National Technical University of Athens
School of Electrical and Computer Engineering

Semester Project: Vision-Language-Action models are efficient learners

NEURO-FUZZY CONTROL AND APPLICATIONS

Ac. year 2024–2025, 9th semester

Elias Kallioras 03120050

1 Introduction

Humans, by nature, are curious to uncover meaning in their existence. From ancient times up until today we try to understand what built the very foundations we take as granted, yet are the biggest mysteries in the universe. The concept of consciousness is certainly among these hard questions that have been in the minds of some of the smartest individuals to ever exist, yet we are still clueless to what mechanisms are behind the element that distinguishes us from the rest of the food chain, making us what we are today. As an effort to grasp these concepts we have been known to try to replicate human consciousness outside the bounds of the biology of nature. These efforts rely on the computational advancements humanity has seen in the last century. These efforts are based upon the assumption that consciousness is computable, which is a debatable and complex topic. The Perceptron (1957-1958) by Frank Rosenblatt is often considered one of the first serious computational efforts to replicate aspects of human consciousness and intelligence.

Recent advances in the field of Machine Learning and in the hardware that supports it have allowed for the rise of Large Language Models (LLMs), which are agents trained on huge amounts of data from the Internet, and are fitting for ”next word prediction” tasks. A wide variety of tasks can be formulated as next word prediction problems, since many things around us are in nature sequential. For example, videos, language, even time are sequential in the sense that for event at $t + 1$ to occur, the event at t must occur first. Moreover, this gave space to the rise of Visual Language Models (VLMs), that can generate meaningful image features after they have been trained on billions of image/caption pairs. Based on this we can build agents that seem to reason and have common sense, and thus cannot always be easily distinguished from a human agent. One could then argue that these agents are conscious, as they present inherent abilities to reason about spatial patterns and partially occluded objects in images, are very good at responding to ambiguous or mentally challenging prompts, and can even understand frustration of the user when they fail to answer correctly.

Of course, these advancements made their way into robotics, where it has always been a dream to build a generalist agent - a robotic agent that can act just like a human would. This is not an easy task, as humans are very good at understanding language instructions and executing them with near perfect hand coordination and precision. On top of that the language instructions may be ambiguous, may require the ability of spatial awareness, or may involve a long chain of reasoning and acting to achieve a final goal. To give en example, lets imagine a human tasked with the prompt: ”I’m thirsty”. The average human can break this into smaller sequential tasks that will result in giving a solution to the initial task prompt, e.g (1) get a glass and place it under faucet, (2) open faucet slightly, (3) wait until glass is almost full, (4) close faucet, (5) serve glass of water. This chain of thought reasoning, inherent in our brains is not trivial. Towards making robot agents have this level of understanding, recent works have been focused on Vision-Language-Action models (VLAs). VLAs are based on the dual-stream hypothesis from the field of cognitive neuroscience, where it is theorized that visual information is processed along two distinct pathways in the primate brain. The Ventral Stream (”What” pathway) is responsible for understanding the essence of the task given, and the Dorsal Stream (”Where/How” pathway) is responsible for how that task must be executed. This theory is backed up by cases where patients with ventral stream damage (visual agnosia) can’t recognize objects but can still reach for and grasp them accurately. Conversely, patients with dorsal stream damage struggle with visually guided actions but retain object recognition. A known example is the ”place the letter in the post” task: Patient D.F., who suffered ventral stream damage from carbon monoxide poisoning, could accurately post a letter through a slot by orienting her hand correctly but was unable to consciously perceive whether the slot was horizontal or vertical when asked. This demonstrated the dissociation between the dorsal stream’s unconscious visuomotor control and the ventral stream’s conscious visual perception.

However, VLAs don’t literally implement separate dorsal and ventral processing streams like biological systems. Instead, they computationally approximate this dual functionality through unified transformer architectures. The main idea behind VLAs is to use a VLM backbone, consisting of a vision encoder and language model, which already possesses powerful generalization capabilities and operates under free-form language. This VLM is then adapted to generate robot actions directly as tokens in the output vocabulary, treating actions like words. The vision encoder processes visual scenes to extract semantic features (analogous to ventral stream object recognition), while the language model integrates this visual understanding with linguistic instructions to plan appropriate actions (echoing dorsal stream visuomotor control). This allows robot agents to take as input a video feed and a task in free-form language and directly output robot actions.

2 Related Work

VLA models are the state of the art in robotics at the time of writing [7, 3, 14]. Older methods for any sort of autonomous robotic task relied heavily on geometry aware policies that either know, or need to explore the object geometry to grasp it and move it [2, 9]. They also needed thousands of expert demonstrations to be able to achieve good performance, which was not only costly as these expert policies are crafted carefully via teleportation, but also time consuming. But even with thousands of episodes of the same task, these robot policies could not generalize well what was learned to novel tasks [15], showing complete performance degradation even in slight changes in the environment [6].

In contrast, VLAs overcome all of these limitations inherently, through their architecture. The common baseline for most recent VLA models is a powerful VLM backbone. VLMs have been proven to generalize well in unseen domains, due to the enormous amount of data they were trained on [1, 13, 10]. They also possess nice properties like spatial reasoning. On top of that, they operate with natural, free form language instructions rather than being constrained to rigid, template-based prompts. Creating similar models but for the 3D world is near impossible, because VLMs were trained on the scale of the Internet (billions of images and captions) [11]. Thus, we would like to inherit their nice properties in a smart way, without having to gather these near impossible amounts of data for the 3D world. VLAs achieve this by adding on, and finetuning these VLMs, making them output robot actions directly, as if they were words that describe what they see [7]. This architecture allows VLAs to reason about spatial properties, occluded objects, and long horizon tasks, just like LLMs and VLMs are able to do. Moreover, their inherent generalizability allows them to complete novel tasks with very little expert demonstrations, and without ever having any knowledge about object geometries in the scene [11].

