VidMan: Exploiting Implicit Dynamics from Video Diffusion Model for Effective Robot Manipulation

Youpeng Wen[1*], **Junfan Lin**[2*], **Yi Zhu**[3], **Jianhua Han**[3],
Hang Xu[3], **Shen Zhao**[1†], **Xiaodan Liang**[12†]
[1]Shenzhen Campus of Sun Yat-Sen University,
[2]Peng Cheng Laboratory, [3]Huawei Noah's Ark Lab

Abstract

Recent advancements utilizing large-scale video data for learning video generation models demonstrate significant potential in understanding complex physical dynamics. It suggests the feasibility of leveraging diverse robot trajectory data to develop a unified, dynamics-aware model to enhance robot manipulation. However, given the relatively small amount of available robot data, directly fitting data without considering the relationship between visual observations and actions could lead to suboptimal data utilization. To this end, we propose **VidMan** (**Vid**eo Diffusion for Robot **Man**ipulation), a novel framework that employs a two-stage training mechanism inspired by dual-process theory from neuroscience to enhance stability and improve data utilization efficiency. Specifically, in the first stage, VidMan is pre-trained on the Open X-Embodiment dataset (OXE) for predicting future visual trajectories in a video denoising diffusion manner, enabling the model to develop a long horizontal awareness of the environment's dynamics. In the second stage, a flexible yet effective layer-wise self-attention adapter is introduced to transform VidMan into an efficient inverse dynamics model that predicts action modulated by the implicit dynamics knowledge via parameter sharing. Our VidMan framework outperforms state-of-the-art baseline model GR-1 on the CALVIN benchmark, achieving a 11.7% relative improvement, and demonstrates over 9% precision gains on the OXE small-scale dataset. These results provide compelling evidence that world models can significantly enhance the precision of robot action prediction. Codes and models will be public.

1 Introduction

In the rapidly advancing field of robotics, accurately predicting and executing precise actions based on sensory inputs is crucial. While traditional approaches [1–5] for robot manipulation often rely on labor-intensive hand-engineered features and models prone to errors, data-driven methods [6–8] offer promising solutions. However, the challenge lies in the difficulty and cost of acquiring high-quality robotic data. Recent advancements [9–12], particularly those utilizing large-scale online video data for learning a video generator, demonstrate significant potential in comprehending complex physical dynamics of the real world. These models, trained on diverse datasets [13, 8], possess a nuanced understanding of the world, suggesting the feasibility of amalgamating and leveraging varied robot visual trajectory data [14–16] to develop a unified dynamics-aware model for enhanced robot manipulation. Yet, achieving this unification poses challenges; merely fitting data without considering the relationship between visual observations and actions could lead to suboptimal utilization of the data. Hence, there is a pressing need to develop efficient training mechanism and model architecture

---

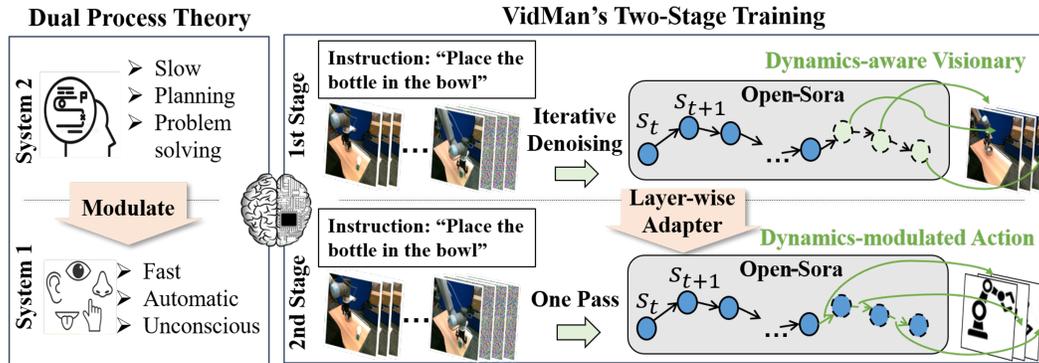*Equal contribution, †Corresponding author

Figure 1: VidMan's two-stage training paradigm mirrors dual process theory: its first stage (like System 2) pre-trains on understanding environment dynamics through video diffusion, forming a foundation for accurate action prediction, while its second stage (like System 1) was adapted from the first stage to leverage the learned dynamics knowledge for rapid, low-level action inference.

that can effectively leverage existing cross-robot and cross-scene data to enhance action prediction accuracy.

As shown in Fig. 1, to optimize the utilization of diverse robot data [8], we draw upon insights from neuroscience's dual process theory [17–19], which unveils the complex mechanisms of information processing and decision-making in the human brain. This theory distinguishes between two cognitive processes: System 1, responsible for rapid, intuitive responses based on immediate sensory inputs, and System 2, which involves slower, long-horizon planning grounded in abstract concepts and understanding of world dynamics [20]. Inspired by these insights, we adopt a two-stage paradigm for robot learning, exemplified by our innovative framework, terms as **VidMan** (**Vid**eo diffusion for robot **Man**ipulation). VidMan leverages the power of the video diffusion generation method Open-Sora [21] for robot imitation learning, tapping into the awareness of long-horizon dynamics inherent in video diffusion models to achieve more nuanced and dynamics-modulated robot action prediction. By learning different facets of data at distinct stages, VidMan acquires an inductive bias of inverse dynamics of robot control, wherein actions are the outcomes of state sequences, significantly enhancing the method's generalization performance, especially under scenarios with limited data.

Specifically, our VidMan employs a two-stage training mechanism, akin to the principles of dual process theory, to enhance stability and significantly improve data utilization: 1) In the first stage, namely the *Dynamics-aware Visionary Stage*, VidMan undergoes pre-training on the Open X-Embodiment [8] dataset (OXE) using a video diffusion model to predict future trajectories based on historical observations and language instructions. This stage involves the robot learning the dynamics of state transitions from data and accurately perceiving the current environmental state, enabling the model to develop a deep understanding of the environment's dynamics, forming a robust foundation for accurate action prediction; 2) In the second stage, dubbed the *Dynamics-modulated Action Stage*, VidMan incorporates a flexible yet powerful layer-wise self-attention adapter [22] to seamlessly integrate the pre-trained knowledge from the first stage into action prediction. Through shared neural architecture and parameters with the dynamics-aware visionary stage, this phase transforms VidMan into an implicit inverse dynamics model that infers dynamics-modulated actions without explicitly generating future visual trajectories, rendering it suitable for real-world robot control scenarios.

The performance of VidMan has been evaluated against SOTA baselines on the CALVIN [14] benchmark, where it achieves a 11.7% relative improvement. In addition, VidMan has shown notable effectiveness on the OXE small-scale dataset, achieving over 9% precision gains. This improvement is particularly evident when the data from the target robot is small, underscoring the effective data utilization of our method. Extensive ablation studies have been conducted to analyze the effects of various design decisions within our method. These experimental results suggest that VidMan represents a meaningful advancement in robotics, providing a valuable tool for developing more capable and responsive robotic systems.

2 Related Work

Language-guided Robot Manipulation. Language-guided robot manipulation has emerged as an elastic and straightforward approach to instructing robots to perform various tasks [14, 23]. Some existing methods [24, 25, 10, 26–30] leverage large language models (LLMs) to plan over the task domain and pass instructions to low-level action policies to generate robot actions. The hierarchical 2D policies [31–33] predict latent features or images of subgoals given a language instruction, which they feed into lower-level subgoal-conditioned policies. 3D policies [34, 35] combine 3D representations with diffusion objectives to learn manipulation from demonstrations using depth maps and camera extrinsics. Some methods [36, 37] also utilize 3D [38–40] or 2D [41–43] detection to identify objects and use constrained optimization methods to control robot operations Another line of work learns language-conditioned policies from unstructured play data, which contains demonstrations with and without language labels [44, 32]. These methods leverage sequence-to-sequence conditional variational auto-encoders to generate latent plans, which are then used to condition the action policy.

