AtomicVLA: Unlocking the Potential of Atomic Skill Learning in Robots

Supplementary Material

Contents

A.1. Video Demonstration

Please refer to the video file in the attachment for a quick overview of AtomicVLA.

A.2. Future Work and Limitations

Most current vision-language-action (VLA) models are typically trained and evaluated on individual tasks. In this work, we investigate skill interference arising from multi-skill joint training through controlled experiments and introduce a Skill-Gated Mixture-of-Experts (SG-MoE) framework to construct a scalable atomic skill library, thereby exploring the potential of VLA models in long-horizon tasks and continual learning. Although this paradigm shows clear promise, many advantages remain insufficiently explored.

- AtomicVLA relies on a task planning module that produces accurate atomic skill abstractions and on a set of well trained skill experts. The skill router relies on the VLM to produce accurate atomic skill abstractions during task execution, a capability constrained by the VLM's reasoning and planning fidelity. Recent studies such as Embodied Brain [3, 6, 7] and $\pi_{0.5}$ [2] indicate that combining large scale web data with embodied experience can effectively train VLMs that are capable of skill decomposition and task planning while enabling the construction of a high quality expert skill library, which can further enhance the performance of AtomicVLA.
- By decoupling skill learning, AtomicVLA substantially mitigates interference during multi-skill training and demonstrates strong adaptability to new skills. However, acquiring new tasks still requires collecting substantial human demonstration data for imitation learning(IL). Notably, recent works like $\pi^*_{0.6}$ [1], SimpleVLA-RL [4] and VLA-RL [5] have shown that reinforcement learning(RL) can effectively train VLA models and achieve strong performance. Integrating a pre-trained skill expert library with reinforcement learning(RL) may empower AtomicVLA to generalize to novel tasks under few-shot or even zero-shot settings.

Table 1. **Atomic skill distribution in the LIBERO dataset.**

Atomic Skill	Count
Pick	2462
Place	761
Open	201
Close	152
Turn	175

A.3. Additional Details

A.3.1. Training Setup

For all experiments, we construct the skill library using one shared expert together with multiple skill experts. Each skill expert follows the Gemma architecture, where the feedforward module is implemented with an independent SwiGLU activated MLP. All skill experts are randomly initialized at the beginning of training to enable disentangled skill representations and support incremental learning. The model configuration is width = 2048, mlp dim = 4096, depth = 18, num heads = 8, and head dim = 256. Building on this configuration, the learning rate follows $CosineDecaySchedul$ with a warm up phase of 1,000 steps, a peak learning rate of 2.5×10^{-5}, and a final learning rate of 5×10^{-6}. The optimizer is AdamW with a gradient clipping norm = 1.0.To stabilize training, an exponential moving average (decay = 0.999) is used throughout optimization.

Following this setup, we train the model for 100k iterations on both the LIBERO and Calvin simulation platforms, and for 30k iterations in real world robotic experiments, with a batch size of 64. All training is performed on 8 $\times$ H200 GPUs, and inference is conducted on a single NVIDIA RTX RPO6000 GPU.

A.3.2. Simulations Setting

LIBERO Setting. We use the public dataset provided by LIBERO and convert it into the Lerobot format for all experiments. Following the data processing method introduced in Sec. 3.5, we perform fine grained annotation and organize the collected data into five atomic action abstractions:

Role:

You are an expert in robotics data analysis. You are analyzing a video clip of a robot performing a task, based on given task instructions. The clip was detected based on the robot's movement patterns and segmented into basic skill segments. Your goal is to determine the task progress of the current segment and identify the specific atomic actions.

Input:

1. The complete task instructions and coarse labels for the video clip.
2. Image frames from a video clip (sampled every three frames)

Instruction: Turn on the stove and put the moka pot on it
Coarse label: Turn

Task:

Your task is to provide a complete task chain based on the task instructions, and analyze the current task progress and corresponding atomic tasks and actions based on the coarse labels and video content.

