Early Fusion Helps Vision Language Action Models Generalize Better

Huang Huang[1]*† **Fangchen Liu**[1]*† **Letian Fu**[1]* **Tingfan Wu**[2]

Mustafa Mukadam[2] **Jitendra Malik**[1] **Ken Goldberg**[1] **Pieter Abbeel**[1]

Abstract: Recent advances in Vision-Language-Action (VLA) models can enable robots to perform a wide range of tasks based on language or goal-based instructions. These VLA models typically encode text and images into disjoint tokens, generating actions that align with the given instructions. This requires the VLA models to simultaneously perform vision-language understanding and precise closed-loop control, resulting in significant challenges for them to generalize to new environments. However, contrastive pre-trained VLMs, such as CLIP, already possess vision-language alignment capabilities, which are underutilized by current VLA models. In this paper, we propose Early Fusion VLA (EF-VLA), a novel VLA architecture that exploits CLIP's vision-language understanding by performing *early fusion*, extracting fine-grained vision-language tokens relevant to the task instructions before passing them to the transformer policy. EF-VLA keeps the VLM frozen, allowing it to effectively perform unseen tasks without requiring fine-tuning, which often reduces generalization capabilities. Simulation and real-world experiments suggest that EF-VLA outperforms state-of-the-art VLA models on diverse tasks, with significant generalization capabilities in unseen environments.

1 Introduction

Recent advancements in Large Language Models (LLMs) and Vision-Language Models (VLMs) have inspired the exploration of scaling datasets and computational resources for vision-language-action (VLA) models [1, 2, 3, 4]. Different input modalities are usually encoded into separate tokens: multi-view images encoded via visual feature extractors, along with tokenized language instructions, optionally with the robot's proprioceptive states, are fed into a transformer-based robot policy for end-to-end action generalization. This approach requires the policy network to connect the vision and language information and conduct precise robot control, which often presents significant challenges, especially in unseen environments.

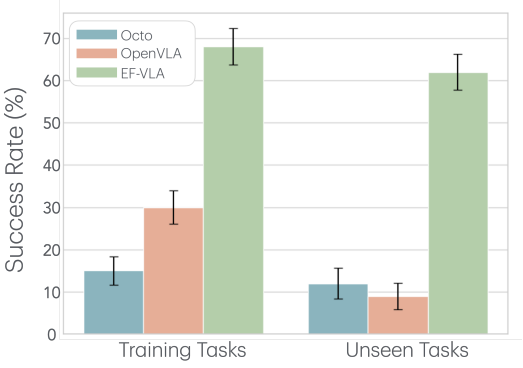

Figure 1: Real-world Robot Experiments. Early Fusion VLA (EF-VLA) demonstrates significantly higher success rates on both training and unseen real-world tasks compared to the state-of-the-art models, Octo and OpenVLA. Notably, EF-VLA exhibits better generalization to unseen objects, maintaining strong performance across a variety of novel tasks. Error bars represent the standard error calculated over 100 runs across 10 training tasks and 70 runs across 7 unseen tasks.

Numerous works [5, 4] have demonstrated the benefits of using pre-trained vision encoders or vision-language models in robotics. While these approaches already use the rich visual features extracted from pre-trained vision encoders, the policy network—often a fine-tuned language model or a transformer trained from scratch—must still learn to associate the language instructions with the visual information. However, models like CLIP [6] and SigLIP [7] are already trained to align image and text instructions, with an impressive

