SimpleVLA: Building State-of-the-Art VLAs the Simple Way

Ankit Goyal, Hugo Hadfield, Xuning Yang, Valts Blukis, Fabio Ramos
NVIDIA

***Abstract*— Vision-Language-Action models (VLAs) hold immense promise for enabling generalist robot manipulation. However, the best way to build them remains an open question. Current approaches often add complexity, such as modifying the existing vocabulary of a Vision-Language Model (VLM) with action tokens or introducing special action heads. Curiously, the simplest strategy of representing actions directly as text has remained largely unexplored. This work introduces SimpleVLA to investigate this idea. We find that SimpleVLA is not only effective; it is surprisingly powerful. With the right design, SimpleVLA outperforms more involved models. On LIBERO, a popular benchmark for evaluating VLAs, SimpleVLA outperforms all existing methods trained on the same robotic data, including $\pi_0.5$-KI, OpenVLA-OFT and SmolVLA. Furthermore, without large-scale robotics-specific training, it outperforms methods trained on large-scale robotic data, like $\pi_0.5$-KI, π_0, GR00T-N1 and MolmoAct. These findings also translate to the real world, where SimpleVLA outperforms SmolVLA, a VLA model pre-trained on large-scale real data. This paper summarizes our unexpected findings and spells out the specific techniques required to unlock the high performance of this simple yet potent VLA design. Visual results, code, and trained models are provided at: https://simplevla.github.io/.**

I. Introduction

Following the success of Large Language Models (LLMs) in text processing and Vision-Language Models (VLMs) in handling both visual and textual inputs, a natural next step is to explore Vision-Language-Action models (VLAs), i.e. systems that not only understand visual and textual information, but also predict actions for robotic agents. VLAs are typically built by modifying a base VLM to predict actions. However, it is still unclear what the 'correct' way to do this is, if there is one at all. Recent research has taken various approaches, which we broadly categorize into three families, as shown in Figure 2: (1) Discrete Token VLAs, (2) Generative Action Head VLAs, and (3) Custom Architecture VLAs.

Discrete Token VLAs. It is one of the initial strategies popularized by models such as RT-2 [24] and OpenVLA [11]. Robot actions, originally continuous, are discretized into bins; each bin is then assigned a token from the VLM vocabulary, using either new or infrequent tokens. The model is then trained to predict these action tokens using the same cross-entropy loss as used to train the base VLM. Although straightforward, this approach has two main limitations: (i) it restricts the resolution of the action space, since fine-grained control can require thousands of bins, which conflicts with sharing the text vocabulary; and (ii) it compromises the pretrained language understanding of the VLM by repurposing its vocabulary for actions. Given these limitations, such VLAs do not perform as well as other alternatives. (see Tab. I)

Generative Action Head VLAs. Another common strategy is to attach an action generation head on top of the VLM, as done by methods like π_0 [2] or SmolVLA [19]. The VLM is fine-tuned to predict a latent vector, which is then decoded into actions using a generative model such as a diffusion process or flow matching. While this method improves action fidelity, it also introduces a new neural network that needs to be finetuned. This often leads to a decline in the language understanding and grounding capabilities of the underlying VLM [9], and introducing a non-pretrained action head may compromise generalization of the overall system.

Custom Architecture VLAs. Beyond the above categories, some methods propose architectural modifications or custom tokenizers tailored to action prediction. For instance, OpenVLA-OFT [10] introduces a specialized ACT head. Another example is π-FAST [16] that create a special tokenization scheme for actions using discrete cosine transform (DCT). π-FAST can also be considered a discrete token VLA, but for the purposes of this work, we classify them as custom VLA as it involves a custom tokenization scheme. While these custom methods are effective, they typically involve significant architectural changes, additional param-

Fig. 1: **Schematic representation of SimpleVLA.** SimpleVLA converts a VLM into a VLA by prompting the VLM to predict action as text. This strategy is surprisingly effective and achieves state-of-the-art results akin to alternatives.

