Spatial Representation Distillation and Knowledge Routing for Vision-Language-Action Models

Jungin Park[1,2,*] **Chaoran Zhu**[3,*] **Changjae Oh**[3] **Kwanghoon Sohn**[2,†]

Abstract

Understanding three-dimensional physical structure has become essential for vision-language-action models (VLAs). However, existing approaches either rely on explicit geometric inputs at inference time or adopt a single visual pathway, potentially compromising the visual-semantic knowledge inherited from pre-trained models. We present SPARK-VLA, **Spa**tial **R**epresentation distillation and **K**nowledge routing for **VLA**s, a preserve-specialize-route framework for spatially adapting VLAs without requiring external 3D inputs or experts at inference. SPARK-VLA consists of two key components: (1) spatial representation distillation, which transfers 3D understanding from a frozen 3D expert to a LoRA-adapted visual branch by aligning attention responses while preserving the original visual pathway, and (2) a lightweight knowledge router, which selectively forwards action-relevant visual tokens to the language model using routing supervision derived from action-loss gradients. Across LIBERO, LIBERO-Plus, CALVIN, and real-world manipulation, SPARK-VLA achieves strong performance with only a 0.5B language backbone and approximately 53M trainable parameters, providing an effective and parameter-efficient approach to 3D-aware VLA learning. Code is available at https://github.com/park-jungin/spark-vla.

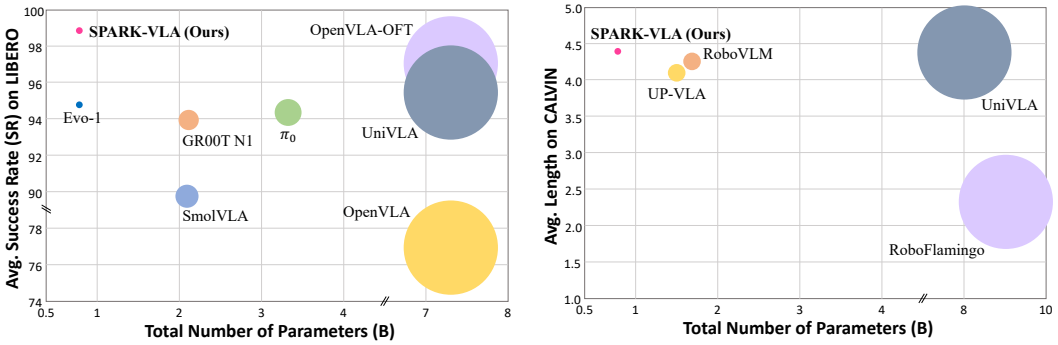
1 Introduction

Vision-language-action models (VLAs) learn to generate robot actions from visual observations and language instructions, offering a unified approach to perception, instruction understanding, and control. Recent systems have benefited from large-scale vision-language pre-training, robot demonstrations, and increasingly expressive action decoders [59, 20, 5, 48, 56]. Nevertheless, recognizing the objects involved in an instruction does not necessarily provide the geometric information required to manipulate them. Reaching, grasping, and placing an object also depend on its position, orientation, extent, and spatial relationship to the robot and surrounding objects. Even with multiple camera views, aggregating 2D visual features insufficiently yields geometrically consistent representations across viewpoints [52, 13, 12]. In other words, understanding and acting in the physical world is a fundamental capability for embodied intelligence, requiring joint modeling of perception, language, action, and *three-dimensional spatial structure*.

Recent studies have increasingly suggested that spatial awareness remains a key bottleneck for reliable robot manipulation and have investigated explicit 3D geometric priors, such as depth maps [3], point clouds [42], or other geometric cues [39, 57], and implicit alignment between 2D and 3D representations [26, 21] with pre-trained vision-language models (VLMs). Despite their promise, both directions have important limitations: (1) Explicit 3D integration typically depends on extra sensing or reconstructed geometric inputs that are not uniformly available across robot platforms and datasets, and (2) implicit alignment often distorts the original 2D representation space, which can weaken the perceptual capability inherited from strong vision-language backbones [58, 27].

[1] DGIST, South Korea. [2] Yonsei University, South Korea. [3] Queen Mary University of London, UK.
[*] Equally contributed. [†] Corresponding author: khsohn@yonsei.ac.kr

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Performance comparison on LIBERO (b) Performance comparison on CALVIN (ABC→D)

Figure 1: We depict (a) the averaged success rate for LIBERO and (b) the average length for CALVIN, with respect to the total number of parameters. The center point and the size of circles indicates the averaged performance and the size of LLMs in each model, respectively.

We propose SPARK-VLA (**Spa**tial **R**epresentation distillation and **K**nowledge routing for **VLA**s), a framework that enhances the 3D spatial understanding of a pre-trained VLM while preserving its original perceptual capability. Specifically, SPARK-VLA establishes a spatial pathway with low-rank adapters [15] *parallel* to the original visual pathway within the pre-trained vision backbone. During training, the original pathway remains frozen, while the adapted pathway learns geometry-aware representations from a pre-trained 3D experts (*e.g.*, VGGT [45]) through attention response distillation [36]: the adapted spatial query is shared with the expert's keys and values, allowing the spatial pathway to learn the geometry-aware information aggregated for the query.

SPARK-VLA further introduces an *action-driven knowledge router* to combine these complementary pathways. The router predicts instruction-conditioned scores over generic and spatial tokens and selectively forwards informative tokens to the language model, determining whether each region is represented by generic tokens, spatial tokens, both, or neither. To train the router without manually annotated supervision, we derive routing targets from the gradients of the action loss with respect to the router gates that indicate the usefulness of the tokens for action prediction. Together, preservation of the original pathway, specialization through spatial distillation, and action-driven routing provide a unified approach to integrating geometric knowledge into pre-trained VLAs.

Experimental results on simulation benchmarks, including LIBERO [29] and CALVIN [33], show that SPARK-VLA with only a 0.5B language backbone and 53M trainable parameters achieves performance comparable to or even better than substantially larger VLAs, as shown in Figure 1. In addition, SPARK-VLA generalizes robustly to challenging settings such as LIBERO-Plus [10] without any additional training. Experiments in real-world environments further demonstrate that our model achieves high sample efficiency under data-constrained conditions, outperforming much larger baselines.

2 Related Work

Vision-language-action models (VLAs). VLAs have emerged as a promising paradigm for generalist robot learning by jointly modeling visual perception, language understanding, and action generation [9, 59, 20, 11, 24, 35, 5]. Recent advances have explored diverse action generation strategies, including autoregressive action tokenization [20, 38], diffusion-based policies [8, 48, 14], and flow matching [5]. Alongside these developments, several studies have investigated efficient adaptation and compact architectures [19, 46, 41, 16], as well as reasoning and long-horizon manipulation [56, 58, 47]. Despite their progress, transferring visual-semantic knowledge to robotic control does not necessarily provide the fine-grained geometric information required for precise manipulation. We addresses this gap through implicit spatial knowledge adaptation.

Spatial understanding in VLAs. Spatial reasoning is essential for robotic manipulation, particularly when actions depend on object geometry, relative positions, and physical interactions. Existing approaches have incorporated explicit geometric information, such as depth and 3D representations,

to improve spatial understanding [39, 57, 3, 42]. More recently, several studies have explored implicit spatial learning by transferring knowledge from pre-trained geometric models or aligning visual representations with spatial priors [21, 26, 1]. Other approaches have emphasized preserving pre-trained semantic representations during efficient VLA adaptation [58, 27]. However, incorporating geometric knowledge into an existing visual pathway introduces potentially competing requirements: acquiring spatial information while retaining generic visual-semantic knowledge. Our SPARK-VLA preserves the original visual pathway and specializes a parallel LoRA-adapted pathway through attention response distillation [36], allowing spatial specialization without modifying the preserved representation.

Visual token selection and routing for action generation. Recent work has shown that selective visual processing is important for efficient and effective action generation in VLAs [51, 25, 23, 22, 44, 31]. Existing methods exploit temporal redundancy through token caching and pruning [51, 31], identify task-relevant information through instruction-conditioned selection [23, 22], or incorporate action-aware importance into token reduction [44, 25]. These approaches primarily focus on identifying informative tokens within a given visual representation. In contrast, SPARK-VLA addresses a different selection problem: determining which knowledge source should represent each spatial position when both generic and spatially specialized features are available. Our action-driven router jointly ranks tokens from the two pathways using supervision derived from downstream action loss.

3 Method

3.1 Motivation and overview

Existing efforts to enhance vision-language-action models (VLAs) with spatial intelligence generally follow two directions: introducing explicit 3D inputs, such as depth maps or point clouds, or implicitly injecting spatial knowledge into the visual representation. While the former often depends on additional sensing or reconstructed 3D signals, the latter can weaken the generic visual-semantic capability inherited from strong pre-trained vision-language backbones by forcing a single visual pathway to absorb both general perception and spatial reasoning.

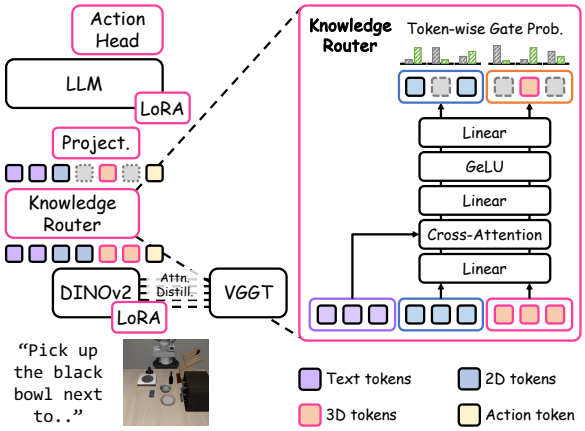


Figure 2: Overview of SPARK-VLA. Trainable modules are represented in the red boxes.

To address this issue, we propose SPARK-VLA, which preserves the original visual pathway while introducing a lightweight spatially adapted pathway. During training, we use a pre-trained 3D expert (*e.g.*, VGGT [45]) to distill spatial knowledge into the adapted branch. To effectively use the two branches, we further introduce a lightweight knowledge router that takes visual tokens, conditions on the language instruction, and predicts token-wise gate probabilities indicating which tokens are useful for action generation. The selected tokens are then forwarded to the language model for action prediction, allowing SPARK-VLA to exploit complementary generic and spatial cues while suppressing redundant or noisy representations. Note that the 3D expert is used only during training and is discarded at inference time. The overall architecture of our SPARK-VLA is shown in Figure 2.