* Equal Contribution,†Alphabetical Order,[1]UC Berkeley,[2]FAIR, Meta

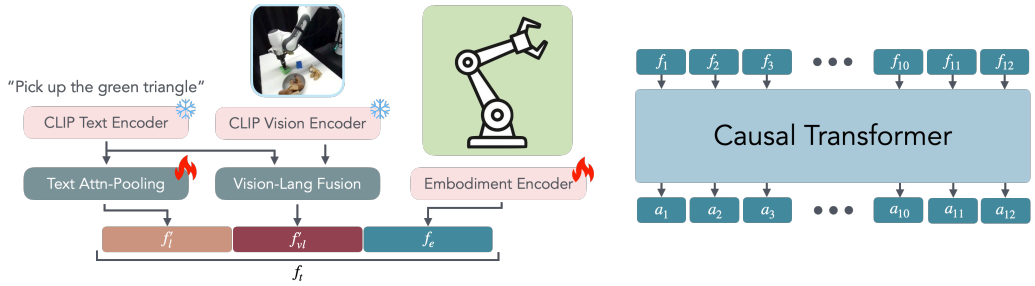

Figure 2: Model architecture of EF-VLA. At each timestep t, vision and language features are extracted by a pre-trained CLIP model and fused into a set of tokens f_{vl} (see Figure 3). The fused vision-language tokens f_{vl} and the text tokens f_l are each processed through separate attention pooling layers, producing two single tokens f'_{vl} and f'_l, respectively. The robot's proprioception is encoded by an embodiment encoder to generate the embodiment representation f_e. The tokens f'_l, f'_{vl}, and f_e are then concatenated along the channel dimension to form f_t, which serves as input to a causal transformer. Based on a context window of 12 steps, the model autoregressively predicts the next 12 actions (a_t) at each step.

performance on various downstream tasks, including image classification, image-text retrieval. It can even perform more fine-grained tasks like open-vocabulary segmentation, by extracting dense patch-level correspondence in recent works [8, 9, 10]. Given the capabilities of these VLMs, it's redundant for the policy network to learn the vision-language alignment from scratch, particularly since robot datasets are far less semantically diverse compared to large vision-language datasets [11] where these VLMs are trained on. Additionally, despite the effort these large VLAs to generalize to unseen tasks, there still exists a performance discrepancy between training tasks and unseen tasks. Some prior works such as OpenVLA [4] have shown that fine-tuning the vision encoder is critical for improving its performance on new tasks. However, fine-tuning, especially for language-aligned encoders like CLIP, introduces a critical trade-off: it can impair generalization and long-tail classification performance [12, 13, 9], posing notable over-fitting issues.

We seek to preserve the generalization capabilities of VLMs for effective performance under unseen scenarios. To this end, we propose Early Fusion VLA (EF-VLA), a novel VLA architecture that exploits CLIP's vision-language understanding by performing *early fusion*, fusing vision and language tokens before, rather than in, the policy network (*late fusion*). Furthermore, the extracted vision-language information should contain dense spatial information for guiding the robot policy to generate accurate actions. To extract dense and semantically meaningful vision-language features, we adopt the architecture from ClearCLIP [9], where we directly use the clean text-patch correspondence as our frozen vision-language representations, preserving the inherent vision-language understanding ability of the pre-trained VLM to a large extent.

Figure 2 provides an overview of EF-VLA. In addition to the ClearCLIP model, EF-VLA also encodes language instructions and the robot's proprioceptive states. The policy network receives the fused vision-language token, a language token, and the proprioception token to autoregressively predict actions in a causal transformer. Importantly, we keep the CLIP model frozen during training to preserve its pre-trained powerful vision-language alignment. Both physical and simulation experiments show that EF-VLA significantly outperforms existing VLA models, demonstrating superior generalization to novel objects and environments with minimal performance degradation (Figure 1).

To summarize, our contributions are:

1. we propose EF-VLA, a VLA model that performs fine-grained early-fusion of vision and language information. It leverages a pre-trained CLIP model with ClearCLIP architecture to extract fine-grained vision-language features for effective performance on robotic tasks.
2. EF-VLA can outperform the state-of-the-art VLA models and its ablations on diverse robot manipulation tasks. More significantly, EF-VLA can perform unseen tasks in a zero-shot manner without the need to finetune vision encoders, which maximally preserves and leverages the superior generalization capabilities of pre-trained vision-language models.

2 Related Work

2.1 Vision Language Pre-training

Vision-language pre-training (VLP) seeks to improve the performance of downstream tasks that involve both vision and language by training models on extensive datasets of image-text pairs. A prominent class of vision-language models leverages contrastive learning [14, 15, 16, 6, 17, 18, 7]. Among them, CLIP [6], which was trained on a private WIT-400M dataset of image-text pairs, demonstrates impressive zero-shot capabilities across various downstream tasks, including image-text retrieval and image classification through text prompts. Furthermore, CLIP shows potential for application in broader fields such as decision-making and robotics, where robots are required to perform language-specified tasks based on visual inputs. Recent early-fusion approaches, exemplified by BLIP [19, 20], extract visual features using a language-aligned vision model and apply multilayered cross-attention between encoded language features and visual features. The resulting features are then passed into a language model. However, many researchers have observed that fine-tuning or even applying additional layers on top of CLIP (instead of using raw CLIP features) [12, 9] may result in models with weaker reasoning capabilities compared to vanilla CLIP.