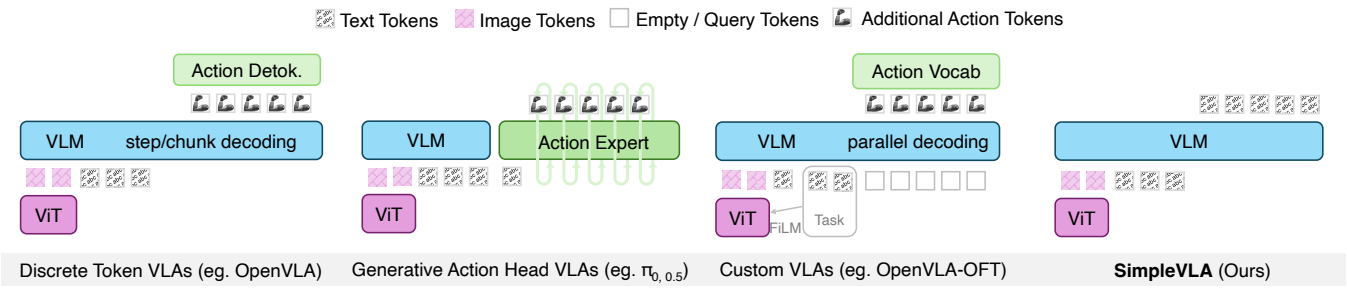

Fig. 2: **Families of methods for building VLAs.** We categorize existing VLAs into three categories: Discrete Token VLAs, Generative Action Head VLAs and Custom Architecture VLAs. In this work, we propose the SimpleVLA family where the VLM is prompted to directly predict action as text. Unlike other methods, SimpleVLA requires no change to the underlying VLM.

eters, or custom training pipelines.

Despite the success of these methods, we ask if there is a simpler alternative. One that does not require changing the VLM's vocabulary or introducing any new architectural components. Have we ruled out predicting actions as text? Why not represent actions (e.g., coordinates, joint angles) as numerical strings and generate them using the VLM's native text generation capability? This approach does not require new tokens, no vocabulary modifications, and no architectural changes. It maintains the integrity of the VLM while offering arbitrary resolution in the action space. Given the extensive effort devoted to optimizing training recipes for various VLA designs, one may ask: what if we instead focus on the simplest architecture?

We evaluate such a design, which we refer to as SimpleVLA. Contrary to expectations from prior literature, we find that this simple formulation is highly competitive–achieving performance on par with other alternatives, while requiring no change to the underlying VLM architecture. On the widely adopted LIBERO [14] benchmark, it outperforms all methods that have been trained with the same amount of robotic action data. Further, SimpleVLA outperforms popular methods that have been pretrained with large-scale action data including π-0 [2], GR00T-N1 [1], π-Fast [9], OpenVLA [11], Octo [21] and MolmoAct [12]. We find that these findings also translates to the real-world where SimpleVLA outperforms SmolVLA [19] which has previously achieved state-of-the-art results.

In order to achieve state-of-the-art performance with this design, a careful training and testing recipe is required. For example, we find that during training, random masking of the action text improves performance. Similarly, during testing, it is helpful to ensemble previous predictions. In summary, our contributions are as follows:

1: We demonstrate that a simple VLA design that requires no change to the VLM architecture can achieve state-of-the-art results akin to popular alternatives;
2: We devise the training and testing recipe that achieves state-of-the-art performance with the simple VLA design.

II. Related Work

Our work builds on recent advances in Vision-Language-Action models and the broader field of robot learning.

Vision-Language-Action Models. The paradigm of adapting pre-trained Vision-Language Models (VLMs) for robotic control has recently gained significant traction. A predominant approach involves representing continuous actions as discrete tokens. This strategy, employed by influential models like RT-2 [24] and OpenVLA [11], discretizes the action space into a finite number of bins and maps each bin to a token within the VLM's vocabulary. While this allows for a straightforward integration of action generation into the language modeling objective, it introduces a trade-off between action resolution and vocabulary size, and can potentially corrupt the semantic meaning of repurposed tokens.

Another prominent family of methods avoids altering the VLM's vocabulary by introducing auxiliary action heads. Models such as π_0 [2] and SmolVLA [19] finetune the VLM to output a latent embedding, which is then decoded into a continuous action by a separate generative model, like a diffusion policy or a flow-matching network. While this preserves the VLM's original vocabulary and allows for high-fidelity actions, it increases model complexity and can sometimes lead to a degradation of the VLM's language grounding capabilities [9].