3.2 Spatial representation distillation

To incorporate spatial understanding into the VLA's visual encoder without disrupting its original visual-semantic capability, we introduce a spatially adapted branch built on the pre-trained visual encoder.

Dual-branch visual encoding. Let f_θ^v denote the visual encoder of a pre-trained vision-language model (*e.g.*, Qwen2.5-VL [2]) and $f_{\theta+\Delta\theta}^v$ denote its LoRA-adapted counterpart, where only the

low-rank parameters $\Delta\theta$ are trainable. Given an RGB observation $\mathbf{I}_t$, SPARK-VLA extracts two complementary representations:

$$\mathbf{Z}_v = f_\theta^v(\mathbf{I}_t), \qquad \mathbf{Z}_p = f_{\theta+\Delta\theta}^v(\mathbf{I}_t), \tag{1}$$

where $\mathbf{Z}_v$ preserves generic visual-semantic knowledge and $\mathbf{Z}_p$ is adapted to capture spatial reasoning. During training, we further employ a pre-trained 3D encoder f_ϕ^p as a frozen teacher that provides spatial supervision to the adapted branch. In this way, the original branch preserves the pre-trained visual representation, while the adapted branch learns to encode additional spatial knowledge.

Attention response distillation. Instead of directly forcing the adapted branch to mimic the output representation of the 3D teacher, we distill *attention responses* [36]. Our key idea is to preserve compatibility with the original transformer structure by teaching the adapted branch *how to attend* in a spatially informed manner. At the l-th layer of $f_{\theta+\Delta\theta}^v$, we obtain the query, key, and value embeddings, denoted by $Q_v^{(l)}$, $K_v^{(l)}$, and $V_v^{(l)}$. From the frozen 3D teacher, we extract the corresponding key and value embeddings, denoted by $K_p^{(l)}$ and $V_p^{(l)}$. We then compute two attention responses using the same query embedding $Q_v^{(l)}$:

$$\mathbf{A}_v^{(l)} = \mathrm{Attn}(Q_v^{(l)}, K_v^{(l)}, V_v^{(l)}), \qquad \mathbf{A}_p^{(l)} = \mathrm{Attn}(Q_v^{(l)}, K_p^{(l)}, V_p^{(l)}), \tag{2}$$

where $\mathrm{Attn}(\cdot)$ denotes the scaled dot-product attention. Using the same query while replacing the key-value pairs allows the teacher response to specify how the adapted visual branch should interact with the attention mechanism to remain compatible with spatial knowledge.

The attention response distillation objective for spatial representations is defined as:

$$\mathcal{L}_{\text{attn}} = \sum_l \left\| \mathbf{A}_v^{(l)} - \mathbf{A}_p^{(l)} \right\|_1, \tag{3}$$

where $\|\cdot\|_1$ denotes the ℓ_1 loss. By aligning attention responses across layers, the adapted branch acquires geometry-aware behavior from the 3D teacher while retaining the original visual pathway for generic scene understanding.

3.3 Knowledge routing

While $\mathbf{Z}_v$ and $\mathbf{Z}_p$ provide complementary generic and spatial cues, not all tokens are equally useful for a given instruction and action prediction. Directly feeding all tokens from both branches into the LLM may introduce redundant or noisy information, especially when spatially adapted tokens are irrelevant to the current instruction. We therefore introduce a lightweight knowledge router that feeds instruction- and action-relevant visual tokens into the LLM.

Router architecture. We first collect the visual tokens from the generic and spatial branches and denote the merged token set by

$$\mathbf{Z} = [\mathbf{Z}_v; \mathbf{Z}_p] = \{\mathbf{z}_n\}_{n=1}^N, \tag{4}$$

where N is the total number of visual tokens. As shown in Figure 2, the knowledge router consists of (1) a linear layer to project $\mathbf{Z}$ into the same dimensional space with the text embeddings $\mathbf{H} = \{\mathbf{h}_m\}_{m=1}^M$; (2) a cross-attention layer to compute the correlations between the visual and text representations; and (3) a lightweight MLP (Linear-GeLU-Linear). The output of the last linear layer followed by a Sigmoid activation serves as the gate probabilities $\mathbf{p}$:

$$\mathbf{p} = \sigma(\mathrm{Router}(\mathbf{Z}, \mathbf{H})/\tau) = \{p_n\}_{n=1}^N, \tag{5}$$

where $\sigma(\cdot)$ is a sigmoid function and τ is a temperature parameter to control the sensitivity.

Token selection and knowledge fusion. Once the gate probabilities $\mathbf{p}$ are obtained, we select the top-k tokens among all N tokens:

$$\mathcal{S} = \mathrm{TopK}(\mathbf{p}, k), \qquad k = \max(k_{\min}, \mathrm{round}(\rho N)), \tag{6}$$

where $\mathrm{TopK}(\mathbf{p}, k)$ returns the indices of the k largest gate probabilities, ρ indicates the routing ratio, and $k_{\min}$ prevents overly sparse routing. We then project the selected tokens to the LLM's input space by fusing them according to their spatial positions. At each spatial position, the routed representation is constructed from the selected tokens among its generic and spatial pair. When both tokens are

selected, their embeddings are concatenated and projected into the LLM input space. If only one token is selected, it is concatenated with a zero embedding of the same dimension and passed through the same projection. Positions for which neither token is selected are discarded. This position-aware fusion preserves the alignment between generic and spatial knowledge, while producing a compact set of visual tokens with a unified input format for the LLM.

Gradient-based gate supervision. The main challenge in training the knowledge router is that there is no supervision signal indicating which visual tokens should be preserved. Instead of relying on heuristic salience measures [44, 31] or learning token selection only through downstream action optimization [23, 22], we derive routing targets from the gradient of the action loss with respect to the router gate probabilities. Given the action loss $\mathcal{L}_{\text{act}}$, the gradient of $\mathcal{L}_{\text{act}}$ *w.r.t.* the gate probability p_n, *i.e.*, $g_n = \partial \mathcal{L}_{\text{act}} / \partial p_n$, measures the influences of the n-th gate on action prediction. A negative gradient indicates that increasing the gate probability would reduce the action loss, suggesting that the corresponding token is useful. Therefore, we define the importance score as

$$s_n = \text{ReLU}(-g_n). \tag{7}$$

Since only routed tokens are forwarded to the LLM, the instantaneous gradients provide supervision only for the currently selected subset. We therefore maintain a momentum-based moving average of token importance scores, allowing tokens that are not selected in the current step to still contribute to router training through their accumulated utility. The detailed update rule is provided in Section A.4.

From $\bar{s}_n$, we construct a gating target $\mathbf{y} = \{y_n\}_{n=1}^{N}$:

$$y_n = \begin{cases} 1, & n \in \text{TopK}(\bar{\mathbf{s}}, k), \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

where $\bar{\mathbf{s}} = \{\bar{s}_n\}_{n=1}^{N}$ denotes the momentum-updated importance score. This gating target enables the router to learn action-aware token selection without explicit token-level annotations.

Routing objectives. We train the knowledge router with the classification problem. To handle the imbalance between selected and discarded tokens, we adopt the focal binary cross-entropy loss [28]:

$$\mathcal{L}_{\text{cls}} = \frac{1}{N} \sum_{n} \alpha_{y_n} (1 - p_t)^{\gamma} \Big[- y_n \log p_n - (1 - y_n) \log(1 - p_n) \Big], \tag{9}$$

where $p_t = y_n p_n + (1 - y_n)(1 - p_n)$. To further stabilize routing behavior, we additionally impose a budget regularization term that encourages the average routing ratio to match the target ratio ρ and an entropy regularization term that controls the uncertainty of the gate probabilities:

$$\mathcal{L}_{\text{bud}} = \left| \frac{1}{N} \sum_{n} p_n - \rho \right|, \qquad \mathcal{L}_{\text{ent}} = -\frac{1}{N} \sum_{n} \Big[p_n \log p_n + (1 - p_n) \log(1 - p_n) \Big]. \tag{10}$$

The final router objective is defined as

$$\mathcal{L}_{\text{router}} = \mathcal{L}_{\text{cls}} + \lambda_{\text{bud}} \mathcal{L}_{\text{bud}} + \lambda_{\text{ent}} \mathcal{L}_{\text{ent}}, \tag{11}$$

where λ_{bud} and λ_{ent} control the impact of budget constraint and entropy regularization, respectively. By combining the knowledge routing with gradient-based pseudo supervision, the proposed knowledge router learns to preserve action-relevant visual tokens from both the generic and spatial branches while suppressing redundant or noisy representations.

3.4 Training objective

SPARK-VLA is trained with three objectives: the action loss, the spatial representation distillation loss, and the knowledge router loss:

$$\mathcal{L}_{\text{spark}} = \mathcal{L}_{\text{act}} + \lambda_{\text{attn}} \mathcal{L}_{\text{attn}} + \lambda_{\text{router}} \mathcal{L}_{\text{router}}, \tag{12}$$

where λ_{attn} balances the contribution of the spatial distillation term. We use ℓ_1 regression for the action loss. Please see Section A.2 for training details.

Table 1: Comparisons with state-of-the-art methods on LIBERO. * incorporate extra depth or point cloud information from other sensors. Bold indicates the best performances among tiny-scale models.