Pre-training for Robot Manipulation. The field of pre-training for robot learning has garnered significant attention in recent years [9, 26, 11, 45]. Some methods aim to learn useful visual representations through masked image modeling [9] and contrastive learning [11]. Previous research [46, 7, 47, 45, 25, 48] has focused on empowering robots and other agents with the capability to comprehend and execute language instructions, typically by learning policies conditioned on language. GR-1 [9] and RoboFlamingo [26] use a GPT-style framework to model action prediction as a token prediction task in CALVIN dataset [14], and achieve good results. Our approach is most similar to GR-1 in that it utilizes GPT-style framework to predict both video and action, while ours directly uses video's ability to capture future information to predict long sequences of actions.

3 Preliminaries

Robot Dynamics and Inverse Dynamics. The dynamics of a robot are typically characterized by the forward dynamics and the inverse dynamics. The forward dynamics describe how the current state and action determine the next state. Formally, given a state space $\mathcal{S}$ and an action space $\mathcal{A}$, the forward dynamics can be represented as the conditional probability distribution: $P(s_{t+1} \mid s_t, a_t)$, where $s_t, s_{t+1} \in \mathcal{S}$ are the robot state at timestep t and $t+1$, respectively, and $a_t \in \mathcal{A}$ is the action taken. Conversely, inverse dynamics describe the probability of an action given a transition from one state to another. The inverse dynamics are particularly important in scenarios where only passive observation data is available, such as data collected from the internet, which often lacks explicit action information. Formally, the inverse dynamics are given by:

$$P(a \mid s_t, s_{t+1}). \tag{1}$$

Observations to States. In many real-world applications, direct access to the state space $\mathcal{S}$ is infeasible, and instead, sequences of image observations are provided. Let $\mathcal{O}$ denote the observation space, *e.g.*, image, proprio. To infer the underlying sequence of states $S = \{s_1, s_2, \ldots, s_T\}$ from a sequence of observations $O = \{o_1, o_2, \ldots, o_T\}$. This process can be formally described by finding the sequence of states that maximizes the posterior probability given the observations:

$$S^* = \arg\max_S P(S \mid O). \tag{2}$$

Using Bayes' theorem, this can be further expressed as:

$$S^* = \arg\max_S P(O \mid S)P(S), \tag{3}$$

where $P(O \mid S)$ is the likelihood of the observations given the states, and $P(S)$ is the prior probability of the sequence of states. This formulation is essential in various applications, including hidden Markov models (HMMs) and other state estimation techniques. With more observations, the likelihood $P(O \mid S)$ becomes more peaked around the true state sequence. This is because more data points allow for better discrimination between different state sequences.

4 Method

In this paper, we aim to enhance the precision of robot action prediction by exploiting the intricate dynamics encoded within robot visual trajectories. We propose VidMan, a novel framework leveraging
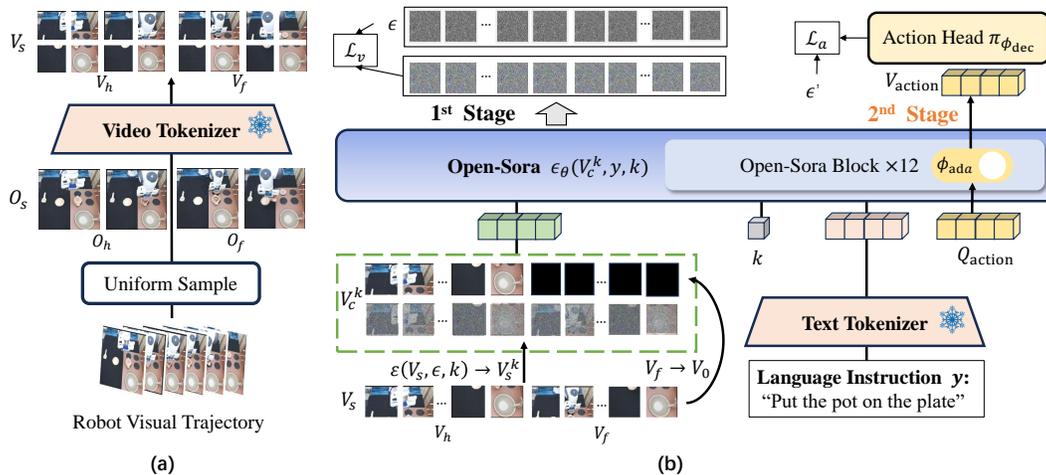
Figure 2: **Overview of VidMan.** (a) We use Video Tokenizer to tokenize the uniform sampled robot visual trajectory O_s to video tokens V_s. (b) In the 1st Stage, we concatenate the video tokens processed through the diffusion process with the historical tokens along the channel dimension to form V_c^k. V_c^k along with the language tokens and diffusion step k are fed into Open-Sora for video prediction training. In the 2nd Stage, we use a learnable action token through a layer-wise adapter applied to the output of the Open-Sora Block to obtain tokens V_{action} that integrate future frame information. V_{action} are then fed into the Diffusion Action Head $\pi_{\phi_{dec}}$ for action prediction training.

<u>Video</u> diffusion for robot <u>Man</u>ipulation. VidMan employs a dual-stage training strategy: in the first stage, the *Dynamics-aware Visionary Stage*, we enable the model to forecast and imagine potential future trajectories based on historical observations, leveraging the multi-frame prediction capability of the video diffusion model. Through this stage, the model is optimized to understand the dynamics of the environment. In the second stage, the *Dynamics-modulated Action Stage*, we introduce a lightweight layer-wise adapter to seamlessly integrate the visionary predictive stage with fast, adaptive action prediction. This approach decouples the knowledge of the world and embodiment into distinct processes while ensuring seamless integration through the training and utilization of shared parameters. Overview of our method is shown in Fig. 2. In the following sections, we will formulate each of these stages.

4.1 Dynamics-aware Visionary Stage

To endow the model with the knowledge of world dynamics, we formulate this knowledge acquisition stage as future image trajectory generation, which captures dynamic priors and predicts future state transitions more accurately according to Equ. (3). Specifically in our context, the goal is to predict future frames based on historical frames and language instructions. To achieve this, we leverage the capabilities of a multi-frame generation framework based on the video diffusion transformer model (VDT) Open-Sora [21], which has shown a remarkable ability to generate diverse and physically authentic successive frames aligned with language instructions. For simplicity, we use VDT to represent Open-Sora in the following content.

To prepare the training data, we utilize a pre-trained video tokenizer [49] to encode successive robot images in a trajectory $O_s = [O_h, O_f]$ into embeddings $V_s = [V_h, V_f]$, where $V_h \in \mathbb{R}^{m \times W \times H \times C}$ and $V_f \in \mathbb{R}^{n \times W \times H \times C}$ stand for embeddings for m history images O_h and n future images O_f, respectively. During the forward diffusion process $V_s^k \leftarrow \varepsilon(V_s, \epsilon, k)$, the noise $\epsilon \sim \mathcal{N}(0; 1)$ is added to trajectory embeddings V_s according to the diffusion step $k \in [1, K]$. The diffusion model is optimized to predict the added noise ϵ from V_s^k given the condition hints. As the condition hints, we pad V_h with zero-valued embeddings V^0 to the same shape as V_s^k, which are concatenated with V_s^k along the channel dimension to form the condition visual embeddings $V_c^k \in \mathbb{R}^{(m+n) \times W \times H \times 2C}$. And the language instruction y is encoded by an text tokenizer [50] to obtain the language embedding. Mathematically, the denoised diffusion model is optimized according to:

$$\mathcal{L}_v(\theta) = \mathbb{E}_{(V_c^k, y, k)} \left[\left\| \epsilon - \epsilon_\theta(V_c^k, y, k) \right\|_2^2 \right]. \tag{4}$$

In this training stage, we use only a third-person camera to predict the representation. This approach has two main advantages: a) most robot datasets include only third-person view data; b) training with a fixed third-person viewpoint can reduce the influence of view changes and help the model focus on predicting the transitions of the robotic arm itself. Additionally, our method can easily be extended to multiple cameras by simply inputting multiple cameras of viewpoint.