A third category involves more substantial architectural modifications [[10], [16], [20]], such as the specialized action head in OpenVLA-OFT [10] or the custom action tokenization via Discrete Cosine Transform in π-FAST [16], which often require intricate training pipelines.

Our proposed method, SimpleVLA, explores a conceptually simpler alternative: representing actions directly as text. By representing numerical actions (e.g., end-effector coordinates) as strings, we leverage the VLM's native text generation capabilities without any architectural modifications. The closest to our method is LLARVA [15] that learns to predict action as text. However, LLARVA employs a two-stage process, first generating a 2D trajectory plan before predicting the final action. In contrast, our work demonstrates that direct, end-to-end generation of action strings can achieve state-of-the-art performance. The key to our success

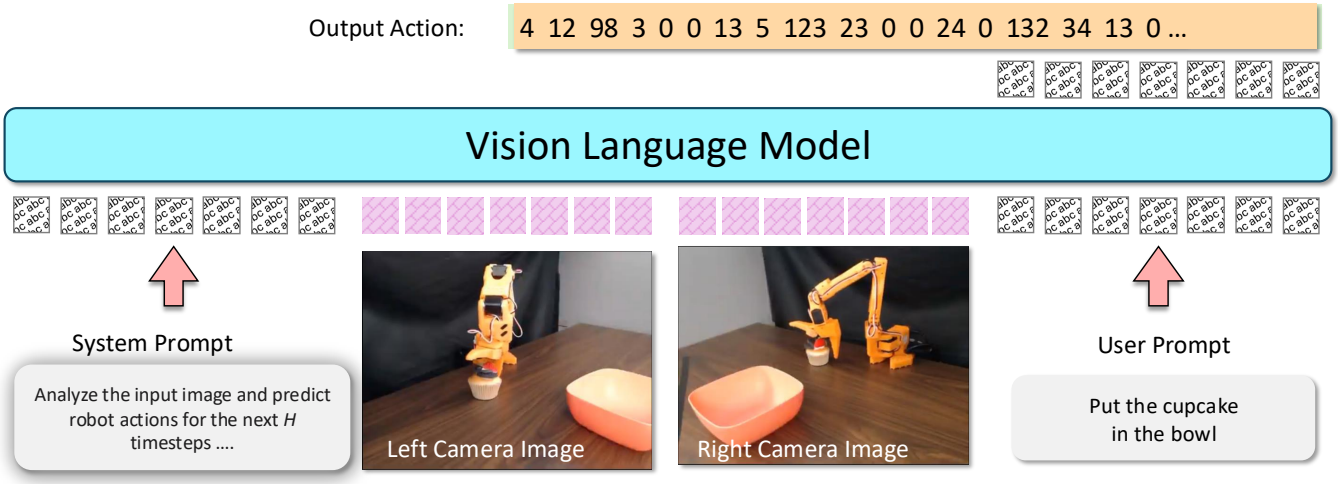

Fig. 3: **Our proposed SimpleVLA**. It creates a VLA without making any changes to the underlying VLM. It takes a system prompt, language instruction, and images as input, and outputs actions represented as space-separated integers.

lies in a carefully designed training and inference recipe, including action token masking and prediction ensembling, a critical component not explored in LLARVA.

Another close work is HAMSTER [13], which proposes a hierarchical vision-language-action model. The first stage of HAMSTER using a VLM to predict a 2D action trajectory in text. Our design is similar but we predict the complete robot action (like joint pose or end-effector delta) as text.

Robot Learning Policies. Learning robotic policies from demonstrations is a well-established field that predates the recent rise of VLAs. Unlike VLAs, these methods typically train policies from scratch on in-domain data without leveraging large pretrained vision and language models. A leading example is Diffusion Policy [4], which models the action space using a conditional diffusion process and has shown strong performance across various manipulation tasks.

Another line of research focuses on improving sample efficiency and spatial reasoning by incorporating explicit 3D representations into the policy architecture [[8], [7], [17], [6]]. Models like RVT [8], RVT-2 [7], ManiFlow [22] and Act3D [6] leverage 3D scene information to learn more robust and generalizable policies.