Method	Spatial SR (%)	Object SR (%)	Goal SR (%)	Long SR (%)	Average SR (%)	LLM Size	Train. Params.
Large-scale models $\geq$ 7B							
Octo [11]	78.9	85.7	84.6	51.1	75.1	-	93M
OpenVLA [20]	84.9	88.4	79.2	53.7	76.5	7B	111M
OpenVLA-OFT [19]	97.6	98.4	97.9	94.5	97.1	7B	111M
Spatial Forcing [21]	99.4	99.6	98.8	96.0	98.5	7B	291M
UniVLA [47]	95.4	99.0	93.6	94.0	95.5	7B	8B
CogVLA [23]	98.6	98.8	96.6	95.4	97.4	7B	98M
1B $\leq$ Small-scale models $<$ 7B							
4D-VLA [54]	88.9	95.2	90.9	79.1	88.6	3.8B	4.2B
π_0 [5]	96.8	98.8	95.8	85.2	94.2	2B	3.3B
π_0-FAST [38]	96.4	96.8	88.6	60.2	85.5	2B	3.3B
SpatialVLA [39]	88.2	89.9	78.6	55.5	78.1	3B	3.4B
NORA [16]	92.2	95.4	89.4	74.6	87.9	3B	4B
SmolVLA [41]	93.0	94.0	91.0	77.0	88.8	1.7B	450M
GR00T N1 [4]	94.4	97.6	93.0	90.6	93.9	1.5B	700M
VLANeXt [50]	99.0	99.2	96.6	94.6	97.4	2B	2.7B
Tiny-scale models $<$ 1B							
Diffusion Policy [8]	78.3	92.5	68.3	50.5	72.4	-	263M
VLA-Adapter [46]	99.6	**99.6**	**98.2**	96.4	**98.5**	0.5B	598M
Evo-1 [27]	92.7	97.7	96.3	92.3	94.8	0.5B	770M
SPARK-VLA (**Ours**)	**99.6**	**99.6**	98.0	**96.6**	**98.5**	0.5B	53M

4 Simulation Experiments

4.1 Experimental setting

Dataset. We evaluate SPARK-VLA on three benchmarks, including LIBERO [29], LIBERO-Plus [10], and CALVIN [33].

- **LIBERO** [29] is a widely used benchmark for lifelong robot manipulation that contains four task suites, namely LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long, where each suite includes 10 language-conditioned manipulation tasks with 50 human demonstrations per task. These suites are designed to probe different types of generalization, including spatial reasoning, object-level transfer, goal-conditioned behavior, and long-horizon compositional manipulation.
- **LIBERO-Plus** [10] extends LIBERO with controlled perturbations over seven dimensions, including object layouts, camera viewpoints, robot initial states, language instructions, lighting conditions, background textures, and sensor noise, thereby providing a substantially more challenging testbed for generalization under realistic variations. To further examine robustness beyond the standard LIBERO setting, we additionally evaluate the trained model on LIBERO-Plus without re-training.
- **CALVIN** [33] is a long-horizon language-conditioned manipulation benchmark built on four simulated environments. CALVIN contains 34 manipulation tasks and 1,000 language instructions, and evaluates a policy by the average number of successfully completed sub-tasks in multi-step task sequences. Following the standard protocol, we report results on the benchmark's compositional generalization settings to measure long-horizon control and transfer to unseen environments.

Backbone and implementation details. We follow the implementation of [20] and [21] to build SPARK-VLA on top of OpenVLA-OFT [19]. Unlike the original setting, we use Prismatic VLM [17] with Qwen2.5-0.5B [2] as the language backbone, and adopt only DINOv2 [34] as the visual encoder without SigLIP [53]. On top of this backbone, SPARK-VLA keeps the original visual pathway frozen, augments it with a LoRA-based spatial branch, and introduces a lightweight knowledge router for selecting action-relevant visual tokens. We use VGGT [45] as the pre-trained 3D expert, which is kept frozen and used only during training for spatial representation distillation. We provide more details in Section A.1.

Table 2: Comparisons with state-of-the-art methods on LIBERO-Plus. Bold indicates the best performances.

Method	Suite	Camera SR (%)	Robot SR (%)	Language SR (%)	Light SR (%)	Background SR (%)	Noise SR (%)	Layout SR (%)	Total SR (%)	LLM Size	Train. Params.
Large-scale models ≥ 7B											
OpenVLA [20]	Avg.	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6	7B	111M
OpenVLA-OFT [19]	Avg.	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6	7B	111M
UniVLA [47]	Avg.	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9	7B	8B
1B ≤ Small-scale models < 7B											
π_0 [5]	Avg.	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6	2B	3.3B
π_0-FAST [38]	Avg.	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6	2B	3.3B
NORA [16]	Avg.	2.2	37.0	65.1	45.7	58.6	12.8	62.1	39.0	3B	4B
VLANeXt [50]	Avg.	**71.6**	**63.4**	**81.4**	93.7	91.8	89.5	77.7	**80.1**	2B	2.7B
Tiny-scale models < 1B											
VLA-Adapter [46]	Avg.	36.4	37.4	73.8	71.0	76.6	57.2	70.2	59.0	0.5B	598M
SPARK-VLA (**Ours**)	Avg.	68.5	59.8	72.7	**94.6**	**95.9**	**89.6**	**78.8**	78.6	0.5B	53M

Table 3: Comparisons with state-of-the-art methods on CALVIN benchmark. Bold indicates the best performance.

Method	Task	Tasks Completed in a Row					Avg. Len ↑	LLM Size	Train. Params.
		1	2	3	4	5			
MCIL [32]	ABCD→D	0.377	0.027	0.002	0.000	0.000	0.40	-	-
RT-1 [7]	ABCD→D	0.844	0.617	0.438	0.323	0.227	2.45	-	-
Robo-Flamingo [24]	ABCD→D	0.964	0.896	0.824	0.740	0.660	4.09	7B	1.4B
GR-1 [49]	ABCD→D	0.949	0.896	0.844	0.789	0.731	4.21	-	-
UP-VLA [55]	ABCD→D	0.962	0.921	0.879	0.842	0.812	4.42	-	1.5B
RoboVLMs [30]	ABCD→D	0.967	0.930	0.899	0.865	0.826	4.49	1.3B	1.7B
UniVLA [47]	ABCD→D	**0.985**	**0.961**	0.931	0.899	**0.851**	**4.63**	7B	8B
SPARK-VLA (**Ours**)	ABCD→D	0.979	0.960	**0.934**	**0.900**	0.843	4.61	0.5B	53M
MCIL [32]	ABC→D	0.304	0.013	0.002	0.000	0.000	0.31	-	-
Robo-Flamingo [24]	ABC→D	0.824	0.619	0.466	0.331	0.235	2.47	7B	1.4B
SuSIE [6]	ABC→D	0.870	0.690	0.490	0.380	0.260	2.69	-	-
GR-1 [49]	ABC→D	0.854	0.712	0.596	0.497	0.401	3.06	-	-
MoDE [40]	ABC→D	0.962	0.889	0.811	0.718	0.635	4.01	-	740M
UP-VLA [55]	ABC→D	0.928	0.865	0.815	0.769	0.699	4.08	1.3B	1.5B
RoboVLMs [30]	ABC→D	0.980	0.936	0.854	0.778	0.704	4.25	1.3B	1.7B
Seer [43]	ABC→D	0.963	0.916	0.861	0.803	0.740	4.28	-	315M
Dita [14]	ABC→D	0.945	0.825	0.728	0.613	0.500	3.61	-	221M
UniVLA [47]	ABC→D	**0.989**	0.948	**0.890**	0.828	**0.751**	4.41	7B	8B
SPARK-VLA (**Ours**)	ABC→D	0.985	**0.955**	0.888	**0.842**	0.750	**4.42**	0.5B	53M

4.2 Results in simulation environment

In this section, we compare our SPARK-VLA with state-of-the-art VLAs on simulation benchmarks. Please refer to Section A for additional analysis, including the effectiveness of each component of SPARK-VLA, the impact of hyperparameters, and visualization.

Results on LIBERO. Table 1 shows the performance evaluated on the LIBERO benchmark. We categorize the baselines according to the model size: *Large-scale models* with more than 7B parameters, *small-scale models* having parameters between 1B and 7B, and *tiny-scale models* with less than 1B parameters. In addition, we specify the LLM size and the number of trainable parameters. SPARK-VLA achieves an average success rate of 98.5% with only a 0.5B language backbone and 53M trainable parameters, which is the best average performance. Compared with VLA-Adapter [46], SPARK-VLA obtains the same average success rate while requiring about 11× fewer trainable parameters. Comparison with Evo-1 [27] shows SPARK-VLA improves the average success rate by 3.7% with 14× fewer trainable parameters, further demonstrating the effectiveness of the proposed method. In addition, SPARK-VLA outperforms larger models [54, 5, 39, 16, 41, 4, 50, 11, 20, 19, 47, 23, 44]

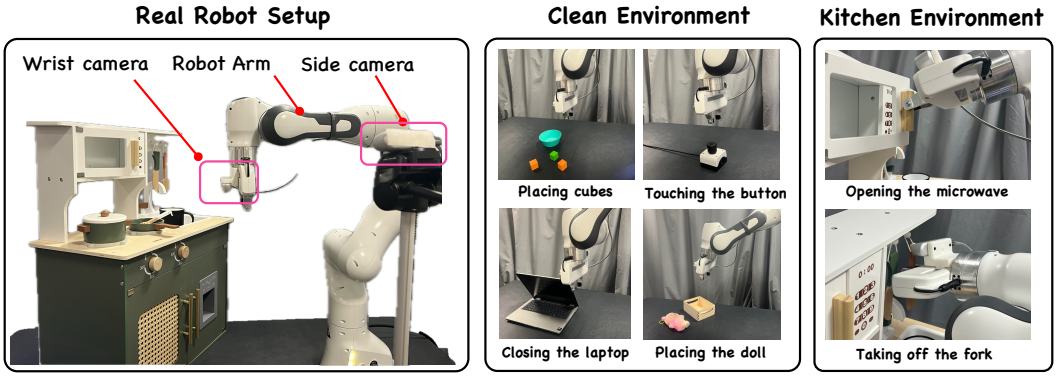


Figure 3: Overview of our real-world robot setup and tasks. We use a 7-DoF robotic arm with two cameras for manipulation. We evaluate six distinct tasks across a clean environment and a more complex kitchen environment.

despite using a substantially smaller language backbone and far fewer trainable parameters. Overall, the results suggest that SPARK-VLA achieves a favorable balance between model scale, trainable parameter efficiency, and task performance in the simulation environment.