4.2 Dynamics-modulated Action Stage

According to Equ. (1), actions can be accurately predicted given the states with the inverse dynamics model. One straightforward approach to combine an inverse dynamics model with the VDT learned in the first stage is to separately construct an inverse dynamics model that maps from image observations to actions. During deployment, this model could predict actions from the generated observations of the dynamics-aware visionary stage. However, by this means, actions are only predictable after the VDT conducts a time-consuming iterative denoising diffusion process, which is not ideal for high-frequency robot control. Moreover, the accuracy of the actions is heavily dependent on the accuracy of the predicted observations. Since not all pixels are important for predicting actions, this method could introduce unnecessary bias and time costs. Additionally, learning a separate inverse dynamics model from scratch does not leverage the pre-trained parameters of the VDT.

To address these issues, we propose directly adapting the VDT into an inverse dynamics model. In this way, the dynamics knowledge and implicit states learned in VDT can be seamlessly leveraged to facilitate the prediction of actions. Below, we introduce a lightweight adapter module that effectively transforms the VDT into an implicit inverse dynamics model. The output of this adapted model is then decoded into a sequence of actions using a diffusion-based action head.

Implicit Inverse Dynamics Adapter. To transform the VDT into an inverse dynamics model, we incorporated a layer-wise adapter which is inspired by [22] after each layer in VDT. Each adapter includes a multi-head self-attention and a feed-forward network (FFN) with a gated mechanism. We use h learnable action tokens Q_{action}, concatenating them with the features ouput from each layer of the VDT, and input them into the layer-wise adapter. This fuses the knowledge of each layer of the VDT to produce h final action embeddings V_{action}. Since we only reuse VDT parameters without its observation generation function, we disable the iterative denoising process by using a fixed diffusion step $k \leftarrow K$, which turns V_s^k into pure Gaussian noise. Formally, the action embeddings are fused with the introduce layer-wise adapter parameterized by ϕ:

$$V_{\text{action}} = \epsilon_{(\theta, \phi_{\text{ada}})}(V_c^K, y, K, Q_{\text{action}}), \tag{5}$$

where $\epsilon_{(\theta, \phi_{\text{ada}})}$ represents the VDT parameters incorporated with the layer-wise self-attention adapter with parameters ϕ_{ada}. The flexibility of the layer-wise adapter allows for the integration of domain-specific knowledge into the action embedding. For example, if the downstream robotic manipulation task includes proprioceptive information, we can use a proprioception embedder to convert it into tokens, which are then concatenated with the output of each layer of the VDT to serve as the additional keys and values for the layer-wise adapter. Please refer to Appendix A.2 for more details.

Diffusion-based Action Head. The fused action embeddings, V_{action}, are subsequently translated into low-level control signals. To achieve this, we utilize a diffusion-based action head [51], $\pi_{\phi_{\text{dec}}}$, responsible for decoding action embeddings into executable action signals, such as determining the 7 degrees of freedom (DoF) for the end-effector pose and gripper status. Similar to the dynamics-aware visionary stage, the objective of action prediction is:

$$\mathcal{L}_a(\theta, \phi_{\text{ada}}, \phi_{\text{dec}}) = \mathbb{E}_{(V_{\text{action}}, l)} \left[\left\| \epsilon' - \pi_{\phi_{\text{dec}}}\left(\varepsilon(V_{\text{action}}, \epsilon', l), l \right) \right\|_2^2 \right], \tag{6}$$

where $\epsilon' \sim \mathcal{N}(0, 1)$, $l \in [1, L]$ stands for the diffusion step and ε denotes the forward diffusion process. Note that this diffusion-based action head is relatively small compared to the VDT, making its computational cost and time consumption almost negligible.

5 Experiment

In this section, we conduct comprehensive experiments to validate the effectiveness of our methods, along with in-depth ablation studies.

5.1 Experiment Settings

5.1.1 2-stage Training Setting

Settings of dynamics-aware visionary stage. We initialized our model with the weights from Open-Sora's [21] 16x256x256 text-to-video model, which was trained on internet data [52–54]. Since we concatenated historical frames along the channel dimension, we append a zero matrix to the parameters of the tokens embedder layer in Open-Sora accordingly. We use Open X-Embodiment Dataset [8] to train our model, which was constructed by pooling 60 existing robot datasets from 34 robotic research labs around the world. We referred to the Octo [7] codebase to selected 25 datasets from it, which are heterogeneous not just in terms of the robot type, but also in the sensors (e.g., including or not including wrist cameras) and labels (e.g., including or not including language instrctions). Unless otherwise specified, we sampled video sequences from trajectories at intervals of 3, resulting in 4-frame video sequences, with 2 historical frames and 2 future frames. At this stage, we only used images from the third-person camera with 256×256 resolution. We use a maximum of $K = 1000$ diffusion step.

Settings of dynamics-modulated action stage. In this training stage, we used data from both the Open X-Embodiment (OXE) and CALVIN [14] datasets for training. We additionally incorporated images from a wrist camera as extra observations. For the data in OXE, we predicted 12 action steps, whereas for the CALVIN data, we predicted 10 steps. For the diffusion policy head, we set the noise addition steps to 100. We used a 224×224 third-person camera and a 224×224 wrist camera. In this stage, we set VDT's diffusion step to the maximum, i.e., $k = K$, and used pure noise as V_s^K. The VDT does not conduct the iterative denoising process. As for the diffusion-based action head, we set the maximum diffusion step as $L = 100$. More details can be found in Appendix A.2

5.1.2 Benchmark and Baselines

Simulation Evaluation: The CALVIN benchmark utilizes the PyBullet[55] simulator and involves a Franka Panda Robot arm interacting with various environments labeled A, B, C, and D. Each environment includes a desk, a sliding door, a drawer, a button controlling an LED, a switch for a lightbulb, and three colored blocks (red, blue, and pink). These environments differ in desk textures and object positions. CALVIN provides 24 hours of unstructured tele-operated play data, with 1% annotated with language descriptions. Each instruction chain consists of five sequential language instructions for execution. Evaluation follows a zero-shot generalization setup, training models on environments A, B, and C and testing on D. Performance metrics include success rates and average completion of sequential tasks, as per prior studies[26, 9]. Notably, CALVIN lacks a motion planner, requiring all models to predict robot pose trajectories.

Offline Evaluation: We also report offline metrics, including the average of xyz accuracy and euler angle accuracy (Avg xyz ang) and MSE for end-to-end action prediction on Bridge [56], Taco Play [57], Cable Routing [58] and Autolab UR5 [59], which are presented in OXE [8]. Following Octo [7], we use continuous action space. XYZ accuracy measures the precision of the robot's predicted 3D position (X, Y, Z coordinates) compared to the ground truth values during evaluation. Euler angle accuracy measures the precision of the robot's predicted orientation angles (rotations around X, Y, and Z axes) compared to the ground truth values during offline evaluation. Specifically, XYZ accuracy refers to whether we predicted the XYZ delta within 0.5 radians and 50% of the norm while in motion. Euler angle accuracy indicates whether we predicted the rotation delta within 0.5 radians during movement. Additionally, we reported the mean squared error (MSE) which reflects how well each model predicts the actions.