Google Deepmind is a frontier in this emerging generalist agent race that came to life with the rise of VLAs, laying the ground with some strong foundations like RT-1 [4] and RT-2 [3]. RT-1 fuses language into a vision backbone via FiLM, uses TokenLearner to reduce the fused feature maps to a compact set of learned tokens, and autoregressively predicts tokenized robot actions with a decoder-only Transformer. RT-2 expresses robot actions as text tokens within a VLM: RT-2-PaLI-X reuses numeric tokens, while RT-2-PaLM-E replaces 256 least-frequent vocabulary tokens with action symbols (”symbol tuning”), and detokenizes them to continuous controls at inference. Building on VLM-style control, OpenVLA [7] is a 7B open-source vision-language-action model that uses an open VLM backbone (Llama-2 with DINOv2+SigLIP features) and is pretrained on ∼970k demonstrations drawn from the Open-X Embodiment collection of public robot datasets. By releasing code, weights, and fine-tuning recipes, OpenVLA substantially democratizes VLA research, complementing prior open efforts such as Octo [14] and the Open-X Embodiment dataset [5]. These early approaches lack the ability to be easily finetuned and deployed, making them less practical for real applications. More recent efforts are focused on making VLAs more parameter efficient, while maintaining or even improving performance. SmolVLA [12] addresses these limitations through a compact architecture combining a lightweight vision encoder with a 2B parameter language backbone, proving that careful model design can achieve competitive performance with a fraction of the computational cost.

In this work we will do a comparative study between SmolVLA and OpenVLA. Both of these models were trained on real robot data from teleoperated expert demonstrations. All experiments in this study are conducted in a simulated environment, thus the need arises for the models to be able to adapt to this real-to-sim transition. Furthermore, we benchmark these models on a different robot than the one they where trained on. Through this we examine their ability to adapt to completely novel setups (different camera views, robot, objects) than the ones they where trained on. We also examine their zero-shot generalizability, and their ability to absorb expert demonstrations on novel tasks. To address these challenges, this work investigates the following questions:

1. What is the sample efficiency of VLA models when adapting to completely new robotic setups (different embodiment, camera views, and environment)?
2. How does model size (specifically VLM backbone capacity) affect zero-shot and transfer learning capabilities to novel manipulation tasks?
3. What are the energy-performance tradeoffs between large-scale and compact VLA architectures during fine-tuning?

3 Experimental Setup

3.1 Simulation Environment

As mentioned earlier, both models in this comparative study are originally trained on real data only coming from a variety of different robot setups and camera views. In this study we will be finetuning and evaluating these models in the LIBERO simulation benchmark [8] using a 7-DoF Franka Emika Panda arm with a wrist camera mounted on the end effector. More specifically, we will focus on the LIBERO-Spatial benchmark, which consists of 10 language prompts, each corresponding to a different task, that involve spatial reasoning. These are:

1. Pick up the black bowl between the plate and the ramekin, and place it on the plate.
2. Pick up the black bowl next to the ramekin, and place it on the plate.
3. Pick up the black bowl from the table center, and place it on the plate.
4. Pick up the black bowl on the cookie box, and place it on the plate.
5. Pick up the black bowl in the top drawer of the wooden cabinet, and place it on the plate.
6. Pick up the black bowl on the ramekin, and place it on the plate.

7. Pick up the black bowl next to the cookie box, and place it on the plate.
8. Pick up the black bowl on the stove, and place it on the plate.
9. Pick up the black bowl next to the plate, and place it on the plate.
10. Pick up the black bowl on the wooden cabinet, and place it on the plate.

The setup for LIBERO-Spatial is shown in Figure 1. In each rollout (instance) of the benchmark, the object positions and initial conditions of the arm joints are picked at random from a set of possible initial conditions, predefined by the benchmark. This means that during evaluation, each configuration is similar, yet different from what the model has seen during training.