Results on LIBERO-Plus. Evaluation on LIBERO-Plus measures the robustness of VLAs under diverse visual perturbations. As shown in Table 2, SPARK-VLA achieves an average success rate of 78.6%, substantially outperforming the tiny-scale baseline (*i.e.*, VLA-Adapter [46]) by 19.6%. Moreover, SPARK-VLA is competitive with larger models and achieves the best performance under light, background, noise, and layout perturbations. Compared with the performance on LIBERO, the drop is especially small under background and light perturbations, suggesting that the proposed knowledge router effectively suppresses task-irrelevant or noisy visual tokens while preserving action-critical information. Although VLANeXt [50] achieves a slightly higher total performance, it requires a 2B language backbone and 2.7B trainable parameters, while our SPARK-VLA provides comparable robustness with more efficient model configurations. We provide per-task and per-perturbation results with additional analysis in Section A.5.

Results on CALVIN. In Table 3, the performance on ABCD→D shows a general long-horizon language-conditioned manipulation capability, *i.e.*, training the model on the four different environments and evaluating on the environment D. In contrast, the performance on ABC→D provides the zero-shot evaluation setting in which the models are trained on the environments A, B, and C, and evaluated on the new environment D. In ABCD→D, SPARK-VLA achieves an average length of 4.61, which is comparable to UniVLA [47] with 4.63, while obtaining the best success rates for completing three and four tasks in a row. In the more challenging ABC→D setting, SPARK-VLA achieves the best average length of 4.42 and the best success rates for two- and four-task completion. The results show that SPARK-VLA provides strong long-horizon manipulation and zero-shot environment generalization with only a 0.5B language model and 53M trainable parameters.

5 Real-world Experiments

5.1 Experimental setting

Setup. To evaluate the effectiveness of SPARK-VLA in the real world, we design two scenarios with different levels of complexity. The first scenario consists of four manipulation tasks with a clean background: (1) *placing cubes*, (2) *closing the laptop*, (3) *placing the doll*, and (4) *touching the button*. These tasks evaluate basic manipulation skills, including pick-and-place and articulated-object interaction. We further construct a more challenging kitchen scenario with cluttered objects and task-relevant background distractors. As shown in Figure 3, the robot must localize target objects among distractors and perform spatial reasoning from dual-camera observations. This scenario includes two tasks: (1) *opening the microwave* and (2) *taking off the fork*.

We use a 7-DoF Franka Emika Panda robot with a two-finger gripper equipped with a wrist camera and a side camera for visual observation. We train the model separately for the two scenarios using

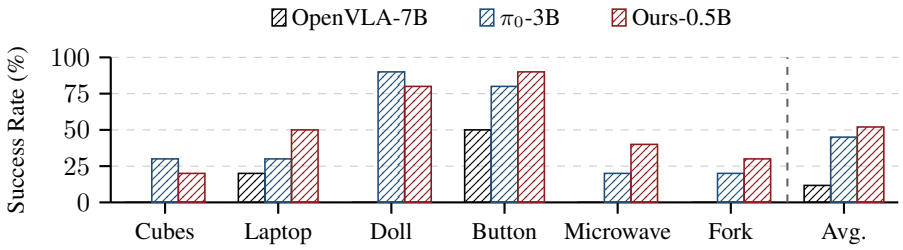


Figure 4: Performance comparison on real-world tasks.

a total of 150 human-collected demonstration trajectories recorded at 15Hz. Our SPARK-VLA is initialized from the pre-trained Qwen2.5VL [2]. Since this model is not originally trained on robotics-specific data, we follow the training protocol of π_0 [5] and VLANeXt [50]: we first pre-train the model on the DROID dataset [18] for 100k steps, and then fine-tune on our real-world demonstrations with LoRAs for 20k steps. During evaluation, we measure the performance on each task over 10 episodes and report the average success rate.

Baselines. We compare our SPARK-VLA with two standard VLA baselines:

- **OpenVLA** [20] is a standard 7B VLA model that predicts actions through direct action regression. We include OpenVLA as a representative large-scale VLA baseline, as it has shown strong performance on simulated manipulation benchmarks [29].
- π_0**-Droid** [5] is a 3B VLA model that combines a vision-language backbone with a flow-matching action expert for continuous control. We use the publicly available checkpoint that was finetuned on DROID [18], which contains Franka robot demonstrations and is therefore well-aligned with our real-world setup.

5.2 Results in real-world environments

Figure 4 reports the real-world manipulation performance across six tasks. OpenVLA-7B shows limited transfer to our real-world setup, achieving an average success rate of 11.7%. The stronger π_0-3B baseline improves the average success rate to 45.0%, especially on relatively clean tasks such as *placing the doll* and *touching a button*. In comparison, SPARK-VLA achieves the best average success rate of 52.0% with only a 0.5B language backbone. Notably, SPARK-VLA improves over π_0-3B on four out of six tasks, including *closing the laptop*, *touching a button*, *opening the microwave*, and *taking off the knife*. The gains on the kitchen tasks are particularly meaningful, as they require target localization under cluttered backgrounds and spatial reasoning from dual-camera observations. These results demonstrate that the proposed spatial representation distillation and knowledge routing are effective not only in simulation but also in real-world manipulation.

6 Conclusion

We presented SPARK-VLA, a simple and effective framework for enhancing spatial understanding in vision-language-action models. Instead of relying on explicit 3D inputs at inference time or directly reshaping the original visual representation, SPARK-VLA preserves the pre-trained visual pathway and introduces a LoRA-adapted spatial branch learned from a frozen 3D expert through attention response distillation. To effectively integrate generic and spatial cues, we further introduced a lightweight knowledge router that selects instruction- and action-relevant tokens from both branches using gradient-derived supervision. Extensive experiments on simulated and real-world manipulation benchmarks demonstrate that SPARK-VLA achieves strong performance with a compact 0.5B language backbone and a small number of trainable parameters, showing that spatial knowledge can be effectively transferred to VLAs without sacrificing deployment simplicity.

Broader impact Our work contributes toward more efficient and spatially aware robot policies that can operate without additional 3D sensors during inference. This can reduce hardware dependency and make VLA-based robot learning more accessible across different platforms. By improving

spatial reasoning with a compact model, SPARK-VLA may also help lower the computational cost of deploying embodied agents in practical environments. At the same time, more capable robotic systems should be deployed with careful consideration of safety, reliability, and human oversight, especially in environments involving people or fragile objects.

Limitations Despite its effectiveness, SPARK-VLA still relies on a pre-trained 3D expert during training, and the quality of spatial distillation may depend on the teacher model and the visual diversity of the training data. In addition, the router is trained with pseudo supervision derived from action-loss gradients, which can be noisy when the action model is not sufficiently optimized. We mitigate this issue with warm-up training and momentum-based score accumulation, but more principled routing supervision remains an interesting direction. Finally, although SPARK-VLA shows promising real-world performance, our real-world evaluation is still limited in task diversity and scale. Extending the framework to broader manipulation scenarios, more diverse embodiments, and dynamic human-centered environments is an important future direction.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2025-00515741) and the AI Seoul Tech Research Support Program of the Seoul Future Foundation.

References

[1] Ali Abouzeid, Malak Mansour, Qinbo Sun, Zezhou Sun, and Dezhen Song. Geoaware-vla: Implicit geometry aware vision-language-action model. *arXiv preprint arXiv:2509.14117*, 2025.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[3] Vineet Bhat, Yu-Hsiang Lan, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami. 3d cavla: Leveraging depth and 3d context to generalize vision language action models for unseen tasks. *arXiv preprint arXiv:2505.05800*, 2025.

[4] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

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

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

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

[10] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[11] Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. In *RSS*, 2024.

[12] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3d object manipulation. In *CoRL*, 2023.

[13] Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. Epipolar transformers. In *CVPR*, 2020.

[14] Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *ICCV*, 2025.

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

[16] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

[17] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *ICML*, 2024.

[18] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[19] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *RSS*, 2025.

[20] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[21] Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025.

[22] Wei Li, Renshan Zhang, Rui Shao, Zhijian Fang, Kaiwen Zhou, Zhuotao Tian, and Liqiang Nie. Semanticvla: Semantic-aligned sparsification and enhancement for efficient robotic manipulation. In *AAAI*, 2026.

[23] Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. In *NeurIPS*, 2025.

[24] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. In *ICLR*, 2024.

[25] Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shutao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration. *arXiv preprint arXiv:2506.12723*, 2025.

[26] Tao Lin, Gen Li, Yilei Zhong, Yanwen Zou, Yuxin Du, Jiting Liu, Encheng Gu, and Bo Zhao. Evo-0: Vision-language-action model with implicit spatial understanding. *arXiv preprint arXiv:2507.00416*, 2025.

[27] Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, et al. Evo-1: Lightweight vision-language-action model with preserved semantic alignment. In *CVPR*, 2026.

[28] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[29] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

[30] Huaping Liu, Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, and Hanbo Zhang. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2025.

[31] Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Vla-pruner: Temporal-aware dual-level visual token pruning for efficient vision-language-action inference. *arXiv preprint arXiv:2511.16449*, 2025.

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

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

[34] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *TMLR*, 2024.

[35] Abby OâĂŹNeill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *ICRA*, 2024.

[36] Jungin Park, Jiyoung Lee, and Kwanghoon Sohn. V-lynx: Token interface alignment for video+x llms. In *ICML*, 2026.

[37] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[38] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. In *RSS*, 2025.

[39] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[40] Moritz Reuss, Jyothish Pari, Pulkit Agrawal, and Rudolf Lioutikov. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning. *arXiv preprint arXiv:2412.12953*, 2024.

[41] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[42] Lin Sun, Bin Xie, Yingfei Liu, Hao Shi, Tiancai Wang, and Jiale Cao. Geovla: Empowering 3d representations in vision-language-action models. *arXiv preprint arXiv:2508.09071*, 2025.

[43] Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *ICLR*, 2025.

[44] Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*, 2025.

[45] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025.

[46] Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *AAAI*, 2026.

[47] Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *ICLR*, 2026.

[48] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

[49] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *ICLR*, 2024.

[50] Xiao-Ming Wu, Bin Fan, Kang Liao, Jian-Jian Jiang, Runze Yang, Yihang Luo, Zhonghua Wu, Wei-Shi Zheng, and Chen Change Loy. Vlanext: Recipes for building strong vla models. *arXiv preprint arXiv:2602.18532*, 2026.

[51] Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. In *NeurIPS*, 2025.

[52] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018.

[53] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023.

[54] Jiahui Zhang, Yurui Chen, Yueming Xu, Ze Huang, Yanpeng Zhou, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, Xingyue Quan, Hang Xu, et al. 4d-vla: Spatiotemporal vision-language-action pretraining with cross-scene calibration. In *NeurIPS*, 2025.