In contrast to these approaches, SimpleVLA aligns with the VLA paradigm by building directly on the powerful, pre-trained representations of a VLM. Our findings indicate that by properly harnessing the VLM, our simple approach can outperform specialized methods like Diffusion Policy on benchmark tasks using only the in-domain action data.

III. Method

A. Background

Vision-Language Models. Vision-Language Models (VLMs) are a class of neural networks designed to process and reason about information from both visual and textual modalities. Typically, they are comprised of a pretrained vision encoder (e.g., a Vision Transformer) that extracts visual features from an image, and a Large Language Model (LLM) that processes textual information. The visual features are projected into the LLM's embedding space, allowing the model to jointly condition on both the image and a text prompt to generate a coherent textual output. [1]

For this work, we build our system upon a publicly available, state-of-the-art VLM. Specifically, we employ the 3-billion-parameter Qwen-VL-2.5 [18] model, although our method is applicable to any other VLM. Several factors motivate our choice. Qwen-VL-2.5-3B demonstrates highly competitive performance for its model size. As a relatively smaller VLM, it is computationally efficient, which facilitates faster training and inference. Furthermore, its open-weight nature promotes accessibility and reproducibility.

B. Method: SimpleVLA

We introduce SimpleVLA, a simple design for building Vision-Language-Action models. Unlike alternatives, SimpleVLA preserves the integrity of the underlying VLM: it does not introduce new tokens, alter the existing vocabulary, or add any new neural network layers. Despite its simplicity, and contrary to expectations from prior literature, SimpleVLA is as performant as more involved alternatives. However, achieving this performance relies on a careful recipe. Three key components of this recipe are action decoding, ensemble prediction, and masked action augmentation.

Input. SimpleVLA inherits the input structure of the underlying VLM, which consists of a *System Prompt*, *Images*, and a *Task Instruction*. The system prompt specifies the high-level goal of the VLM. During fine-tuning, we use the following prompt, where H, D, and B are chosen based on the data.

System Prompt. *Analyze the input image and predict robot actions for the next H timesteps. Each action has D*

[1] Some VLMs can also generate outputs in other modalities, such as images. However, for clarity in this paper, we refer to VLMs as models that produce only text.

Models	Large-scale act. pre-train	VLA Type	Spatial	Object	Goal	Long	Avg.	Avg. rank
Diffusion Policy [4], [11]	✗	N/A	78.3	92.5	68.3	50.5	72.4	6.5
π_0-FAST (Paligemma) [2], [19]	✗	Custom	87.0	63.0	89.0	48.0	71.8	6.0
SmolVLA (0.24B) [19]	✗	Gen Head	87.0	93.0	88.0	63.0	82.8	5.3
SmolVLA (2.25B) [19]	✗	Gen Head	93.0	94.0	91.0	77.0	88.8	4.0
OpenVLA-OFT [10]	✗	Custom	94.3	95.2	91.7	86.5	91.9	2.8
$\pi_{0.5} - KI$ [5]	✗	Gen Head	96.6	97.2	94.6	85.8	93.3	2.3
SimpleVLA (Ours)	✗	Simple	**97.0**	**97.8**	**96.2**	**87.6**	**94.7**	**1.0**
Octo [21]	✓	Gen Head	78.9	85.7	84.6	51.1	75.1	8.8
OpenVLA [11]	✓	Dis. Tok.	84.7	88.4	79.2	53.7	76.5	8.0
π_0-FAST [16]	✓	Custom	90.0	86.0	95.0	73.0	86.0	6.5
Molmo Act [12]	✓	Dis. Tok.	87.0	95.4	87.6	77.2	86.8	6.5
GR00T-N1 [1]	✓	Gen Head	94.4	97.6	93.0	90.6	93.9	4.5
π_0 [2]	✓	Gen Head	96.8	**98.8**	95.8	85.2	94.2	3.3
$\pi_{0.5} - KI$ [5]	✓	Gen Head	**98.0**	97.8	95.6	85.8	94.3	3.0
OpenVLA-OFT [10]	✓	Custom	97.6	98.4	**97.9**	**94.5**	**97.1**	**1.5**
SimpleVLA (Ours)	✗	Simple	97.0	97.8	96.2	87.6	94.7	2.8