2.2 Vision Language Action Models

In recent years, there has been a surge of interest in developing robot foundation models, largely inspired by the success of large language models (LLMs) and vision-language models (VLMs) [21, 22, 23, 24, 25, 26, 6, 20]. A key hypothesis driving this trend is that more capable robot foundation models can emerge by scaling up robot datasets, increasing model capacity, and co-training or pre-training models on vision and language datasets. This has led researchers in the robot learning community to train robot foundation models, investigate pre-training strategies, and iterate on model designs [27, 5, 4, 3, 28, 29, 30, 1, 31, 32].

Many existing VLMs [33, 34, 35] use a "late-fusion" approach, where visual features and languages are directly passed into the LLM to generate answers. Similarly, the majority of Vision-Language-Action (VLA) models also opt for late-fusion, where language, vision, and robot proprioception data are separately encoded by modality-specific feature extractors before being fed into a single transformer policy. This method has shown promise in many language-conditioned multi-task learning models [29, 5, 28, 30, 1, 31], including current open-source state-of-the-art models such as Octo [3] and OpenVLA [4]. In contrast to the late-fusion approach, "early-fusion" combines vision and language inputs before feeding them into the language model or during visual feature extraction. Early works such as FiLM [36] encode text information and fuse these features into each block of a ResNet [37]. RT-1 [27], one of the first language-conditioned robot models, uses FiLM to encode text information for action generation. However, FiLM and RT-1 need to learn the language-vision alignment from task data, thus cannot leverage pre-trained models such as CLIP [6], where visual features are already aligned with text.

Inspired by ClearCLIP [9], EF-VLA distinguishes itself by using a similarity-based fusion between visual patch features and text token features from CLIP while also incorporating additional text tokens and robot embodiment tokens as inputs to the robot policy. This approach allows us to leverage the strengths of fine-grained features from the pre-trained vision-language models while maintaining the flexibility to incorporate robot-specific information.

3 Method

We propose Early Fusion VLA, a vision-language-action model for learning a robot manipulation policy through early fusion on the vision-language features. We first describe how EF-VLA employs early-fusion between the vision and language modalities, then provide a more detailed explanation of the model architecture.

3.1 Vision-Language Early Fusion

EF-VLA utilizes a pre-trained CLIP for vision-language fusion. Consider a ViT-based CLIP vision encoder [6] consisting of a series of residual attention blocks. Each of these blocks takes as input a collection of visual tokens X, and outputs the feature X_{out} as shown below:

$$q = \text{Proj}_q(\text{LN}(X)), \quad k = \text{Proj}_k(\text{LN}(X)), \quad v = \text{Proj}_v(\text{LN}(X)) \tag{1}$$

$$X_{\text{sum}} = X_{\text{res}} + X_{\text{attn}} = X + \text{Proj}(\text{Attn}(q, k, v)) \tag{2}$$

$$X_{\text{out}} = X_{\text{sum}} + \text{FFN}(\text{LN}(X_{\text{sum}})) \tag{3}$$

Proj, LN, and FFN denote linear projection matrix, layer norm [38], and feed-forward network respectively. A recent work ClearCLIP [9] shows improved training-free open-vocabulary segmentation performance by using CLIP's last self-attention block's attention feature X_{attn} instead of the CLIP's output feature X_{out}, resulting in segmentation with less noise. Inspired by ClearCLIP, we use a parameter-free method to extract task-relevant CLIP features.

In EF-VLA, we extract text per-token features from CLIP's language encoder f_l (m tokens). For the visual features, motivated by the improved ability to capture text-aligned semantics in the visual features as shown in ClearCLIP, we specifically utilize the attention output X_{attn} from the last vision attention layer, rather than the CLIP's output feature X_{out}, denoting it as f_v (n tokens), where n is the total number of patch tokens from ViT. To illustrate the effectiveness of this approach, we provide a visualization in Figure 6 that demonstrates how using X_{attn} enhances the alignment between visual features and language semantics.