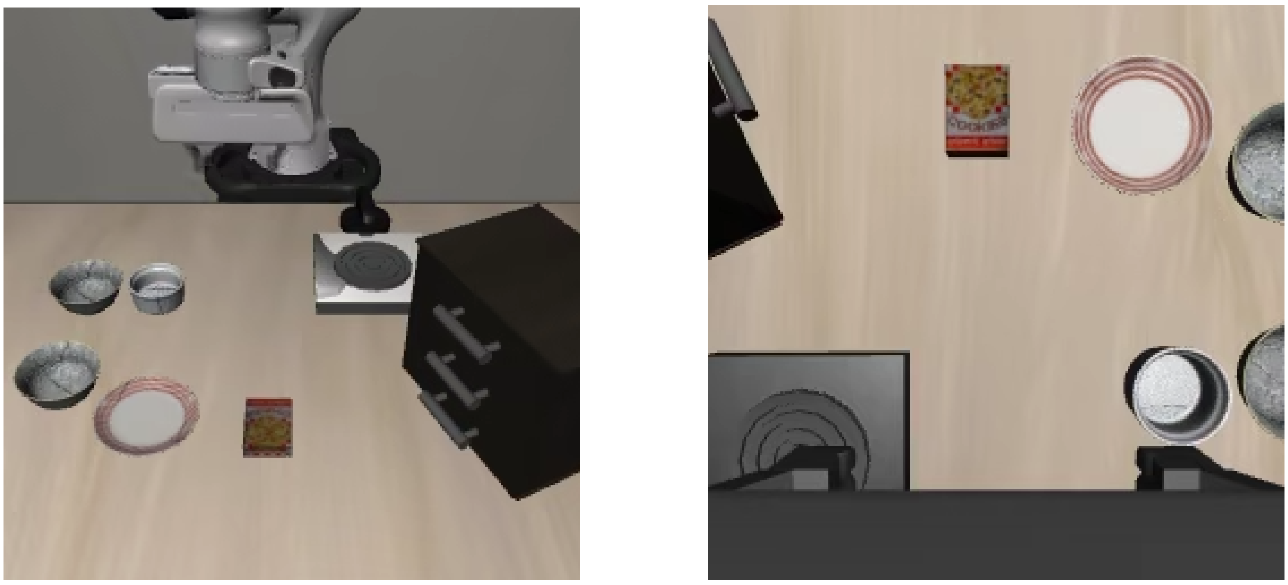

(a) Context (3rd person) camera view. (b) Wrist camera view.

Figure 1: The LIBERO-Spatial benchmark setup with a Franka Emika Panda 7-DoF arm.

3.2 Models

OpenVLA: OpenVLA [7] is built upon the Prismatic VLM framework, using a SigLIP vision encoder fused with a Llama 2 (7B) language model through a learned projector module. The architecture treats robot actions as text tokens, directly predicting 7-DoF continuous actions (end effector position, rotation, and gripper state) in a normalized action space. OpenVLA was trained on the Open X-Embodiment dataset [5], comprising over 970,000 episodes across 20 different robot embodiments and more than 1,000 distinct tasks. The training process involves first pretraining the vision-language backbone on web-scale image-text data, then fine-tuning end-to-end on the robot demonstration data, where actions are discretized into bins and treated as tokens in the language model's vocabulary.

In order to adapt the model to the environment that we described, which differs from its training setup, we need to fine-tune it on some expert demonstrations of the LIBERO-Spatial benchmark. In order to do that, we use a LIBERO-Spatial HuggingFace dataset, and bring it in RLDS format, which OpenVLA expects as input. Note that OpenVLA was trained on 3rd person view cameras (context cameras) only. For this reason we wont be using the wrist camera images for its training, as it may compromise performance. Low Rank adaptation enable us to finetune such a big model effectively with a relatively small training budget. OpenVLA was finetuned in Google Colab on an NVIDIA A100 40GB GPU.

For inference the OpenVLA project offers the option to use 4 or 8-bit quantization to save resources, at the cost of some performance loss. In their work they find that when using 8-bit quantization the performance (measured as ”Bridge Success”, meaning success rate on the BridgeData V2 dataset) drops significantly (58.1 $\pm$ 5.1%), while with 4-bit quantization the achieve performance identical to one with full float-16 accuracy at a fraction of the compute (71.9 $\pm$ 4.7% with 7.0 GB compared to 71.3 $\pm$ 4.8% with 16.8 GB). This counterintuitive performance drop in 8-bit quantization is explained because in the BridgeData V2 evaluation set the robot uses a non-blocking feedback loop, and the 8-bit quantization makes the control frequency drop significantly (1.2Hz on a A5000 GPU), which significantly changes the system dynamics compared to the training dataset for the 5Hz non-blocking controller used in the BridgeData V2 tasks [7]. Interestingly, the simpler and faster 4-bit quantization allows for a frequency comparable to the one with float-16 (3Hz on the A5000). This limitation is not present in our setup, as the we do assume no action chunking, and the benchmarks are run with a blocking controller. This allows us to use 8-bit quantization on inference and run the model on a single NVIDIA 3060 12GB GPU.

SmolVLA: SmolVLA [12] is designed as an efficient alternative to larger VLA models, using a smaller vision-language architecture while maintaining competitive performance. The model employs a SigLIP vision encoder paired with a smaller 2B parameter language model (Phi-3), connected through a projection layer. SmolVLA adopts a similar action tokenization scheme to OpenVLA, discretizing continuous 7-DoF actions into bins and predicting them as tokens. The training follows a multi-stage approach: vision-language pretraining on image-text pairs, followed by fine-tuning on the Open X-Embodiment dataset. Despite having significantly fewer parameters than OpenVLA (2B vs 7B), SmolVLA demonstrates comparable performance on manipulation benchmarks while offering substantially faster inference speeds and lower computational requirements, making it more practical for real-world deployment.

SmolVLA is compact enough to be fully trained (keeping the Vision encoder and Language model frozen) or fine-tuned on a single NVIDIA 3060 12GB GPU with a batch size of 32 compared to the 256 (64 for fine-tuning) used by the authors. For inference we create a similar evaluation script as with OpenVLA, and evaluate on the LIBERO-Spatial benchmark with a blocking controller.

3.3 Experiments

We begin by evaluating zero-shot generalization for both base models, meaning we run inference on the LIBERO-Spatial benchmark modifying the model outputs to fit our setup and using the models as they were trained originally, without any fine-tuning. We then fine-tune both models using 434 episodes of expert demonstrations from the LIBERO-Spatial benchmark. We also experiment with training SmolVLA from scratch on the same data, while keeping the Vision encoder and Language model frozen.

To test the ability of these models to absorb information and adapt to novel setups, we fine-tune using only 100 and 10 expert demonstrations on the LIBERO-Spatial benchmark. We also measure their generalization and learning abilities by evaluating the fine-tuned models on LIBERO-Object, a completely novel benchmark compared to what the models have been fine-tuned on. We then lightly fine-tune both models on 10 expert demonstrations from LIBERO-Object (1 demonstration for each task) and re-evaluate. Additionally, we analyze the training energy efficiency versus performance trade-offs between these models during fine-tuning. Considering the multifaceted analysis of performance metrics, training efficiency, and computational requirements, we conclude with evidence-based recommendations regarding the optimal model choice for different deployment scenarios.

