamics Model

Our model can perform forward dynamics predictions $\mathbf{O}_{t+1} = f_{\text{forward}}(\mathbf{O}_t, \mathbf{A}_t)$. To evaluate its effectiveness, we use it to guide the behavior of a pretrained policy model, such as the DP-C. We evaluate this approach in a block-pushing environment, where the model is trained to push one block to a specified square and another block to a second square, each randomly assigned. During testing, we aim to control the policy to complete specific tasks, such as pushing the red block to the red square (*R-R*) or the red block to the green square (*R-G*). The evaluation considers four distinct settings, as illustrated in Figure 7. Each setting is tested 10 times with varying initial positions of the objects and the robot. Notably, a perfect policy model in this setup would achieve a maximum average success rate of 50%, since training only requires one block to be pushed to any square, while testing specifies exact target assignments for each block.

At each step, we sample 100 trajectories of 16 future actions using DP-C. The sampled actions, along with historical observations, are input into our model, which predicts future observations by functioning as a forward dynamics model. For



Input Condition
Future Observation Prediction
GT
UVA 1 Step
UVA 8 Steps
UniPi
"Cup arrangement"
"Towel fold"
"Put both the cream cheese box and the butter in the basket"
Libero 10
"Put both moka pots on the stove"
Libero 10

Fig. 8: **Video Generation Results on Validation Set.** UVA generates high-quality videos that closely match the ground truth, with 8 autoregressive steps further enhancing detail compared to a single autoregressive step. In contrast, UniPi occasionally produces blurry (Cup arrangement) or mismatched images (Towel fold and Libero10 left) and may fail to generate some objects (Libero10 right, the second moka pot). With the similar computational resources, UVA consistently outperforms UniPi.

each trajectory, we calculate a reward based on the predicted observations, selecting the trajectory with the highest reward. The reward is computed as the distance between the blocks and their target squares, which are identified from the predicted frames. We then execute the first 6 steps of the selected trajectory and resample new trajectories until the task is completed or the episode ends. DP-C can complete the tasks with a success rate of 38% in Table V. By leveraging our model to guide the policy, the success rate increases to 60%.

For comparison, we evaluate the performance of using a ground-truth simulator to render the sampled trajectories and select the best ones. Even with the ground-truth simulator, the success rate is limited to 75% due to suboptimal sampled trajectories and errors in object detection. While our model performs worse than the simulator, it still significantly en-

	Position ↓	**Rotation ↓**
UniPi Inverse Dynamics	1.92 cm	2.21°
UVA	**0.75** cm	**1.11°**
Visual Inertial SLAM	0.41 cm	0.30°

TABLE VI: **Inverse Dynamics.** L2 distances between predicted actions and ground truth actions from Mocap. The table compares positions and rotation results of UVA, the UniPi inverse dynamics model, and a well-engineered SLAM system [5]. UVA outperforms UniPi inverse dynamics model, while SLAM achieves the best accuracy but is more complex to implement.

hances performance and guides the pretrained policy models to complete required tasks.

VIII. UVA as a Inverse Dynamic Model

In this section, we evaluate the effectiveness of the proposed method in an inverse dynamics setting, where actions are inferred as $\mathbf{A}_t = f_{\text{inverse}}(\mathbf{O}_t, \mathbf{O}_{t+1})$ using UMI data. For the UMI Cup Arrangement data, the robot's actions are naturally aligned with camera movements, enabling straightforward evaluation of action prediction accuracy using the ground truth camera poses obtained from motion capture (Mocap). Notably, our model had never encountered this test data during training.

Baselines: As a comparison, we evaluate the actions (i.e., camera pose) generated by the inverse dynamics model used in UniPi [12] and a well-engineered SLAM system used in UMI [5], where the SLAM system requires an additional mapping. The UMI actions in the training data are derived from SLAM.