Baselines: On the CALVIN benchmark, we compare our approach to the hierarchical 2D policies of MCIL [31], HULC [32], and SuSIE [33], which predict latent features or subgoal images based on language instructions and feed them into lower-level subgoal-conditioned policies. These methods can train the low-level policy using the full CALVIN dataset, rather than being restricted to the language-annotated subset. We also compare against large-scale 2D transformer-based policies like RT-1 [47], RoboFlamingo [26], and GR-1 [9], which pretrain on extensive interaction or observational (video-only) data. Additionally, to better highlight the performance of our method, we compare with the 3D methods, 3D Diffusion Policy [34] and 3D Diffuser Actor [35]. Both share a similar goal of combining 3D representations with diffusion objectives to learn manipulation from demonstrations using depth maps and camera extrinsics. As for OXE benchmark, we compared our method with

RT-1-X [8] and Octo [7]. RT-1-X is an openly available generalist robot policy which are trained on OXE using RT-1 [47] framework. Octo is a transformer-based diffusion policy that supports flexible task and observation definitions, pretrained on OXE. These baselines are used to validate the benefit of our two-stage training strategy since both RT-1-x and Octo are optimized with all data in a train-from-scratch manner. To demonstrate the effectiveness of using diffusion models in learning dynamics priors and modeling states in the first stage, we implemented a GPT-style (VidMan-GPT) policy, as a baseline. This baseline autoregressively predicts the next images and actions in the first stage. This allows us to compare, in a fair manner, the benefits of simultaneously predicting multiple frames in a diffusion-like manner versus predicting only the next frame in a GPT-like fashion for modeling dynamics priors and state representations. We only predict a single frame and the corresponding action. We discarded the diffusion scheduler and used random masking [60] to reconstruct images to calculate the video loss.

5.2 Comparison Results

Table 1: **Zero-shot long-horizon evaluation on CALVIN.** *All* denotes that the model is trained on the entire dataset, including visual data without language annotations, while *Lang* refers to training on only the language-labeled data. Our method outperforms the hierarchical 2D policies (MCIL [31], HULC [32] and SuSIE [33]) and large-scale 2D transformer-based policies (RT-1 [47] RoboFlamingo [26] and GR-1 [9]), while also remaining competitive compared to 3D-based policies (3D Diffusion Policy [34] and 3D Diffuser Actor [35]).

Method	Training Data	Tasks completed in a row					
		1	2	3	4	5	Avg. Len.
3D Diffusion Policy [34]	Lang	28.7	2.7	0	0	0	0.31
MCIL [31]	All	30.4	1.3	0.2	0	0	0.31
HULC [32]	All	41.8	16.5	5.7	1.9	1.1	0.67
RT-1 [47]	Lang	53.3	22.2	9.4	3.8	1.3	0.9
RoboFlamingo [26]	Lang	82.4	61.9	46.6	33.1	23.5	2.48
SuSIE [33]	All	87	69	49	38	26	2.69
GR-1 [9]	Lang	85.4	71.2	59.6	49.7	40.1	3.06
3D Diffuser Actor [35]	Lang	93.8	80.3	66.2	53.3	41.2	3.35
VidMan (Ours)	Lang	91.5	76.4	68.2	59.2	46.7	3.42

Multi-Task Performance. We first conducted the initial stage of training on OXE, then utilized CALVIN's static camera and wrist camera as the two input sources for the second stage of training on CALVIN. Following the setup in GR-1, we used the portion of the CALVIN dataset that includes language instruction labels, while also incorporating CALVIN's proprioceptive state data as additional input into the layer-wise adapter. Tab. 1 compares the performance of mainstream methods such as SuSIE, RT-1, RoboFlamingo, GR-1, and 3D Diffuser Actor against VidMan. Compared to hierarchical 2D policies like SuSIE, which struggle to effectively leverage language information at the low level despite being able to train on the entire dataset ("All"), these methods generally underperform in terms of generalization compared to end-to-end policies like GR-1 and RoboFlamingo. Our method adopts an end-to-end training approach, outperforming the best hierarchical architecture, SuSIE, by 0.73 in Avg. Len., while using less data. When compared to large-scale 2D transformer-based policies such as GR-1 and RoboFlamingo, which also use an end-to-end output approach, our method is highly comparable. Thanks to pretraining on the large-scale robotics dataset OXE and utilizing the Open-Sora architecture, which is more vision-friendly, our method outperforms GR-1 by a clear margin on average length (11.7% relative improvement). In comparison to 3D Diffuser Actor, which leverages depth information to predict actions, our model remains competitive due to the ability to pretrain on large-scale datasets without depth information, which are easier to collect.

Offline Performance. To evaluate offline performance, in this part, the second stage of our model is optimized with OXE dataset. We evaluate performance on in-distribution tasks. Specifically, we conduct two parts of evaluation: 1) the evaluation on domains that have small-scale sub-datasets in OXE (Taco Play, Cable Routing and AUTOLab UR5), where we would expect pretraining with larger datasets can significantly improve performance; 2) the evaluation on domains with large-scale dataset (Bridge), where we expect further improvement to be more challenging. We compare our results with RT-1-X, Octo-small, Octo-base, and VidMan-GPT on the four sub-datasets. We compared the average
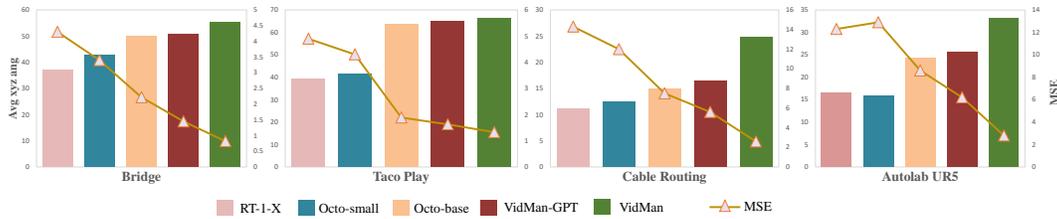
Figure 3: Offline Performance. The average accuracy (Avg xyz ang) of xyz accuracy and angle accuracy and MSE correspond to the left and right y-axes of the graph respectively. All models were trained on OXE and validated on offline performance across four datasets. VidMan outperformed Octo-base [7] by 5.6% on Bridge, 2.6% on Taco Play, 9.9% on Cable Routing, and 9.0% on Autolab UR5. Additionally, Our method also shows improvements over the VidMan-GPT approach.

Table 2: **Ablation Studies on Key Factors of VidMan..** We conduct finetune experiments on CALVIN. Average length is used. Best practice settings are marked in gray .

(a) **2-stage training.** Training only the action prediction in the second stage is crucial.

Setting	CALVIN
co-train	2.70
action-only	3.42

(b) **Pretrain type.** Video pretraining on domain-specific data is important.

Type	CALVIN
w/o Pre-train	2.89
w/ Ego4d	3.29
w/ OXE	3.42

(c) **Effect of layer-wise adapter.** Layer-wise adapter is important for performance improvements.