Finally, in a brief ablation study we experiment with action chunking on SmolVLA (OpenVLA was not trained and does not support action chunking mechanisms) and how it affects performance to simulate a more realistic scenario where a non-blocking controller would be used that would require higher control loop frequencies. Additionally, we investigate OpenVLA's robustness to visual perturbations by testing its performance when the camera feed is rotated 90 degrees—a transformation not encountered during training—to assess generalization to novel viewing angles.

4 Results

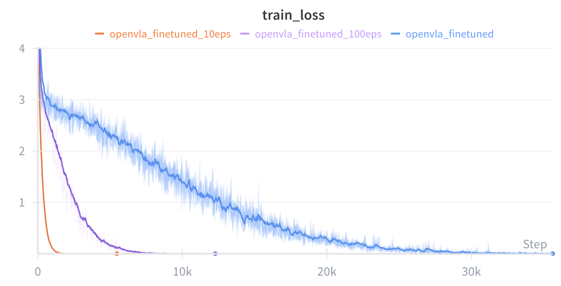
4.1 Training

We train both models on a variety of different fine-tuning regimes. This shows us how effectively each model learns as more expert demonstrations are shown to it. In detail, we aim to gradually increase the number of examples shown to the model and observe how effectively it can learn, especially on limited resources. To achieve this we experiment with these training regimes:

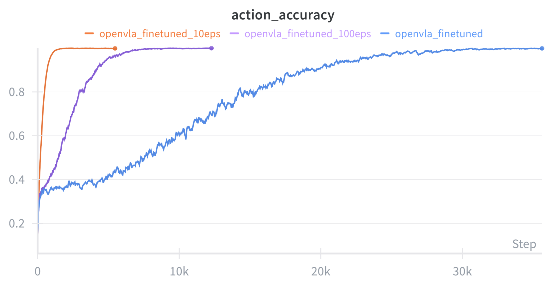
- **Zero-shot**: Using only pretrained weights and evaluating directly on the new setup and benchmarks without any fine-tuning.
- **10 episode few-shot fine-tuning**: Introducing 10 expert demonstrations on the new setup, corresponding to 1 demonstration per task.
- **100 episode few-shot fine-tuning**: Fine-tuning on 100 random expert demonstrations chosen uniformly from the complete training set.
- **Full fine-tuning**: Fine-tuning the pretrained model on the whole dataset (434 expert demonstrations for LIBERO-Spatial).
- **Scratch (for SmolVLA only)**: Keeping the Vision encoder and language model frozen, and training from scratch on the whole LIBERO-Spatial benchmark. We only do this for SmolVLA, as from scratch training for OpenVLA is too computationally heavy.

We remain conservative on the training budget for each model due to computational constraints. Interestingly, we find that this otherwise normal training strategy of stopping when the loss curve flattens and is near zero, can actually hurt performance in our case. This agrees with what the authors of OpenVLA find: ”Typical LLM or VLM training runs complete at most one or two epochs through their training dataset. In contrast, we found it important for VLA training to iterate through the training dataset significantly more times, with real robot performance continually increasing until training action token accuracy surpasses 95%” [7].

For OpenVLA, we follow the authors’ guidance and monitor action token accuracy in addition to loss, continuing training until this metric exceeds 95%. However, SmolVLA provides no such metric or training recommendations in its documentation, specifying only fixed step counts for training from scratch versus fine-tuning from pretrained weights. For instance, the SmolVLA authors recommend training for 200k steps from scratch and 20k steps for fine-tuning from pretrained weights. We only follow the latter, as training for 200k steps exceeds our computational budget. Furthermore, this guidance proves insufficient for few-shot fine-tuning scenarios, where the model learns much faster due to the limited demonstrations. The authors provide no adaptive stopping criterion for such cases, leaving unclear when training should be terminated for optimal performance. We observe that stopping training based solely on loss convergence—a conservative approach we adopt due to computational constraints—likely results in suboptimal performance for SmolVLA. This suggests that SmolVLA may benefit from extended training similar to OpenVLA. The training curves for both models can be seen in Figures 2 and 3



(a) Train loss curves for different fine-tuning regimes.



(b) Action accuracy curves for different fine-tuning regimes.

Figure 2: Training curves for OpenVLA on different expert demonstrations subsets of the LIBERO-Spatial benchmark.

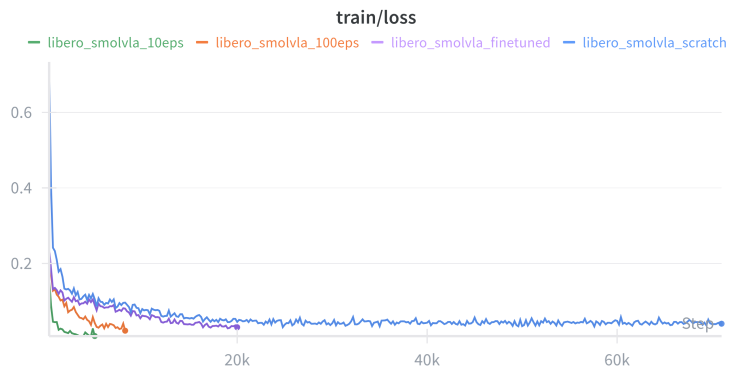


Figure 3: Training curves for SmolVLA on different expert demonstrations subsets of the LIBERO-Spatial benchmark.