Results: For the actions predicted by each method, we compute the L2 distance to the ground truth actions from Mocap, as shown in Table VI. The UniPi inverse dynamics model predicts actions from two consecutive images, which may lead to discontinuities in the predicted actions over time. In contrast, our method predicts 16 actions simultaneously, resulting in more consistent and temporally coherent predictions. The action errors produced by SLAM were 0.41 cm for position and 0.30 degrees for rotation. While UVA exhibited slightly higher errors than SLAM, it still demonstrated strong performance with position errors under 1 cm and rotation errors around 1 degree. These results highlight the generalization capability of UVA for action prediction, even on unseen data, and suggest that it could serve as a viable alternative to SLAM, which is difficult to calibrate and suffers from a high failure rate.

IX. Discussion

Summary: We propose a unified video-action model that jointly models and separately decodes video and actions. This design enables us to fully leverage video data as additional supervision, resulting in stronger performance and fast action prediction by skipping video decoding during inference. The framework inherently supports masking training, allowing it to fulfill various robotics functions, including acting as a policy, video model, forward and inverse dynamics model, and a combined policy and planner. By fully utilizing video and action data in diverse configurations, our model reduces overfitting to specific tasks, outperforms previous methods, and demonstrates versatility for multi-purpose applications.

Limitation and Future Work: One limitation of our framework is that it does not currently leverage large amounts of actionless video data, which could provide valuable additional supervision. As a result, our method occasionally achieves only comparable performance to the DP-UMI on real-world tasks. We believe that pretraining the model on web-scale video datasets could significantly enhance its generalization capabilities, and we leave this exploration for future work. Furthermore, our model can be naturally extended to predict modalities beyond video and action, such as sound and force, by incorporating additional diffusion heads, offering a more comprehensive and versatile framework. This remains a promising direction for future research.

Acknowledgment

We would like to thank Huy Ha for his valuable advice on video recording and website design. We also thank Amber Xie, Austin Patel, Jennifer Grannen, Vincent de Bakker, John So, Max Du, and Vidhi Jain for their important feedbacks on the paper draft. We are grateful to Mengda Xu, Suneel Belkhale, Xiaomeng Xu, Fanqi Lin, Lirui Wang, and Tianhong Li for helpful discussions. We would like to express our gratitude to Chi Cheng, Zhenjia Xu, Chuer Pan, Zeyi Liu, Huy Ha, Fanqi Lin, Yingdong Hu, and Zhaxizhuoma for their contributions to the shared UMI dataset. Finally, we want to thank everyone who contributed their computing resources to help us train the models.

This work was supported in part by the Toyota Research Institute, NSF Award #1941722, #2143601, #2037101, #2132519, ONR Project #N00014-22-1-2293 and the DARPA TIMAT project. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

[1] Dosovitskiy Alexey. An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*, 2020.

[2] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[3] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video Generation Models as World Simulators, 2024.

[4] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[5] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[6] Joao Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[7] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked Generative Image Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

[8] Haoxuan Che, Xuanhua He, Quande Liu, Cheng Jin, and Hao Chen. Gamegen-x: Interactive open-world game video generation. *arXiv preprint arXiv:2411.00769*, 2024.

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[10] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal Manipulation Interface: In-the-Wild Robot Teaching Without In-the-Wild Robots. *arXiv preprint arXiv:2402.10329*, 2024.

[11] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive Video Generation Without Vector Quantization. *arXiv preprint arXiv:2412.14169*, 2024.

[12] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming Transformers for High-Resolution Image Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021.

[14] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit Behavioral Cloning. *Conference on Robot Learning (CoRL)*, November 2021.

[15] Kaifeng Gao, Jiaxin Shi, Hanwang Zhang, Chunping Wang, and Jun Xiao. ViD-GPT: Introducing GPT-Style Autoregressive Generation in Video Diffusion Models. *arXiv preprint arXiv:2406.10981*, 2024.

[16] Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. Emu Video: Factorizing Text-to-Video Generation by Explicit Image Conditioning. *arXiv preprint arXiv:2311.10709*, 2023.

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[19] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen Video: High Definition Video Generation with Diffusion Models. *arXiv preprint arXiv:2210.02303*, 2022.

