

A APPENDIX

A.1 IMPLEMENTATION DETAILS

In this section, we outline the implementation details of our framework.

We employ a MAE-pretrained ViT-B (He et al., 2022) as the vision encoder. At each timestep, images are captured from two views: eye-on-hand and eye-on-base. Each image is processed by the vision encoder to produce 196 latent vectors, which represent local patch information, along with a [CLS] token that encodes the global representation of the image. Directly inputting all 197 tokens into the transformer backbone would create a significant computational burden, particularly when processing long histories. Moreover, many image details are redundant for accomplishing manipulation tasks. To address this, we utilize the Perceiver Resampler (Alayrac et al., 2022) to condense the image representations and extract task-relevant features. The Perceiver Resampler employs learnable latent vectors with a shape of (num_latents, dim), where num_latents is significantly smaller than the number of image tokens; in our implementation, num_latents = 6. Through Perceiver Attention, these latent vectors condense the input image features, along with the [CLS] token, to form the final image tokens.

The robot state consists of the arm state and the gripper state. The arm state includes the end-effector position and its rotation in Euler angles, resulting in a six-dimensional representation. The gripper state is a binary value indicating whether the gripper is open or closed. We tokenize the robot state using an MLP. Specifically, the gripper state is first converted into a one-hot encoding. The one-hot encoding of the gripper state and the arm state are then each passed through separate linear layers. The outputs are concatenated and passed through a final linear layer to produce the state token.

Language instructions are encoded using the CLIP ViT-B/32 text encoder (Radford et al., 2021) and projected through a linear layer to generate the language token.

At each timestep, we append $[FRS]$ and $[INV]$ tokens to read out foresight and actions. Specifically, 18 $[FRS]$ tokens are appended to extract representations for two views, while 3 $[INV]$ tokens are appended to predict actions across three steps, ensuring temporal action consistency and robustness to idle actions.

After passing through the transformer backbone, the action and image latents generated by the $[INV]$ and $[FRS]$ tokens are input to the action decoder and image decoder to predict actions and images for conditional visual foresight and inverse dynamics prediction.

The action decoder is an MLP that decodes the action latent into a seven-dimensional action vector. First, the action latent is processed by a linear layer followed by a ReLU activation function. Then, it passes through a second linear layer with ReLU activation. The output is fed into two independent decoders: the arm action decoder and the gripper action decoder. The arm action decoder maps the high-dimensional vector to a six-dimensional output through a linear layer, applying a Tanh activation function to constrain the arm action within the range $[-1, 1]$. The gripper action decoder also employs a linear layer to map the latent vector to a one-dimensional output, applying a Sigmoid activation function to constrain the gripper action between $[0, 1]$. A gripper action value of 0.5 or higher is interpreted as 1 (closed), while a value below 0.5 is interpreted as 0 (open).

For image decoding, following (He et al., 2022), we use a vision transformer (ViT) as the image decoder. The image decoder receives the image latent and mask tokens from the transformer backbone as input. Positional information is provided through fixed sine-cosine positional encodings. The inputs are processed by multiple transformer encoder blocks. Finally, a linear layer predicts the pixels for each patch, generating the image that represents the predicted future state.

We present relevant hyperparameters during both pretraining and finetuning in Table 5.

Our model overall contains 316M parameters, where only 65M is tunable. For all simulation results, we use 8 4090 GPUS to pre-train and fine-tune. The pre-training for CALVIN ABC-D requires about 40 hours and for LIBERO-LONG, it requires about 30 hours. The fine-tuning for CALVIN ABC-D requires about 24 hours and for LIBERO-LONG, it requires 6 hours.

Table 5: Training hyperparameters.

