WorldAgen: Unified State-Action Prediction with Test-Time World Model Training

Chi Wan[1], Kangrui Wang[1,*], Yuan Si[1], Pingyue Zhang[1], Manling Li[1]

[1]**Northwestern University** [*]**Project Lead**

How can vision-language-action (VLA) models adapt to new environments where world dynamics shift? While recent research has combined world modeling and action prediction to improve VLA performance, existing methods largely rely on pretraining on static datasets and lack mechanisms for active adaptation at deployment time. As a result, these models often struggle to generalize to unseen environments with novel object configurations and dynamics. We present **WorldAgen**, a unified framework that jointly learns world modeling and action prediction while enabling **Test-Time Training (TTT)** for adaptation to new environments. WorldAgen employs a shared Transformer backbone with two heads: (1) a **world model head** that predicts future states from past state-action trajectories, and (2) a **policy head** that predicts actions conditioned on task instructions and current states. We design a **Mixed Unidirectional Attention Mask** to disentangle the learning dynamics of these two heads within a single architecture. At test time, WorldAgen samples exploratory actions, collects ground-truth state transitions, and performs lightweight TTT updates to refine its world model. This online adaptation enhances the model's understanding of the environment and in turn leads to more accurate action predictions. Experiments on the CALVIN and LIBERO benchmarks demonstrate that our base model achieves strong performance. More importantly, applying TTT to the world model head during inference yields additional performance gains, confirming the effectiveness of online world-model adaptation in novel environments.

Website: https://worldagen.github.io

Code: https://github.com/mll-lab-nu/WorldAgen

Models: https://huggingface.co/MLL-Lab/worldagen

1. Introduction

Vision-language-action (VLA) models have emerged as a powerful and popular paradigm for robotic manipulation (Ma et al., 2025; Din et al., 2025), enabling agents to follow natural language instructions and act directly from raw visual observations. Recent work has explored joint state-action prediction techniques (Tian et al., 2024; Guo et al., 2024; Liu et al., 2025; Xian and Gkanatsios, 2023; Wolf et al., 2025), allowing models not only to predict the next actions but also to anticipate how their actions will transform the environment. This joint formulation

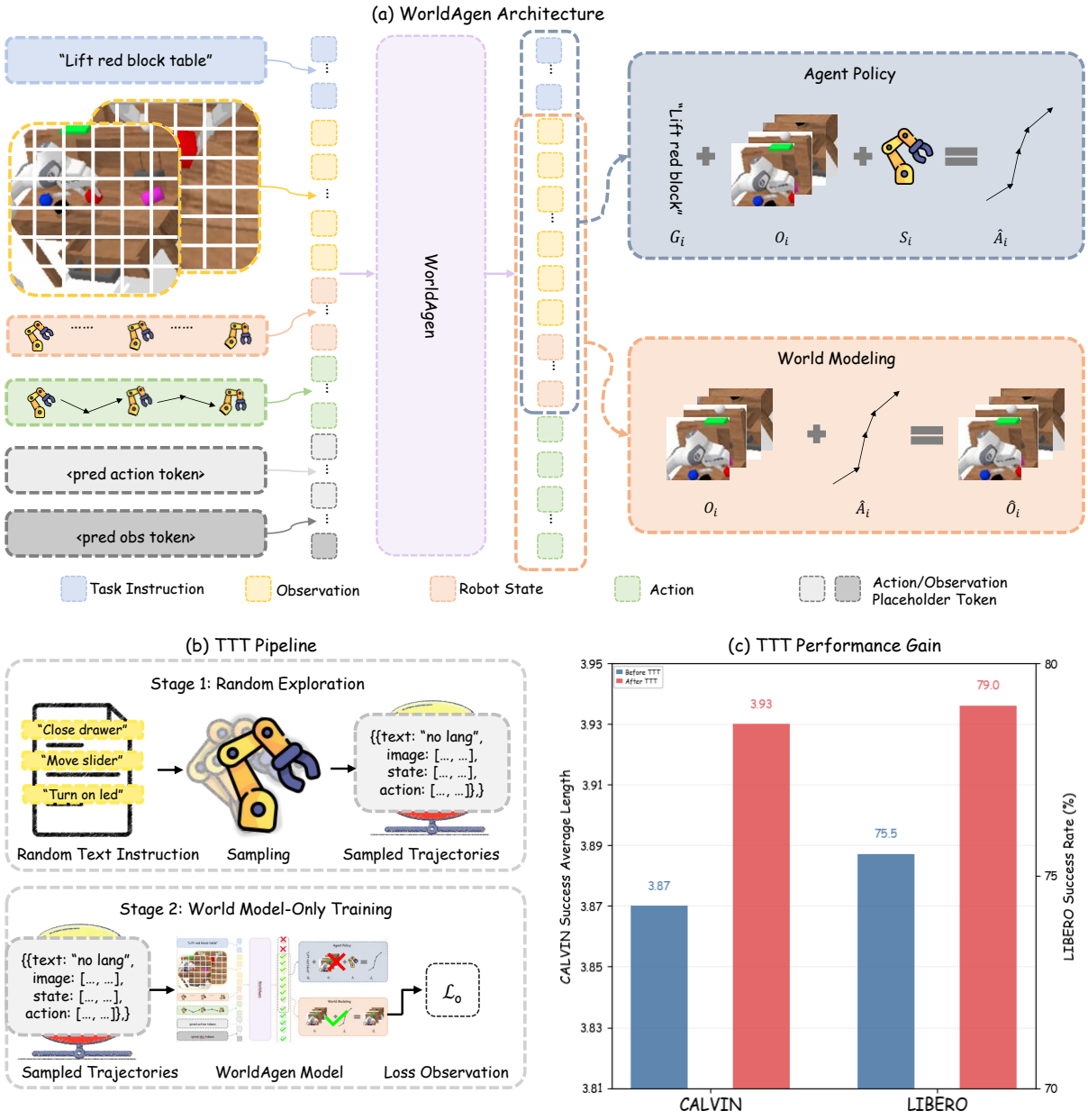

Figure 1 | Overview of WorldAgen. (a) WorldAgen Architecture. WorldAgen unifies an agent policy head for task-conditioned action prediction and a world modeling head for task-agnostic state prediction within a single Transformer backbone. At trajectory unit i, the agent receives the task instruction G_i, the observation chunk O_i, and the robot state chunk S_i, the action placeholder A_p and observation placeholder O_p. Then it predicts the action chunk $\hat{A}_i$. After that, the world model predicts the future observation chunk $\hat{O}_i$ to refine the shared representation of environment dynamics. (b) Two-stage TTT pipeline. Stage 1 performs random exploration in the new environment to collect unlabeled trajectories. Stage 2 adapts the world model using LoRA-based fine-tuning on the observation loss $\mathcal{L}_o$, improving environment modeling without modifying the agent policy. (c) TTT Performance We tested WorldAgen with TTT strategy on CALVIN and LIBERO benchmark, which show great performance gain.

has improved data efficiency and scene understanding, leading to stronger performance across manipulation benchmarks.

However, we argue that current methods remain fundamentally limited by training on static datasets (Kim, Finn, and Liang, 2025; Li et al., 2025; Kim et al., 2024). Once pretrained, these models lack mechanisms to adapt their internal representations of world dynamics when deployed in novel environments. In realistic settings, distribution shifts such as new object layouts, lighting conditions, or physical properties are inevitable, and static world modeling fails to capture these variations (Wu et al., 2023b; Lanier et al., 2025). This leads to a key question: **How can we enable VLA models to actively adapt their understanding of the environment during test time?**

As shown in Figure 1, we address this question with WorldAgen, a unified framework that combines joint state-action prediction with a novel TTT strategy:

- **Joint State-Action Modeling.** WorldAgen uses a shared Transformer backbone with two heads. The world model head predicts future states from historical state-action trajectories, while the policy head predicts actions conditioned on task instructions. By training these tasks jointly, WorldAgen aligns scene understanding with action prediction.
- **Test-Time Training for Scene Adaptation.** During test time, WorldAgen executes exploratory actions to collect ground-truth state transitions. It then performs lightweight, LoRA-based TTT updates to refine its world model. These updates improve the internal representation of the environment, indirectly boosting the agent's ability to generate effective actions in novel scenarios, which is depicted in.