TABLE I: **Performance on Libero.** We report the success rate for four LIBERO suites, for two category of models, with and without large scale action pretraining. Best performance is highlighted in **bold** and second best in underline. SimpleVLA outperforms all models without large scale action pretraining, achieving the highest average success rate and best rank. Further, SimpleVLA, without any large-scale action pretraining, outperforms popular VLAs including $\pi_{0.5} - KI$, π_0 and GR00T-N1 that have been pretrained with large scale action data.

dimensions. Output a single sequence of $H \times D$ integers (0 - B each), representing the H timesteps sequentially. Provide only space-separated numbers. Nothing else.

Similar to the underlying VLM, SimpleVLA can take one or more images as input, depending on the setup. For our simulation experiments, we use the third person and wrist camera image as input, like the baselines. For real experiments, we use the left and right camera images as shown in Figure. 3. We also experiment with an alternative image input design where, instead of providing images as separate entities, we tile them into a single composite image. In our experiments, we find that both designs exhibit similar performance (see Table II). Lastly, the input includes the task instruction, for example: "put the banana on the plate."

Action Decoding. SimpleVLA produces actions as text. To simplify this task, we ask the VLM to output actions as integers. Specifically, the original continuous action values are first normalized to a fixed integer range (e.g., [0,1000]). The VLM is then tasked with generating an integer for each action dimension. The maximum value of this range can be tuned based on the dataset and the desired action resolution. Notably, unlike discrete token-based VLAs, this approach allows for arbitrary resolution without altering the model's vocabulary.

Ensemble Prediction. SimpleVLA employs the prediction ensembling technique introduced by the Action-Chunking Transformer (ACT) [23], which has also been adopted by other state-of-the-art VLAs like OpenVLA-OFT [10]. At each inference step, the VLM predicts a sequence of n future actions. Consequently, for the current time step t, there are n available predictions for the action: one made at the current step t, another made at step t–1 (as the second action in its predicted sequence), and so on, back to the prediction made at step t–n+1. In our design, we average these n predictions to produce the final, more stable action at time step t.

Masked Action Augmentation. Another component of our recipe is a training augmentation we introduce, which we call Masked Action Augmentation. VLMs produce text auto-regressively, meaning each generated token is conditioned on the previously generated tokens. During training, we randomly mask out characters in the target action string. This procedure forces the VLM to reason about the action based on the visual observation and instruction, rather than simply relying on auto-completing a numerical sequence it has started to generate.

Training Details. We train SimpleVLA by performing a full fine-tuning of the base VLM. The model is trained to generate the target action strings using a standard cross-entropy loss over the vocabulary. For optimization, we use the Adam optimizer and train the model for 64 epochs with a batch size of 192 and a learning rate of 5e-6. Training takes approximately 32 hours on 8 A100 GPUs.

IV. Experiments

A. Setup

We evaluate our model in both real-world and simulated environments to thoroughly assess its performance. **Real-World.** For our real-world evaluation, we use the SO-100 robot and the LeRobot framework. We train and test policies on four distinct manipulation tasks: reorienting a block, pushing an apple, picking and placing a banana, and picking and placing a cupcake. For each task, we collect 100 demonstrations for training. The learned policies are then evaluated across varied initial conditions of the objects to test for robustness.

Simulation. In simulation, we use the LIBERO benchmark [14], a widely adopted benchmark for comparing VLA models. LIBERO consists of four suites: Spatial, Object,