Hyperparameters	Pre-training	Fine-tuning
Batch size	640 (LIBERO and CALVIN) 2048 (Real)	512
Learning rate	1e-4	1e-3
Optimizer	AdamW	AdamW
Learning rate schedule	Cosine decay	Cosine decay
Training epochs	30 (LIBERO and Real) 20 (CALVIN)	40 (LIBERO and Real) 20 (CALVIN)
History length	7 (LIBERO and Real) 10 (CALVIN)	7 (LIBERO and Real) 10 (CALVIN)
Action chunk length	3	3

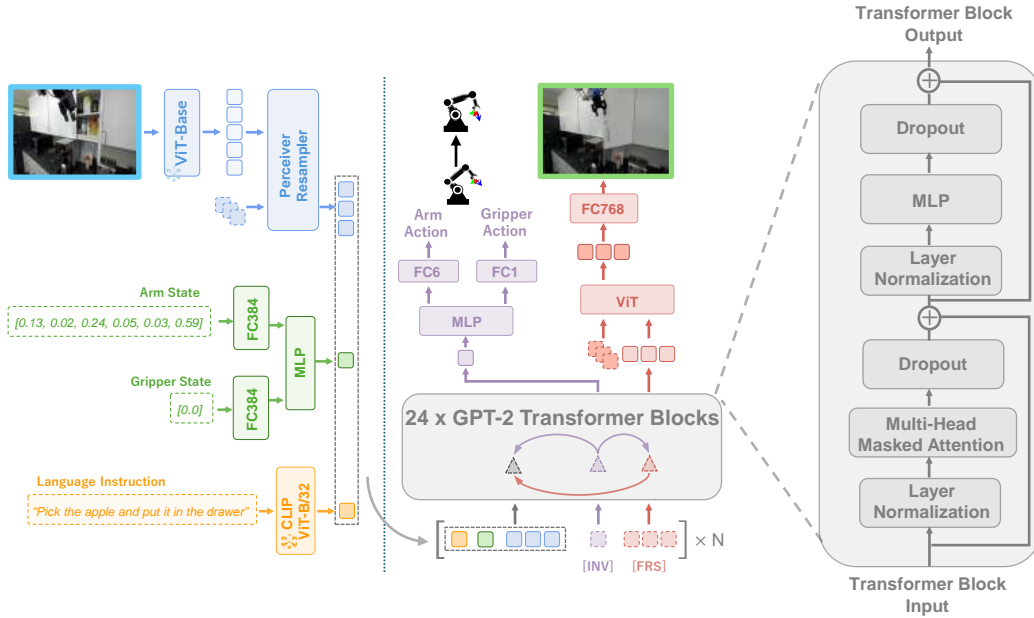


Figure 6: Network Architecture.

Table 6: Hyperparameters for the transformers in our policy.

	Hidden size	Number of layers	Number of heads
image encoder	768	12	12
perceiver resampler	768	3	8
transformer backbone	384	24	12
image decoder	384	2	16

A.2 NETWORK ARCHITECTURE

As can be seen in Figure 6, Seer consists of the following modules: image encoder, perceiver resampler, robot state encoder, language encoder, transformer backbone, action decoder and image decoder. Here we describe each module in detail:

- **image encoder:** It is a MAE pre-trained ViT-Base (He et al., 2022). Details can be seen in Table 6.
- **perceiver resampler:** It is a module to reduce the number of image tokens efficiently. Details can be seen in Table 6.
- **robot state encoder:** It consists of linear layers and some MLPs, projecting robot states into a latent space.
- **language encoder:** It is a CLIP (Radford et al., 2021) ViT-B/32 text encoder.
- **transformer backbone:** It takes image tokens, language tokens, robot states tokens, [INV], [FRS] as inputs. It comprises 24 layers of GPT-2 transformer blocks, with a hidden size 384 and 12 heads.
- **action decoder:** It involves MLPs and linear layers to decode the action latent into seven-dimensional action vector.
- **image decoder:** It is a ViT-based transformer followed by a linear layer. Details can be seen in Table 6.

A.3 BASELINE IMPLEMENTATION

In the simulation benchmark, we report the scores for Roboflamingo, Susie, GR-1, and the 3D Diffusor Actor from their respective papers. For MTACT and OpenVLA, we reproduce the results using the official code. For MVP and MPI, we replace the vision encoder in our policy with their pretrained versions. Thanks to the strong design of our policy, MVP and MPI show competitive performance, though they only approach the results of our policy without pretraining.

A.4 LIBERO-LONG EXPERIMENT DETAILS

LIBERO (Liu et al., 2024) is a novel benchmark for lifelong learning in robot manipulation, comprising four task suites: LIBERO-SPATIAL, LIBERO-OBJECT, LIBERO-GOAL, and LIBERO-100. The first three task suites are designed to disentangle the transfer of declarative and procedural knowledge, while LIBERO-100 consists of 100 tasks involving entangled knowledge transfer. LIBERO-100 includes 100 tasks that require diverse object interactions and versatile motor skills. LIBERO-100 is divided into 90 short-horizon tasks (LIBERO-90) and 10 long-horizon tasks (LIBERO-LONG). We use LIBERO-90 as the pretraining dataset, while LIBERO-LONG is utilized for the downstream finetuning and evaluation.

