APPENDIX

A. Method Details

1) Annotation: To generate structured reasoning annotations, we employ the video understanding capabilities of Gemini-2.5-Pro [19]. Our process begins with the LeRobot-formatted LIBERO-10[3] and LIBERO-90[4] demonstration datasets. These have been pre-filtered to exclude unsuccessful trajectories and contain agentview image observations corrected via the script provided by the OpenVLA authors.

For each demonstration episode, we compile the agentview images into a 10 fps video. We then prompt Gemini-2.5-Pro with the task instruction, the corresponding video, and the prompt template shown in Figure 7. The model's task is to generate a sequence of textual sub-plans and identify the concluding timestep for each one.

This output is then structured into the following format to serve as the intermediate reasoning signal for our model:

- *Plans*: <all text plans in the task>
- *What has been done*: <completed plans>
- *Now I need to do*: <the next text plan to execute>

Since the end of one sub-plan marks the beginning of the next, we use the generated timesteps to align and supervise the model's reasoning generation at the start of each new action sequence.

Task:
You will be presented with a short video clip of a robot performing certain manipulation tasks on the table. Your task is to reason about the task to give the plans and monitor the task progress along the way.

The **task goal** is to {task_instruction}.

The **plans** should be a list of action steps to complete the task goal where each step is a brief action description (e.g, pick up sth, place it somewhere, close sth, open sth, move sth to somewhere else, etc.) based only on task goal without examining the video.

The **critical moments** should be a list of time (precise to millisecond) and the frame number (10 fps) in the video for completing each plan step. The keys are "time" and "frame" in critical moments. Please identify the exact moment the step finishes and make sure the list of critical moments is the same length as plans. The time should be aa.b format and be precise about b as milliseconds.

The **output** should be a JSON object with the following fields:
plans: ...
critical moment: [...]

Note: Please follow the JSON output format strictly and do not include any other text in your response. Also keep the plans precise and brief.

Now, please look at the following video carefully.

Fig. 7: **Prompt Template for Reasoning Annotations**

2) Training: We follow the recipe in [8] to train our base reasoning VLA model π_0-reason. The hyperparameters used for training are listed in Table I. The training of π_0-reason takes around 20 hours on 8 A100 GPU or 10 hours on 8 H100 GPU. For our baseline π_0, we use the same hyperparameters used for LIBERO finetuning from openpi repository[5]. For learning the Q function in π_0-V-GPS, we use the original code from the paper [16] and select Implicit Q Learning (IQL) [25] algorithm to learn the Q function from offline demonstration dataset LIBERO-100.

[3]https://huggingface.co/datasets/physical-intelligence/libero

[4]https://huggingface.co/datasets/jesbu1/libero_90_lerobot/tree/main

[5]https://github.com/Physical-Intelligence/openpi

TABLE I: **Hyperparameters**. The Hyperparameters we use to train and evaluate π_0-reason.

Hyperparameter	Value
Training Batch Size	128
Validation Batch Size	64
Training Steps	30000
Max Token Length	415
Text Loss Coefficient	0.5
Action Loss Coefficient	0.5
Image Size	$(224, 224, 3)$
Prediction Horizon	16
Action Horizon	5
Image View	Wrist-view and Agent-view

3) Policy Steering:

Hypothesize. In this stage, SEAL generates multiple action sequences in parallel conditioned on the current text plan. The base reasoning model, π_0-reason, uses special tokens (<act> and <think>) to switch between planning and acting. We leverage this by using the <think> token to signal the completion of an action sequence. Specifically, we generate a batch of $K = 10$ action sequences concurrently, using KV cache to accelerate inference. As soon as any sequence in the batch generates a <think> token, it is asynchronously passed to the VLM verifier. While verification runs as a background process, π_0-reason continues to generate subsequent action tokens for the remaining sequences in the batch. This parallel process terminates as soon as the VLM confirms any one of the proposed sequences as successful.

Predict. We maintain $K = 10$ vectorized environments to simulate the physical outcomes of the hypothesized actions. Each of the K action sequences generated during the *Hypothesize* stage is executed in its own dedicated simulation environment. Following the selection of a successful sequence by the *Verify* stage, we synchronize all parallel environments by setting their states to match the final state of the successful simulation. This ensures that the system proceeds from a single, validated world state before planning the next sub-task.

Verify. We use **GPT-4o** as a Vision-Language Model (VLM) verifier to approximate a ground-truth reward function. To provide the necessary context, we supply the verifier with the initial agentview image of the task (to establish initial object locations) along with the final agentview and wrist-view images from the completed action sequence. These images are combined with the corresponding textual sub-plan using the prompt template shown in Figure 8.