The models converge significantly faster when trained on fewer examples. This exposes a key limitation as discussed earlier: performance may continue improving well beyond loss convergence, yet without adaptive metrics like OpenVLA's action token accuracy, determining optimal stopping points for few-shot fine-tuning remains unclear. Note also that for SmolVLA from scratch training we stop earlier than the 200k steps recommended by the authors due to computational limitations. In the following subsection, we analyze the trade-offs between model performance, training efficiency, few-shot adaptation, transfer learning capability, and computational cost.

4.2 Zero-Shot and Fine-Tuned models

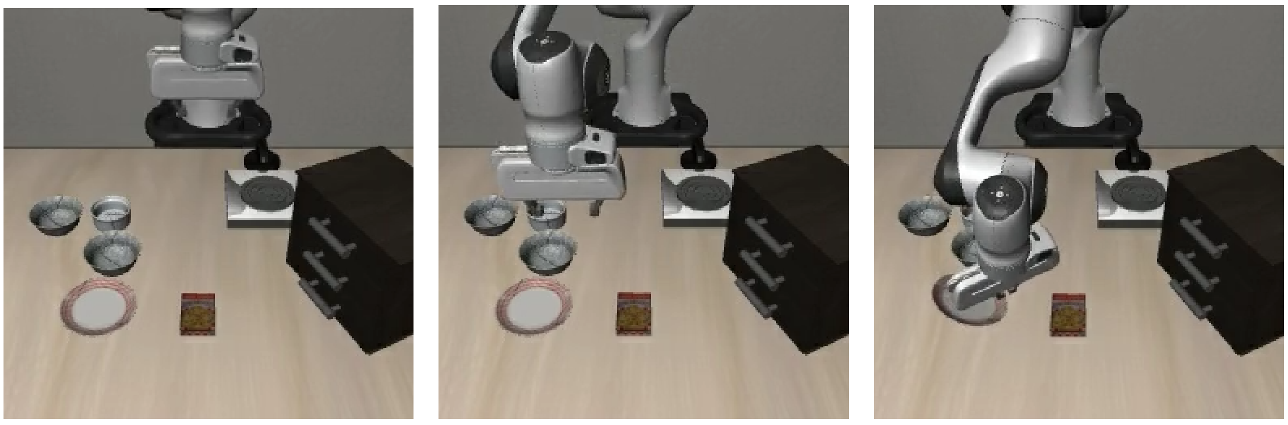
We compare performance across five settings as discussed earlier: zero-shot evaluation using only pretrained weights, few-shot fine-tuning with 10 or 100 demonstrations denoted as (10 Ep and 100 Ep), full fine-tuning on the complete dataset (denoted as Full), and training from scratch without pretraining (while keeping Vision encoder and Language model frozen, with SmolVLA only due to compute constraints). This evaluation method reveals the size of data needed for a VLA model to adapt to completely novel setups like ours. Results can be seen in Table 1.

Model	**Fine-tuning**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Overall**
OpenVLA	Zero-shot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	10 ep	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
	100 ep	45.0	10.0	15.0	35.0	40.0	20.0	45.0	45.0	45.0	10.0	31.0
	Full	78.0	82.0	74.0	74.0	54.0	62.0	80.0	76.0	62.0	64.0	71.0
SmolVLA	Zero-shot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	10 ep	0.0	0.0	0.0	0.0	18.0	4.0	2.0	70.0	0.0	0.0	9.4
	100 ep	72.0	66.0	76.0	86.0	60.0	30.0	64.0	38.0	66.0	38.0	59.6
	Full	70.0	84.0	90.0	94.0	84.0	82.0	98.0	76.0	62.0	54.0	79.4
	Scratch	62.0	86.0	76.0	94.0	92.0	78.0	100.0	84.0	72.0	82.0	82.6

Table 1: Success rates (%) for OpenVLA and SmolVLA across different fine-tuning regimes on LIBERO-Spatial tasks. SmolVLA is evaluated on 500 total roll-outs (50 per task $\times$ 10 tasks) while OpenVLA is evaluated on 200 total roll-outs (20 per task $\times$ 10 tasks).

We notice that both models perform poorly under zero-shot evaluation. When examining the robot behavior in the simulation we see that it moves randomly for both models, with no signs pointing to it trying to complete the tasks at hand. We attribute this to the fact that both models were originally trained on real environments only, and with completely different camera setups, making the real-to-sim gap too large for the model to be able to generalize.

However, with just 10 episodes on this completely new environment (only 1 demonstration per task) we can see that both models are completing some tasks successfully. Most tasks still have a 100% failure rate, but when examining the actual roll-outs we see that the robots behavior shows a clear patter. Across both models, the robot consistently attempts to move its end-effector toward the target object specified in the task. An example can be seen in Figure 4.