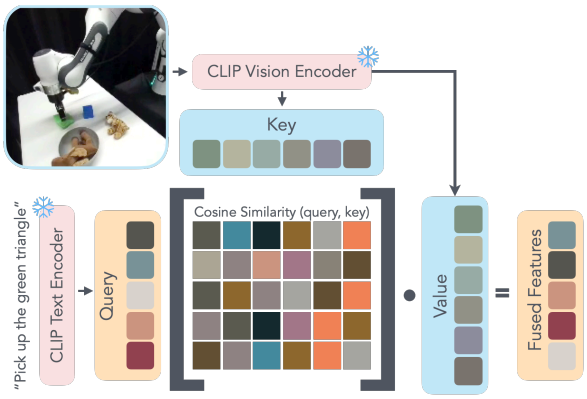

Figure 3: Vision-Language Early Fusion We calculate the similarity between the visual patch features and per-token language features, then take the softmax over the patch feature dimension. Intuitively, this give a distribution of semantic similarity over all spatial locations. We then multiply the visual patch features to retrieve the visual semantic features that correspond to each token in the sentence.

Since the language features and the visual features have different dimensions, CLIP uses a matrix per modality to project the network's output feature to the same latent dimension, denoted as w_l and w_v for language and vision respectively. We normalize the text and visual features for vision-language fusion. The text features are normalized using the final layer normalization: $\hat{f}_l = \text{LN}_{\text{final}}(f_l)w_l$. Similarly, the visual features are normalized using the post-attention layer normalization: $\hat{f}_v = \text{LN}_{\text{post}}(f_v)w_v$. We apply L2 normalization to both text and visual features: $\hat{f}_l = \hat{f}_l / \|\hat{f}_l\|_2$ and $\hat{f}_v = \hat{f}_v / \|\hat{f}_v\|_2$ as in standard CLIP.

With the normalized features, we perform temperature-weighted attention:

$$f_{vl} = \text{softmax}(\hat{f}_l \hat{f}_v^\top / \tau, \text{dim=1}) \hat{f}_v \tag{1}$$

where τ is the temperature parameter. Same as in CLIP [6], τ is learnable and is clipped between 0 and 100. The resulting feature $f_{vl} \in \mathbb{R}^{m \times d}$ are the fused vision-language tokens, where each row is a linear combination of normalized visual features $\hat{f}_v$. Intuitively, the *softmax* serves as a selection function, where patch features relevant to a particular language token are selected, and a weighted average of these patches is calculated to provide cues to where the robot policy should pay attention to. A smaller τ sharpens the *softmax*, concentrating the selection on the patch with the most similar feature, while a larger τ produces a smoother, more evenly distributed selection across patches. Critically, all parameters except the τ are *frozen* throughout the training.

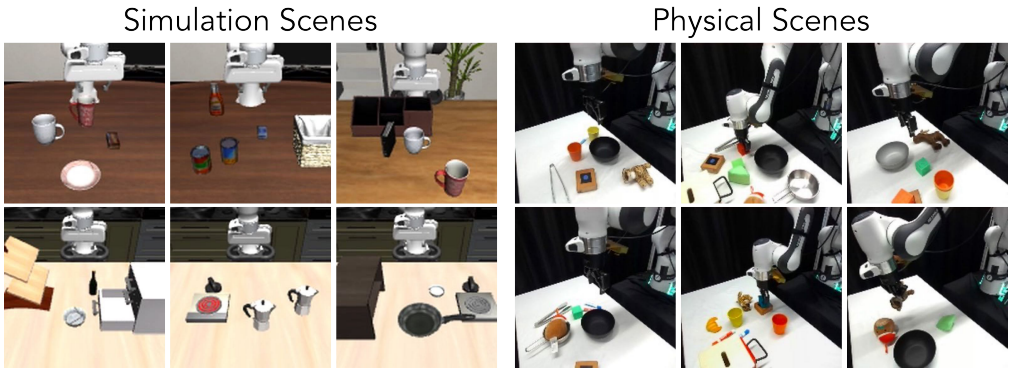

Figure 4: Example scenes in the simulation (left) and in the physical environments (right) using a Franka robot.

3.2 Model Architecture

Policy Network Input We compress the fused vision-language features f_{vl} into a single token for each camera. To achieve this, we apply a *learnable* cross-attention pooling operation to each camera's f_{vl} to obtain a single feature f'_{vl}. To facilitate both early and late fusion of language features for better instruction following capabilities, we additionally employ another *learnable* cross-attention pooling on the text features f_l, resulting in a single text token $f'_l \in \mathbb{R}^{d_l}$. The robot's proprioceptive state is encoded through an FFN to extract an embodiment feature f_e. At time step t, we concatenate the embodiment feature f_e with the perception feature f'_l and f'_{vl} along the channel dimension to create a single token f_t. This token serves as input to a policy network for action prediction.