[20] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video Diffusion Models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[21] Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video Prediction Policy: A Generalist Robot Policy with Predictive Visual Representations. *arXiv preprint arXiv:2412.14803*, 2024.

[22] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*, 2024.

[23] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B Tenenbaum. Learning to act from actionless videos through dense correspondences. *arXiv preprint arXiv:2310.08576*, 2023.

[24] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive Image Generation Without Vector Quantization. *arXiv preprint arXiv:2406.11838*, 2024.

[25] Junbang Liang, Ruoshi Liu, Ege Ozguroglu, Sruthi Sudhakar, Achal Dave, Pavel Tokmakov, Shuran Song, and Carl Vondrick. Dreamitate: Real-World Visuomotor Policy Learning via Video Generation. *CoRL*, 2024.

[26] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024.

[27] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking Knowledge Transfer for Lifelong Robot Learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[28] Fangchen Liu, Hao Liu, Aditya Grover, and Pieter Abbeel. Masked Autoencoding for Scalable and Generalizable Decision Making. *Advances in Neural Information Processing Systems*, 35:12608–12618, 2022.

[29] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei,

Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models from Natural Language Supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2021.

[31] Ilija Radosavovic, Baifeng Shi, Letian Fu, Ken Goldberg, Trevor Darrell, and Jitendra Malik. Robot Learning with Sensorimotor Pre-Training. In *Conference on Robot Learning*, pages 683–693, 2023.

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, 2021.

[33] Abhishek Sharma, Adams Yu, Ali Razavi, Andeep Toor, Andrew Pierson, Ankush Gupta, Austin Waters, Aäron van den Oord, Daniel Tanis, Dumitru Erhan, Eric Lau, Eleni Shaw, Gabe Barth-Maron, Greg Shaw, Han Zhang, Henna Nandwani, Hernan Moraldo, Hyunjik Kim, Irina Blok, Jakob Bauer, Jeff Donahue, Junyoung Chung, Kory Mathewson, Kurtis David, Lasse Espeholt, Marc van Zee, Matt McGill, Medhini Narasimhan, Miaosen Wang, Mikołaj Bińkowski, Mohammad Babaeizadeh, Mohammad Taghi Saffar, Nando de Freitas, Nick Pezzotti, Pieter-Jan Kindermans, Poorva Rane, Rachel Hornung, Robert Riachi, Ruben Villegas, Rui Qian, Sander Dieleman, Serena Zhang, Serkan Cabi, Shixin Luo, Shlomi Fruchter, Signe Nørly, Srivatsan Srinivasan, Tobias Pfaff, Tom Hume, Vikas Verma, Weizhe Hua, William Zhu, Xinchen Yan, Xinyu Wang, Yelin Kim, Yuqing Du, and Yutian Chen. Veo. 2024.

[34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015.

[35] Yang Song and Prafulla Dhariwal. Improved Techniques for Training Consistency Models. *arXiv preprint arXiv:2310.14189*, 2023.

[36] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling Through Stochastic Differential Equations. *arXiv preprint arXiv:2011.13456*, 2020.

[37] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency Models. *arXiv preprint arXiv:2303.01469*, 2023.

[38] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[39] Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards Accurate Generative Models of Video: A New Metric & Challenges. *arXiv preprint arXiv:1812.01717*, 2018.

[40] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. *arXiv preprint arXiv:2408.14837*, 2024.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017.

[42] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable Length Video Generation from Open Domain Textual Descriptions. In *International Conference on Learning Representations*, 2022.

[43] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling Autoregressive Video Models. *arXiv preprint arXiv:1906.02634*, 2019.

[44] Wenming Weng, Ruoyu Feng, Yanhui Wang, Qi Dai, Chunyu Wang, Dacheng Yin, Zhiyuan Zhao, Kai Qiu, Jianmin Bao, Yuhui Yuan, et al. ART-V: Auto-Regressive Text-to-Video Generation with Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7395–7405, 2024.

[45] Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked Trajectory Models for Prediction, Representation, and Control. In *International Conference on Machine Learning*, pages 37607–37623, 2023.