Our work reframes world modeling from a passive pretraining objective into an active test time adaptation mechanism, bridging the gap between offline training and real-world generalization.

In sum, we make the following contributions:

- **Unified joint state-action prediction architecture.** We present a unified Transformer-based structure that integrates world modeling and action prediction, enabling shared representations and work separately via a Mixed Unidirectional Attention Mask.
- **Active test-time adaptation for VLA models.** We introduce a TTT paradigm that transforms world modeling from a static pretraining into an active, test time adaptation mechanism, allowing VLA models to refine their scene understanding on the fly.
- **Empirical validation across challenging benchmarks.** Our baseline achieves performance comparable to, or even better than, state-of-the-art methods. Furthermore, by fine-tuning the world modeling component with only a small number of samples at test time, our method attains state-of-the-art results on both the CALVIN and LIBERO benchmark.

We believe these results demonstrate that continuous adaptation during test time, rather than merely scaling model size or pretraining data, is a key ingredient for robust and generalizable robotic manipulation.

2. Method

2.1. Task Formulation

We frame our task as a Partially Observable Markov Decision Process (POMDP). Each rollout trajectory is divided into a sequence of trajectory units indexed by i. A trajectory unit is defined as $U_i = (G_i, O_i, S_i, A_i, A_p, O_p)$, where G_i denotes the task instruction, and O_i, S_i, and A_i denote the observation, robot state, and action chunks, respectively, accompanied by an action placeholder A_p and an observation placeholder O_p.

Each chunk (i.e., O_i, S_i, and A_i) consists of a sequence of time-indexed elements, including observations, states, or actions extracted from the raw trajectory τ. Each element (i.e., o_t, s_t, a_t) is further tokenized into a sequence of tokens that serves as the actual model input.

In unit i, the model first predicts an action chunk $\hat{A}_i$ conditioned on the instruction G_i, the observation chunk O_i, and the state chunk S_i. It then predicts the next observation chunk conditioned on (O_i, S_i, A_i) during training or $(O_i, S_i, \hat{A}_i)$ during inference[1]. Executing the predicted actions advances the environment and produces the next unit U_{i+1}. A summary of the key notations used in the WorldAgen framework is provided in Table 12 in Appendix.

2.2. Joint State–Action Prediction

WorldAgen contains two predictive components: a task-conditioned agent model and a task-agnostic world model. Both models share a single Transformer backbone, enabling a unified representation across tasks, as illustrated in Figure 1(a).

We maintain two types of histories:

Agent history:

$$h_i^a = (G_i, O_x, S_x, \hat{A}_x)_{x=i-k}^{i-1},$$

which contains the task instruction G_i along with the observation, robot state, and predicted action chunks from the previous k segments. This history is used by the agent model for task-aware action prediction.

World history:

$$h_i^w = (O_x, S_x, \hat{A}_x)_{x=i-k}^{i-1},$$

which includes the observation, robot state, and predicted action chunks from the previous k segments, and is used by the world model to predict task-agnostic environment dynamics.

We define the two models at trajectory segment U_i as follows:

Agent Model. The agent model predicts the next action chunk $\hat{A}_i$ conditioned on the current observation chunk O_i, robot state chunk S_i, task instruction G_i, and agent history h_i^a:

$$p_a\big(\hat{A}_i \,\big|\, G_i, O_i, S_i, h_i^a\big)\,.$$

World Model. The world model predicts the next observation chunk $\hat{O}_i$ conditioned on the current observation chunk O_i, robot state chunk S_i, predicted action chunk $\hat{A}_i$, and world history h_i^w:

$$p_w\big(\hat{O}_i \,\big|\, O_i, S_i, \hat{A}_i, h_i^w\big)\,.$$

[1] A history buffer h is also included in the conditioning; detailed formulation is provided in a later section.

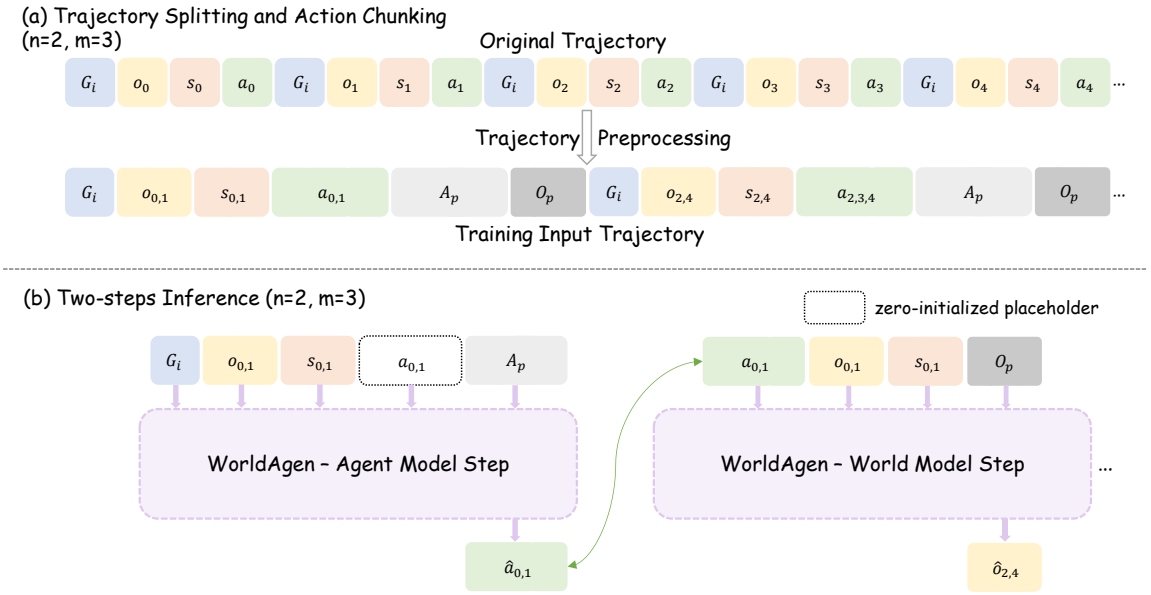

Figure 2 | **Trajectory Splitting, Chunking, and Two-Step Inference.** Example with observation chunk length $n = 2$ and action chunk length $m = 3$. (a) *Trajectory Splitting and Chunking:* a raw trajectory is divided into trajectory units U_i, where each unit contains an observation chunk O_i and a robot state chunk S_i obtained via uniform subsampling from m consecutive observations and states, and an action chunk A_i obtained by grouping m actions. (b) *Two-Step Inference:* the agent model first initializes a zero action placeholder A_p and then replaces it with the predicted action chunk $\hat{A}_i$. The world model subsequently predicts the next observation chunk $\hat{O}_i$, conditioned on the updated trajectory unit.

During training, we adopt teacher forcing and replace all predicted actions $\hat{A}_i$ with the ground-truth actions A_i.

2.3. Trajectory Splitting and Action Chunking

Predicting multiple actions at once has been shown to improve the execution efficiency of real-world robots Intelligence et al. (2025). However, consecutive observation frames often contain highly redundant visual information, and predicting long observation sequences slows down both training and deployment. To balance these factors, WorldAgen supports different action chunk lengths m and observations chunk length n. We further propose a robust trajectory splitting and action chunking scheme that flexibly adapts to different experimental configurations.