Setting	CALVIN
w/o adapter	1.54
freeze	2.98
no freeze	3.42

values of xyz accuracy and angular accuracy (Avg xyz ang), the higher the better, as well as the MSE (the lower the better). As shown in Fig. 3, it can be observed that our method outperforms the state-of-the-art open-source method, Octo, in both evaluation settings. Particularly on the small-scale sub-datasets CableRouting and Autolab UR5, our method improves the offline average xyz angle accuracy by 9.9% and 9.0% over Octo, respectively. The key difference between Octo and our method is in the optimization approach. Octo employs a co-training strategy, utilizing all data at once. However, our two-stage training strategy demonstrates a significant improvement, especially in domains with limited data. This improvement is attributed to our approach's ability to utilize the training data more efficiently by implicitly leveraging the inductive bias of inverse dynamics. Regarding Taco Play, the improvement we achieved is less notable. This is likely because the Taco Play scenes are relatively simple, and various methods tend to achieve similar performance levels in such environments. Additionally, our method shows improvements of 5.6% on the large-scale dataset Bridge which also demonstrates the effectiveness of our method. We have also achieved significant improvements over our own baseline, VidMan-GPT, across all four datasets, demonstrating the superiority of our model architecture. Details can be found in Tab. 7

5.3 Extra Ablation Studies

In this section, we conduct ablation studies for VidMan to answer the following questions: 1) Can co-training obtain a similar performance to our two-stage training? 2) Whether pretrained with general video data on the internet help improve the performance? 3) How much performance is gained by using the layer-wise self-attention block? 4) Does increasing the frame sampling interval positively impact action prediction? In the following, we answer each of these questions.

The importance of two-stage training. In our second stage, we only use action as a supervision signal to learn an implicit inverse dynamics model $P(a_t \mid s_t, s_{t+1})$ in Equ. (1). In other words, it suggests that we should not co-train with both actions and frames generation at the same time, otherwise a joint distribution $P(a_t, s_{t+1} \mid s_t)$ is learning. To validate this viewpoint, we conducted experiments where the frames generation loss $\mathcal{L}_v$ is also used in the second stage along with $\mathcal{L}_a$. The result of this variant is presented as "co-train" in Tab. 2a. In comparison to using only $\mathcal{L}_a$ in the second stage, namely "action-only" in the table, the average score of "co-train" degrades severely and approaches the performance of baseline methods in Tab. 2a. This further supports the effectiveness of our two-stage training strategy.

Table 3: **Effect of Frame Sampling Interval.** Setting the frame sampling interval to 3 is effective.

| interval | FID↓ | FVD↓ | Bridge Dataset | | | CALVIN |
			MSE ↓	xyz ↑	angle ↑	Avg. Len. ↑
1	29.5	327	2.80	42.8	46.6	2.24
2	32.1	345	1.29	48.1	51.2	2.82
3	38.4	376	**0.89**	**51.5**	**58.2**	**3.42**
4	51.9	422	1.20	48.2	53.1	3.03

Pretrained with general video data. To evaluate whether pretraining with general video data (not specific to robotics) from the internet helps improve performance, we conducted two control experiments. "w/o Pre-train" refers to our model without video pretraining, "w/ Ego4d" represents pretraining during the first stage using the general dataset Ego4d [13] for video prediction, and "w/ OXE" refers to pretraining using the robot-specific dataset OXE. For the Ego4d dataset, we followed the same processing pipeline as GR-1. As shown in Tab. 2b, pretraining with robot-specific video data resulted in a 0.53 increase in the average task length on CALVIN compared to "w/o Pre-train", while pretraining with general data ("w/ Ego4d") slightly decreased performance by 0.13. This suggests that training on web-based general data can yield marginal improvements, and pretraining in robotics domains significantly boosts performance.

Effect of layer-wise adapter. To measure the effect of the layer-wise adapter, we removed it (w/o adapter) and directly extracted the observed information into the action policy head by concatenating learnable action token to observation tokens before Open-Sora blocks. We found that this significantly reduced performance, comparing row 1 and row 3 in Tab. 2c. Additionally, we found that training only the layer-wise adapter while keeping the Open-Sora blocks fixed leads to faster convergence, as shown in Fig. 4. The training loss of the "freeze" curve converges faster than the "not freeze" curve. If both the adapter and Open-Sora blocks are trained, better accuracy can be achieved, by comparing row 2 and row 3 in Tab. 2c. In this sense, we can quickly adapt to a specific robotic scenario with the adapter being trained. In brief, layer-wise adapter can provide good performance. If Open-Sora blocks' parameters are not fixed, better performance can be achieved; if the parameters are fixed, faster convergence can be obtained.
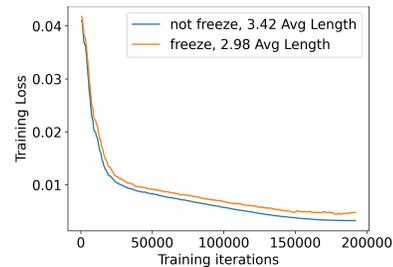

Figure 4: Efficiency comparison between two types of training.

Effect of frame sampling interval. We compare the effectiveness of different frame sampling intervals (*i.e.* 1, 2, 3, and 4) on Bridge Dataset and CALVIN. With a larger frame interval, a longer historical information our model can perceive and a longer future information needed to be predicted. The total length of the four experimental frames is 4, including 2 historical frames and 2 future frames. Results are shown in Tab. 3. It is observed that increasing the intervals from 1 to 3 improves performance on Bridge and CALVIN. The potential reason is that consecutive frames are very similar, and predicting frames that are farther away from the current step helps the robot gain a better understanding of the future. However, when the interval increases from 3 to 4, this improvement saturates quickly. We speculate that predicting frames too far into the future may not offer effective guidance for immediate local action prediction.

6 Conclusion

In this paper, we propose VidMan, a novel framework utilizing video diffusion models for robot imitation learning, which addresses the limitations of current GPT-style paradigms in real-time applications. By combining a Dynamics-aware Visionary Stage, which develops a deep understanding of environment dynamics through pre-training on the Open X-Embodiment dataset, with a Dynamics-modulated Action Stage that efficiently integrates this knowledge into action prediction, VidMan achieves both high precision and computational efficiency. This two-stage approach, ensures robust and rapid action generation, significantly improving performance on benchmarks like CALVIN and the OXE dataset. In the future, we will expand VidMan to be able to perceive more dimensions of information.

References

[1] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1386–1383. IEEE, 2017.

[2] Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943. IEEE, 2014.

[3] Ester Martinez-Martin and Angel P Del Pobil. Vision for robust robot manipulation. *Sensors*, 19(7):1648, 2019.

[4] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *The International Journal of Robotics Research*, 41(7):690–705, 2022.

[5] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[6] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[7] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. `https://octo-models.github.io`, 2023.

[8] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[9] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.

[10] Haoran He, Chenjia Bai, Ling Pan, Weinan Zhang, Bin Zhao, and Xuelong Li. Large-scale actionless video pre-training via discrete diffusion for efficient policy learning. *arXiv preprint arXiv:2402.14407*, 2024.

[11] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[12] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

[13] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[14] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[15] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[16] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[17] Philip M Groves and Richard F Thompson. Habituation: a dual-process theory. *Psychological review*, 77 (5):419, 1970.

[18] Jonathan St BT Evans. Dual process theory: Perspectives and problems. *Dual process theory 2.0*, pages 137–155, 2017.

[19] Bertram Gawronski and Laura A Creighton. Dual process theories 14. *The Oxford handbook of social cognition*, page 282, 2013.

[20] Keith E Stanovich. *Who is rational?: Studies of individual differences in reasoning*. Psychology Press, 1999.

[21] Zangwei Zheng, Xiangyu Peng, and Yang You. Open-sora: Democratizing efficient video production for all, March 2024. URL https://github.com/hpcaitech/Open-Sora.

[22] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[23] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[24] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[25] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[26] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[27] 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.