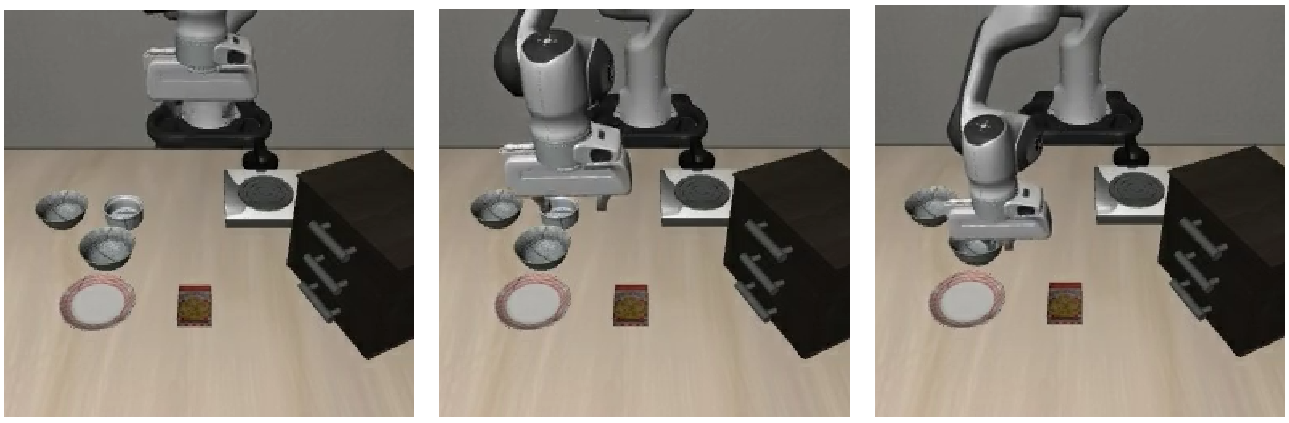


(a) Initial position. (b) EE above object. (c) Grasping failed.

Figure 4: **TASK**: ”Pick up the black bowl between the plate and the ramekin, and place it on the plate”. We observe the robot confidently moving the end-effector towards the object of interest, keeping the correct orientation needed to successfully grab the object, but eventually failing to pick it up.

For SmolVLA, we notice an unusually higher performance on tasks that involve phrases like "top of" or "on", and a 0.0% success rate on the ones where the task includes spatial relations like "next to" or "between". This may be related to the spatial relations found in the training set of the base model. This performance disparity likely stems from some demonstrations containing more diverse or complex action sequences, potentially overshadowing simpler examples during training. We find that at this level of few-shot learning where demonstrations are very few, SmolVLA tends to rely heavily on the initial selection of examples, as well as the total steps used in training. For OpenVLA we observe a more stable and predictable behavior with this fine-tuning regime, without the erratic task-specific performance spikes, but with lower overall success rates.

We notice a big performance leap when fine-tuning with 100 episodes. These episodes are picked at random, and thus could result in distributions that work in favor of some tasks more than others. We see this in both models, with performance dropping ∼30-35% in some tasks compared to others. Despite this variability, SmolVLA demonstrates superior learning efficiency, achieving a 50.2% performance improvement from its 10-episode baseline compared to OpenVLA's more modest 30% improvement over the same interval. In this training regime, we also notice that robot movements become more precise and consistent across episodes, positioning the end effector in the right position and orientation to successfully grasp objects of interest. However, we notice that the robot seems to often "dive" to pick up an object, rather than carefully lowering the end-effector to pick it up. This often results in it hitting the object of interest hard. An example is shown in Figure 5



(a) Initial position. (b) EE above object. (c) Grasping successful.

Figure 5: **TASK**: "Pick up the black bowl between the plate and the ramekin, and place it on the plate". We observe the robot confidently moving the end-effector towards the object of interest, and picking it up successfully.

The next step in our evaluation process is to fine-tune the pretrained models on the whole 434 episodes available on the LIBERO-Spatial dataset used. While for SmolVLA this is a relatively short procedure, taking ∼7.7 hours, the same cannot be said for OpenVLA which requires ∼14.5 hours of GPU time. Performance-wise, SmolVLA remains superior, although OpenVLA is not far behind. Both models fall short in matching the performance reported in their papers. OpenVLA's success rate (71%) has a large margin from the $84.7 \pm 0.9\%$ success rate reported in the paper. Similarly, SmolVLA achieves a 79.4% overall success rate, while the authors report 93.0% for LIBERO-Spatial. We attribute this performance gap to potential differences in the pre-processing of the LIBERO benchmark data. Regarding robot behavior, we notice that the robot has become much more careful with its motions, avoiding to "dive" to grasp the object of interest, and instead approaches them smoothly and picks them up successfully. A successful episode can be seen in Figure 6

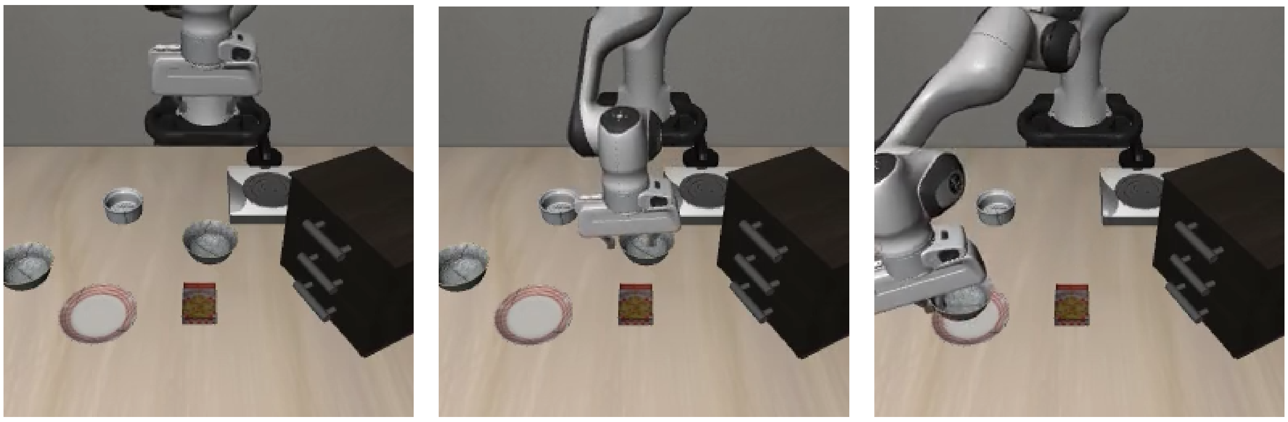

(a) Initial position. (b) Grasping successful. (c) Task completion.