Concretely, given a rollout $\tau = \{(o_t, s_t, a_t)\}_{t=0}^{T-1}$ with T timesteps, we transform it into a sequence of trajectory units $\{U_i\}_{i=0}^{I-1}$ with I units. The initial unit U_0 aggregates the first n timesteps of the rollout and is paired with the first action chunk $A_0 = A_{0\to n-1} = [a_0, a_1, \ldots, a_{n-1}]$. For units U_i with $i > 0$, we uniformly sample n timesteps to construct the current observation

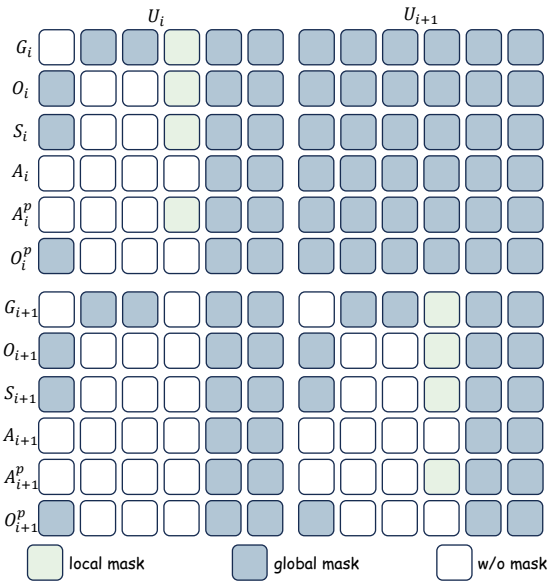

Figure 3 | Proposed Mixed Unidirectional Attention Mask: combines local and global masking. For each trajectory unit, WorldAgen applies a local mask that prevents the action chunk A_i from attending to the corresponding action placeholder A_p. A global mask is further used to construct a task-agnostic world model head, so that the observation placeholder O_p will not attend to the task instruction G_i. General structure conforms to strictly causal prediction.

chunk O_i, state chunk S_i, and associated action chunk A_i:

$$
\begin{aligned}
O_i &= O^n_{n+(i-1)m \to n+im-1} \\
&= \left[o_{n+im},\, o_{n+im+\lfloor \frac{1}{n}m \rfloor},\, \ldots,\, o_{n+im+\lfloor \frac{n-1}{n}m \rfloor}\right], \\
S_i &= S^n_{n+(i-1)m \to n+im-1} \\
&= \left[s_{n+(i-1)m},\, s_{n+(i-1)m+\left\lfloor \frac{1}{n}m \right\rfloor},\right. \\
&\quad \left. \ldots,\, s_{n+(i-1)m+\left\lfloor \frac{n-1}{n}m \right\rfloor}\right], \\
A_i &= A_{n+(i-1)m \to n+im-1} \\
&= \left[a_{n+(i-1)m},\, a_{n+(i-1)m+1},\, \ldots,\, a_{n+im-1}\right].
\end{aligned}
$$

This procedure keeps each observation chunk fixed to length n, reducing redundant visual information and improving runtime efficiency.

Figure 2 (a) illustrates an example with $n = 2$ and $m = 3$. Algorithm of trajectory splitting and action chunking for variable length of action, observation and robot state chunk is shown in Algorithm 1 in Appendix , where UNIFORMSUBSAMPLE(a, b, n) uniformly samples n distinct indices from the integer interval $\{a, \ldots, b\}$ without replacement and returns them in ascending order, which we then use to construct the observation and state chunks.

2.4. Mixed Unidirectional Attention Mask

To ensure that the world modeling and agent policy tasks are trained jointly without information leakage, we introduce a mixed unidirectional attention masking mechanism as shown in Figure 3. This mechanism integrates two complementary components:

Local mask: ensures that the current observation chunk O_i, the robot proprioception chunk S_i, the task instruction G_i, and the action placeholder A_p cannot access the action chunk A_i. This prevents information leakage within a single timestep.

Global mask: guarantees that the world model head remains invisible to the task instruction G_i across all time steps. Additionally, the global mask enforces mutual invisibility among the task instruction and observations to avoid cross-modal leakage.

Meanwhile, the general structure conforms to strictly causal prediction, such that both the world model and policy heads cannot access any future tokens.

2.5. Pretraining

We pretrain WorldAgen using large-scale trajectory data collected from robot demonstrations. We split trajectory using our proposed robust splitting method. During pretraining, we jointly optimize the agent model and world model using teacher forcing:

$$\mathcal{L} = \mathcal{L}_a + \lambda \mathcal{L}_o,$$

where $\mathcal{L}_a$ is the cross-entropy loss for predicting future action chunks, $\mathcal{L}_o$ is the reconstruction loss for predicting future observation chunks, and λ is a weighting factor balancing the two objectives.

It is worth noting that during training, we feed the ground-truth action chunk (teacher forcing), whereas during testing, the model performs autoregressive prediction by feeding the previously predicted action chunk at each timestep.

2.6. Two-Step Inference

At test time, WorldAgen runs in a two-step inference loop over trajectory units as shown in Figure 2 (b). For unit i, we first perform the agent model step. We construct the input sequence with a zero-initialized action placeholder A_p and feed it to the agent model, which predicts the current action chunk $\hat{A}_i$. The predicted actions then overwrite the placeholder A_p.

Next, we perform the world model step. Conditioned on the task instruction G_i, the current observation chunk O_i and the updated action chunk A_i (same as $\hat{A}_i$), the world model predicts the next observation chunk $\hat{O}_i$. During testing, these two steps are applied alternately to roll out the whole trajectory .

2.7. Test-Time Training

While joint pretraining improves world understanding, distribution shifts in novel environments can still degrade performance. To address this, we introduce a two-stage Test-Time Training strategy that adapts the world model online, refining its understanding of environment dynamics and indirectly improving downstream action prediction. The whole pipeline is given in Figure 1 (b).

Stage 1: Random Exploration and Data Collection. At deployment, the agent first performs free exploratory rollouts in the target environment. During this phase, it records trajectories of (O_i, S_i, A_i). To ensure that adaptation focuses on environment dynamics rather than task-specific

Method	T1	T2	T3	T4	T5	Avg. Len. ↑
RoboFlamingo	82.4	61.9	46.6	33.1	23.5	2.47
SuSIE	87.0	69.0	49.0	38.0	26.0	2.69
GR-1	85.4	71.2	59.6	49.7	40.1	3.06
3D Diffusor Actor	92.2	78.7	63.9	51.2	41.2	3.27
CLOVER	96.0	83.5	70.8	57.5	45.4	3.53
Seer	93.0	82.4	72.3	62.6	53.3	3.64
Seer-Large	92.7	84.6	76.1	68.9	60.3	3.83
WorldAgen	96.3	87.7	76.8	67.3	59.1	3.87
WorldAgen-TTT	**96.6**	**88.5**	**78.5**	**68.7**	**60.5**	**3.93**

Table 1 | Performance comparison on the CALVIN benchmark. We report the success rate (%) for completing 5 consecutive tasks and the average sequence length (ASL).

instructions, all collected trajectories are relabeled with a generic "no-lang" token in place of language instructions.

Stage 2: World Model Adaptation. Using the collected trajectories, we adapt only the shared backbone with LoRA-based parameter-efficient fine-tuning, while keeping the policy head frozen. The update rule is:

$$\theta'_w \leftarrow \theta_w - \eta \nabla_{\theta_w} \mathcal{L}_o,$$

where θ_w are the parameters of the world model head and η is the learning rate.

This targeted update improves world understanding ability and enhances the shared representations , which are in turn leveraged by the agent model for action prediction.

By decoupling adaptation from the task goal and restricting it to the world model, **our TTT procedure enables robust scene adaptation without requiring additional task-specific annotations, paving the way for scalable test-time adaptation in VLA models.**

3. Experiments

3.1. Datasets

CALVIN. CALVIN (Mees et al., 2022) is an open-source simulated benchmark designed for learning long-horizon language-conditioned robot manipulation tasks. The benchmark requires agents to solve complex manipulation tasks by understanding a series of unconstrained language instructions in sequence.

LIBERO. LIBERO (Liu et al., 2023a) is a comprehensive benchmark for lifelong learning in robot manipulation that emphasizes knowledge transfer across diverse tasks.