Policy Network and Action Head Our policy model is a transformer consists of 4 layers and 8 heads, with a hidden dimension of 512. Fed by the combined features from the perception and embodiment, the model generates an action a_t. The model is trained with a context length of 12 steps. For each output token at a given timestep, we use an FFN to predict the next 12 actions. More details about our model architecture can be found in Appendix B. When executing the predicted actions, we employ temporal ensembling [39] in conjunction with receding horizon control [40]. Through experimentation, we determined that an action horizon of 8 steps yields optimal performance.

4 Experiments

We address two problem classes: language-conditioned multi-task learning and zero-shot generalization in unseen environments. In language-conditioned multi-task learning, the policy must perform the correct task from multiple possible tasks in the same scene based on a given language instruction. In zero-shot generalization, the policy receives a language description of an unseen task and is required to perform it in a novel environment. In this section, we first introduce our experimental setup in Section 4.1. We compare EF-VLA against several baseline and ablation models in simulation and real-world in Section 4.2 and Section 4.3. In Section 4.4, we further investigate EF-VLA's capabilities by scaling up models.

4.1 Environment Setup

Simulation Environment We use the LIBERO benchmark [41] for simulation evaluation. Specifically, we use LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-90 as the pre-training dataset, which contains 120 tasks with diverse objects, scene layouts, and language instructions. Each simulation task has 50 demonstrations. We evaluate EF-VLA's capabilities on both in-distribution tasks and unseen tasks. The in-distribution tasks are the 30 tasks in the original LIBERO-Spatial, LIBERO-Object, and LIBERO-Goal, which can evaluate the model's multi-task learning capabilities. In addition, we also construct 10 novel tasks, where we modify the language instructions and corresponding objects of 10 original LIBERO-90 tasks. For the 10 unseen tasks, we follow the same convention in LIBERO [41] about object initialization and goal configuration by defining task bddl files. Example scenes in the simulation are shown in the left column of Figure 4.

Method	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	Unseen
EF-VLA w.o. CLIP vision	59% ± 7.3%	62% ± 7.8%	68% ± 6.3%	29% ± 8.7%
LF-VLA	72% ± 9.2%	51% ± 7.4%	**76% ± 8.4%**	28% ± 11%
EF-VLA w.o. f_e	62% ± 6.3%	58% ± 9.1%	61% ± 8.7%	48% ± 7.8%
EF-VLA w.o. f_l'	61% ± 9.9%	47% ± 9.4%	57% ± 10.3%	49% ± 9.9%
EF-VLA (Ours)	**71% ± 7.3%**	**64% ± 9.2%**	73% ± 9.4%	**59% ± 7.4%**

Table 1: Simulation results on LIBERO. We evaluate EF-VLA and baselines on 300 trials on in-distribution tasks, and 100 trials on unseen tasks.

Method	Training Tasks	Unseen Tasks
Finetuned Octo	15% ± 3.4%	12% ± 3.6%
EF-VLA w.o. CLIP vision	17% ± 2.9%	11% ± 2.5%
Finetuned OpenVLA	30% ± 3.9%	9%±3.1%
LF-VLA	29% ± 3.7%	4% ± 1.6%
EF-VLA (Finetune CLIP)	26% ± 4.0%	15% ± 3.9%
EF-VLA w.o. f_e	40% ± 4.0%	29% ± 4.3%
EF-VLA w.o. f_l'	57% ± 4.4%	53% ± 4.6%
EF-VLA (Ours)	**68% ± 4.3%**	**62% ± 4.2%**

Table 2: Physical results on 100 trials on in distribution training tasks and 70 trials on unseen tasks. EF-VLA achieves similar success rate on the in distribution training tasks and unseen tasks, significantly outperforming the baselines, highlighting the benefits of using early fusion and a frozen pre-trained VLM.

Real Robot Environment For real-robot evaluation, we assess models on pick-and-place tasks with varying objects and target locations. Using a Franka robot, we collected a dataset of 10 tasks, each with 50-80 human tele-operated demonstrations, totaling 724 demonstrations. We evaluate 10 training tasks and 7 unseen tasks, where unseen tasks involve novel object and placement combinations. Example scenes in real are shown in the right column of Figure 4. Each trial involves randomizing object locations and the two distractor objects. For unseen tasks, novel objects or object-placement combinations are introduced. We generate 10 randomized scenes per task, resulting in 100 trials for training tasks and 70 for unseen tasks. Performance is scored based on task completion: 0.5 for picking the correct object and 1 for also placing it correctly. Failure to pick the correct object scores 0. Models have 30 seconds per trial, except OpenVLA, which gets 60 seconds due to its slower inference. Success rate averages are reported with standard error across all trials. The full lists of simulation and real-world environments can be found Appendix A.