[55] Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. Up-vla: A unified understanding and prediction model for embodied agent. In *ICML*, 2025.

[56] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *CVPR*, 2025.

[57] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[58] Zhongyi Zhou, Yichen Zhu, Xiaoyu Liu, Zhibin Tang, Junjie Wen, Yaxin Peng, Chaomin Shen, and Yi Xu. Chatvla-2: Vision-language-action model with open-world reasoning. In *NeurIPS*, 2025.

[59] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023.

Appendix

A Simulation Environment

Table A.1: Hyperparameters for simulated experiments.

Hyperparameter	Value
GPUs	NVIDIA RTX A6000 48GB $\times$ 4
Batch size per GPU	16
Initial learning rate / Final learning rate	5e-5 / 5e-8
Warm-up with Equation (13)	2k steps
Learning rate scheduler	Cosine annealing lr scheduler
Input images	1 exo-view + 1 wrist view
Image size	224×224
LoRA rank	32
Use proprioception	Yes
Use FiLM	No
Trainable parameters	53M total: 23.2M (LLM LoRAs) + 12.6M (vision LoRAs) + 4.8M (vision proj.) + 4.3M (knowledge router) + 7.3M (action head) + 0.8M (proprio. proj.)
Momentum parameter β in Equation (14)	0.9
λ_{attn}, λ_{router}, λ_{bud}, λ_{ent}	1.0, 0.5, 0.05, 0.001

A.1 Default setting

Table A.1 shows a default setting used in our experiments.

A.2 Training regime

Warm-up and joint optimization. Reliable router supervision requires meaningful spatial representations and a reasonably trained action prediction head. We therefore adopt a two-stage training procedure. In the first stage, we warm up the model without the router objective:

$$\mathcal{L}_{\text{warm}} = \mathcal{L}_{\text{act}} + \lambda_{\text{attn}}\mathcal{L}_{\text{attn}}. \tag{13}$$

This stage trains the LoRA-adapted spatial branch with the attention response distillation loss while optimizing the action head with the action loss. After warm-up, we enable the knowledge router and optimize the full objective in Equation (12). In this stage, the router is trained with pseudo targets derived from action-loss gradients, while the spatial branch continues to receive distillation supervision from the frozen 3D expert. This procedure stabilizes routing training by first establishing geometry-aware visual features and meaningful action gradients, and then learning to select tokens that are useful for action generation.

Inference without 3D experts. At inference time, the pre-trained 3D expert is completely removed. Given an RGB observation $\mathbf{I}_t$ and a language instruction, SPARK-VLA extracts the generic visual representation $\mathbf{Z}_v$ from the original visual encoder and the spatial representation $\mathbf{Z}_p$ from the LoRA-adapted branch. The knowledge router predicts token-wise gate scores conditioned on the visual and language features, selects the top-ranked tokens according to the target keeping ratio, and fuses the selected generic and spatial tokens before forwarding them to the language model for action prediction. Thus, SPARK-VLA benefits from spatial knowledge distilled during training while requiring neither depth maps, point clouds, nor external 3D experts at inference time.

A.3 Component analysis

In this section, we further analyze the impact of each component in our SPARK-VLA, including LoRA layers in the LLM and the vision encoder, projector to connect the visual tokens to the LLM, the usage of the vision encoder, attention distillation, and knowledge routing. In Table A.2, we compare the performance and the number of trainable parameters according to each component. Note that the action head with ℓ_1 regression is identically trained on all cases unless we specify.

Baseline. We start from the simplest LoRA tuning by applying the LoRA layers only to the LLM in Prismatic Qwen2.5-0.5B [2]. The results in Table A.2-(a) show that applying the LoRAs only to the LLM is insufficient to adapt the pre-trained VLMs to manipulation, showing less than 5% of the average success rate.

LoRAs to the vision encoder. We apply the additional LoRA layers to the vision encoder, *i.e.*, DINOv2 [34] and SigLIP [53], managing about 28.7M additional trainable parameters. Similar to the baseline, Table A.2-(b) suggests applying the LoRAs to the model fails to adapt the pre-trained VLM to the manipulation tasks.

Projector. In addition to the LoRAs, training the projector (with 4.8M parameters) to project the visual token embeddings to the LLM's input space yields meaningful results, showing 72.4% of the average success rate. Note that we train a new projector from scratch instead of re-using the projector in the pre-trained VLM.

Usage of vision encoder. In the original prismatic VLM, two vision encoders are employed to complement with each other for two different types of visual information. Specifically, DINOv2 [34] and SigLIP [53] provide localization-oriented and language-aligned visual information, respectively [17]. Therefore, removing SigLIP from the pre-trained VLM weakens its language-aligned semantic grounding. In practice, the model without SigLIP performs worse than using both vision encoders, suggesting two complementary information is essential for instruction following manipulation, as shown in Table A.2-(d).

Attention distillation. We transfer the spatial representations using attention response distillation from VGGT [45] to the LoRA-adapted path in the DINOv2. Table A.2-(e) shows that injecting spatial knowledge into the pre-trained vision encoder significantly improves the performance of (d) and even outperforms (c), which employs two vision encoders, with only a single vision encoder. This result suggests the consistent conclusion with the recent 3D VLAs [27, 21].

Knowledge routing. While attention response distillation enables the model to produce the generic visual and spatial representations, redundant or noisy information may be included. Our full model with the knowledge router trained using the action loss-driven supervision feeds forward the instruction- and action-relevant visual tokens to the LLM, yielding the best performance, as shown in Table A.2-(f).

Consequently, all components in our SPARK-VLA are carefully designed to complement each other, requiring minimum additional trainable parameters.

Objectives. We now replace ℓ_1 regression with alternatives for action generation, *i.e.*, the OpenVLA-style next-token prediction and the π_0-style diffusion policy. Compared to regression, next-token prediction reformulates action generation as discrete token modeling, making it naturally compatible with autoregressive language models. With next token prediction, the performance can be affected by action discretization errors and the limited resolution of the action vocabulary.

Diffusion policy, on the other hand, directly models continuous action trajectories and can capture multi-modal distributions through an iterative denoising process, which is beneficial for ambiguous or long-horizon manipulation, but introduces additional sampling cost at inference time.

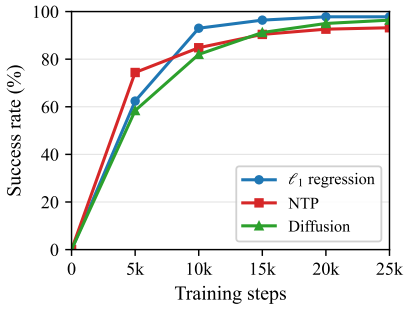


Figure A.1: Performance according to the action generation objective. NTP and Diffusion denote next token prediction and diffusion policy, respectively.

In Figure A.1, we depict the performance evaluated on LIBERO-Goal according to the training steps. We evaluate the model every 5k training steps and report the success rate (%). Next-token prediction shows stronger early learning, achieving 74.4% at 5k iterations, thanks to its autoregressive nature, while the performance saturates at 93.2%. Diffusion policy starts lower at 58.4%, but improves steadily as training proceeds and reaches 96.4% at 25k iterations, benefiting from continuous trajectory modeling and its ability to capture multi-modal action distributions. Overall, ℓ_1 regression converges fastest and achieves the best final performance in our setting, while next-token prediction favors early optimization and diffusion provides stronger late-stage continuous control.

Table A.2: Ablation study for each component in our SPARK-VLA.

Method	Spatial SR (%)	Object SR (%)	Goal SR (%)	Long SR (%)	Average SR (%)	Train. Params.
(a) only LLM LoRAs	-	-	-	-	< 5%	31.4M
(b) + Vision LoRAs	-	-	-	-	< 5%	60.1M
(c) + Projector	82.6	84.4	77.4	45.0	72.4	64.9M
(d) - SigLIP	62.4	76.8	58.4	37.2	58.7	48.8M
(e) + Attention Distill.	95.8	96.0	94.0	90.6	94.1	48.8M
(f) + Knowledge Router	99.6	99.6	98.0	96.6	98.5	53.1M

Table A.3: Comparison with different spatial supervision strategies on LIBERO-Goal.

Spatial supervision	Success rate (%)
Action loss only	81.6
Feature matching	94.8
Relational matching	95.6
Attention-map matching	96.4
SPARK-VLA **(ours)**	**98.0**

Spatial supervision. We investigate different strategies for transferring geometric knowledge from the pre-trained 3D expert to the spatial pathway. Specifically, we consider four alternatives to our attention-response distillation: (1) action-only adaptation; (2) feature matching; (3) relational matching; and (4) attention-map matching. Action-only adaptation learns spatial representations solely from the downstream action objective, without explicit geometric supervision. Feature matching directly aligns the student and teacher embeddings, while relational matching transfers pairwise similarities between visual tokens. Attention-map matching instead aligns the query-key attention distributions, capturing where spatial information is aggregated. Unlike these approaches, our attention response distillation additionally transfers the teacher's aggregated value representations, providing geometry-aware contextual information conditioned on the VLA queries.

We compare these objectives on LIBERO-Goal while keeping the remaining architecture and training configurations unchanged. As shown in Table A.3, action-only adaptation achieves a success rate of 81.6%, whereas feature and relational matching improve the performance to 94.8% and 95.6%, respectively. Attention-map matching further increases the success rate to 96.4%, indicating the benefit of transferring query-conditioned relationships beyond static feature alignment. Our attention-response distillation achieves the highest success rate of 98.0%, outperforming feature matching by 3.2% and attention-map matching by 1.6%. These results suggest that transferring both the attention distribution and its aggregated spatial content is more effective than supervising either latent representations or attention patterns alone.

Knowledge fusion. We explore different strategies for integrating generic visual representations with spatially specialized features. Specifically, we compare our action-driven knowledge router with two alternative fusion mechanisms: direct concatenation and feature-wise linear modulation (FiLM) [37]. Direct concatenation retains all tokens from both pathways and combines their representations without distinguishing their relevance to the downstream task. FiLM-based fusion instead uses the generic features to predict channel-wise affine transformations of the spatial features, allowing the two representations to interact through feature modulation. However, neither approach explicitly determines which knowledge source is more useful at each spatial position. In contrast, our knowledge router selects tokens from the generic and spatial pathways according to their estimated contribution to action prediction, enabling task-dependent integration of complementary visual information.