3.2. Implementation Details

Test-Time Training TTT represents a crucial component of WorldAgen that enables adaptive performance improvements during inference. In the TTT phase, we apply LoRA fine-tuning to

Method	Avg. Success ↑	Put soup and box in basket	Put box and butter in basket	Turn on stove and put pot	Put bowl in drawer and close it
MT-ACT	41.0	30.0	50.0	75.0	85.0
MVP	68.2	83.3	**90.0**	80.0	88.3
MPL	77.3	66.6	86.6	96.6	95.0
OpenVLA	54.0	35.0	95.0	65.0	45.0
Seer	78.7	80.0	**90.0**	91.7	81.7
WorldAgen	75.5	70.0	75.0	95.0	100
WorldAgen-TTT	**79.0**	**85.0**	75.0	**95.0**	**100**

Put mugs on left and right plates	Pick book and place it in back	Put mug on plate and put pudding to right	Put soup and sauce in basket	Put both pots on stove	Put mug in microwave and close it
20.0	75.0	0.0	0.0	10.0	65.0
46.7	63.3	45.0	78.3	**60.0**	46.7
83.3	83.3	56.6	86.6	40.0	78.3
40.0	80.0	60.0	45.0	20.0	55.0
85.0	65.0	**86.7**	88.3	51.7	**66.7**
85.0	90.0	60.0	100	45.0	35.0
90.0	**90.0**	50.0	**100**	45.0	**60.0**

Table 2 | Performance comparison on LIBERO benchmark. We report the success rate (%) across different manipulation tasks.

the backbone of Qwen3. Specifically, we apply LoRA to the attention projection layers (q_proj, k_proj, v_proj, o_proj) and the MLP projection layers (gate_proj, up_proj, down_proj) of Qwen3.

The test-time training procedure, including sampling and LoRA adaptation, can be executed on an RTX 4090 GPU and require approximately 8 minutes per task for CALVIN and 2 minutes per scene. More details can be found in Appendix.

3.3. Results

Our experimental results demonstrate the effectiveness of WorldAgen across both CALVIN and LIBERO benchmarks.

CALVIN As shown in Table 1, we compare WorldAgen with recent VLA baselines, including RoboFlamingo Li et al. (2023), SuSIE Black et al. (2023), GR-1 Wu et al. (2023a), 3D Diffusor Actor Ke, Gkanatsios, and Fragkiadaki (2024), and CLOVER Bu et al. (2024). WorldAgen follows a GR-1-style GPT video architecture, but introduces the Mixed Unidirectional Attention Mask to separate task instruction from world modeling and employs a lightweight TTT strategy to enhance world-modeling capability. Our method achieves consistent performance improvements across five consecutive tasks, indicating that ***strengthening world modeling via TTT improves environment understanding and boosts long-horizon manipulation performance.***

LIBERO As shown in Table 2, we further evaluate WorldAgen on the LIBERO benchmark against multi-task VLA baselines such as MT-ACT Bharadhwaj et al. (2024), MVP Xiao et al. (2022), MPI Zeng et al. (2024), OpenVLA Kim et al. (2024) and Seer Tian et al. (2024). WorldAgen

attains the best or on-par performance across most LIBERO subsets, further supporting that test-time adaptation of the world model leads to stronger environment understanding and improved robustness to distribution shifts.

We observe improvements or maintained performance across almost every task compared to the baseline. These consistent gains across diverse manipulation scenarios mirror our findings on CALVIN, demonstrating ***robustness and generalization ability*** of our TTT approach across different benchmarks and scenario dynamics.

3.4. Ablation Study

To better understand the key components contributing to performance, we conduct ablation studies on (1) the contribution of world modeling to policy learning, (2) the effect of LoRA parameterization during TTT, (3) the impact of the amount of randomly sampled data used for TTT, and (4) transfer from simulator to the real robot. We further analyze the impact of trajectory splitting and preprocessing, image and action chunk lengths, and different Transformer backbones, and compare LoRA-based TTT with full fine-tuning while visualizing performance before and after TTT. All ablation and additional experimental results are provided in the Appendix.

3.4.1. World Modeling Ability

Dataset	World Modeling	Avg. Success ↑
CALVIN	×	2.96
	✓	**3.87**
LIBERO	×	46.5%
	✓	**78.0%**

Table 3 | Ablation study on world modeling. We compare models with and without image prediction to evaluate the contribution of world modeling to agent policy learning.

To validate the contribution of world modeling to policy learning, we compare models with and without image prediction capability by including or removing image-prediction tokens in the input as shown in Table 3. World modeling consistently improves performance: on CALVIN, the average success length increases from 2.961 to 3.87 (+30.7%), and on LIBERO, the success rate rises from 46.5% to 78.0% (+67.7%). These results indicate that ***world modeling yields richer representations of environment dynamics, leading to more effective action prediction and policy learning.***

3.4.2. LoRA Configuration

We study the effect of LoRA rank on TTT performance while fixing the learning rate and training data size as shown in Table 4. Performance only varies within 0.02 across ranks, indicating that ***under fixed hyperparameters, TTT performance is largely insensitive to the LoRA rank.*** We therefore use a LoRA rank of 128 in our main experiments, which slightly outperforms other settings while remaining computationally efficient.

LoRA Rank	Avg. Len. ↑
16	3.928
32	3.918
64	3.918
128	**3.930**
256	3.923

Table 4 | Ablation study on LoRA rank during TTT in CALVIN dataset. All other parameters are kept constant.

Test Time Training Data	Avg. Lens. ↑
6	3.871
90	3.922
204	**3.928**
340	3.917

Table 5 | Ablation study on TTT data sampling volume. We vary the amount of test-time training data, defined as the product of number of samples and repeat times.

3.4.3. TTT Data Sampling Volume

We study how the amount of test-time training data affects performance as shown in Table 5. As the number of TTT samples increases from 6 to 204, the average success score improves from 3.871 to 3.928, indicating that more diverse exploration data benefits world-model adaptation. However, further increasing the data to 340 slightly reduces performance to 3.917, suggesting diminishing returns and mild overfitting to image generation rather than action prediction. These results indicate that ***moderately increasing TTT data improves performance, but excessive data can be counterproductive, highlighting the need for an appropriate TTT data budget.***

3.4.4. Simulator to Real World

	w/o TTT	w/o Noise	w/ Noise
CALVIN	3.87	3.93	3.90
LIBERO	75.5	79.0	78.0

Table 6 | Performance comparison under Gaussian noise perturbation.

We add Gaussian noise (std = 0.1) to the observations to simulate real-world conditions, such as camera distortion and sensor noise, as shown in Table 6.

This experiment indicates that ***even under noisy conditions, our method consistently outperforms the baselines, demonstrating its robustness and strong adaptability to real-world uncertainty.***

4. Related Work

Vision-Language-Action Models. Vision-Language-Action (VLA) models unify perception, language, and control for robotic manipulation (Sapkota et al., 2025; Din et al., 2025). Early

systems such as RT-1 and RT-2 established the effectiveness of large-scale transformer policies, while PaLM-E and LLaVA further incorporated multimodal grounding (Brohan et al., 2023; Zitkovich et al., 2023; Driess et al., 2023; Liu et al., 2023b). However, most VLA models directly predict actions without explicitly modeling world dynamics, limiting generalization to novel environments (Zhang et al., 2025).

World Models. World models learn predictive dynamics to support planning and decision-making (Sutton, 1990; Hafner et al., 2025). Neural approaches such as World Models and Dreamer improve sample efficiency and long-horizon reasoning (Ha and Schmidhuber, 2018; Hafner et al., 2022, 2025). In robotics, however, these models are typically used for state prediction or model-based planning and remain largely decoupled from language-grounded action generation (Sakagami et al., 2023).