[46] Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. *CoRL*, 2024.

[47] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video Generation Using VQ-VAE and Transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[48] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

X. Supplementary Materials

In this section, we first introduce the autoregressive video generation process in §X-A and then show more details of the simulation benchmarks (§X-B) and real-world benchmarks (§X-C) we used for evaluation. We finally report the inference speed details in §X-D.

A. Autogregressive Video Generation

Our autoregressive video generation is based on the methods from [7] and [24], originally designed for image generation, which we have extended to video generation. In [7], images are first converted into discrete visual codes using VQGAN [13]. During training, a subset of these visual codes is randomly masked and the model is trained to reconstruct them. During inference, the entire image is generated from an empty mask. This masked training approach enables the model to generate images autoregressively. The number of autoregressive steps can be adjusted during inference, with more steps leading to improved performance, as demonstrated in their paper. However, the visual discretization in VQGAN and training with discrete visual codes often leads to information loss, resulting in lower-quality images. Li et al. [24] address this limitation by using continuous latent representations instead of discrete code for image tokens. Their approach models each visual token's probability using a diffusion model, eliminating the need for vector quantization. This method has shown improved performance over previous approaches.

Our method is similar to [24] in that it predicts continuous latent representations. These representations are then used as conditions of the diffusion heads to decode actions and video observations. The autoregressive generation process is shown in Figure 9. If the autoregressive step is set to 1, the entire video is generated in a single pass. Otherwise, with a predefined number of steps, the method generates the video autoregressively, completing the process in the specified number of steps.

Our model builds on the pretrained model (MAR-B) released by [24] but has undergone substantial modifications for the joint video and action modeling. Please check out code for details.

B. Simulation Benchmarks

Single-Task Evaluation: For this setting, different policies are trained for different tasks.

- PushT [9, 14]: requires the agent to push the gray "T" to align it with the target "T" at the center of the scene. The average success rate over 50 rollouts is reported in the main paper.
- Toolhang [29]: requires a robot to insert a hook into a base and then hang a wrench. It is one of the most challenging tasks in RoboMimic. We evaluate the average success rate over 50 environments with different random seeds.

Multi-Task Evaluation: In multi-task evaluation, we train one policy for multiple task goals defined by image or text.

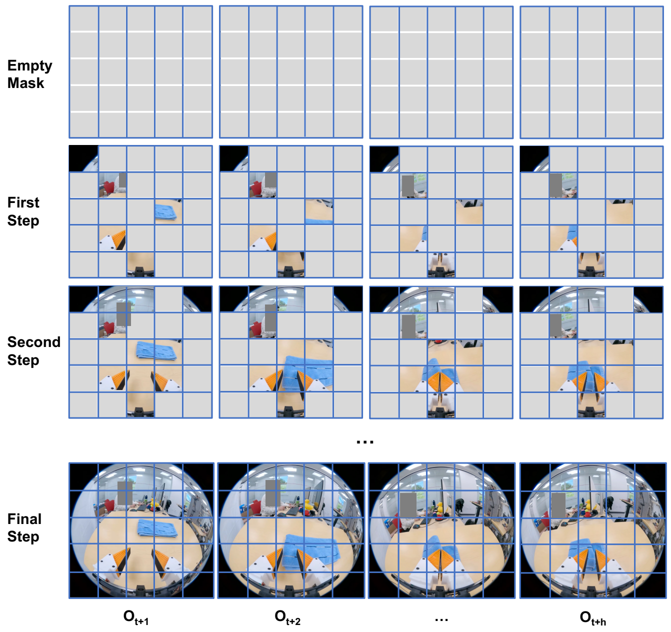


Fig. 9: **Autoregressive Video Generation.** Our method generates a video starting from an empty mask. Given a specified number of generation steps, the model produces a set of tokens at each autoregressive step, progressively constructing the video. If the generation step is set to 1, the entire video is generated in a single pass.