4.2 EF-VLA V.S. Late-fusion VLA

To evaluate if the early fusion in EF-VLA can better leverage the semantic understanding capabilities of the pre-trained VLMs, we consider three baselines with late-fusion architectures, including two state-of-the-art open-sourced VLA models and one late fusion variant of EF-VLA:

1. Octo [3], an open-sourced transformer-based policy trained from scratch on 800K trajectories from the Open X-Embodiment dataset [1].
2. OpenVLA [4], a fine-tuned Prismatic-7B [35] VLM on the Open X-Embodiment dataset.
3. LF-VLA: a late fusion variant of EF-VLA where the text tokens, vision tokens are passed to an attention pooling layer separately to obtain independent tokens, which are then concatenated with the embodiment feature f_e as the input to the transformer.

As Octo and OpenVLA are pre-trained on a real robotics dataset, we evaluate both models in the physical environments. For fair comparisons, we train LF-VLA from scratch and fine-tune Octo and OpenVLA on our real robot dataset using the same amount of learning steps. The physical experiment results are reported in Table 2. For more details about model training and architectures, please refer to Appendix B.

In both training and unseen tasks, Octo struggles with identifying the correct object and placement, resulting in a low success rate. We attribute this to two factors: first, Octo lacks a pre-trained VLM like CLIP and instead trains its vision encoder from scratch using a robotic dataset (OXE [1]), which lacks the semantic diversity of larger datasets like LAION [11]. Second, EF-VLA's early-fusion strategy on CLIP's visual and text representations leads to better vision-language alignment, improving its visual grounding and generalization, even with a smaller dataset. While OpenVLA and LF-VLA perform better than Octo on training tasks, they still fail to generalize on unseen tasks, likely due to the challenges late-fusion architectures face in learning generalizable vision-language connections on small datasets. In contrast, EF-VLA benefits from early-fused features from the pre-trained VLM.

We also compare LF-VLA with EF-VLA in simulation as shown in Table 1. On LIBERO-Spatial and LIBERO-Goal, LF-VLA and EF-VLA work similarly well. That's because the task semantics in LIBERO-Spatial and LIBERO-Goal can be easily distinguished. However, on LIBERO-Object, LF-VLA is worse than EF-VLA because the objects are very similar, and LF-VLA cannot accurately find the correct object to interact with. On unseen tasks, EF-VLA can outperform LF-VLA by a large margin, which is aligned with the real-world experiments.

4.3 Ablations on Model Design

We consider the following ablations on the design choices of EF-VLA. Full details about model training and architectures can be found in Appendix B.

1. EF-VLA w.o. f_e: EF-VLA without the embodiment representation f_e. The concatenated text token f_l' and fused vision-language token f_{lv}' are passed as the input to the transformer.
2. EF-VLA w.o. f_l': EF-VLA without the text token f_l'. Only f_{lv}' and f_e are concatenated as the input to the transformer.
3. EF-VLA w.o. CLIP vision: EF-VLA using a small VIT to train from scratch instead of a frozen pre-trained CLIP vision encoder.
4. EF-VLA (Finetune CLIP): EF-VLA with the CLIP initialized from the pre-trained weight and fine-tuned end to end on the robotic dataset.

Simulation Results Table 1 presents the simulation results of EF-VLA and other ablations. On the in-distribution tasks, EF-VLA w.o. CLIP vision and EF-VLA work similarly well given sufficient demonstrations, but EF-VLA w.o. CLIP vision drops 51% on unseen tasks, which shows the benefits of using a pre-trained VLM for better generalization capabilities.

The performance of EF-VLA w.o. f_e drops about 10% on both the in-distribution and unseen tasks, indicating that f_e is beneficial for task completion as it provides explicit spatial information of the robot. EF-VLA w.o. f_l' is also noticeably worse, especially for LIBERO-Object and LIBERO-Goal. We hypothesize this is due to the object are not very realistic in simulation, so the early fusion in CLIP's may highlight multiple objects or wrong objects. f_l' can provide complementary information for the transformer to interact with the correct objects.