We structure the prompt to elicit step-by-step reasoning, guiding the VLM to analyze whether the final state shown in the images matches the objective of the sub-plan. Empirically, we find this structured prompting significantly improves verification accuracy compared to direct queries. However, we note two primary failure modes: the VLM

TABLE II: **In-Distribution Performance on LIBERO-10**. Success rates (%) are averaged over 50 trials for each method, conditioned on the training dataset. Our method, **SEAL**, generally outperforms baselines across all training data configurations. For each training dataset and each task, the highest success rate is marked as bold.

Task ID	Success Rate (%)											
	Training Dataset (LIBERO-10)				Training Dataset (LIBERO-100-Basket)				Training Dataset (LIBERO-100)			
	π_0	π_0-V-GPS	π_0-reason	**SEAL**	π_0	π_0-V-GPS	π_0-reason	**SEAL**	π_0	π_0-V-GPS	π_0-reason	**SEAL**
0	**96**	92	90	**96**	88	**96**	90	**96**	92	88	90	**100**
1	**100**	98	98	**100**	**100**	**100**	**100**	**100**	94	92	96	**100**
2	88	94	98	**100**	94	92	80	**98**	**90**	88	62	84
3	**100**	**100**	**100**	**100**	96	**98**	96	**98**	86	96	98	**100**
4	76	74	96	**98**	72	78	84	**88**	80	**98**	**98**	**98**
5	92	**100**	98	**100**	96	98	98	**100**	92	**98**	**98**	**98**
6	74	80	84	**98**	78	80	80	**96**	82	94	96	**98**
7	96	**100**	90	98	88	**92**	**92**	**92**	90	90	98	**100**
8	66	64	66	**88**	44	60	50	**70**	54	56	80	**90**
9	96	94	98	**100**	90	94	92	**100**	86	88	96	**100**
Average	85	90	92	**96**	85	89	86	**94**	85	87	89	**97**

Given the observation images of a sampled action plan (the first image is the initial state, the second image is the final state from the agentview and the last image is the the final state from wrist view, The gripper is at the bottom of the wrist view image), you are an external verifier. Help the robot agent describe the action and determine if it achieves the specified subtask.

Subtask: {TASK_INSTRUCTION}.

Your verification must follow this logical process:
Identify the target object in the first image. The target object is present in the first image. If you don't see it, look again for the target object because it is occluded behind other items. Pay attention to the shape (e.g. can or box). Try your best to find the most relevant object in the first image if you think there is no exact match as the target object. Describe where it is located.

If the subtask is about picking, analyze both the wrist view and the agent view of the final state to check:
a) For objects not a pot or a bowl, Is the robot's gripper holding the target object with right shape (box vs can) ? Is the object staying secure between the gripper? For a pot or a bowl, is the gripper holding any part of the pot or the bowl (check if any part is between the gripper in the wrist view, not considering security as long as it is in contact)?

If the subtask is about placing, analyze the wrist view and the agent view of the final state to check:
b) Is the object in/on the target place? If object is not visible, compare the first and second image to see whether target object disappears from the table. If the object is a mug, the final pose should not be tilted in the agentview. If the object is a pot, it should be in contact with the red region in the agentview.

If the subtask is about closing, analyze the wrist view and the agent view of the final state to check:
c) Is the object fully closed?

Determine the outcome based on this intermediate thinking.
Finally, provide your verification in the format "Verification: Success" or "Verification: Failure".

Fig. 8: **Prompt Template for VLM Verification**

TABLE III: **Task Instructions for LIBERO-10**. The ten long-horizon tasks and their corresponding language instructions.

Task ID	Task Instruction
0	put both the alphabet soup and the tomato sauce in the basket.
1	put both the cream cheese box and the butter in the basket.
2	turn on the stove and put the moka pot on it.
3	put the black bowl in the bottom drawer of the cabinet and close it.
4	put the white mug on the left plate and put the yellow and white mug on the right plate.
5	pick up the book and place it in the back compartment of the caddy.
6	put the white mug on the plate and put the chocolate pudding to the right of the plate.
7	put both the alphabet soup and the cream cheese box in the basket.
8	put both moka pots on the stove.
9	put the yellow and white mug in the microwave and close it.

sometimes struggles to correctly assess gripper-object contact, likely because the robot's `wrist-view` images are out-of-distribution for GPT-4o's training data, and its accuracy decreases in scenes with significant object occlusion.