As shown in Table A.4, direct concatenation achieves a success rate of 94.0%, while FiLM-based fusion slightly improves the performance to 94.8%. Our action-driven router further increases the success rate to 98.0%, outperforming the two alternatives by 4.0 and 3.2 percentage points, respectively. Interestingly, combining all available features does not necessarily lead to better action prediction, suggesting that generic and spatial representations may contain redundant or task-irrelevant

Table A.4: Comparison with different knowledge fusion strategies on LIBERO-Goal.

Fusion strategy	Success rate (%)
Direct all-token concatenation	94.0
FiLM-based branch fusion	94.8
Action-driven knowledge router (ours)	**98.0**

information. These results highlight the importance of selectively integrating the two knowledge sources, rather than uniformly combining their representations.

A.4 Hyperparameters

In this section, we provide additional analysis corresponding to the hyperparameters in our SPARK-VLA. Note that all experiments in this section are conducted on the LIBERO-Goal suite.

Momentum-based moving average. In our SPARK-VLA, only routed tokens are forwarded to the LLM, making tokens that are not selected in the current training step cannot directly affect the action loss and thus do not provide reliable instantaneous gradient signals. In other words, the supervision would be biased toward the currently routed subset, as the routing targets were derived only from the current gradients. To mitigate this issue, we maintain a momentum-based moving average of the token importance scores:

$$\bar{s}_n \leftarrow \beta \bar{s}_n + (1-\beta) s_n, \tag{14}$$

where β is the momentum coefficient and $s_n = \mathrm{ReLU}(-\partial \mathcal{L}_{\mathrm{act}} / \partial p_n)$ is the instantaneous importance score. The accumulated score $\bar{s}_n$ provides a stable estimate of token utility across training steps, allowing previously informative tokens to remain considered even when they are not selected in the current step. We use $\bar{\mathbf{s}} = \{\bar{s}_n\}_{n=1}^{N}$ to construct the routing target by top-k selection.

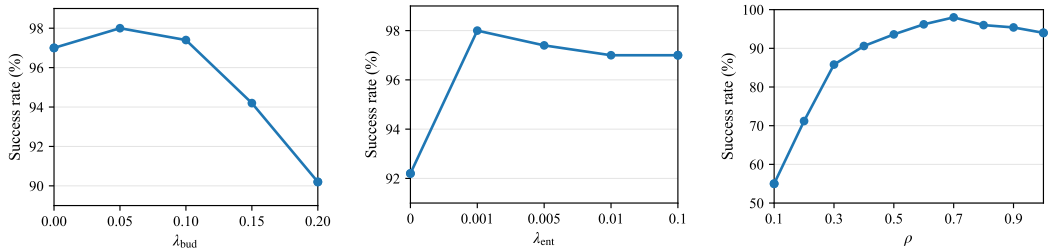
Table A.5: Performance according to the momentum parameter β on LIBERO-goal.

Momentum	0.0	0.9	0.99	0.999	0.9999
Avg. SR (%)	94.0	98.0	97.6	97.2	97.0

In Table A.5, we compare the performance evaluated on the LIBERO-Goal suite according to the value of the momentum parameter β. With $\beta = 0.0$, the router is trained using only the instantaneous importance scores, resulting in a lower success rate of 94.0%. This indicates that routing supervision becomes unstable when it depends solely on the gradients from the currently selected tokens. Introducing momentum substantially improves performance, achieving the best success rate of 98.0% with $\beta = 0.9$. This demonstrates that accumulating the token importance scores across training steps provides more reliable routing targets. However, overly large momentum values slightly degrade performance, as the accumulated scores are updated too slowly and may not sufficiently reflect the current policy and scene distribution. We therefore use $\beta = 0.9$ as the default setting.

Coefficient λ_{bud}. The budget loss encourages the average router gate probability to match the target routing ratio ρ. In Figure A.2a, we evaluate different values of λ_{bud} from 0.0 to 0.2 with an interval of 0.05 on the LIBERO-Goal suite. As λ_{bud} increases, the optimization places more emphasis on satisfying the global routing budget, which reduces the flexibility of token selection. As a result, the router is encouraged to match the average gate probability to ρ rather than to preserve locally important tokens for each instruction and action. This overly strict budget constraint suppresses useful deviations that could otherwise be controlled by entropy regularization, leading to degraded performance. In particular, increasing λ_{bud} to 0.2 reduces the success rate to 90.2%.

Coefficient λ_{ent}. The entropy loss regularizes the router to prevent gate probability collapse and maintain adaptive token selection. We report the performance with different values of λ_{ent} in Figure A.2b. When $\lambda_{\text{ent}} = 0$, the router is regularized only by the budget loss, resulting in a lower success rate of 92.2%, as discussed above. Introducing entropy regularization stabilizes routing by preventing early saturation of gate probabilities and improving robustness to noisy pseudo targets from gradient-based supervision. As λ_{ent} increases, the entropy term encourages more confident,



(a) Effectiveness of λ_{budget} (b) Effectiveness of λ_{ent} (c) Effectiveness of ρ

Figure A.2: Ablation study for each hyperparameter. We report the performance evaluated on LIBERO-Object according to the values of (a) the budget coefficient λ_{bud}, (b) the entropy coefficient λ_{ent}, and the target keeping ratio ρ.

Table A.6: Per-task and per-perturbation performance evaluated on LIBERO-Plus. Bold indicates the best performances.

Method	Suite	Camera SR (%)	Robot SR (%)	Language SR (%)	Light SR (%)	Background SR (%)	Noise SR (%)	Layout SR (%)	Total SR (%)	LLM Size	Train. Params.
Large-scale models $\geq$ 7B											
OpenVLA-OFT [19]	Spatial	88.3	40.0	80.5	98.3	97.3	96.3	93.9	84.0	7B	111M
	Object	38.9	25.4	99.0	73.7	97.6	72.3	71.8	66.5		
	Goal	62.0	25.2	53.2	**93.9**	**92.5**	75.2	59.1	63.0		
	Long	38.7	38.2	87.0	89.4	86.8	63.5	76.9	66.4		
	Avg.	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6		
1B $\leq$ Small-scale models $<$ 7B											
VLANeXt [50]	Spatial	**94.4**	**83.4**	**86.7**	**99.3**	98.4	98.0	92.2	**92.8**	2B	2.7B
	Object	**73.2**	**67.8**	**84.2**	98.3	94.8	92.2	73.0	**82.1**		
	Goal	**71.1**	**50.6**	**66.3**	92.5	81.9	**90.2**	66.8	**72.7**		
	Long	47.7	**51.9**	88.5	84.7	92.0	77.5	78.8	72.8		
	Avg.	**71.6**	**63.4**	**81.4**	93.7	91.8	89.5	77.7	**80.1**		
Tiny-scale models $<$ 1B											
SPARK-VLA (**Ours**)	Spatial	92.6	82.0	76.9	**99.3**	**99.2**	**98.3**	**93.2**	90.1	0.5B	53B
	Object	69.4	63.6	78.5	**99.3**	**98.4**	**94.3**	**75.2**	81.3		
	Goal	66.2	45.7	58.5	84.2	85.4	**90.2**	**68.2**	70.3		
	Long	**48.2**	50.9	78.1	**94.9**	**94.8**	**78.0**	**80.1**	**72.9**		
	Avg.	68.5	59.8	72.7	**94.6**	**95.9**	**89.6**	**78.8**	78.6		

near-binary gate decisions. This sharpening is beneficial at moderate levels, but an overly large λ_{ent} makes the routing decisions too rigid and reduces the flexibility of token selection during training. We obtain the best performance with $\lambda_{\text{ent}} = 0.001$.

Target keeping ratio ρ. The target keeping ratio ρ controls the proportion of visual tokens forwarded to the LLM. The model completely fails when $\rho = 0.0$, since no visual information is provided for action prediction. As shown in Figure A.2c, increasing ρ consistently improves performance from 55.0% at $\rho = 0.1$ to 96.2% at $\rho = 0.6$, indicating that sufficient visual context is necessary for reliable manipulation. The best performance is achieved at $\rho = 0.7$, reaching 98.0%, which suggests that preserving a moderately large but selective set of tokens provides the best balance between visual sufficiency and noise suppression. When ρ is further increased, the success rate gradually decreases to 96.0%, 95.4%, and 94.0% at $\rho = 0.8$, 0.9, and 1.0, respectively. This drop shows that forwarding excessive tokens weakens the effect of routing, as redundant or instruction-irrelevant representations can interfere with action prediction.