The policy use images from both fixed and gripper cameras to observe the environment, which were resized to 224x224 pixels. We also incorporated the robot state to help the policy understand the robot’s self-state, including the position and orientation of the end effector and the gripper state indicating the width between the grippers. The action space consists of a seven-dimensional vector: six dimensions represent arm actions, and one dimension indicates the gripper’s open/close state. The arm action represents the 6D pose (position and orientation) of a controlled frame. This frame is located between the fingers of robots.

A.5 CALVIN ABC-D EXPERIMENT DETAILS

CALVIN (Mees et al., 2022) is a simulated benchmark designed for learning long-horizon, language-conditioned tasks. Its goal is to enable the development of agents capable of solving various robotic manipulation tasks using only onboard sensors and instructions provided in natural human language. The tasks in CALVIN are complex, involving long sequences and intricate language instructions. It supports flexible configurations of sensor suites. Agents are evaluated in a zero-shot manner on novel language instructions and unfamiliar environments.

The CALVIN benchmark includes four distinct but structurally similar environments—Env A, B, C and D. Each environment features a Franka Emika Panda robot arm equipped with a parallel gripper, as well as a desk with a sliding door and a drawer that can be opened and closed. And there are several objects, such as blocks and buttons, on the desk. To more effectively assess the generalization of the learned policies, each environment features distinct textures, and objects like the sliding door, drawer, and light button, are placed in different positions.

CALVIN offers rich observations for robot learning. We use images from both fixed and gripper-mounted cameras, resized to 224x224 pixels, along with robot state information, which includes end-effector position, orientation, and gripper state (open/close). The action space is a 7-dimensional vector: six dimensions correspond to end-effector displacement (position and orientation), and one dimension controls the gripper’s open/close state. The unstructured data includes exploratory and sub-optimal behaviors, comprising approximately 2.4 million interaction steps and 40 million short-horizon windows. Data from Env A, B, and C, which lacks language annotations, is used to pretrain the policy, while data with language annotations is used for downstream task learning. Env D is reserved for policy evaluation.

A.6 REAL WORLD EXPERIMENT DETAILS

A.6.1 DETAILED TASK SETTING

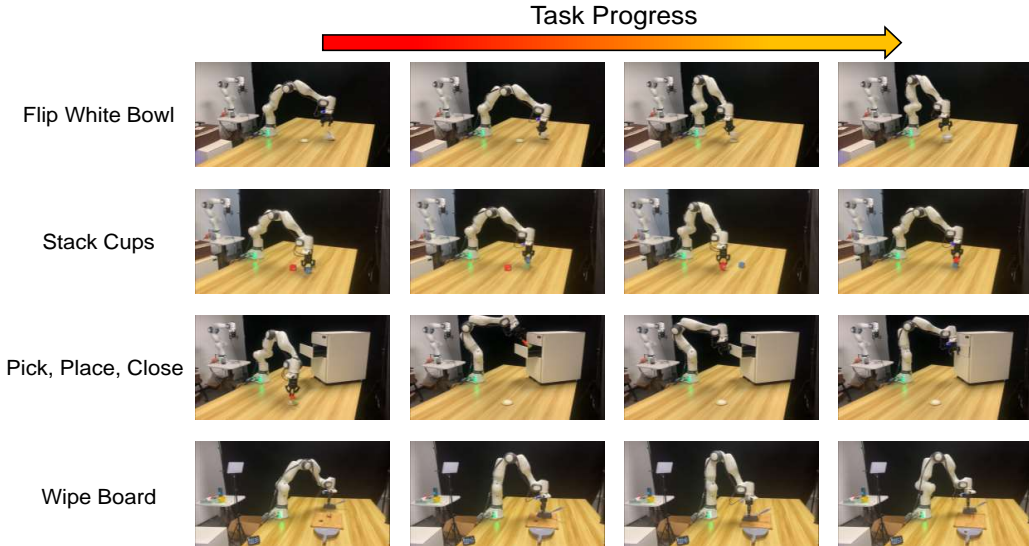


Figure 7: Task Progress.