4) Runtime Analysis: To analyze the computational cost and scalability of our method, we measure the average inference time per step while varying the number of parallel samples (K). Fig. 6 presents this runtime scaling, breaking down the total time into its two primary components: Batch Action Sampling and VLM Verification

As shown in the figure, the total inference time increases with the number of samples. The runtime grows from 147 ms for a single sample to 347 ms for $K = 10$ samples.

- *Batch Action Sampling*: The cost of Batch Action Sampling scales efficiently and sub-linearly. Its duration increases from 86 ms at $K = 1$ to 184 ms at $K = 10$. This demonstrates the effectiveness of our batching implementation, where a tenfold increase in samples results in only a ˜2.1x increase in sampling time.
- *VLM Verification*: In contrast, the time spent on VLM Verification scales more sharply, especially at higher sample counts. This component's average cost per step grows from 61 ms for one sample to 163 ms for ten samples. Although the VLM verification runs in the background, each query still takes 7-10 seconds for GPT-4o to generate the answer because of the thinking mode.

The results indicate a trade-off between potential performance gains from more samples and the resulting inference latency. While our system supports parallel sampling efficiently, the VLM verification step becomes a more significant bottleneck as the number of samples increases.

B. Benchmark Details

1) In-distribution Tasks: We evaluate on ten long-horizon manipulation tasks from the LIBERO-10 benchmark [12] as our in-distribution test set. These tasks are well-suited for Vision Language Action Models (VLAs) with intermediate reasoning capabilities due to their long-horizon nature. The instructions for each task are listed in Table III.

We evaluate all methods on these tasks after training them on three different datasets. The per-task success rates are detailed in Table II, and the average performance is reported in Figure 2a of the main paper.

2) Out-of-Distribution (OOD) Tasks: Following [22], we extend the LIBERO-10 benchmark with four out-of-distribution (OOD) variations, categorized as semantic or

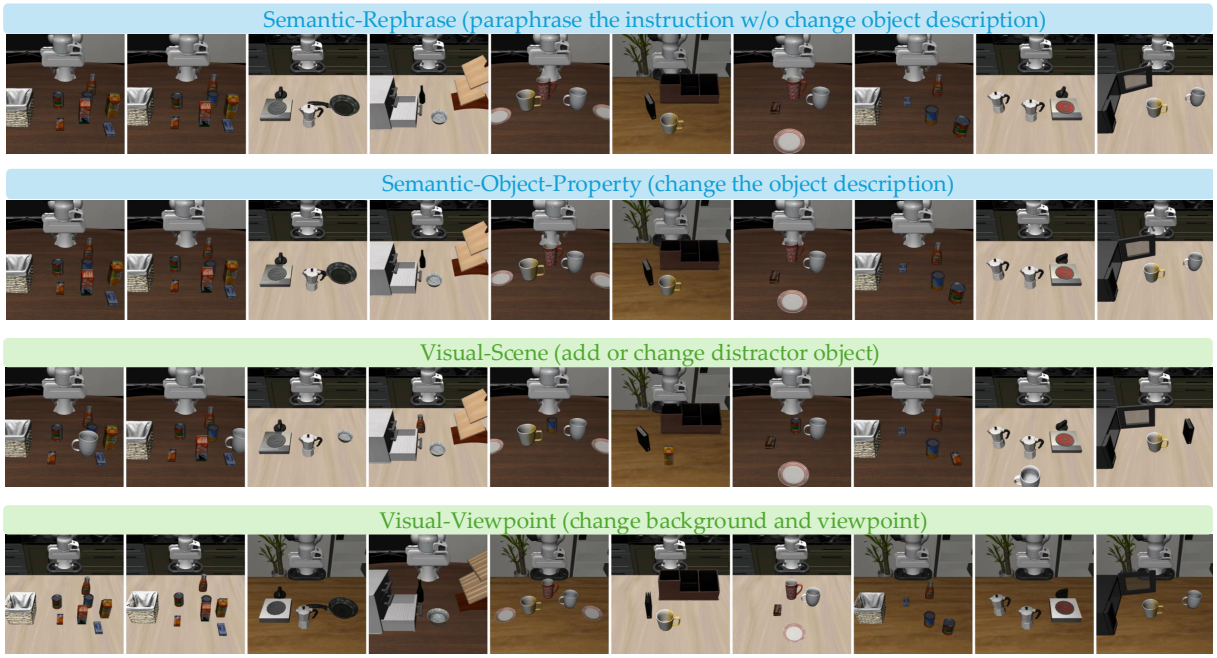

Fig. 9: **Visualizations of Initial States for OOD Tasks**. The top row shows the semantic OOD shift of Lang-Rephrase from LIBERO-10. The second row corresponds to semantic shifts with Lang-Object-Property. These two rows are the same as LIBERO-10 since they have no visual changes. The third row shows the *Visual-Scene* shift with object substitutions or additions. The bottom row shows the *Visual-Viewpoint* shift with a new background and camera angle.

visual shifts.