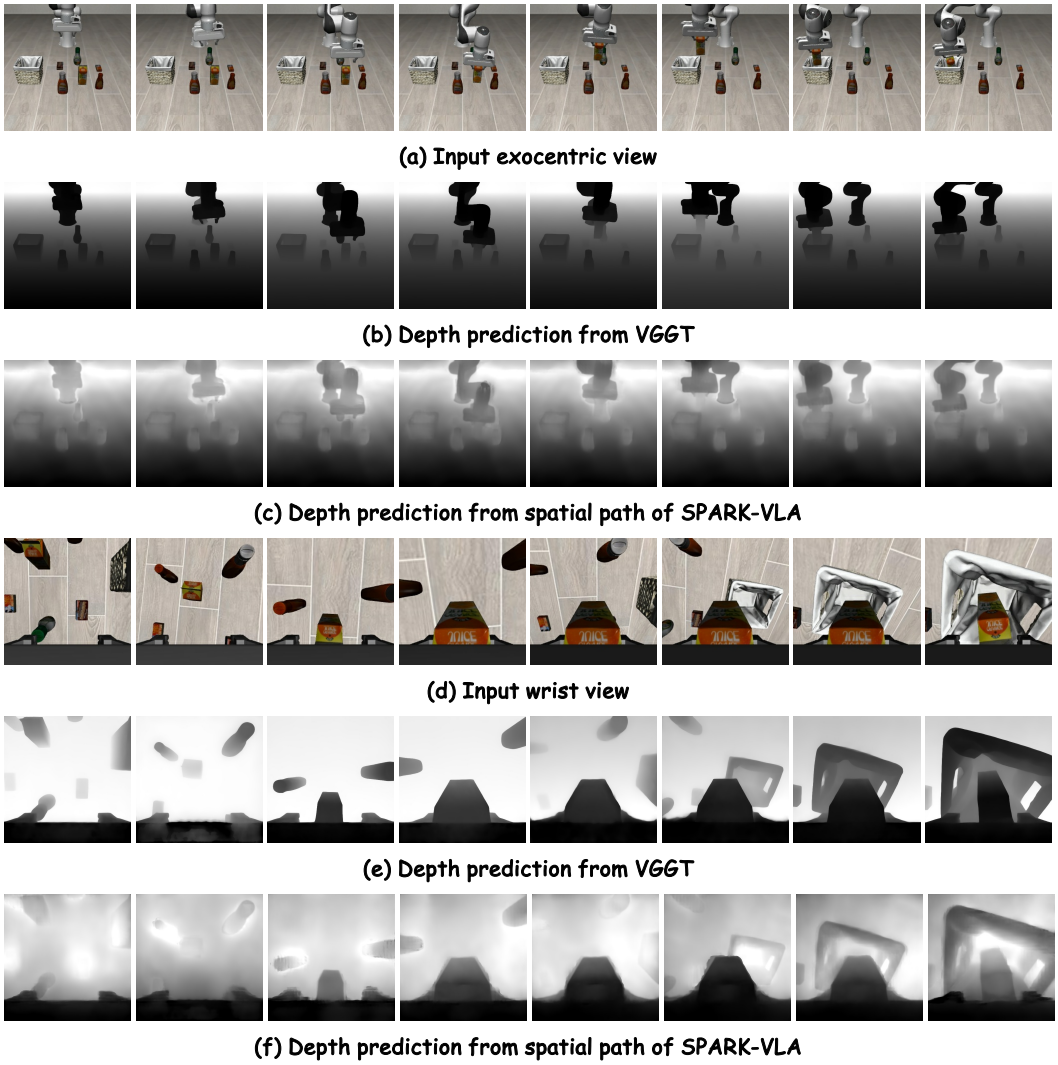


Figure A.3: Qualitative analysis of learned spatial representations. From given exocentric and wrist view observations, we extract depth maps using the original VGGT and the spatial path in our SPARK-VLA.

A.5 Additional analysis on LIBERO-Plus

Table A.6 provides per-task and per-perturbation results on LIBERO-Plus. Overall, SPARK-VLA achieves strong robustness across all four suites with only a 0.5B language backbone and 53M trainable parameters. On the Spatial suite, SPARK-VLA obtains 90.1% total success rate and achieves the best performance under light, background, noise, and layout perturbations. This suggests that the proposed spatial representation distillation and knowledge routing are particularly effective when manipulation requires robust spatial reasoning under visual changes.

On the Object and Goal suites, SPARK-VLA remains competitive with larger models, although VLANeXt or OpenVLA-OFT shows stronger performance under some camera, robot, and language perturbations. In contrast, SPARK-VLA consistently performs well under appearance-level perturbations such as light, background, and noise. For example, on the Object suite, SPARK-VLA achieves the best success rates under light, background, noise, and layout perturbations, despite using substantially fewer trainable parameters than VLANeXt. On the Long suite, SPARK-VLA achieves the best total success rate of 72.9%, slightly outperforming VLANeXt, and obtains the best performance under camera, light, background, noise, and layout perturbations. These results indicate that selectively routing task-relevant generic and spatial tokens is especially beneficial for

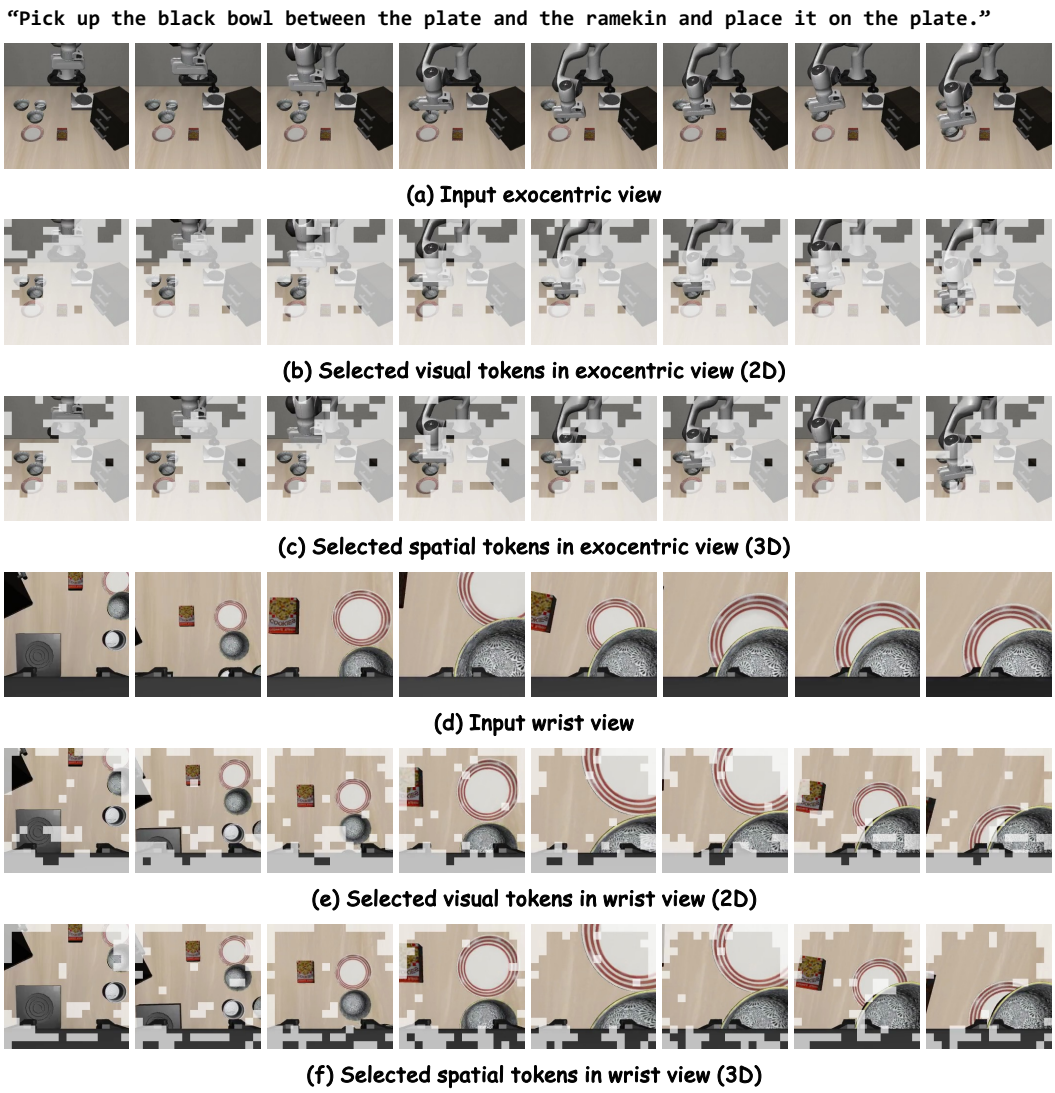


Figure A.4: Qualitative analysis of selected tokens. From given exocentric and wrist view observations, the knowledge router in our SPARK-VLA selectively forwards instruction- and action-relevant visual (2D) and spatial (3D) tokens to the LLM. We mask out the tokens, which are not selected in the router.

long-horizon manipulation, where irrelevant visual changes can accumulate errors across multiple action steps.

In summary, while larger models can be more robust to certain perturbation types such as language and robot appearance changes, SPARK-VLA shows strong robustness to visual perturbations that affect scene appearance and spatial layout. This supports our main observation that knowledge routing helps suppress task-irrelevant visual variations while preserving action-critical evidence.

A.6 Visualization

Does spatial representation distillation indeed transfer 3D information into the vision encoder? We present a new implicit adaptation strategy to inject spatial knowledge to the pre-trained vision encoder. To examine this question, we use VGGT [45] as a diagnostic model: we replace its original DINOv2 encoder with our LoRA-adapted vision encoder, while keeping the remaining components of VGGT fixed, and predict depth maps from the resulting representations. If the adapted encoder has acquired spatial knowledge, the modified VGGT should still recover meaningful scene geometry

despite the encoder replacement. As shown in Figure A.3, the predicted depth maps preserve the global scene structure and object boundaries, demonstrating that the adapted encoder captures geometry-aware representations through the proposed attention response distillation. This suggests that spatial representation distillation injects useful 3D understanding into the vision encoder, rather than merely improving action prediction through task-specific adaptation.

Which tokens are preserved and excluded in the knowledge router? We further analyze the behavior of the proposed knowledge router to examine which visual tokens are preserved for action generation. Specifically, we visualize (a) the original exocentric video and the selected tokens from the exocentric view in (b) the original visual and (c) spatial branches. We also visualize (d) the original wrist view video and the selected tokens from the wrist view in (e) the original visual and (f) spatial branches. As shown in Figure A.4, the router exhibits view-dependent selection behavior. In the exocentric view, it preserves only a sparse subset of tokens around task-relevant objects, interaction regions, and spatial context, while discarding large background areas. This is because the exocentric camera provides a wide and relatively stable scene-level view, where many tokens correspond to static background or redundant global context.

In contrast, the router selects a denser set of tokens from the wrist view. Unlike the exocentric view, the wrist camera is closely coupled with the end-effector and directly observes fine-grained manipulation cues, such as object pose, gripper-object contact, local geometry, and occlusion changes. Therefore, a larger portion of wrist view tokens can contribute to action prediction, especially when precise control depends on local spatial evidence near the manipulated object. This indicates that the router does not simply prune tokens uniformly across views, but adaptively allocates the visual budget according to action utility. Across both views, the original branch tends to preserve semantically informative regions, while the spatial branch emphasizes geometrically relevant areas around objects and contact regions. These visualizations suggest that knowledge routing selectively balances sparse contextual evidence from the exocentric view with dense local manipulation evidence from the wrist view.

B Real-World Environment

We present demos of our real-world experiments for the six tasks mentioned in the main paper, showing the execution process in Figure B.5. We also provide addtional real-world experiments and analysis in the following sections.