Thought process and examples :

For task chain, it is a list of multiple atomic tasks.
For each atomic task, the formatting and constraints is:
1. Output one imperative sentence starting with an action verb.
2. Use exactly one verb from this set: [Pick/Place/Turn/Open/Close/Push/Pull/Adjust].
3. Focus on the final positions of the manipulated objects to avoid errors or repeated identification of atomic actions.
4. Sentence length limit: no more than 15 words.
5. Specify the manipulated object and key attributes (color, category, location/support surface/container).
6. For Place/Move, specify the destination or target container.
7. Describe only one atomic action and the final action (no multi-step sequences, no plans or intentions).
8. If this is final step(N/N), considering the completeness of the task, please avoid erroneous judgments such as "open" or "pick."
For atomic action, it must be one verb from this set: [Pick/Place/Turn/Open/Close/Push/Pull/Adjust].

Examples:
1. The task chain is [pick up the yellow cup, place the yellow cup in microwave, close the microwave], This is step 1/3, pick up the yellow cup, and the atomic abstraction is pick.
2. The task chain is [pick up the butter, place the butter in the basket], This is step 2/2, place the butter in the basket, and the atomic abstraction is place.
3. The task chain is [open the top drawer, pick up the block, place the block into the top drawer], This is step 2/3, open the top drawer, and the atomic abstraction is open.

Instructions:

Based on the text and video information above, please provide the task chain for this task, as well as the task progress and atomic actions corresponding to the video clip. Your judgment should be as detailed and accurate as possible, with reasoning supported by the video clip and task instructions. If the coarse label is incorrect, ignore it and provide the correct label.

Output Format:

The task chain is <the list of atomic tasks>, This is step x/N, <current atomic task>, and the atomic abstraction is <choose one atomic action>.

: The task chain is [turn on the stone, pick up the moka pot, place the moka pot on the stone], This is step 1/3, turn on the stone, and the atomic abstraction is turn.

Figure 1. **The prompts and examples of the InternVideo2.5.**

$Pick$, $Place$, $Open$, $Close$, and $Turn$. The data distribution for these action categories is presented in Tab. 1. All skills are trained in a mixed manner, and therefore maintaining balanced data becomes essential. To achieve this, we increase the sampling frequency of the less represented actions, specifically $Open$, $Close$, and $Turn$, in order to equalize the data distribution and prevent insufficient training of the corresponding skill experts. For a fair comparison, AtomicVLA is consistent with the evaluation of the baseline method, testing each task 50 times and reporting the average results.

Calvin Setting. We use the task ABC-D public dataset provided by Calvin and divide the data according to the instruction annotations and the corresponding frame intervals. Each trajectory is capped at 64 frames and is converted into the Lerobot format for our experiments. Following the data processing method introduced in Sec.3.5, we perform fine grained annotation and organize the data into eight atomic action abstractions: $Rotate$, $Push$, $Move$, $Open\&Close$, $Lift$, $Place$, $Turn$, and $Stack$. Based on these categories, we construct a skill expert library consisting of 8 skill experts. Building on this configuration, we ensure fair comparison by keeping the AtomicVLA evaluation protocol consistent with that of the baseline methods. In this setting, the robot executes 1,000 task sequences, and each sequence contains five consecutive tasks. We report

Table 2. **List of tasks and prompts used in our real-world experiments.**

Task Type	Task Prompt
Long-horizon Tasks	
Objects in plate	Place all blocks on the table into a green plate.
Object into drawer	Open the top drawer and place the block inside.
Object into microwave	Place the plate into the microwave and close the door.
Short Tasks	
Grasp	Grasp the block from the table.
Stack	Stack the red block on the orange block.
Close	Close the microwave on the table.
Press	Press the button on the table.
Open	Open the top drawer.
Complex Scenes	
Objects in plate	Put the pepper and corn into the green plate.
Objects in plate	Put the carrot and cucumber into the green plate.
Objects in plate	Put the potato and eggplant into the green plate.

Table 3. **Performance under Different Settings.**

Settings	π_0	SG	MoE	One-hot encoding	ours
Params	3.24B	3.24B	4.18B	4.17B	4.17B
Succ (%)	85.2	89.2	88.6	92.4	**95.2**

the average success rates together with the average length of the completed sequences.

A.3.3. Real-world Setting

Hardware. Our real-world experimental setup consists of a Franka Research3 robotic arm with two Realsense D435i cameras: one mounted on the wrist to provide a first-person perspective, and the other positioned opposite the robotic arm to offer a third-person view.