Figure 6: **TASK**: "Pick up the black bowl between the plate and the ramekin, and place it on the plate". We observe the robot confidently moving the end-effector towards the object of interest, picking it up successfully without hitting it, and placing it on the plate.

Finally, we also train SmolVLA from scratch on the LIBERO-Spatial examples, keeping its VLM backbone frozen. This yields a 3.2% increase in overall accuracy, but the results reveal a trade-off pattern: while some tasks show performance gains when training from scratch, other tasks exhibit equally significant performance degradation. We attribute this to the model losing prior knowledge acquired during pretraining, which had contributed to performance on those specific tasks. As for robot behavior, we do not observe any noticeable changes compared to the one fine-tuned on pretrained weights.

4.3 Zero-Shot and Few-Shot Transfer Learning

As a result of the fine-tuning regimes presented above, the models have now adapted their capabilities to our setup. This doesn't just mean they have learn to copy given examples, but they have also developed a deeper, more meaningful understanding of the environment dynamics and setup. In this subsection we will prove this, by experimenting on their transfer learning capabilities to a completely new task.

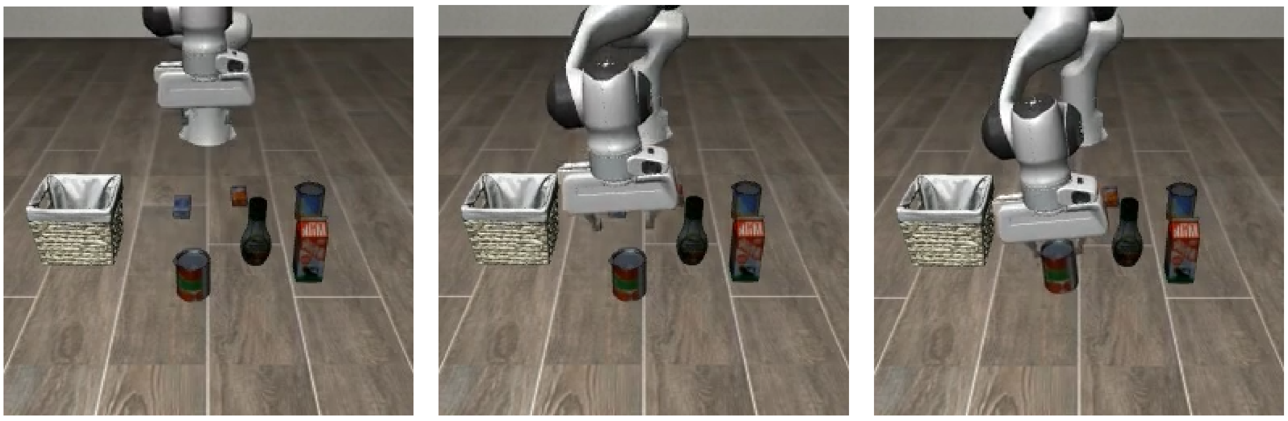
The task we will be using for this is the LIBERO-Object task, which again consists of 10 tasks that involve semantic reasoning. To set a baseline, we first evaluate zero-shot performance for both base models on this benchmark. Both base models move the robot randomly, not showing any signs of trying to complete the tasks. They achieve a 0.0% success rate in this setting.

However, after fine-tuning on LIBERO-Spatial like we described in the previous section, a crucial qualitative difference emerges in their zero-shot behavior on LIBERO-Object tasks. While both models still achieve 0.0% task completion, OpenVLA now demonstrates meaningful semantic understanding by consistently moving toward target objects (sometimes even grasping them correctly but failing to complete the pick-up motion), which is a behavior absent in the initial zero-shot evaluation (Figure 7). This suggests that fine-tuning on LIBERO-Spatial has indeed adapted the VLM capabilities to understand environment dynamics, enabling partial generalization to completely novel object manipulation tasks without any prior exposure. SmolVLA, however, doesn't follow this behavioral pattern and continues to move the arm in ways unrelated to the task. We attribute this to the significantly smaller VLM backbone that SmolVLA has (2B) compared to OpenVLA (7B). This means that in the larger VLM backbone important cues emerge that are inherited from the large scale pretraining of the VLM on the Internet, while the smaller VLM backbone lacks these traits.

This essentially means that through our fine-tuning on LIBERO-Spatial, we have enabled models with sufficiently rich VLM backbones to leverage their capabilities more effectively. Fine-tuning on LIBERO-Spatial allows the model to learn not only the specific tasks of that benchmark, but also how to understand and interact with the new environment it is operating in.

OpenVLA's zero-shot behavior suggests it is close to successfully completing these novel tasks and appears to need only a small additional push to achieve full task completion. To test this hypothesis, we fine-tune both models on just 10 episodes (1 per task) from LIBERO-Object. For fairness of comparison, while SmolVLA doesn't exhibit this emergent semantic understanding behavior due to its smaller VLM backbone, we also fine-tune it on the same data to demonstrate that it requires more examples to generalize to new tasks. Results are shown in Table 2.

As expected, OpenVLA is able to complete tasks with just 10 episodes of fine-tuning, effectively utilizing its rich VLM backbone. On the contrary, SmolVLA struggles to move the end-effector toward the correct object. This suggests much grater transfer learning capabilities for models like OpenVLA that are equipped with a larger VLM backbone.



(a) Initial position. (b) EE above object. (c) Grasping successful.