- **Semantic OOD**: The task instruction is altered while the scene remains the same.
  - *Lang-Rephrase*: The instruction is paraphrased using different verbs and sentence structures, but the object descriptions remain the same. The new instructions are in Table IV.
  - *Lang-Object-Property*: The object descriptions are replaced with different but equivalent references (e.g., "yellow and white mug" becomes "middle mug"). The new instructions are in Table V.
- **Visual OOD**: The visual properties of the scene are changed while the instruction remains the same.
  - *Visual-Scene*: A non-target object is replaced, or a distractor object is added to the scene (e.g., a wine bottle is replaced with a ketchup bottle). See the third row of Figure 9 for visualizations.
  - *Visual-Viewpoint*: The background and camera pose are changed. We randomly select one of two alternate backgrounds available in LIBERO (Kitchen, Living Room, Study Room) that was not used in the original task. The camera pose is adjusted to keep all objects in view. See the fourth row of Figure 9 for visualizations.

We evaluate all methods trained on the LIBERO-100 dataset against these four OOD settings, as models trained on larger, more diverse data are expected to be more robust and LIBERO-100 is our largest dataset. Per-task results are presented in Table VI, with average performance summarized in Figure 4.

TABLE IV: **Task Instructions for Semantic OOD (Lang-Rephrase)**. The original instructions are paraphrased, altering sentence structure and verbs while keeping object descriptions intact.

Task ID	Task Instruction
0	place the alphabet soup in the basket and the tomato sauce as well.
1	pick up both the cream cheese box and the butter and place them in the basket.
2	switch the stove on and set the moka pot on the stove.
3	move the black bowl to the bottom drawer and close the drawer of the cabinet.
4	place two mugs on the plates, left one is the white mug and the right one is the yellow and white mug.
5	grab the standing book and transfer it to the back compartment of the caddy.
6	set the white mug on the plate with chocolate pudding to be placed to the right.
7	put both the can of soup and the box of cheese in the basket.
8	transfer both moka pots from the table to the stove.
9	place the yellow and white mug inside the microwave, then shut the door.

TABLE V: **Task Instructions for Semantic OOD (Lang-Object-Property)**. The way objects are described is changed, while the core task and sentence structure remain similar.

Task ID	Task Instruction
0	put both the can of soup and can of sauce in the basket.
1	put both the box of cheese and the box of butter in the basket.
2	turn on the cooktop and place the moka machine on it.
3	put the middle bowl to the lowest drawer and close it.
4	put the pure white cup on the left plate and put the other one with the yellow handle on the right plate.
5	pick up the right book and put it in the rear part of the caddy.
6	put the pure white cup on the middle plate and put the brown chocolate to the right of the plate.
7	move the two objects, alphabet soup and cream cheese box, to the basket.
8	put both the moka coffee makers on the cooktop.
9	put the middle mug inside the microwave and close the door.

3) Behavior Composition Tasks: To further test generalization, we introduce a suite of novel *behavior composition* tasks. These tasks require reusing skills learned from training demonstrations in combinations not seen during training. For instance, if a model has seen demonstrations for "put both

TABLE VI: **OOD Performance on LIBERO-10 Variants**. Success rates (%) are averaged over 50 trials for each method trained on the LIBERO-100 dataset. Our method, **SEAL**, demonstrates stronger robustness to both semantic and visual OOD shifts. For each OOD shift and each task, the highest success rate is marked as bold.