[28] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[29] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

[30] Zhijian Huang, Tao Tang, Shaoxiang Chen, Sihao Lin, Zequn Jie, Lin Ma, Guangrun Wang, and Xiaodan Liang. Making large language models better planners with reasoning-decision alignment. *arXiv preprint arXiv:2408.13890*, 2024.

[31] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[32] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022.

[33] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

[34] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024.

[35] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[36] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[37] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[38] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11444–11453, 2020.

[39] Yang Cao, Yihan Zeng, Hang Xu, and Dan Xu. Coda: Collaborative novel box discovery and cross-modal alignment for open-vocabulary 3d object detection. In *NeurIPS*, 2023.

[40] Yang Cao, Yihan Zeng, Hang Xu, and Dan Xu. Collaborative novel object discovery and box-guided cross-modal alignment for open-vocabulary 3d object detection. *arXiv preprint arXiv:2406.00830*, 2024.

[41] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[42] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[43] Lewei Yao, Jianhua Han, Youpeng Wen, Xiaodan Liang, Dan Xu, Wei Zhang, Zhenguo Li, Chunjing Xu, and Hang Xu. Detclip: Dictionary-enriched visual-concept paralleled pre-training for open-world detection. *Advances in Neural Information Processing Systems*, 35:9125–9138, 2022.

[44] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

[45] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[46] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

[47] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[48] Vikash Kumar, Rutav Shah, Gaoyue Zhou, Vincent Moens, Vittorio Caggiano, Abhishek Gupta, and Aravind Rajeswaran. Robohive: A unified framework for robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[49] 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.

[50] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[51] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[52] Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, and Sergey Tulyakov. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. *arXiv preprint arXiv:2402.19479*, 2024.

[53] Alexandros Stergiou and Ronald Poppe. Adapool: Exponential adaptive pooling for information-retaining downsampling. 2021.

[54] Wenjing Wang, Huan Yang, Zixi Tuo, Huiguo He, Junchen Zhu, Jianlong Fu, and Jiaying Liu. Videofactory: Swap attention in spatiotemporal diffusions for text-to-video generation. *arXiv preprint arXiv:2305.10874*, 2023.

[55] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.

[56] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[57] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task agnostic offline reinforcement learning. 2022.

[58] Jianlan Luo, Charles Xu, Xinyang Geng, Gilbert Feng, Kuan Fang, Liam Tan, Stefan Schaal, and Sergey Levine. Multi-stage cable routing through hierarchical imitation learning. *arXiv pre-print*, 2023. URL https://arxiv.org/abs/2307.08927.

[59] Lawrence Yunliang Chen, Simeon Adebola, and Ken Goldberg. Berkeley UR5 demonstration dataset. https://sites.google.com/view/berkeley-ur5/home.

[60] 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.

[61] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[62] 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. PMLR, 2021.

[63] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[64] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

[65] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[66] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[67] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.

A Appendix

The outline of the Appendix is as follows:

- Negative impacts and limitations.
- Details of the network and training process.
- Description and usage guide for OXE.
- Additional experimental results.
- Visualizations of video prediction and action prediction.

A.1 Negative Impacts and Limitations

Potential Negative Social Impact. Our method has no ethical risk on dataset usage and privacy violation since all the benchmarks are publicly available and transparent.

Limitations and Future Works. Our current model operates solely on 2D vision and does not possess 3D perception capabilities. This limitation affects its performance on tasks that demand accurate 3D spatial understanding. Our future work will prioritize the integration of 3D perception into the model. Our model's capacity to comprehend complex human instructions is limited. It currently lacks the sophistication of a CLIP's language encoder. We are considering state-of-the-art Large Language Models (LLMs), such as LLama [61], as a promising avenue for future exploration to enhance these capabilities. The model's perception is not fine-grained. It processes the entire image as a single input for action prediction. We believe that incorporating fine-grained auxiliary inputs, such as object-bounding boxes or masks, could significantly enhance the model's performance. We are exploring ways to integrate such detailed inputs to refine the model's perceptual abilities.

A.2 Network and Training Details

Model details. For the video autoencoder, we utilized Stabilityai's VideoAutoencoderKL [49] to encode the video into embeddings. For the text encoder, we employed CLIP's text encoder [62]. For the diffusion model, we used STDiT-XL/2 [21].

Training details. The video diffusion transformer of VidMan contains 12 layers (we reduced the original number of layers in Open-Sora to 12) and 16 heads with a hidden size of 1152. While training on OXE for video prediction in stage 1, we used 16 Nvidia V100 32GB GPUs, with a batch size of 24 per GPU resulting in a total batch size of 384. This stage will cost 42 hours. While training on OXE in stage 2, we used 16 Nvidia V100 32GB GPUs, with a batch size of 18 per GPU. We employed gradient accumulation, updating the parameters every 5 steps, resulting in a total batch size of 1440. This process took 32 hours. While training on CALVIN in stage 2, we used 8 Nvidia V100 32GB GPUs with a total batch size of 400. We use the AdamW [63] optimizer with an inverse square root decay learning rate schedule [64], with weight decay of 0.1. Training hyperparameters are shown in Tab. 4

Table 4: Training Hyperparameters

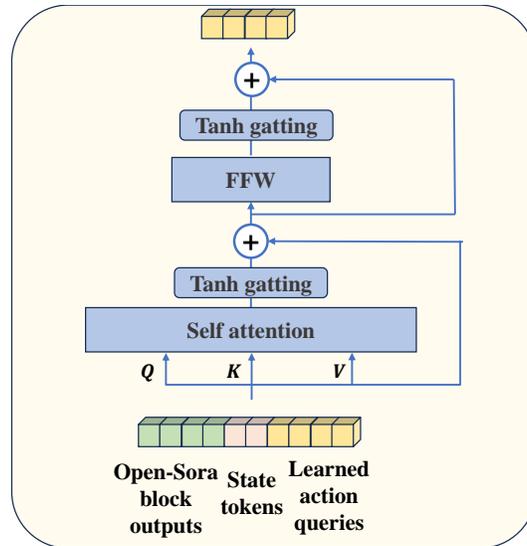| Hyperparameters	1st Stage	2nd Stage OXE Training	2nd Stage CALVIN Training
batch size	384	1440	400
learning rate	2e-5	1e-4	1e-4
dropout	0.1	0.1	0.1
optimizer	AdamW	AdamW	AdamW
weight decay	0.1	0.1	0.1
lr schedule	None	inverse square root decay	inverse square root decay
training steps	100k	300k	20 epochs

Figure 5: Our model utilizes a layer-wise adapter, which includes a self-attention layer and a feed-forward network (FFN). This block uses a gating mechanism to distill the information extracted by the Open-Sora block into the action query.

Layer-wise adapter. As shown in Fig. 5, following [22], our model employs a layer-wise self attention block, which comprises a self-attention layer and a feed-forward network (FFN). This block incorporates a gating mechanism to refine the information extracted by the VDT block into the action query.

A.3 More Details of Open X-Embodiment Datasets

We trained VidMan using a curated selection from the Open X-Embodiment Dataset [8], which is a rich repository of robot learning datasets. Our training data encompasses a variety of robot forms, environments, and tasks, reflecting a broad spectrum of robot types, sensor configurations, and labeling styles, including the presence or absence of language instructions.

Inspired by the approach taken in Octo [7], we began by filtering out datasets that did not include image streams or those that did not utilize delta end-effector control. From the remaining datasets, we carefully selected those with the highest diversity and task relevance, omitting any that were too repetitive, had low-resolution images, or were too specialized.

We then divided the chosen datasets into two groups: "more diverse" and "less diverse", based on the variety of tasks and environments they represented. To ensure a balanced training, we assigned a higher weight to the "more diverse" datasets. Additionally, we adjusted the weight of some large datasets that had an abundance of data points to maintain a balanced mix.