Evaluation Tasks. In the real world, we collected three long-horizon tasks and five short tasks, and additionally gathered three long-horizon tasks in more complex scenarios to evaluate the performance of AtomicVLA. We employed Gello [9] to control the Franka arm and record demonstration data. We collected 100 trajectories per long-horizon task and 50 per short task. The results reported in this paper were obtained using a multi-task mixed training protocol. Each task was evaluated 20 times with randomized object placements, and the average performance across these trials was reported as the final test result. The full list of tasks is presented in Tab. 2.

A.3.4. Continual Learning Setting

We conducted experiments on continual learning for short tasks. Specifically, we used four tasks for mixed training, iterating for 20k steps. Then, we applied "open the top drawer" as a new skill for continual learning, fine-tuning on the weights learned from the four tasks. We used a learning rate of 5×10^{-6} and iterated for 7k steps, and reported the results by averaging over 20 validation runs for each of the five tasks.

A.3.5. Data Generation Setting

We use principal component analysis to obtain precise video segmentation and coarse labels. By analyzing the motion changes across five consecutive frames, we determine the dominant motion axis. Specifically, the threshold for the translation axis($\Delta x, \Delta y, \Delta z$) is set to 3 cm, the threshold for the rotation axis(Δroll, Δpitch, Δyaw) is set to 0.05 radians, and the gripper change(ΔGrip) threshold is set to 0.1.

In Fig. 1, we provide detailed prompts and examples for VLM (InternVideo2.5 [8]). The VLM analyzes video clips and generates task chains, task progress, and atomic actions based on the input text instructions.

A.4. Additional Results

More ablations. As shown in Tab. 3, we supplement our experiments with a baseline conditioned on skills (SG). While this improves performance, it still underperforms our architecture, as skill interference arises when multiple skills are learned jointly. Additionally, while there are only 5 discrete skills, our experiments (Tab. 3) show that one-hot encoding underperforms our method.

Table 4. **Results on Complex Scenes.**

Method	Pepper/Corn	Carrot/Cucumber	Potatoe/Eggplant	Avg.
$\pi_{0.5}$ [2]	25	40	35	33.3
AtomicVLA*	**40**	**45**	**45**	**43.3**

Table 5. **Success rates for all evaluated tasks on CALVIN ABC-D dataset.**

Task Name	SR (%)	Task Name	SR (%)	Task Name	SR (%)
rotate blue block right	97.4	lift red block table	99.4	lift blue block table	99.4
move slider right	100.0	lift pink block table	94.5	place in drawer	100.0
lift red block slider	99.3	move slider left	100.0	rotate red block left	98.5
place in slider	98.6	turn on lightbulb	100.0	push pink block left	93.5
turn off lightbulb	100.0	rotate blue block left	100.0	lift blue block slider	95.6
turn off led	98.8	push blue block left	94.2	lift pink block drawer	100.0
push into drawer	86.0	turn on led	100.0	rotate pink block right	98.6
lift blue block drawer	100.0	stack block	98.4	unstack block	98.6
close drawer	100.0	push pink block right	33.8	push blue block right	22.2
lift pink block slider	97.8	push red block right	29.2	rotate pink block left	100.0
open drawer	100.0	push red block left	89.9	lift red block drawer	100.0
rotate red block right	97.3				

Table 6. **Parameter and inference-time.**

Experts	π_0	K=5	K=8	K=12
Params	3.24B	4.17B	4.81B	5.65B
Act	71ms	92 ms	126 ms	160 ms
Think	-	104 ms	104 ms	104 ms

Results on Complex Scenes. As shown in Tab. 4, we report the performance of AtomicVLA* and $\pi_{0.5}$ on three additional real-world experiments designed to evaluate its ability to handle complex scenes and grasp irregular objects. AtomicVLA* achieved an average accuracy of 43.3%, which is 10% higher than the $\pi_{0.5}$ average. In addition, when picking corn, due to the color being similar to the background of the table, AtomicVLA* was able to make multiple corrections as it approached the target, resulting in a 15% improvement.

Detail Results on Calvin. As shown in Tab. 5, we report the performance of AtomicVLA* on the 34 evaluation tasks of the Calvin ABC-D dataset. The results indicate that the model achieves success rates close to 100 percent on most tasks. However, performance on several $Push\ blocks\ right$ tasks is considerably lower, with average success rates only between 20 and 30 percent. Building on this observation, we find that in the training set the relevant blocks are typically placed near the center of the table. In contrast, during evaluation the blocks are often positioned on the right side of the table. This distribution shift leads the model to push the block in the correct direction while failing to push it far enough to satisfy the success criterion, which results in task failure and prevents the execution of subsequent steps.