Flip White Bowl: In this task, the white bowl is randomly placed on the table within a 40cm \times 40cm square, and the coaster is randomly placed within a 15cm \times 15cm square. The robot needs to pick up the white bowl and put it on the coaster. In the generalization test, 1 to 3 bowls with identical shapes, sizes and materials are added to disturb the policy. Success rate (SR) is recorded as 100% only when the white bowl is placed on the coaster safely. If the bowl is successfully grasped, the score will plus one (+1). If the bowl is successfully placed on the coast, the score will also plus one (+1). The full score in this task is 2.

Stack Cups: In this task, three cups of different sizes are randomly placed on the table within a 40cm \times 40cm square. The robot needs to cover the small cup with the middle one, and cover the middle cup with the big one. Only when all the cups are stacked precisely and in a correct order will the SR be recorded as 100%. The score will plus one (+1) when each primitive action (pick or place) is accomplished. The full score in this task is 4.

Pick, Place, Close: In this task, a drawer with three layers is fixed on the table. During each test, one of three layers is open. A carrot on the coaster is randomly placed within a 20cm \times 20cm square. The orientation of the carrot is randomized. The robot needs to pick the carrot, place it into

a certain layer, and close the drawer. The score will plus one (+1) when successfully (1) picking the carrot, (2) placing the carrot, and (3) closing the drawer. The full score in this task is 3.

Wipe Board: In this task, a board (30cm \times 40cm) and dustpan is fixed on the table. A brush is randomly placed within a 5cm \times 5cm square. 3 to 7 chocolate balls (diameter 1cm) are divided into 1 to 3 clusters and randomly placed on the whole board. Only when all the chocolate balls are swept into the dustpan safely will the SR be recorded as 100%. The score will plus one (+1) when (1) grasping the brush successfully, (2) sweeping partial chocolates into the dustpan, and (3) sweeping all chocolates into the dustpan. The full score in this task is 3.

A.6.2 ADDITIONAL HIGH-PRECISION AND CONTACT-RICH TASKS.

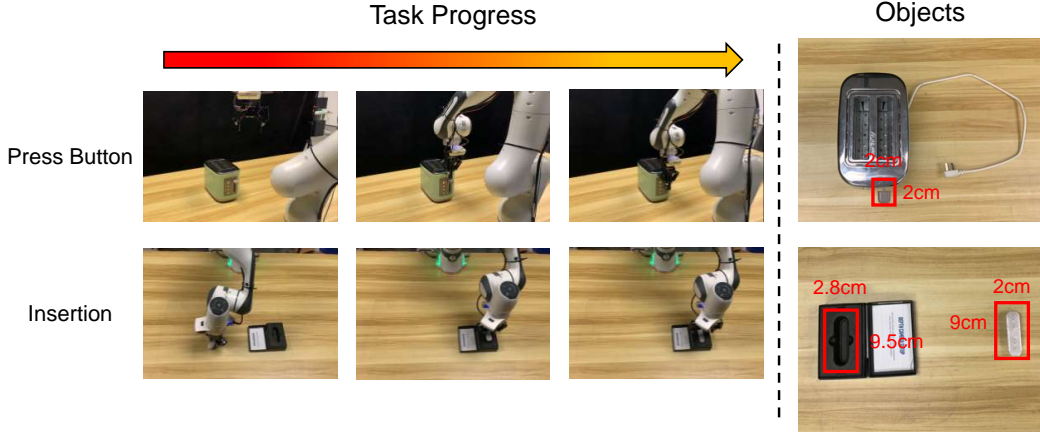


Figure 8: High-precision and contact-rich tasks.

Table 7: Results on additional high-precision and contact-rich tasks.

Method	Demos per Task	Press Button	Insertion
		SR (%) \uparrow / Score \uparrow	SR (%) \uparrow / Score \uparrow
MVP	100	46.7 / 17.0	26.7 / 11.0
Ours (w/o pre-train)	100	40.0 / 13.0	40.0 / 16.0
Ours	100	60.0 / 18.0	60.0 / 19.0

Press Button: In this task, the toaster is randomly placed in a 30cm \times 30cm square. The robot is required to approach the toaster, close the fingers, push the button from a top-down view, and exceed 3/4 of the scale (Figure 8). The score will plus one (+1) when (1) pushing the button successfully with no collision, and (2) exceeding 3/4 of the scale. The full score in this task is 2.