To streamline the training process, we filled in any gaps in the camera channels with zero-padding and standardized the gripper action spaces across all datasets. This standardization meant that a gripper command of +1 consistently signified "the gripper is open", and 0 signified "the gripper is closed".

A.4 Additional Experimental Results

Why use pure noise in stage 2. In the second stage, a key issue is how to utilize the video prediction model pre-trained in the first stage for action prediction. We replaced the noisy images from the first stage with zero embeddings, pure noise, and images without noise, and then concatenated them with historical frames along the channel dimension as input to the VDT. All experiments were trained and tested for offline validation on the Bridge dataset. These experiments used a small batch size of 192 and were trained for 100k iterations with VDT block frozen. The results are shown in Tab. 5. We found that using pure noise during training allows the model to achieve a certain level of performance during evaluation. One possible reason is that using pure noise aligns with the denoising process, effectively distilling the pre-trained model's future perception capabilities into action prediction.

Table 5: V_c^k **type in stage 2.** Concatenating historical frames with noise in the channel dimension is an effective way.

type	MSE↓	xyz ↑	angle ↑
no_noise	11.2	6.6	0.4
pure_noise	**4.8**	**32.7**	**37.6**
pure_zero	12.1	2.3	6.5

Table 6: **Effect of historical frames v.s. future frames.** The length of historical frames is more important than the length of future frames.

| history, future | FID↓ | FVD↓ | Bridge Dataset | | | CALVIN |
			MSE↓	xyz ↑	angle ↑	Avg. Len. ↑
m=1, n=1	28.9	318	4.20	37.0	41.6	2.09
m=2, n=1	27.3	305	1.80	48.9	55	2.80
m=2, n=2	38.4	376	0.89	**51.5**	**58.2**	**3.42**

Effect of historical frames v.s. future frames. To explore whether the length of historical frames or future frames has a greater impact on action prediction performance, we conducted analytical experiments with varying lengths of historical and future frames. We experimented with different settings: 1) history frame n=1, future frame m=1; 2) n=2, m=1; n=2, m=2. We kept the length of historical and future frames consistent across both the first and second training stages. The frame interval was set to 3. Results are shown in Tab. 6. Comparing row 1 and row 2, if the length of historical frames is shortened, there will be a significant decrease in performance on both Bridge and CALVIN. Comparing row 2 and row 3, shortening the length of future frames also leads to a decrease in performance, but not as significant as with the historical frames.

Detailed numerical metrics on OXE data. The numerical metrics in Fig. 3 are displayed in Tab. 7.

Results on RLBench. We also conducted experiments on RLBench [23]. We used 18 tasks from RLBench for multi-task robot learning evaluation. Each task includes 2-60 variations. For example, in the stack blocks task, "stack 2 red blocks" and "stack 4 purple blocks" are considered as two variants. During testing, our model has to handle novel object poses, randomly sampled goals, and randomly sampled scenes with different semantic instantiations of object colors, shapes, sizes, and categories. Each multi-task agent is evaluated independently on all of the tasks. Evaluations are scored either 0 for failures or 100 for complete successes. We report average success rates on 25 evaluation episodes per task for agents trained with 100 demonstrations. On the RLBench benchmark,

Table 7: **Detailed numerical metrics on OXE data.** All models were trained on OXE and validated on offline performance across four datasets. VidMan outperformed Octo-base [7] by 5.6% on Bridge, 2.6% on Taco Play, 9.9% on Cable Routing, and 9.0% on Autolab UR5. Additionally, Our method also shows improvements over the VidMan-GPT approach.

| Method | Bridge | | | | Taco Play | | | |
	MSE↓	xyz ↑	angle ↑	avg xyz ang ↑	MSE↓	xyz ↑	angle ↑	avg xyz ang ↑
RT-1-X	4.3	36.2	38.1	37.2	4.9	36.4	42.1	39.3
Octo-small	3.4	40.3	45.5	42.9	4.3	39.0	44.1	41.6
Octo-base	2.2	45.8	53.8	49.8	1.9	62.5	65.0	63.8
VidMan-GPT	1.5	46.5	55.2	50.9	1.6	62.7	67.0	64.9
VidMan	0.8	52.3	58.4	55.4	1.0	64.3	68.4	66.4

| Method | Cable Routing | | | | Autolab UR5 | | | |
	MSE↓	xyz ↑	angle ↑	avg xyz ang ↑	MSE↓	xyz ↑	angle ↑	avg xyz ang ↑
RT-1-X	14.3	8.7	13.5	11.1	12.3	13.9	19.2	16.6
octo-small	12.0	10.0	15.0	12.5	12.9	13.8	18.0	15.9
octo-base	7.5	12.5	17.2	14.9	8.6	23.2	25.3	24.3
VidMan-GPT	5.6	13.2	19.8	16.5	6.2	24.3	27.0	25.7
VidMan	3.3	22.4	27.1	24.8	3.4	29.4	37.0	33.2

Models	Open Drawer	Slide Block	Sweep to Dustpan	Meat off Grill	Turn Tap	Put in Drawer	Close Jar	Drag Stick	Stack Blocks	Screw Bulb
PerAct	80	72	56	84	80	68	60	68	36	24
RVT	71.2 ± 6.9	81.6 ± 5.4	72.0 ± 0.0	88.0 ± 2.5	93.6 ± 4.1	88.0 ± 5.7	52 ± 2.5	99.2 ± 1.6	28.8 ± 3.9	48.0 ± 5.7
Act3D	93	93	92	94	94	90	92	92	12	47
Ours	94.4 ± 3.6	97.6 ± 4.4	92.8 ± 1.8	90.4 ± 2.2	96.8 ± 3.3	83.2 ± 1.8	88 ± 2.8	84.8 ± 3.5	48 ± 0.0	66.4 ± 2.1

| Models	Put in Safe	Place Wine	Put in Cupboard	Sort Shape	Push Buttons	Insert Peg	Stack Cups	Place Cups	Avg. Success	Inf. Speed
PerAct	44	12	16	20	48	0	0	0	42.7	-
RVT	91.2 ± 3.0	91.0 ± 5.2	49.6 ± 3.2	36.0 ± 2.5	100.0 ± 0.0	11.2 ± 3.0	26.4 ± 8.2	4.0 ± 2.5	62.9	11.6
Act3D	95	80	51	8	99	27	9	3	65.1	-
Ours	67.2 ± 3.3	79.6 ± 0.0	32.8 ± 3.3	48.0 ± 0.0	89.6 ± 1.6	21.6 ± 3.6	18.2 ± 1.8	13.2 ± 1.8	67.4	18.3

Table 8: **Multi-Task Performance on RLBench-100.**

we compared our method with 3 methods: PerAct [65], RVT [66], Act3D [67]. To align with methods like RVT, we also reported the variance of 5 seeds in the evaluation on RLBench.

A.5 Visualization

Visualization of our video prediction. We investigate the video prediction performance of VidMan trained on OXE in stage 1. Qualitative results are shown in Fig. 6. It can be observed that VidMan correctly predicts future frames, although some details (such as occluded objects) are missing. This predictive capability in the second stage can strongly guide action prediction as potential future forecasts.