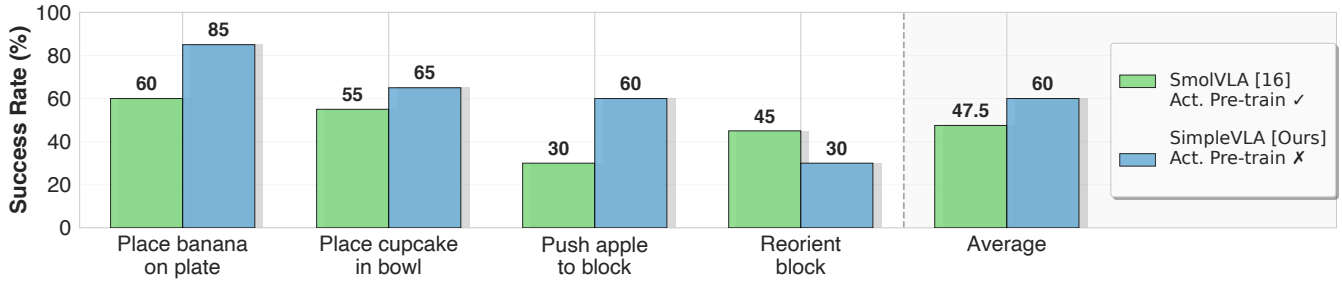

Fig. 4: **Performance on Real.** We compare SimpleVLA to SmolVLA on four different tasks with SO-100. SimpleVLA outpeforms SmolVLA on average. SmolVLA is pretrained with large-scale SO-100 data while SimpleVLA is trained from scratch.

Goal, and Long. Each suite is designed to assess a system's capability along a particular dimension. Each suite contains 10 tasks, and each task is tested over 50 episodes. Performance is reported as the success rate for each individual suite and as an overall average.

B. Baselines

We compare SimpleVLA against several baselines, including Diffusion Policy and a variety of state-of-the-art Vision-Language-Action (VLA) models. Our baselines are drawn from the three categories previously outlined (Sec. I, Figure 2): OpenVLA [11] and MolmoAct [12] from the discrete token-based family; Octo [21], π_0 [2], GR00T-N1 [1], $\pi_{0.5}-KI$ [9] and SmolVLA [19] from the generative action-based family; and π_0-FAST [16] and OpenVLA-OFT [10] from the custom architecture family. By $\pi_{0.5}-KI$, we refer to the work by Dries et al. which builds on top of $\pi_{0.5}$ to show the effectiveness of knowledge insulation in VLAs. A key distinction among these VLA models is their use of large-scale action pretraining. To ensure a fair comparison, our primary analysis focuses on models that, like SimpleVLA, have not undergone such pretraining. However, for completeness, we also report the performance of pretrained models in Table I.

C. Simulation Results

Table. I summarizes the performance of various baselines and SimpleVLA on LIBERO. We find the SimpleVLA outperforms all existing VLA models that, like our, were not pretrained with large-scale robotic data. These include models like $\pi_{0.5}$-KI, OpenVLA-OFT [11], π_0-Fast [16] and SmolVLA [19]. SimpleVLA outperforms these baseline across all the LIBERO suites, outperforming the second best method by 1.4 points on average. This result is highly surprising and runs counter to the expectations set by existing literature. It shows how highly performance VLAs can be built without introducing any change to the underlying VLM.

What's even more surprising is how SimpleVLA stacks up against models that did have the advantage of pretraining on large-scale robotic data. Despite having no large-scale action pretraining, SimpleVLA surpasses the performance of many well-known pretrained models, including $\pi_{0.5}$-KI, π_0, π_0-FAST [16], Octo [21], OpenVLA [11] and GROOT-N1.5 [1] and MolmoAct [12]. Overall, it gets the second best average rank 2.8, trailing only OpenVLA-OFT [10] (average rank 1.5), a custom VLA model. This suggests that proposed simple strategy holds up effectively against the best models that are pretrained with large scale training data.

D. Real-World Evaluation

To validate our approach on physical hardware, we evaluate SimpleVLA in the real world using the LeRobot framework [3]. We compare with SmolVLA [19], a strong baseline that was specifically trained on the large-scale SO-100 dataset and has been shown to outperform popular methods like π_0 [2] and ACT [23] on this platform.

For inference, we use a desktop equipped with a 5090 GPU. Our system streams actions for each timestep, achieving an inference speed of 4 Hz. This performance is achieved using a standard PyTorch implementation. We believe this speed could be significantly increased through techniques such as model distillation or quantization, which we leave as future work. For simplicity, we do not ensemble actions in real, although it is possible to do so but requires 8 simultaneous running instances of the model.