Insertion: In this task, a 2cm \times 9cm camera model is randomly placed in a 20cm \times 20cm square. The robot is required to pick the camera model and insert it into a 2.8cm \times 9.5cm groove without any collision (Figure 8). The score will plus one (+1) when (1) grasping the camera model, and (2) inserting successfully with no collision. The full score in this task is 2.

Results: As can be seen in Table 7, being pre-trained on the DROID dataset brings obvious improvements, compared to the scratch version and previous state-of-the-art baselines. Notably, both tasks require quite precise action predictions and collision-free interactions, showing our model’s potential in high-precision and contact-rich tasks.

A.6.3 REAL-WORLD IMPLEMENTATION DETAILS

During real-world training, we set the sequence length as 7, visual foresight steps and action prediction steps as 3. We use the MAE pre-trained vision encoder ViT-B and set the type of ViT-B

as bfloat16 to speed up inference. We find this quantization won't produce side effects on manipulation tasks. The pre-training dataset, e.g., DROID involves 76K successful trajectories, and the downstream fine-tuning data involves 400 demos. Key hyperparameters are listed in Table 5. We use the 9-th pre-trained checkpoint for fine-tuning and evaluate using the 17-th fine-tuned checkpoint.

For real-world baselines MVP and MPI, we simply replace the MAE pre-trained vision encoder with the MVP pre-trained and MPI pre-trained counterpart respectively in our network. We then fine-tune these two baselines on the downstream tasks and report performances. For OpenVLA, we choose to fully finetune its public released checkpoint with model size 7B pre-trained on OXE. The fine-tuning config is identical to the one trained on Bridge dataset in OpenVLA's public codebase. We fine-tune this large model using 8 A100 GPUs with more than 24 hours and use the checkpoint with lowest average validation loss to evaluate.

A.6.4 DETAILED REAL-WORLD RESULTS

Table 8: Detailed results (SR (%) / Score) in Flip White Bowl.

Case Index	MVP	MPI	OpenVLA	Ours (w/o pre-train, 20 demos)	Ours (20 demos)	Ours (w/o pre-train, 100 demos)	Ours (100 demos)
1	0.00 / 0.00	100 / 2.00	0.00 / 1.00	0.00 / 1.00	100 / 2.00	100 / 2.00	100 / 2.00
2	100 / 2.00	100 / 2.00	100 / 2.00	0.00 / 0.0	100 / 2.00	100 / 2.00	100 / 2.00
3	100 / 2.00	100 / 2.00	0.00 / 1.00	0.00 / 0.0	0.00 / 0.0	100 / 2.00	100 / 2.00
4	100 / 2.00	0.00 / 0.00	100 / 2.00	0.00 / 0.0	0.00 / 0.0	100 / 2.00	100 / 2.00
5	100 / 2.00	0.00 / 0.00	100 / 2.00	100 / 2.00	100 / 2.00	0.00 / 0.00	100 / 2.00
6	100 / 2.00	100 / 2.00	100 / 2.00	0.00 / 0.0	0.00 / 0.0	0.00 / 0.00	100 / 2.00
7	100 / 2.00	100 / 2.00	0.00 / 1.00	0.00 / 0.0	0.00 / 0.0	100 / 2.00	100 / 2.00
8	100 / 2.00	100 / 2.00	100 / 2.00	100 / 2.00	100 / 2.00	0.00 / 1.00	100 / 2.00
9	100 / 2.00	100 / 2.00	100 / 2.00	100 / 2.00	100 / 2.00	100 / 2.00	100 / 2.00
10	100 / 2.00	100 / 2.00	0.00 / 0.00	0.00 / 0.0	0.00 / 0.0	100 / 2.00	0.00 / 0.00
11	100 / 2.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.0	100 / 2.00	0.00 / 0.00	100 / 2.00
12	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	100 / 2.00	100 / 2.00	0.00 / 0.00	0.00 / 0.00
13	100 / 2.00	100 / 2.00	100 / 2.00	0.00 / 0.0	0.00 / 1.00	0.00 / 0.00	100 / 2.00
14	100 / 2.00	100 / 2.00	0.00 / 0.00	0.00 / 1.00	100 / 2.00	100 / 2.00	100 / 2.00
15	0.00 / 0.00	0.00 / 0.00	100 / 2.00	0.00 / 0.0	0.00 / 0.0	100 / 2.00	100 / 2.00

Table 9: Detailed results (SR (%) / Score) in Stack Cups.