Visualization of our offline action prediction. As shown in Fig. 7, we visualized VidMan's results on OXE offline evaluation. It can be observed that our model effectively fits the ground truth actions across various scenarios, potentially demonstrating its applicability to real robot tasks.
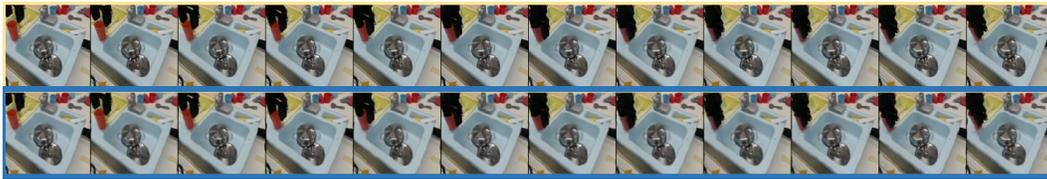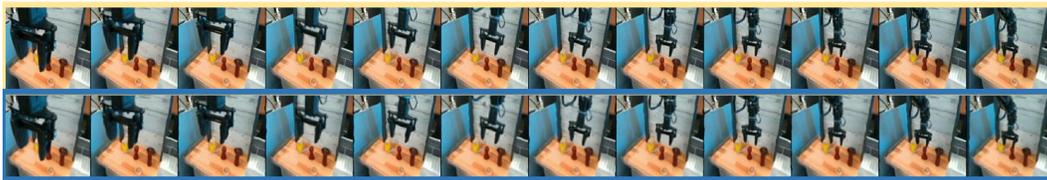
"put cup from anywhere into sink"

"move the silver pot to the top right of the stove"
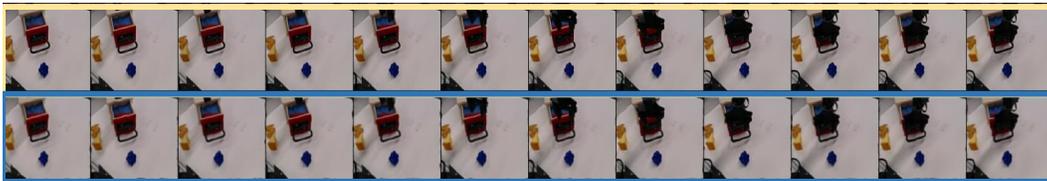
"Close the draw"

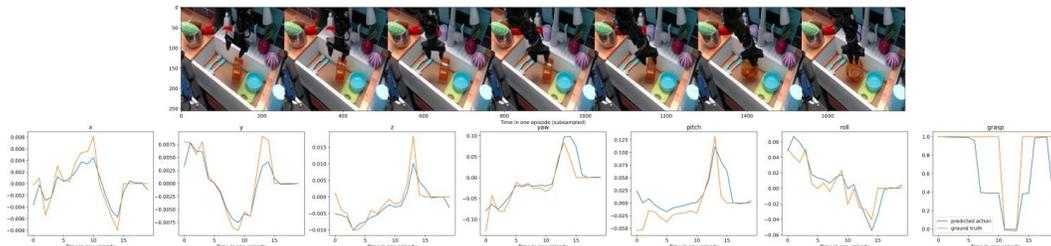"remove the cube from the tower and put it in the middle of the table"
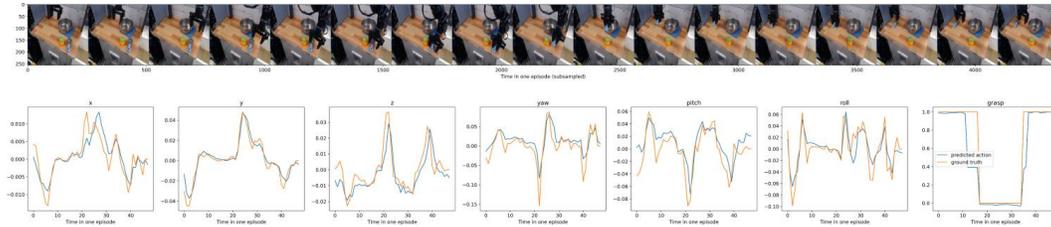
"folding from left bottom to the right side"
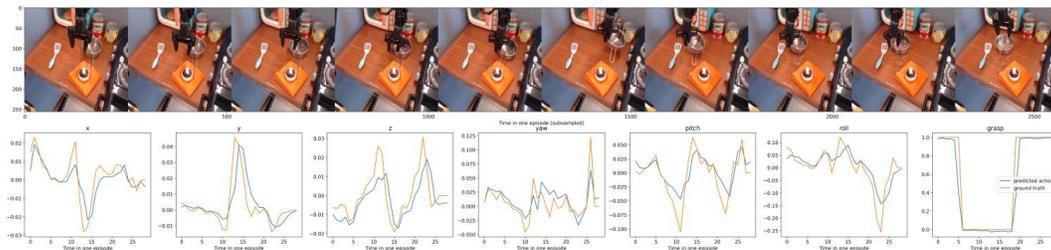
"flip pot upright in sink"

Figure 6: **Video prediction results on OXE.** The images in yellow boxes are ground-truth images; the images in blue boxes are predicted images. The language instruction is placed below the image.
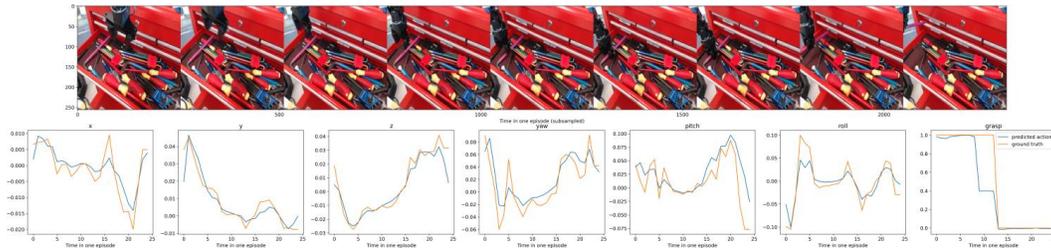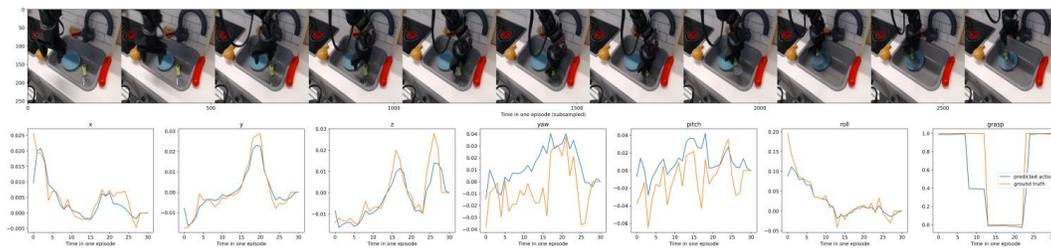
"Flip orange pot upright in sink"

"move the knife from right side to left side"

"move the pot on left upper side of the orange cloth"

"pick up violet Allen key"

"put spatula on plate sink"

Figure 7: **Offline action prediction results on OXE.** The upper part of each group of images shows subsampled frames from an episode, while the lower part displays the true and predicted 7D pose results, including x, y, yaw, pitch, roll, and grasp over time.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We believe that the main claims reflect the areas we are targeting: robot manipulation and imitation learning.

   Guidelines:
   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the Appendix.

   Guidelines:
   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: We do not have theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: See Section 5 and Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We already include the data and instructions in Section 5 and Appendix and the code will be released upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See Section 5 for more details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Our experiments are quite stable with multiple runs. All results are observed via a fixed seed.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

    Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

    Answer: [Yes]

    Justification: See Appendix for more details.

    Guidelines:

    - The answer NA means that the paper does not include experiments.
    - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
    - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
    - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

    Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

    Answer: [Yes]

    Justification: The research is conducted with the NeurIPS code of Ethics in every respect.

    Guidelines:

    - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
    - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
    - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have cited the creators of all existing assets. All the used existing assets are open-source.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.