Figure 4 summarizes the task success rates on four real world tasks. The results show that SimpleVLA outperforms SmolVLA by 12.5 points, despite not being pretrained on the large-scale SO100 dataset. This demonstrates that our method's effectiveness translates from simulation to real.

E. Ablations

We conduct a series of ablation studies on the LIBERO benchmark to analyze the impact of key design components in SimpleVLA. Table II summarizes these results.

a) Action Ensembling: Disabling action ensembling (comparing Row 0 and 1) reveals its significant impact. We find that this technique is a critical component, improving the overall success rate by 2 points.

b) Masked Action Augmentation: Our proposed Masked Action Augmentation provides a modest but consistent benefit. Removing this augmentation (comparing Row 0 and 2) decreases the success rate by 1.2 points.

Row ID	Ensemble Act.	Masked Act. Aug.	Tiled Img.	Act. Res.	Avg. Succ.	Δ perf.
0	✓	✓	✓	1000	94.7	0.0
1	✗	✓	✓	1000	92.0	-2.0
2	✓	✗	✓	1000	93.5	-1.2
3	✓	✓	✓	4000	94.2	-0.5
4	✓	✓	✓	250	93.2	-1.5
5	✓	✓	✗	1000	94.5	-0.2

TABLE II: **Ablations.** We ablate various design choices for SimpleVLA on LIBERO. We report the average success rate over the four LIBERO suites.

c) Action Resolution: The choice of action resolution is an important hyperparameter. For the LIBERO benchmark, we find that a resolution of 1000 is sufficient. Decreasing the resolution to 250 degrades performance, reducing the success rate by 1.5 points, while a higher resolution of 4000 yields no additional performance gains.

d) Image Tiling: When providing multiple image observations to the VLM, one can either tile them into a single composite image or feed them as separate inputs. We find that this decision has no discernible impact on performance.

V. Conclusions and Limitations

In this work, we made the case for a simple VLA design that preserves the integrity of the base VLM without altering its tokenization or introducing new architectural components. We demonstrated that with the right recipe, this approach outperforms more involved strategies—a surprising result that considering prevailing trends in the literature.

Despite these promising findings, our work has limitations that present clear directions for future research. A key area to explore is how SimpleVLA would perform when trained with large-scale action data. Another area of investigation would be to improve inference speed of SimpleVLA using optimization techniques like quantization and distillation.

References

[1] 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.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*, 2025.

[3] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. `https://github.com/huggingface/lerobot`, 2024.

[4] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[5] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

[6] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. *arXiv preprint arXiv:2306.17817*, 2023.

[7] Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. *RSS*, 2024.

[8] 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.

[9] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[10] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[11] 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.

[12] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

[13] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, et al. Hamster: Hierarchical action models for open-world robot manipulation. *ICLR*, 2025.

[14] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[15] Dantong Niu, Yuvan Sharma, Giscard Biamby, Jerome Quenum, Yutong Bai, Baifeng Shi, Trevor Darrell, and Roei Herzig. Llarva: Vision-action instruction tuning enhances robot learning. *arXiv preprint arXiv:2406.11815*, 2024.

[16] 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. *arXiv preprint arXiv:2501.09747*, 2025.

[17] Shengyi Qian, Kaichun Mo, Valts Blukis, David F Fouhey, Dieter Fox, and Ankit Goyal. 3d-mvp: 3d multiview pretraining for robotic manipulation. *CVPR*, 2024.

[18] A Yang Qwen, Baosong Yang, B Zhang, B Hui, B Zheng, B Yu, Chengpeng Li, D Liu, F Huang, H Wei, et al. Qwen2. 5 technical report. *arXiv preprint*, 2024.

[19] 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.

[20] Ishika Singh, Ankit Goyal, Stan Birchfield, Dieter Fox, Animesh Garg, and Valts Blukis. Og-vla: 3d-aware vision language action model via orthographic image generation. *arXiv preprint arXiv:2506.01196*, 2025.

[21] Octo Model Team, 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. *arXiv preprint arXiv:2405.12213*, 2024.

[22] Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, et al. Maniflow: A general robot manipulation policy via consistency flow training. *arXiv preprint arXiv:2509.01819*, 2025.

[23] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[24] 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.