B.1 Additional real-world experiments

To further assess the effectiveness of SPARK-VLA across a broader range of manipulation tasks, we extend the real-robot evaluation with four additional kitchen manipulation tasks: (1) *stacking bowls*, (2) *pouring water into the sink*, (3) *placing a pan in the sink*, and (4) *picking up a rag*. These tasks involve multi-object spatial alignment, container manipulation, precise placement, and deformable-object interaction, expanding our real-world evaluation from six tasks to ten. For these additional experiments, we use a different hareware of Robotiq two-finger gripper and fine-tune both methods for 30k steps, following a training protocol similar to that described in the main paper. Each task is evaluated over 10 trials per method.

As shown in the Figure B.6, SPARK-VLA achieves an average success rate of 65.0%, compared with 55.0% for π_0-3B, with gains of 20% on both stacking bowls and placing the pan in the sink, which require more precise reasoning about relative object position, orientation, and target geometry.

B.2 Generalization to unseen objects and language instructions

To test whether VGGT-based spatial distillation contributes to semantic generalization, we conducted additional real-world experiments to evaluate SPARK-VLA under previously unseen objects and language instructions. We compare SPARK-VLA with the same architecture trained without VGGT-based spatial distillation, denoted as SPARK w/o VGGT. We evaluate the same pick-and-place behavior-grasping a target object and placing it in the sing using one object (seen during training) and five unseen objectrs. The unseen objects vary in color, shape, handle structure, and semantic category. For reproducibility, each object was tested at the same five pre-defined starting locations. Across all

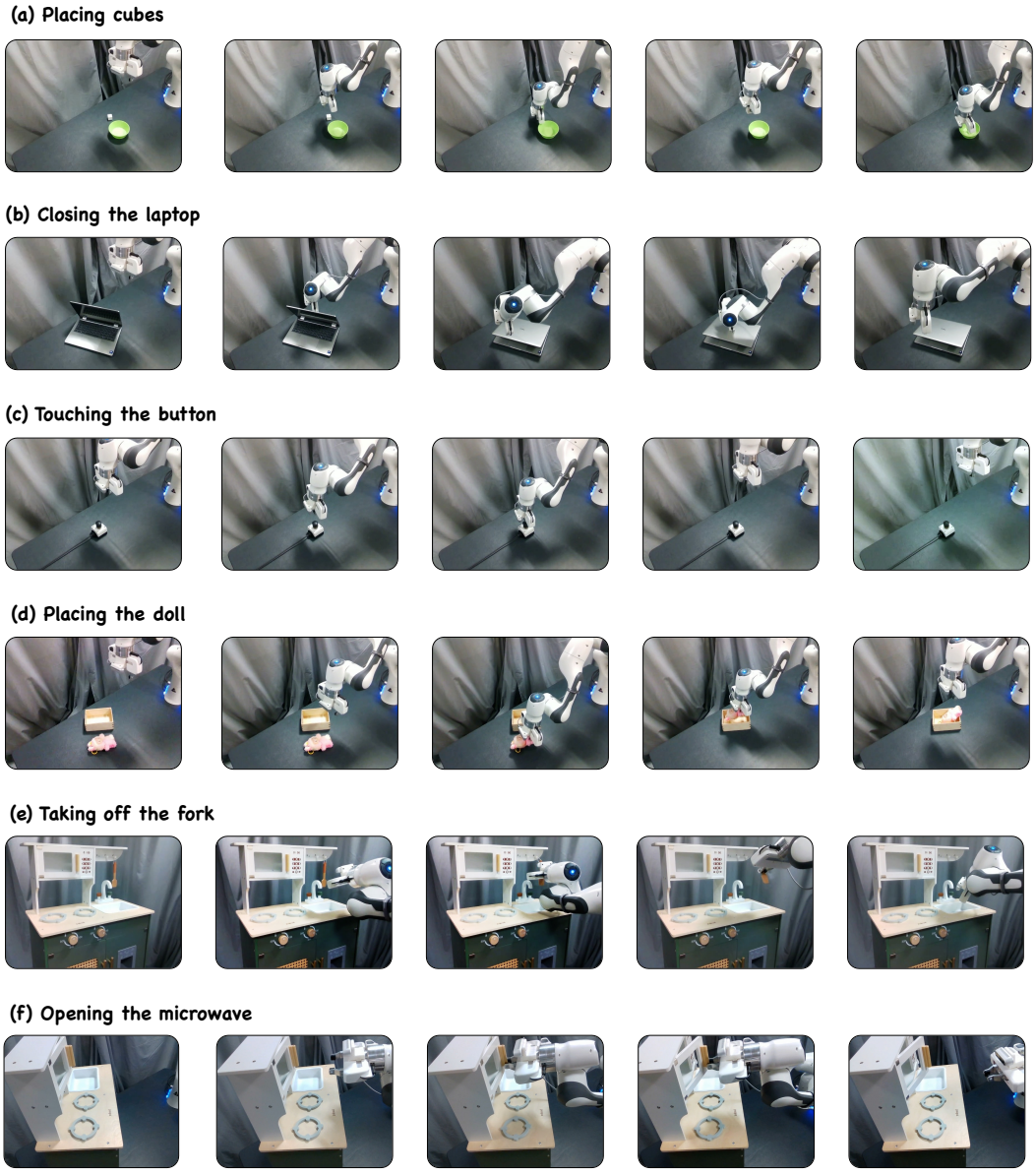


Figure B.5: Qualitative experiments of our method in real-world tasks.

six objects, SPARK-VLA succeeds in 18 of 30 trials (60.0%), compared with 14 of 30 (46.7%) for SPARK w/o VGGT.

We further evaluated nine language instructions that were not used during training. They include unseen paraphrases, spatial attributes, object descriptions, and category names. SPARK-VLA achieves six complete successes and two partial successes, whereas SPARK w/o VGGT achieves four complete successes and no partial successes. These results show that SPARK-VLA can execute several previously unseen object descriptions and linguistic formulations. However, they should not be interpreted as evidence that geometric distillation increases the semantic capacity of the language model itself.

Our failure analysis instead suggests that VGGT-based distillation primarily improves geometric execution after the target object has been identified. In many failures of SPARK w/o VGGT, the robot approached the correct object but closed the gripper too early or too late. We therefore hypothesize that the distilled spatial representation improves object-proximity estimation, grasp timing, and

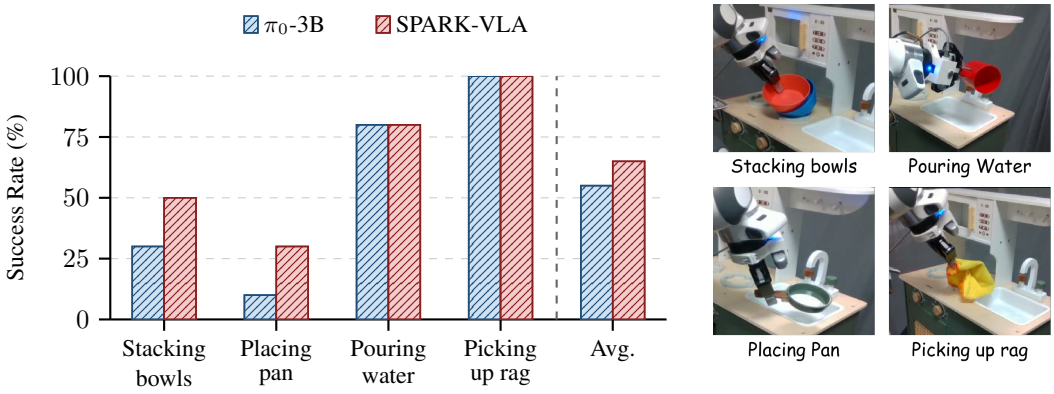


Figure B.6: Success rates on four additional kitchen manipulation tasks (left) and additional real-world examples (right). The success-rate plot reports 10 trials per task per method; Avg. denotes the mean across the four tasks.

Table B.7: Pick-and-place success rates for SPARK-VLA and SPARK w/o VGGT on the training red cup and five held-out objects. Each object is tested at the same five predefined starting locations (five trials per object).

Object	Condition	SPARK-VLA	w/o VGGT
Red cup	Training object	80%	60%
Blue cup	Unseen color	40%	40%
Black cup	Unseen color	60%	40%
Green handleless paper cup	Unseen color and geometry	40%	40%
Small milk carton	Unseen category and geometry	60%	60%
Hand-cream bottle	Unseen category and geometry	80%	40%
Mean	Across six objects	**60.0%**	46.7%

Table B.8: Outcomes for SPARK-VLA and SPARK w/o VGGT on nine language instructions held out from task training. Complete success requires executing the requested manipulation with the correct target; partial success denotes a correct grasp without completing the requested placement.

Language instruction	SPARK-VLA	w/o VGGT
“Pick the cup in the sink.”	Complete	Complete
“Move the cup to the sink.”	Complete	Failure
“Pick the cup and place it in the right-side area.”	Complete	Failure
“Move the red object to the right-side area.”	Partial	Complete
“Get the cup ready to wash in the sink.”	Partial	Failure
“Pick up the handleless green cup and put it in the sink.”	Complete	Complete
“Move the blue food can to the sink.”	Failure	Failure
“Pick up the white hand-cream bottle.”	Complete	Complete
“Pick up the blue-and-orange Capri-Sun juice pouch and put it in the sink.”	Complete	Failure
Complete success	**6/9 (66.7%)**	4/9 (44.4%)

placement accuracy across changes in object appearance and shape. In this sense, the geometric prior complements the pretrained model’s semantic knowledge rather than replacing it.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Yes, the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer:[Yes]

   Justification: We provide the limitations in the main paper.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A].

Justification: The paper aims to provide a new framework for current challenges, not a new theory.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide detailed information to reproduce the main experiments in the appendix.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Both the code and the data are publicly available.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes].

Justification: We provide the hyperparameters in both the main part and in the appendix.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: We provide this information in the Experiment section.

Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide the detailed computer resources in the Appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research is conducted in the paper conforms with the NeurIPS Code of Ethics in every respect.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss the broader impacts in the main paper.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A].

    Justification: The paper does not pose such a risk.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: They are properly cited with the correct license.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A].

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

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A].

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

    Guidelines:

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

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.