Case Index	MVP	MPI	OpenVLA	Ours (w/o pre-train, 20 demos)	Ours (20 demos)	Ours (w/o pre-train, 100 demos)	Ours (100 demos)
1	0.00 / 3.00	100 / 4.00	0.00 / 2.00	0.00 / 0.00	0.00 / 1.00	100 / 4.00	100 / 4.00
2	100 / 4.00	0.00 / 3.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 1.00	100 / 4.00
3	0.00 / 3.00	0.00 / 3.00	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	0.00 / 2.00	100 / 4.00
4	100 / 4.00	0.00 / 1.00	0.00 / 1.00	0.00 / 1.00	0.00 / 0.00	100 / 4.00	100 / 4.00
5	0.00 / 1.00	0.00 / 3.00	0.00 / 2.00	0.00 / 1.00	0.00 / 1.00	100 / 4.00	0.00 / 2.00
6	0.00 / 1.00	100 / 4.00	0.00 / 0.00	0.00 / 1.00	0.00 / 2.00	100 / 4.00	100 / 4.00
7	0.00 / 0.00	100 / 4.00	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	0.00 / 1.00	100 / 4.00
8	0.00 / 0.00	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
9	100 / 4.00	100 / 4.00	0.00 / 0.00	100 / 4.00	0.00 / 0.00	100 / 4.00	100 / 4.00
10	0.00 / 1.00	0.00 / 1.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 2.00
11	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	0.00 / 1.00	0.00 / 1.00	0.00 / 0.00	0.00 / 0.00
12	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
13	0.00 / 0.00	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	0.00 / 0.00	100 / 4.00	0.00 / 2.00
14	0.00 / 1.00	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	100 / 4.00
15	100 / 4.00	0.00 / 1.00	0.00 / 0.00	0.00 / 1.00	0.00 / 1.00	100 / 4.00	100 / 4.00

Table 10: Detailed results (SR (%) / Score) in Pick, Place, Close.

Case Index	MVP	MPI	OpenVLA	Ours (w/o pre-train, 20 demos)	Ours (20 demos)	Ours (w/o pre-train, 100 demos)	Ours (100 demos)
1	100 / 3.00	100 / 3.00	0.00 / 0.00	100 / 3.00	100 / 3.00	100 / 3.00	100 / 3.00
2	100 / 3.00	100 / 3.00	100 / 3.00	0.00 / 0.00	100 / 3.00	100 / 3.00	100 / 3.00
3	100 / 3.00	100 / 3.00	0.00 / 0.00	0.00 / 2.00	100 / 3.00	0.00 / 2.00	100 / 3.00
4	0.00 / 0.00	100 / 3.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 2.00	100 / 3.00
5	0.00 / 0.00	100 / 3.00	0.00 / 2.00	100 / 3.00	100 / 3.00	100 / 3.00	100 / 3.00
6	0.00 / 1.00	0.00 / 0.00	0.00 / 1.00	0.00 / 1.00	0.00 / 1.00	0.00 / 2.00	100 / 3.00
7	100 / 3.00	100 / 3.00	0.00 / 0.00	0.00 / 0.00	100 / 3.00	100 / 3.00	100 / 3.00
8	100 / 3.00	100 / 3.00	0.00 / 0.00	100 / 3.00	100 / 3.00	100 / 3.00	100 / 3.00
9	0.00 / 1.00	0.00 / 0.00	100 / 3.00	0.00 / 0.00	0.00 / 0.00	0.00 / 1.00	0.00 / 2.00
10	100 / 3.00	100 / 3.00	0.00 / 0.00	0.00 / 0.00	0.00 / 2.00	100 / 3.00	100 / 3.00
11	100 / 3.00	0.00 / 0.00	0.00 / 1.00	0.00 / 0.00	0.00 / 1.00	100 / 3.00	100 / 3.00
12	0.00 / 1.00	0.00 / 1.00	0.00 / 1.00	0.00 / 1.00	0.00 / 0.00	100 / 3.00	0.00 / 1.00
13	100 / 3.00	100 / 3.00	0.00 / 0.00	100 / 3.00	100 / 3.00	100 / 3.00	100 / 3.00
14	100 / 3.00	100 / 3.00	0.00 / 2.00	0.00 / 0.00	0.00 / 0.00	100 / 3.00	100 / 3.00
15	0.00 / 1.00	0.00 / 1.00	0.00 / 0.00	0.00 / 0.00	100 / 3.00	100 / 3.00	100 / 3.00