Task ID	Success Rate (%)															
	Semantic OOD (Lang-Rephrase)				Semantic OOD (Lang-Object-Property)				Visual OOD (Visual-Scene)				Visual OOD (Visual-Viewpoint)			
	π_0	π_0-V-GPS	π_0-reason	SEAL	π_0	π_0-V-GPS	π_0-reason	SEAL	π_0	π_0-V-GPS	π_0-reason	SEAL	π_0	π_0-V-GPS	π_0-reason	SEAL
0	86	92	92	**98**	72	80	90	**98**	**96**	88	**96**	**96**	52	30	12	**68**
1	92	96	98	**100**	96	96	98	**100**	**100**	98	**100**	**100**	34	36	72	**84**
2	32	32	82	**96**	66	70	54	**72**	**94**	92	64	90	4	0	8	**10**
3	86	88	96	**98**	90	88	94	**98**	94	96	**100**	**100**	**72**	66	2	32
4	32	14	78	**84**	70	74	84	**90**	90	70	96	**98**	0	8	24	**40**
5	96	94	**100**	**100**	18	**100**	76	96	92	96	98	**100**	30	26	58	**94**
6	86	88	76	**96**	86	**88**	76	84	88	**98**	96	96	34	32	30	**44**
7	86	**92**	74	82	90	90	94	**98**	88	94	94	**100**	20	14	24	**48**
8	62	44	78	**98**	58	54	68	**84**	4	4	68	**98**	10	6	12	**20**
9	74	74	88	**98**	78	70	80	**92**	92	92	96	**100**	**27**	12	8	22
Average	73	71	86	**95**	72	81	81	**91**	84	83	91	**98**	28	23	25	**45**

TABLE VII: **Compositional Generalization Performance**. Success rates (%) are averaged over 50 trials. As the training dataset size and diversity increase from LIBERO-10 to LIBERO-100, our method's performance advantage over baselines becomes more pronounced. Entries marked with '—' are not applicable for that training dataset. For each training dataset and each task, the highest success rate is marked as bold.

Task ID	Success Rate (%)											
	Training Dataset (LIBERO-10)				Training Dataset (LIBERO-100-Basket)				Training Dataset (LIBERO-100)			
	π_0	π_0-V-GPS	π_0-reason	SEAL	π_0	π_0-V-GPS	π_0-reason	SEAL	π_0	π_0-V-GPS	π_0-reason	SEAL
0	28	32	**34**	32	4	8	6	**10**	26	46	30	**82**
1	0	**2**	**2**	0	0	0	0	**4**	20	6	42	**60**
2	—	—	—	—	22	28	**74**	70	22	18	76	**88**
3	—	—	—	—	2	2	18	**28**	14	**18**	2	4
4	—	—	—	—	0	2	6	**12**	0	2	46	**60**
5	—	—	—	—	0	0	**6**	4	6	2	48	**72**
6	—	—	—	—	2	2	**14**	10	24	**34**	0	4
7	—	—	—	—	0	2	4	**6**	30	24	50	**74**
8	—	—	—	—	68	70	82	**90**	54	34	94	**96**
9	—	—	—	—	—	—	—	—	10	12	14	**20**
10	—	—	—	—	—	—	—	—	2	2	14	**38**
11	—	—	—	—	—	—	—	—	2	0	4	**14**
12	—	—	—	—	—	—	—	—	0	4	**80**	78
Average	14	17	**18**	16	11	13	23	**26**	16	16	38	**53**

the alphabet soup and the tomato sauce in the basket" and "put the orange juice in the basket," a compositional task might be "put both the orange juice and the tomato sauce in the basket."

The set of compositional tasks depends on the training dataset. For models trained on *LIBERO-10* dataset, we create two new tasks by recombining objects to put in the basket (Tasks 0-1 in Table VIII). For *LIBERO-100-Basket*, which includes more pick-and-place examples for basket , we use a larger set of nine tasks (Tasks 0-8). For the most diverse *LIBERO-100* dataset, we create the full suite of 13 tasks including more objects like mugs and wine bottle. The complete list of tasks is provided in Table VIII.

We evaluate all methods on their corresponding compositional task suite. Per-task results are in Table VII, and average performance is reported in Figure 2a of the main paper.

TABLE VIII: **Task Instructions for Behavior Composition Tasks**. These new tasks require combinations of skills that were not observed during training.

Task ID	Task Instruction
0	put both the cream cheese and the tomato sauce in the basket.
1	put both the alphabet soup and the butter in the basket.
2	put both the orange juice and the tomato sauce in the basket.
3	put both the milk and the tomato sauce in the basket.
4	put both the orange juice and the butter in the basket.
5	put both the tomato sauce and the butter in the basket.
6	put both the milk and the butter in the basket.
7	put both the ketchup and the cream cheese in the basket.
8	put both the tomato sauce and the cream cheese box in the basket.
9	put the wine bottle in the bottom drawer of the cabinet and close it.
10	put the red mug on the plate and put the chocolate pudding to the right of the plate.
11	put the red mug on the plate and put the chocolate pudding to the left of the plate.
12	put the red mug on the left plate and put the yellow and white mug on the right plate.