Figure 7: **TASK**: "Pick up the alphabet soup and place it in the basket". The robot demonstrates emergent semantic understanding by moving the end-effector toward the target object and successfully grasping it, though it fails to complete the pick-up motion. This behavior emerges without any prior training on similar object manipulation tasks in this setup.

Model	**Fine-tuning**	**Success Rate (%)**
OpenVLA	Zero-Shot	0.0
	10 Episodes	**5.5**
SmolVLA	Zero-Shot	0.0
	10 Episodes	0.0

Table 2: Transfer learning performance on LIBERO-Object tasks. Models were first fine-tuned on LIBERO-Spatial, then evaluated zero-shot and with 10-episode fine-tuning on novel object manipulation tasks.

4.4 Energy Efficiency

Beyond task performance, computational efficiency represents a critical factor for practical deployment of VLA models. To evaluate the energy-performance trade-offs between OpenVLA and SmolVLA, we measured GPU power consumption during training across different fine-tuning regimes and compared the resulting energy costs against achieved performance levels.

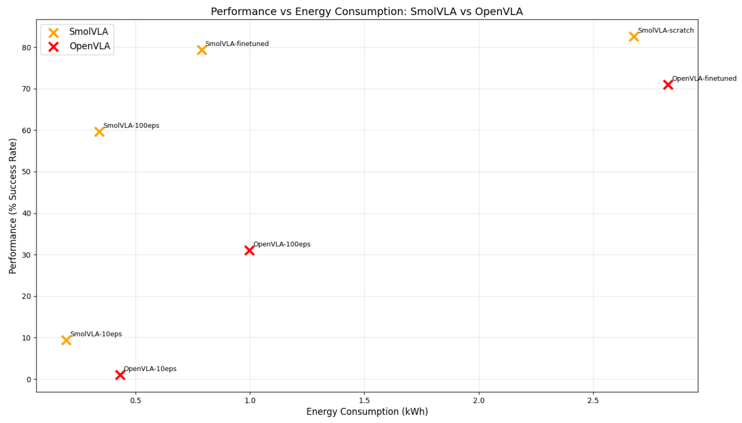


Figure 8: Energy efficiency scatter plot for different fine-tuning.

It should be noted that our SmolVLA evaluation presents the model in a favorable light, as the recommended training protocol specifies 200k steps for training from scratch and 20k steps for fine-tuning pretrained models. While we adhered to the fine-tuning guidelines, our from-scratch training was terminated at approximately 70k steps—roughly one-third of the recommended duration.

Taking this into account, we find that fine-tuning SmolVLA on the full LIBERO-Spatial dataset achieves the optimal balance between performance and energy efficiency among all evaluated configurations.

4.5 Ablation Study

4.5.1 Action chunking

Action chunking refers to the method of predicting multiple sequential actions in a single forward pass, allowing the control loop to query the model less frequently and achieve higher feedback frequencies. This approach is particularly valuable in real-time robotic applications where blocking controllers that wait for individual action predictions are impractical. Non-blocking controllers require relatively high loop frequencies to stay true to the underlying system dynamics, making action chunking an essential technique for practical deployment.

Here we evaluate performance on SmolVLA with and without action chunking (OpenVLA does not support action chunking). Results can be seen in Table 3.

Chunking Strategy	**Fine-tuning**	**Overall (%)**
No action chunking	Full	79.4
	Scratch	82.6
Action chunking = 50	Full	71.0
	Scratch	71.0

Table 3: Success rates (%) for SmolVAL with and without action chunking on LIBERO-Spatial tasks.

We notice a relatively large performance gap emerging when we are using action chunking on SmolVLA. Predicting multiple future actions simultaneously is inherently more challenging than single-step prediction.

4.5.2 Camera Rotation Robustness Evaluation

To assess OpenVLA's robustness to visual perturbations, we evaluate the model's performance when the camera feed is rotated by 90-degree intervals relative to the training setup. The results reveal a complete performance collapse, with success rates dropping to 0.0% across all tasks when any 90-degree rotation is applied. This dramatic sensitivity to camera orientation changes highlights a critical limitation in the model's visual processing capabilities. This suggests that OpenVLA has learned highly specific visual features tied to the exact camera perspective used during training, rather than developing rotation-invariant representations that would enable robust performance under different viewing angles. This sensitivity to camera orientation poses significant challenges for real-world deployment, where camera positioning may vary from the training configuration.

5 Conclusions

To conclude this study, we find that both models exceed at certain aspects, while failing in others. SmolVLA is suitable for fast adaptation to a completely new setup, and is able to outperform larger models, like OpenVLA, in this setting while remaining energy efficient and easily deployable on limited hardware. On the contrary, OpenVLA's larger VLM backbone enables it to transfer prior knowledge from large scale pretraining into completely novel tasks. This fundamental understanding that these large models posses bring them closer to the cognitive abilities of humans, paving the way for robot-human interaction.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, and et al. Flamingo: a visual language model for few-shot learning, 2022.

[2] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, and et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, and et al. Rt-1: Robotics transformer for real-world control at scale, 2023.

[5] Embodiment Collaboration, Abby O'Neill, Abdul Rehman, and et al. Open x-embodiment: Robotic learning datasets and rt-x models, 2025.

[6] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control, 2016.

[7] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas

Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024.

[8] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[9] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics, 2017.

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[11] Ranjan Sapkota, Yang Cao, Konstantinos I. Roumeliotis, and Manoj Karkee. Vision-language-action models: Concepts, progress, applications and challenges, 2025.

[12] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025.

[13] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, and et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

[14] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024.

[15] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Ayzaan Wahid, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation, 2022.