Test-Time Training. Test-time training adapts models to distribution shifts during inference via self-supervised objectives (Sun et al., 2020; Ma, 2024). While TTT has been explored in both vision and NLP (Ye et al., 2023), robotic adaptation has mainly focused on perception modules or low-level policies (Zhao, Queralta, and Westerlund, 2020). Our work instead incorporates TTT into a unified VLA framework by adapting the world model itself during deployment.

5. Conclusion

We introduced **WorldAgen**, a unified vision–language–action framework that jointly learns world modeling and action prediction through a shared Transformer backbone. WorldAgen integrates a world model head that predicts future states and a policy head that predicts task-conditioned actions. At test time, the model performs lightweight Test-Time Training (TTT) using short exploratory rollouts, turning world modeling into an online adaptation signal. This enables the agent to quickly adjust to new environments and leads to consistent performance gains on CALVIN and LIBERO, demonstrating that adaptive VLA systems can benefit substantially from continual test-time interaction.

6. Limitations and Future Work

Our study evaluates only a single model size and architecture, and focuses test-time training exclusively on the world-model component. Future work includes exploring larger and more diverse model architectures, as well as developing joint test-time-training methods that improve both action prediction and world modeling.

References

Bharadhwaj, H.; Vakil, J.; Sharma, M.; Gupta, A.; Tulsiani, S.; and Kumar, V. 2024. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 4788–4795. IEEE.

Black, K.; Nakamoto, M.; Atreya, P.; Walke, H.; Finn, C.; Kumar, A.; and Levine, S. 2023. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*.

Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Dabis, J.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jackson, T.; Jesmonth, S.; Joshi, N. J.; Julian, R.;

Kalashnikov, D.; Kuang, Y.; Leal, I.; Lee, K.-H.; Levine, S.; Lu, Y.; Malla, U.; Manjunath, D.; Mordatch, I.; Nachum, O.; Parada, C.; Peralta, J.; Perez, E.; Pertsch, K.; Quiambao, J.; Rao, K.; Ryoo, M.; Salazar, G.; Sanketi, P.; Sayed, K.; Singh, J.; Sontakke, S.; Stone, A.; Tan, C.; Tran, H.; Vanhoucke, V.; Vega, S.; Vuong, Q.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; Yu, T.; and Zitkovich, B. 2023. RT-1: Robotics Transformer for Real-World Control at Scale. arXiv:2212.06817.

Bu, Q.; Zeng, J.; Chen, L.; Yang, Y.; Zhou, G.; Yan, J.; Luo, P.; Cui, H.; Ma, Y.; and Li, H. 2024. Closed-loop visuomotor control with generative expectation for robotic manipulation. *Advances in Neural Information Processing Systems*, 37: 139002–139029.

Din, M. U.; Akram, W.; Saoud, L. S.; Rosell, J.; and Hussain, I. 2025. Vision Language Action Models in Robotic Manipulation: A Systematic Review. arXiv:2507.10672.

Driess, D.; Xia, F.; Sajjadi, M. S.; Lynch, C.; Chowdhery, A.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; Huang, W.; et al. 2023. Palm-e: An embodied multimodal language model.

Guo, Y.; Hu, Y.; Zhang, J.; Wang, Y.-J.; Chen, X.; Lu, C.; and Chen, J. 2024. Prediction with action: Visual policy learning via joint denoising process. *Advances in Neural Information Processing Systems*, 37: 112386–112410.

Ha, D.; and Schmidhuber, J. 2018. World Models. *CoRR*.

Hafner, D.; Lillicrap, T.; Norouzi, M.; and Ba, J. 2022. Mastering Atari with Discrete World Models. arXiv:2010.02193.

Hafner, D.; Pasukonis, J.; Ba, J.; and Lillicrap, T. 2025. Mastering diverse control tasks through world models. *Nature*, 1–7.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16000–16009.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.

Intelligence, P.; Black, K.; Brown, N.; Darpinian, J.; Dhabalia, K.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; et al. 2025. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*.

Jaegle, A.; Gimeno, F.; Brock, A.; Vinyals, O.; Zisserman, A.; and Carreira, J. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*, 4651–4664. PMLR.

Ke, T.-W.; Gkanatsios, N.; and Fragkiadaki, K. 2024. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*.

Kim, M. J.; Finn, C.; and Liang, P. 2025. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. arXiv:2502.19645.

Kim, M. J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; Vuong, Q.; Kollar, T.; Burchfiel, B.; Tedrake, R.; Sadigh, D.; Levine, S.; Liang, P.; and Finn, C. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. arXiv:2406.09246.

Kojima, Y.; Xu, J.; Zou, X.; and Wang, X. 2025. LoRA-TTT: Low-Rank Test-Time Training for Vision-Language Models. arXiv:2502.02069.

Lanier, J.; Kim, K.; Karamzade, A.; Liu, Y.; Sinha, A.; He, K.; Corsi, D.; and Fox, R. 2025. Adapting World Models with Latent-State Dynamics Residuals. arXiv:2504.02252.

Li, P.; Wu, Y.; Xi, Z.; Li, W.; Huang, Y.; Zhang, Z.; Chen, Y.; Wang, J.; Zhu, S.-C.; Liu, T.; and Huang, S. 2025. ControlVLA: Few-shot Object-centric Adaptation for Pre-trained Vision-Language-Action Models. arXiv:2506.16211.

Li, X.; Liu, M.; Zhang, H.; Yu, C.; Xu, J.; Wu, H.; Cheang, C.; Jing, Y.; Zhang, W.; Liu, H.; et al. 2023. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*.

Liu, B.; Zhu, Y.; Gao, C.; Feng, Y.; Liu, Q.; Zhu, Y.; and Stone, P. 2023a. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36: 44776–44791.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023b. Visual instruction tuning. *Advances in neural information processing systems*, 36: 34892–34916.

Liu, S.; Wu, L.; Li, B.; Tan, H.; Chen, H.; Wang, Z.; Xu, K.; Su, H.; and Zhu, J. 2025. RDT-1B: a Diffusion Foundation Model for Bimanual Manipulation. arXiv:2410.07864.

Ma, J. 2024. Improved self-training for test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 23701–23710.

Ma, Y.; Song, Z.; Zhuang, Y.; Hao, J.; and King, I. 2025. A Survey on Vision-Language-Action Models for Embodied AI. arXiv:2405.14093.

Mees, O.; Hermann, L.; Rosete-Beas, E.; and Burgard, W. 2022. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3): 7327–7334.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Sakagami, R.; Lay, F. S.; Dömel, A.; Schuster, M. J.; Albu-Schäffer, A.; and Stulp, F. 2023. Robotic world models—conceptualization, review, and engineering best practices. *Frontiers in Robotics and AI*, 10: 1253049.

Sapkota, R.; Cao, Y.; Roumeliotis, K. I.; and Karkee, M. 2025. Vision-Language-Action Models: Concepts, Progress, Applications and Challenges. arXiv:2505.04769.

Sun, Y.; Wang, X.; Liu, Z.; Miller, J.; Efros, A. A.; and Hardt, M. 2020. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. arXiv:1909.13231.

Sutton, R. S. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, 216–224. Elsevier.

Tian, Y.; Yang, S.; Zeng, J.; Wang, P.; Lin, D.; Dong, H.; and Pang, J. 2024. Predictive Inverse Dynamics Models are Scalable Learners for Robotic Manipulation. arXiv:2412.15109.

Wolf, R.; Shi, Y.; Liu, S.; and Rayyes, R. 2025. Diffusion Models for Robotic Manipulation: A Survey. arXiv:2504.08438.

Wu, H.; Jing, Y.; Cheang, C.; Chen, G.; Xu, J.; Li, X.; Liu, M.; Li, H.; and Kong, T. 2023a. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*.

Wu, P.; Escontrela, A.; Hafner, D.; Abbeel, P.; and Goldberg, K. 2023b. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, 2226–2240. PMLR.

Xian, Z.; and Gkanatsios, N. 2023. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *Conference on Robot Learning/Proceedings of Machine Learning Research*. Proceedings of Machine Learning Research.