To probe into how pre-training helps visual robot manipulation, we compare tasks success rates and score of MVP, MPI, OpenVLA and ours. Results are shown in Table 8, Table 9, Table 10 and Table 11. Under the same amount of fine-tuning data, our method achieves the best performance. Even only relying on 20% of fine-tuning data, our method still outperforms the scratch one.

Table 11: Detailed results (SR (%) / Score) in Wipe Board.

Case Index	MVP	MPI	OpenVLA	Ours (w/o pre-train, 20 demos)	Ours (20 demos)	Ours (w/o pre-train, 100 demos)	Ours (100 demos)
1	100 / 3.00	100 / 3.00	0.00 / 0.00	0.00 / 0.00	0.00 / 2.00	100 / 3.00	100 / 3.00
2	100 / 3.00	100 / 3.00	0.00 / 0.00	0.00 / 2.00	100 / 3.00	0.00 / 2.00	100 / 3.00
3	100 / 3.00	100 / 3.00	0.00 / 1.00	100 / 3.00	100 / 3.00	0.00 / 0.00	100 / 3.00
4	100 / 3.00	0.00 / 2.00	0.00 / 0.00	100 / 3.00	0.00 / 2.00	100 / 3.00	100 / 3.00
5	0.00 / 2.00	0.00 / 2.00	0.00 / 1.00	0.00 / 2.00	100 / 3.00	100 / 3.00	100 / 3.00
6	0.00 / 2.00	0.00 / 2.00	0.00 / 0.00	0.00 / 2.00	0.00 / 2.00	0.00 / 2.00	0.00 / 2.00
7	0.00 / 2.00	0.00 / 2.00	0.00 / 0.00	0.00 / 1.00	0.00 / 1.00	0.00 / 2.00	100 / 3.00
8	0.00 / 2.00	0.00 / 2.00	0.00 / 0.00	0.00 / 2.00	0.00 / 2.00	100 / 3.00	0.00 / 2.00
9	100 / 3.00	100 / 3.00	0.00 / 0.00	100 / 3.00	100 / 3.00	100 / 3.00	100 / 3.00
10	0.00 / 2.00	0.00 / 2.00	0.00 / 0.00	100 / 3.00	0.00 / 2.00	0.00 / 2.00	100 / 3.00
11	100 / 3.00	0.00 / 2.00	0.00 / 0.00	0.00 / 2.00	0.00 / 2.00	100 / 3.00	100 / 3.00
12	100 / 3.00	0.00 / 2.00	0.00 / 0.00	0.00 / 0.00	0.00 / 2.00	100 / 3.00	0.00 / 2.00
13	0.00 / 2.00	100 / 3.00	0.00 / 0.00	100 / 3.00	100 / 3.00	100 / 3.00	100 / 3.00
14	0.00 / 2.00	0.00 / 2.00	0.00 / 2.00	0.00 / 2.00	0.00 / 2.00	0.00 / 2.00	0.00 / 2.00
15	100 / 3.00	0.00 / 2.00	0.00 / 0.00	0.00 / 0.00	0.00 / 2.00	100 / 3.00	100 / 3.00

A.7 FORESIGHT VISUALIZATION

Figure 9 and Figure 10 presents both the predicted and ground truth future images in real world and simulated environment tasks. To improve training efficiency, we opt for a lightweight ViT-based image decoder instead of a diffusion-based generative model. While the generated images are not of the highest quality and applying normalization on the image label will further reduce the image quality (He et al., 2022), we argue that for manipulation tasks, the image quality is not critical. What matters is that the images sufficiently capture environmental changes and provide adequate guidance for action prediction.