Parameter and inference-time. Tab. 6 shows the parameter counts and inference-time on a single H20 GPU. Even with 12 experts, the inference latency is only 160 ms, which is fully practical for real-world use.

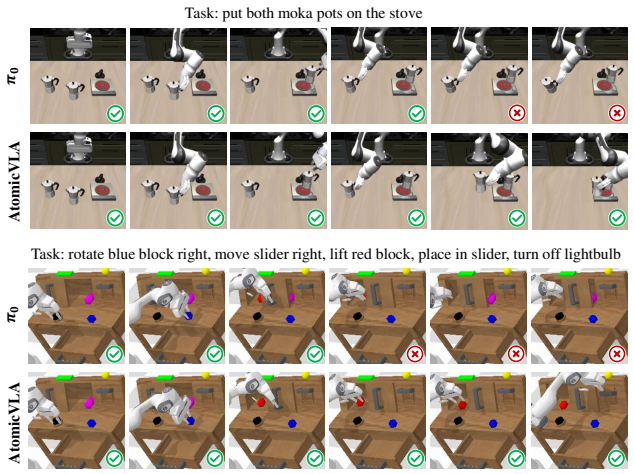

Figure 2. **Demonstrations of LIBERO and Calvin experiments.**

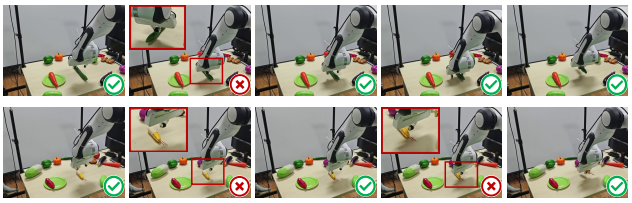

Figure 3. **Error recovery cases in real-world experiments.**

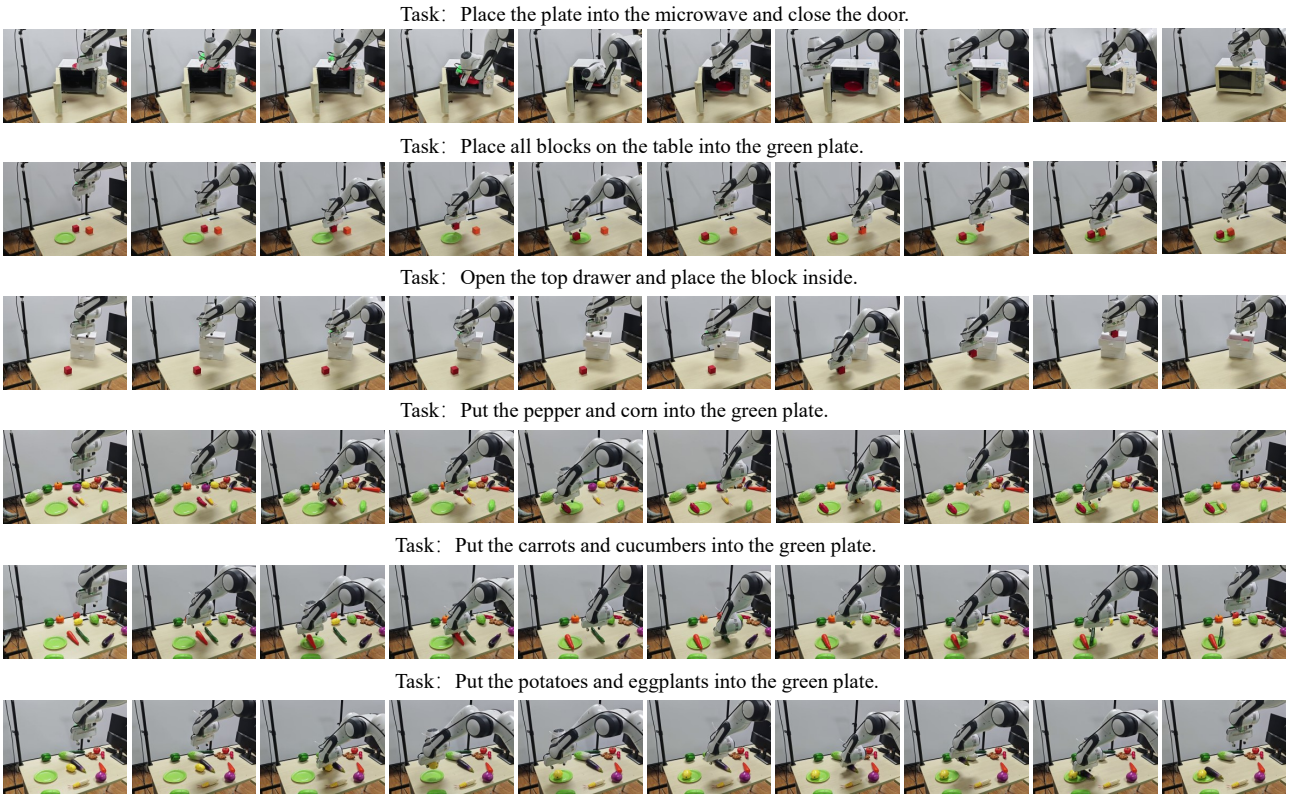

Figure 4. **Demonstrations of real-world experiments(Long-horizon tasks).**

A.5. Additional Visualizations

In Fig. 2, we present a comparison between AtomicVLA and π_0 across simulation environments. Representative task cases are selected from both LIBERO and Calvin. As shown, AtomicVLA successfully completes several task instances where pi0 fails, demonstrating its stronger robustness and execution reliability in simulated settings.

In Fig. 3, we further illustrate AtomicVLA's real-world error recovery capability. When a subtask fails during execution, AtomicVLA automatically replans and corrects its behavior to ensure successful completion of the overall task. Specifically, as highlighted in the red box in the figure, when execution errors occur, such as misgrasps due to inaccurate positioning or visual ambiguity between the target object and the background, AtomicVLA can assess the current task state, generate an updated task plan, and reattempt the failed subtask, thereby ensuring robust completion of the overall task.

Additionally, we show more demonstrations of real-world experiments in Fig. 4. These experiments span a wide spectrum of scenarios, from simple to highly complex tasks and from regular to irregular objects. Across all settings, AtomicVLA consistently exhibits strong performance and robust generalization.

References