Xiao, T.; Radosavovic, I.; Darrell, T.; and Malik, J. 2022. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; Zheng, C.; Liu, D.; Zhou, F.; Huang, F.; Hu, F.; Ge, H.; Wei, H.; Lin, H.; Tang, J.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Zhou, J.; Lin, J.; Dang, K.; Bao, K.; Yang, K.; Yu, L.; Deng, L.; Li, M.; Xue, M.; Li, M.; Zhang, P.; Wang, P.; Zhu, Q.; Men, R.; Gao, R.; Liu, S.; Luo, S.; Li, T.; Tang, T.; Yin, W.; Ren, X.; Wang, X.; Zhang, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Wang, Z.; Cui, Z.; Zhang, Z.; Zhou, Z.; and Qiu, Z. 2025. Qwen3 Technical Report. arXiv:2505.09388.

Ye, H.; Ding, Y.; Li, J.; and Ng, H. T. 2023. Robust Question Answering against Distribution Shifts with Test-Time Adaptation: An Empirical Study. arXiv:2302.04618.

Zeng, J.; Bu, Q.; Wang, B.; Xia, W.; Chen, L.; Dong, H.; Song, H.; Wang, D.; Hu, D.; Luo, P.; et al. 2024. Learning manipulation by predicting interaction. *arXiv preprint arXiv:2406.00439*.

Zhang, J.; Wu, S.; Luo, X.; Wu, H.; Gao, L.; Shen, H. T.; and Song, J. 2025. InSpire: Vision-Language-Action Models with Intrinsic Spatial Reasoning. arXiv:2505.13888.

Zhao, W.; Queralta, J. P.; and Westerlund, T. 2020. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, 737–744. IEEE.

Zitkovich, B.; Yu, T.; Xu, S.; Xu, P.; Xiao, T.; Xia, F.; Wu, J.; Wohlhart, P.; Welker, S.; Wahid, A.; et al. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, 2165–2183. PMLR.

Appendix

Table of Contents

This appendix provides comprehensive supplementary materials for our WorldAgen framework. We present detailed implementation specifications including benchmark descriptions, model architecture components, and trajectory processing methods for variable-length chunks. Additionally, we report extensive experimental results examining the effects of different chunk configurations on baseline performance, and comprehensive TTT analyses comparing LoRA (Hu et al., 2022) versus full fine-tuning approaches across various parameter settings and data sizes.

A. Implementation Details

A.1. Benchmark

LIBERO Benchmark (Liu et al., 2023a) is a comprehensive benchmark for lifelong learning in robot manipulation that consists of four distinct task suites designed to evaluate different aspects of knowledge transfer. The benchmark includes LIBERO-Spatial (10 tasks focusing on spatial relationship transfer), LIBERO-Object (10 tasks emphasizing object-centric knowledge transfer), LIBERO-Goal (10 tasks targeting procedural knowledge generalization), and LIBERO-100 (100 tasks with highly entangled knowledge requirements). Each task suite is accompanied by high-quality human teleoperation demonstrations to support sample-efficient learning.

We use LIBERO-100, which is the most challenging and comprehensive subset, containing 100 diverse manipulation tasks that require the transfer of mixed declarative and procedural knowledge. This suite is further divided into LIBERO-90, consisting of 90 short-horizon tasks used for pretraining policies, and LIBERO-10 (also referred to as LIBERO-Long), which contains 10 long-horizon tasks specifically selected for evaluating downstream lifelong learning performance.

CALVIN Benchmark (Mees et al., 2022) is an open-source simulated benchmark designed for learning long-horizon language-conditioned robot manipulation tasks. The dataset contains 34 manipulation tasks that are more complex than existing vision-and-language datasets in terms of sequence length, action space, and language complexity. The environment features a Franka Emika Panda robot with a parallel-jaw gripper operating in a desktop workspace containing various interactive objects, including a sliding door, drawer, colored blocks, LED, and light bulb that can be manipulated according to unconstrained natural language instructions.

CALVIN provides four different environments (A, B, C, D) that vary in desk colors and object configurations, enabling evaluation across different visual contexts and supporting flexible specification of sensor suites. The benchmark's evaluation protocol requires agents to solve sequences of up to five consecutive tasks by understanding and executing a series of language instructions, such as *open the drawer, pick up the blue block, rotate the block, push the block into the drawer, open the sliding door.* This sequential task completion paradigm makes CALVIN particularly challenging for assessing the generalization capabilities and long-horizon reasoning of language-conditioned manipulation policies.

A.2. Model Architecture

Leveraging the flexible architecture of Transformers, we achieve unified processing of image, language, action, and robot state modalities, supporting variable-length image chunks and action chunks for both input and output.

Encoder In the encoder component, we employ a MAE-pretrained ViT-B (He et al., 2022)

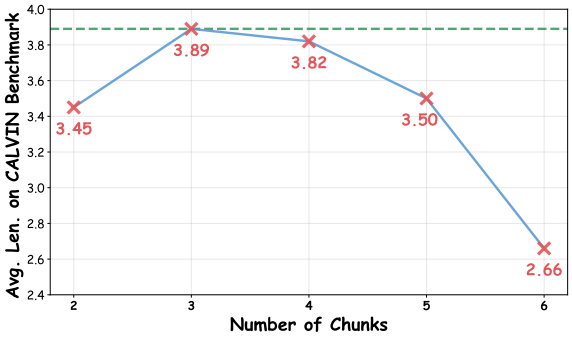

Figure 4 | This figure shows the performance of the WorldAgen model on CALVIN as the number of chunks changes.

as the visual encoder, utilizing dual-view RGB inputs from static and wrist cameras. Similar to Seer (Tian et al., 2024), we incorporate a Perceiver Resampler (Jaegle et al., 2021) to reduce the number of image tokens, thereby decreasing computational load and improving efficiency. The Perceiver Resampler is an attention-based feature compression module that uses a set of learnable query tokens to extract the most important information from a large number of image features. Through this approach, it compresses the originally large number of image tokens into a fixed number of compact representations, preserving essential visual information while significantly reducing the computational complexity of subsequent network layers.

For language processing, we utilize the text encoder from CLIP ViT-B/32. For robot state and action, we employ MLPs to project them into the same embedding space. Both robot state and action are represented as 7-dimensional vectors, where the first 6 dimensions encode the arm state representing the end effector's position and orientation, and the last dimension represents the gripper state indicating the open/close status. We use separate MLP layers to process arm state and gripper state independently, then concatenate the results.

We initialize predicted tokens as zero-filled tensors with the same dimensions as the target actions and images to be predicted, and insert them after the input data at each time step.

Decoder In the decoder, we index the predicted tokens based on the input and slice out the image and action tokens. For image tokens, we employ the Vision Transformer encoder architecture (He et al., 2022) for decoding, followed by a linear layer to predict pixel patches. For the action decoder, similarly, we use an MLP to reduce the action-corresponding vectors to 7 dimensions. For the arm state and gripper state components, we employ different linear layers for decoding. For the gripper state, we apply binary thresholding at 0.5 to represent the gripper's open/close status (0 for closed, 1 for open).

Backbone Qwen3 (Yang et al., 2025) represents the latest advancement in the Qwen large language model family, comprising both dense and Mixture-of-Experts (MoE) architectures with parameter scales ranging from 0.6 to 235 billion. Utilizing Qwen3 as a backbone architecture offers significant advantages for large language model applications, primarily through its innovative parameter efficiency where Qwen3 dense base models achieve performance comparable to much larger Qwen2.5 models while using fewer parameters, with Qwen3-4B matching the performance of Qwen2.5-72B-Instruct. The MoE variants provide exceptional computational efficiency, as Qwen3-MoE base models deliver similar performance to Qwen2.5 dense base models while utilizing only 10% of the active parameters, resulting in significant savings in both training and inference costs.