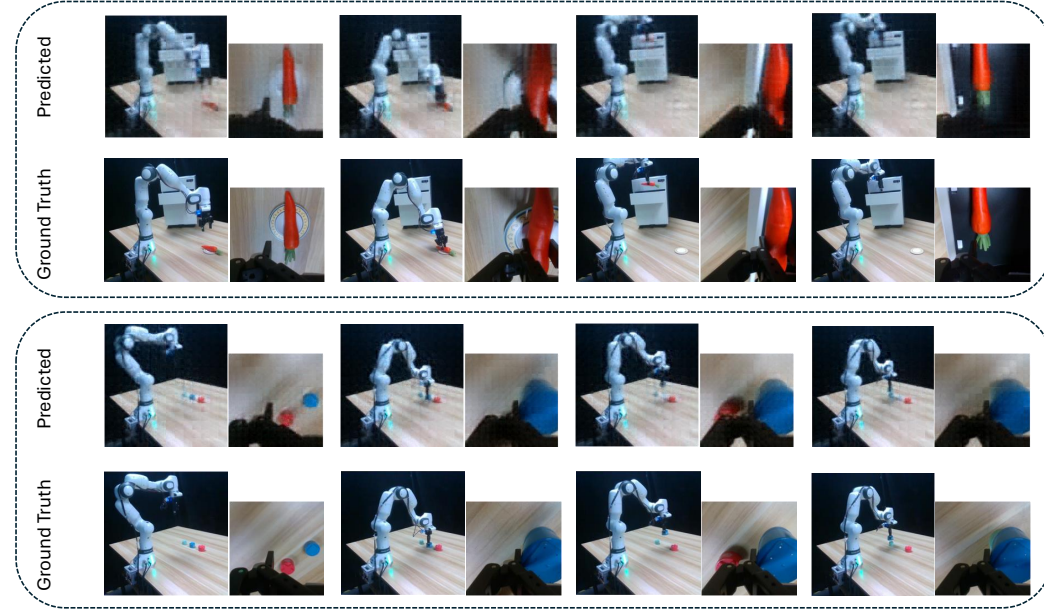


Figure 9: Visualization of predicted images in real world tasks.

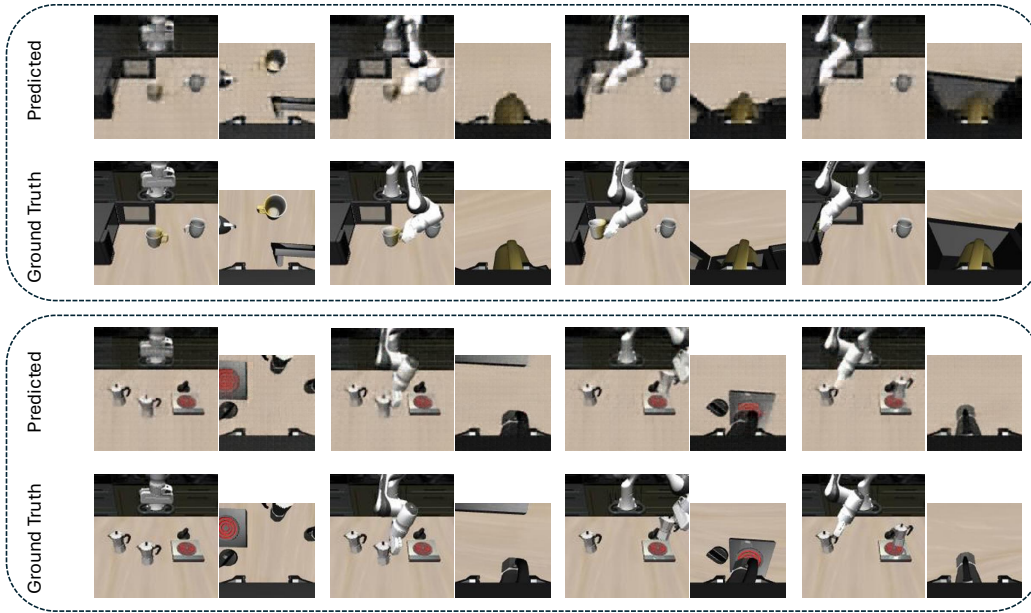


Figure 10: Visualization of predicted images in simulation tasks.