- PushT-M: We extend the *PushT* task to include varying target "T" positions. This setting adds additional uncertainty and presents a greater challenge, requiring the agent to dynamically plan its trajectory and adapt to diverse spatial configurations. We reported the average reward on 50 environments with different random seeds.
- Libero10 [27]: is the most challenging task in the Libero benchmark due to its long-horizon requirements. This set comprises 10 tasks, each accompanied by a language goal description, such as "put the red mug on the left plate and open the bottom drawer of the cabinet". We use the same language encoder, *e.g.*, CLIP, for all methods. Objects are initialized in varying locations within the scene. We test each task three times with different random seeds and report the average rewards across all 10 tasks.

C. Real-world Benchmarks

Training Data: We assess real-world performance using two publicly available benchmarks introduced by [10] and [26]. Both benchmarks collect data using the handheld UMI [10] device. Thanks to the UMI's design, these datasets can be directly utilized to train various models and evaluate them on our robotic setup, which consists of a single ARX X5 robotic arm. We used three tasks for training:

- Cup Arrangement [10]: requires the robot to first rotate the cup with the handle oriented to the left of the robot, then pick up an espresso cup and place it onto a saucer. Success is achieved when the cup is properly placed on the saucer and the handle lies within $\pm 15^{\circ}$ of the exact left alignment.

- Towel Folding [26]: requires the robot to grasp the left edge of the towel and move it to the right, folding it in half.
- Mouse Arrangement [26]: requires the robot to pick up the mouse and place it on the mouse pad.

Single-Task Evaluation: We train a single-task policy on the Cup dataset and directly compare it to the DP-UMI model provided by the authors [10]. Both methods are trained using the same dataset. The cup and saucer are initially placed in varying positions and orientations. Across 20 test rollouts, the robot needs to rotate the cup before placing it on the saucer in 85% of the cases. Additionally, we include challenging scenarios where the cup is positioned at the boundary of the robot arm's reach. We evaluate each method across 20 rollouts with different initial configurations. The average success rate is reported in the main paper.

Multi-Task Evaluation: We train one model with all three tasks and then evaluate the policy performance on each task independently. The three public datasets contain a total of 6,764 episodes. We randomly selected 500 episodes from each dataset and combined them into a dataset with 1500 episodes to train both our model and DP-UMI [10]. To ensure a wide testing distribution, we evaluate 20 different scenarios, including six initial configurations of objects and the robot, six different distractor objects, five distinct backgrounds, and an unseen gripper type (training is conducted with only orange and purple grippers, while testing includes a green gripper).

D. Inference Speed Measurement and Decomposition

For the simulation results in Table I, we measure the inference speed using a server with NVIDIA L40 GPUs for all of the models. For the real-world results in Table II, we measure the inference speed on an NVIDIA RTX 3080 GPU to match real-world deployment scenarios.

We also decompose the inference time of each component in our model under the real-world deployment scenario:

Modules / Tasks	**Inference time (ms) $\downarrow$**
VAE Image Encoder	40
Transformer (Attention)	40
Transformer (Flash Attention)	30
Action Diffusion (16 steps)	15
Action Diffusion (100 steps)	93
Rest of the model	< 1
UVA (16 steps)	95
UVA (16 steps Flash Attention)	85
UVA (100 steps)	173
UVA (100 steps Flash Attention)	163

TABLE VII: **Inference Time Decomposition.** We measure the inference time for each module and the total runtime of UVA as a policy model, using 16 policy steps and 100 diffusion steps.

The pretrained VAE image encoder requires 40ms, while the transformer module takes approximately 40ms, which can be reduced to 30ms using flash attention. The action prediction head takes 15ms with 16 diffusion steps and 93ms with 100 diffusion steps. In our policy learning experiment in Table II, we chose 16 diffusion steps. However, using 100 diffusion steps results in smoother action predictions and a higher success rate, though at the cost of slower performance. In total, UVA requires 95ms to predict an action trajectory with 16 actions when using 16 diffusion steps, and 173ms when using 100 diffusion steps. Incorporating flash attention can reduce 10ms for both.