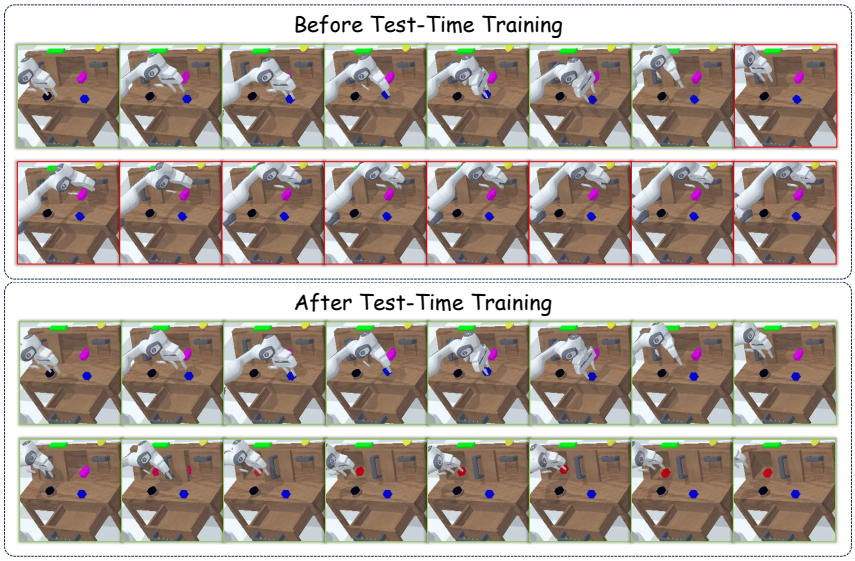

Figure 5 | In the comparison between the trajectories before and after TTT, the red regions indicate failures within the trajectory, while the green regions represent successful portions.

B. Baseline Results

In this section, we present more results of our baseline on CALVIN benchmark, including pretraining, different length and number of chunks and different backbone.

B.1. Pretraining

Given the powerful modeling capability and flexibility of Qwen3 Yang et al. (2025), we adopt it as the network backbone for WorldAgen. Following the configuration established in (Tian et al., 2024), we configure the transformer architecture with 12 transformer heads and 24 transformer layers, resulting in a total network size of 370M parameters with 120M trainable parameters. Model pretraining is performed on a 4×H100 GPU server, taking approximately 60 hours for CALVIN and 5 hours for LIBERO.

B.2. Image and Action Chunk Lengths

WorldAgen offers flexible input-output interfaces, accommodating image and action chunks of variable lengths. We evaluate the effect of different image and action chunk lengths on model performance. The results are shown in Table 7.

Therefore, on CALVIN benchmark, we employ an image chunk length of 1, action chunk length of 5, and trajectory length of 16. For LIBERO, we use an image chunk length of 1, action chunk length of 3, and trajectory length of 7.

We first fix the image chunk length to 1, as adjacent image frames contain redundant information and predicting more images increases computational overhead, thereby reducing model efficiency. By varying the action chunk length, we observe that ***longer action sequences lead to degraded performance***. We attribute this to error accumulation in action predictions, where mistakes in early actions compound over longer sequences. However, excessively short action chunk lengths also result in performance degradation, as the network learns less information from the dataset.

Algorithm 1 Trajectory splitting with variable-length action, observation, and robot state units.

Require: Rollout $\tau = \{(o_t, s_t, a_t)\}_{t=0}^{T-1}$, observation chunk length n, action chunk length m

$\mathcal{U} \leftarrow \emptyset$
for $i = 0$ **to** $I - 1$ **do**
  if $i = 0$ **then**
    $\mathcal{I}^{\text{obs}} \leftarrow \{0, 1, \ldots, n-1\}$
    $O^n_{0\to n-1} \leftarrow \{\, o_t \mid t \in \mathcal{I}^{\text{obs}} \,\}$
    $S^n_{0\to n-1} \leftarrow \{\, s_t \mid t \in \mathcal{I}^{\text{obs}} \,\}$
    $A_{0\to n-1} \leftarrow \{\, a_t \mid t \in \mathcal{I}^{\text{obs}} \,\}$
    $O_i \leftarrow O^n_{0\to n-1}$
    $S_i \leftarrow S^n_{0\to n-1}$
    $A_i \leftarrow A_{0\to n-1}$
    $A^p_i \leftarrow \mathbf{0}^n$ // action placeholder (length n)
    $O^p_i \leftarrow \mathbf{0}^n$ // observation placeholder (length n)
  else
    $a \leftarrow n + (i-1)m$
    $b \leftarrow n + im - 1$
    $\mathcal{I}^{\text{obs}} \leftarrow \textsc{UniformSubsample}(a, b, n)$
    $O^n_{n+(i-1)m\to n+im-1} \leftarrow \{\, o_t \mid t \in \mathcal{I}^{\text{obs}} \,\}$
    $S^n_{n+(i-1)m\to n+im-1} \leftarrow \{\, s_t \mid t \in \mathcal{I}^{\text{obs}} \,\}$
    $A_{n+(i-1)m\to n+im-1} \leftarrow \{\, a_t \mid n + (i-1)m \le t \le n + im - 1 \,\}$
    $O_i \leftarrow O^n_{n+(i-1)m\to n+im-1}$
    $S_i \leftarrow S^n_{n+(i-1)m\to n+im-1}$
    $A_i \leftarrow A_{n+(i-1)m\to n+im-1}$
    $A^p_i \leftarrow \mathbf{0}^m$ // action placeholder (length m)
    $O^p_i \leftarrow \mathbf{0}^n$ // observation placeholder (length n)
  end if
  $U_i \leftarrow \left(G, O_i, S_i, A_i, A^p_i, O^p_i\right)$
  $\mathcal{U} \leftarrow \mathcal{U} \cup \{U_i\}$
end for
return $\mathcal{U}$

On LIBERO, we obtain consistent results, with performance deteriorating as action chunk length increases. Based on these findings, we adopt action chunk lengths of 5 for CALVIN and 3 for LIBERO in our main experiments.

B.3. Number of Chunks

To investigate the effect of the number of chunks, we fix the image chunk length to 1 and the action chunk length to 3, conducting comprehensive experiments with the number of chunks varying from 2 to 6. As illustrated in Figure 4, we observe the **same pattern** as in our experiments with image chunk lengths and action chunk lengths.

A larger number of chunks require the model to predict more action steps, which leads to accumulation of prediction errors. And the decreasing rate becomes larger when the number of chunks increases. However, an excessively small number of chunks limits the model's ability to learn the relationship between world modeling and actions during training.

According to our results, setting the number of chunks to 3 achieves the best performance.

Dataset	I. Chunk	A. Chunk	Avg. Success ↑
CALVIN	1	3	3.82
	1	**5**	**3.87**
	1	7	3.43
	1	9	3.16
	3	5	3.30
	5	5	1.78
LIBERO	**1**	**3**	**78.0%**
	1	5	74.5%
	1	7	47.0%

Table 7 | Ablation study on image and action chunk lengths across CALVIN and LIBERO benchmarks. I. Chunk and A. Chunk represent image chunk length and action chunk length, respectively

We apply this finding to the configuration with image chunk length and action chunk length equal to 1 and 5, thereby achieving state-of-the-art results. This demonstrates

Both the scalability and robustness of our baseline approach, validating that the optimal chunk configuration principles discovered through systematic experimentation can be effectively transferred to different parameter settings to achieve superior performance.

C. Test-Time Training Ablation Study

C.1. TTT Details

For CALVIN, since the test scenarios are relatively uniform within each environment, we do not need to perform TTT for every individual sample. Instead, we conduct adaptation by sampling only on the first sample's scenario. We select 34 text instructions that have appeared in the training set as guidance and perform random exploration for 60 frames. We then uniformly sample 6 times within these 60 frames for each task, which leads to $34 \times 6 = 204$ trajectories in total. Each trajectories have the same length as the pretraining dataset. Following Kojima et al. (2025), we use single-epoch single-step optimization with LoRA rank set to 128, employing the AdamW optimizer with a learning rate of 0.005 and weight decay of 0.01.