[1] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. 1

[2] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. 1, 4

[3] Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, Xinda Xue, Qinghang Su, Huaihai Lyu, Xiaolong Zheng, Jiaming Liu, Zhongyuan Wang, and Shanghang Zhang. Robobrain: A unified brain model for robotic manipulation from abstract to concrete, 2025. 1

[4] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. Simplevla-rl: Scaling vla training via reinforcement learning, 2025. 1

[5] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. 1

[6] Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Xiansheng Chen, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-rft: Reinforcement fine-tuning for visual reasoning of vision language models, 2025. 1

[7] BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Xiansheng Chen, Minglan Lin, Zhiyu Li, Zhou Cao, Pengwei Wang, Enshen Zhou, Yi Han, Yingbo Tang, Xiangqi Xu, Wei Guo, Yaoxu Lyu, Yijie Xu, Jiayu Shi, Mengfei Du, Cheng Chi, Mengdi Zhao, Xiaoshuai Hao, Junkai Zhao, Xiaojie Zhang, Shanyu Rong, Huaihai Lyu, Zhengliang Cai, Yankai Fu, Ning Chen, Bolun Zhang, Lingfeng Zhang, Shuyi Zhang, Dong Liu, Xi Feng, Songjing Wang, Xiaodan Liu, Yance Jiao, Mengsi Lyu, Zhuo Chen, Chenrui He, Yulong Ao, Xue Sun, Zheqi He, Jingshu Zheng, Xi Yang, Donghai Shi, Kunchang Xie, Bochao Zhang, Shaokai Nie, Chunlei Men, Yonghua Lin, Zhongyuan Wang, Tiejun Huang, and Shanghang Zhang. Robobrain 2.0 technical report, 2025. 1

[8] Yi Wang, Xinhao Li, Ziang Yan, Yinan He, Jiashuo Yu, Xiangyu Zeng, Chenting Wang, Changlian Ma, Haian Huang, Jianfei Gao, Min Dou, Kai Chen, Wenhai Wang, Yu Qiao, Yali Wang, and Limin Wang. Internvideo2.5: Empowering video mllms with long and rich context modeling, 2025. 3

[9] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2024. 3