For LIBERO, the evaluation involves 10 different test scenarios in the LIBERO-10 dataset, necessitating a different TTT strategy. We perform scene-level TTT sampling in each individual test scenario to adapt to the specific environmental conditions. Similar to CALVIN, we uniformly sample for 6 times with 6 different random seed for 60 frames exploration, which generates 6×6 samples per scenario. Also we set LoRA rank to 64 and employ single-epoch single-step optimization using the AdamW optimizer with a learning rate of 0.005 and weight decay of 0.01.

C.2. LoRA vs. Full Fine-tuning

We conducted a comparative study between LoRA and Full Fine-Tuning (FFT) on the CALVIN dataset. In our experiments, we set the LoRA rank to 128. Both LoRA fine-tuning and FFT used identical hyperparameters: learning rate $lr = 0.0005$, Adam optimizer, and weight decay of 0.01.

As shown in Table 8, our results reveal that FFT does not improve model performance and

Method	Avg. Len.
WorldAgen	3.87
WorldAgen + LoRA	**3.93**
WorldAgen + FFT	3.85

Table 8 | Comparison of test-time training with LoRA and FFT on CALVIN dataset. Avg. Len. represents the average sequence length.

actually leads to a decline in performance. We attribute this phenomenon to the fact that

FFT causes the network to overfit to the test scenarios, whereas LoRA enables the model to acquire scenario-specific features while preserving the original scene perception capabilities. Therefore, LoRA demonstrates superior performance by striking a better balance between adaptation to new scenarios and retention of pretrained knowledge.

C.3. Scalability and Robustness

To investigate the scalability and robustness of our method, we select three parameters crucial to TTT: number of chunks (N), image chunk length (n), and action chunk length (m). We conducted TTT experiments across different combinations of N, n, and m values.

We choose three representative settings with different configurations to demonstrate the generalizability of our approach across various parameter combinations.

N	n	m	LoRA Rank	Data Size	Before TTT After TTT
3	1	3	64	170	3.89 **3.90**
5	1	3	256	204	3.50 **3.67**
3	1	5	128	204	3.87 **3.93**

Table 9 | TTT experiments under different number of chunks (N), image chunk length (n), and action chunk length (m) configurations.

As demonstrated in the Table 9, TTT consistently improves performance across different settings using only around 200 samples. This validates

The scalability and robustness of our TTT approach and it can effectively adapt to various architectural configurations while maintaining consistent improvement patterns.

C.4. Test-Time Training Data Size

Furthermore, during the grid search process for the experimental configuration with $n = 1$, $m = 3$, $N = 5$, as illustrated in the Figure 6, we discovered that TTT exhibits linear performance improvement with small amounts of data. However, when the data volume becomes excessive, we observe a phenomenon where TTT performance degrades. These experiments were conducted with LoRA rank = 256 and learning rate = 0.0005.

We attribute this phenomenon to the fact that excessive training samples cause LoRA to overfit on the world modeling component, making the model more biased towards predicting

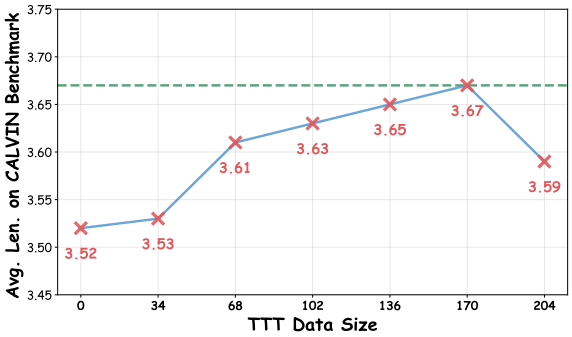

Figure 6 | This figure shows that as the amount of data increases, the TTT effect gradually increases.

world observations rather than executing correct actions. This finding highlights

The importance of carefully balancing the amount of test-time training data to achieve optimal performance without compromising the model's action execution capabilities.

C.5. Comparison Between Backbones

Backbone	CALVIN	LIBERO-10
GPT2	3.32	75.0
Qwen3	3.87	75.5

Table 10 | Performance comparison on different backbones.

We incorporate the GPT-2 Radford et al. (2019) backbone to examine the performance gap across different language backbones. Compared with GPT-2, Qwen3 demonstrates stronger representation capacity and better alignment with multimodal instructions, leading to consistently higher performance, as shown in Table 10.

C.6. LIBERO Subsets

Spatial	Object	Goal	LIBERO-10
93.0	79.5	91.0	79.0

Table 11 | Performance across different subsets in LIBERO.

We further include additional subsets of the LIBERO benchmark to comprehensively evaluate the performance of WorldAgen. The results demonstrate that our method maintains stable and consistent performance across various subsets, indicating strong generalization ability and robustness to task diversity, as shown in Table 11.

C.7. Comparison Visualization

We visualize the comparison before and after applying TTT. As shown in the figure 5, the red regions represent the failed portions of the trajectory, while the green regions denote the successful

units. The selected task sequence is *rotate_blue_block_right*, *move_slider_right*, *lift_red_block_slider*, and *place_in_slider*. It can be observed that before applying TTT, the robot only completed the *rotate_blue_block_right* task and then entered a halted state. ***After incorporating TTT, however, the model demonstrates a significantly enhanced ability to model the environment, thereby successfully accomplishing a series of tasks.***

Table 12 | Summary of important notations used in the WorldAgen framework.

Notation	Symbol	Description
Action	a_t	Robot action at time step t in the raw rollout.
Observation	o_t	Environmental observation at time step t in the raw rollout.
Robot State	s_t	Robot proprioceptive state at time step t in the raw rollout.
Rollout Trajectory	τ	A trajectory with T time steps: $\tau = \{(o_t, s_t, a_t)\}_{t=0}^{T-1}$.
Task Instruction	G_i	Language instruction specifying the task in unit i.
Action Chunk Length	m	Number of actions in each action chunk A_i.
Observation / Robot State Chunk Length	n	Number of observations and robot proprioceptive states in each chunk O_i and S_i.
Trajectory Unit	U_i	The i-th trajectory unit: $U_i = (G_i, O_i, S_i, A_i, A_p, O_p)$.
Action Chunk	A_i	Action chunk in unit i: $A_i = A_{n+(i-1)m \to n+im-1} = \left[a_{n+(i-1)m}, a_{n+(i-1)m+1}, \ldots, a_{n+im-1}\right]$.
Observation Chunk	O_i	Observation chunk in unit i: $O_i = O^n_{n+(i-1)m \to n+im-1} = \left[o_{n+(i-1)m}, o_{n+(i-1)m+\lfloor \frac{1}{n} m \rfloor}, \ldots, o_{n+(i-1)m+\lfloor \frac{n-1}{n} m \rfloor}\right]$.
Robot State Chunk	S_i	Robot state chunk in unit i: $S_i = S^n_{n+(i-1)m \to n+im-1} = \left[s_{n+(i-1)m}, s_{n+(i-1)m+\lfloor \frac{1}{n} m \rfloor}, \ldots, s_{n+(i-1)m+\lfloor \frac{n-1}{n} m \rfloor}\right]$.
Action Placeholder	A_p	Placeholder action tokens.
Observation Placeholder	O_p	Placeholder observation tokens.
Predicted Action Chunk	$\hat{A}_i$	Predicted action chunk: $\hat{A}_i = \left[\hat{a}_{n+(i-1)m}, \hat{a}_{n+(i-1)m+1}, \ldots, \hat{a}_{n+im-1}\right]$.
Predicted Observation Chunk	$\hat{O}_i$	Predicted next observation chunk: $\hat{O}_i = \hat{O}^n_{n+im \to n+(i+1)m-1} = \left[\hat{o}_{n+im}, \hat{o}_{n+im+\lfloor \frac{1}{n} m \rfloor}, \ldots, \hat{o}_{n+im+\lfloor \frac{n-1}{n} m \rfloor}\right]$.