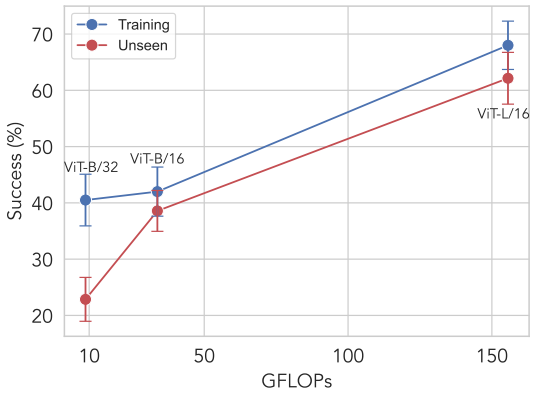

Figure 5: We evaluate EF-VLA's performance with improved vision language features by scaling CLIP. In particular, we train EF-VLA with three CLIP variants with increasing FLOPs: ViT-B/32, ViT-B/16, and ViT-L/16. We report the task performance vs. the inference FLOPs per image on training and unseen tasks. The results suggest that the EF-VLA can benefit from scaling up vision-language model.

Real-world Results Physical results in Table 2 suggest that the performance on both the training tasks and the unseen tasks drop significantly for the ablations compared to EF-VLA.

Similar to the simulation results, the performance of EF-VLA w.o. f_e drops 28% on the training tasks and 33% on the unseen tasks, indicating that f_e is vital for task completion and generalization, likely because it provides a physical grounding for decision-making. Without f_e, the model's understanding of embodied features, possibly linked to the spatial or physical aspects of the task, is severely impaired. EF-VLA w.o. f_l' experiences a performance drop of around 10% on both training and unseen tasks but maintain a decent performance, suggesting that f_l' provides complementary information that may help in more nuanced task understanding, aligned with the simulation results.

While EF-VLA w.o. CLIP vision shows decent performance on in distribution tasks in simulation experiments, it has a significant performance drop of more than 50% on the training and unseen tasks in physical experiments. The results of EF-VLA w.o. CLIP vision is similar to Octo which also trains a vision encoder from scratch on the robotics dataset. This suggests that pre-trained VLM provides more robust and transferable visual representations. Training a vision encoder from scratch can result in poor performance, as it lacks the generalization capabilities learned from large-scale pre-training.

OpenVLA suggests that fine-tuning the vision encoder of the pre-trained VLM on the robotics dataset is crucial for improving the performance of a late fusion VLA. However, we hypothesize that fine-tuning a pre-trained VLM can diminish the general vision-language understanding capabilities of a VLM obtained through pre-training on internet-scale vision language datasets. EF-VLA (Finetune CLIP) shows worse performance on both the training tasks and the unseen tasks. This may be attributed to that a fine-tuned CLIP vision encoder is easier to over-fit on the training data and that a fine-tuned CLIP vision encoder has a degraded vision-language understanding capabilities. The large performance discrepancy between the training tasks and unseen tasks of OpenVLA and EF-VLA (Finetune CLIP) implies a worse vision-langauge generalization ability, showing the benefits of EF-VLA for retaining the vision-language features from a frozen pre-trained VLM. It's worth noting that both early fusion of the vision-language features and the frozen VLM is crucial for learning a VLA that can generalize to unseen tasks, as shown by the worse performance of LF-VLA with a frozen VLM, EF-VLA (Finetune CLIP) that has a fine-tuned VLM and OpenVLA that is a late fusion model with fine-tuned VLM.

4.4 Scaling up Vision-Language Model

The semantic understanding capability of VLMs scales with model capacity and compute [6]. To understand whether EF-VLA can leverage the advances of pre-trained VLMs, we evaluate its performance with three CLIP models with increasing floating point operations per model forward pass: ViT-B/32, ViT-B/16, and ViT-L/14. The task success and the inference FLOPs per image are provided in Figure 5. We observe significant improvements of EF-VLA when scaling up CLIP for training and unseen tasks, indicating that EF-VLA is a scalable approach that effectively utilizes pre-trained vision language models for downstream robotics tasks.

5 Limitations and Conclusions

While EF-VLA shows improved task completion rates compared to existing VLAs, it has limitations. A key challenge is scaling to different morphologies, especially those not easily parameterized by SE(3) transforms (e.g., multi-fingered hands), limiting its adaptability to various robotic platforms and tasks. Additionally, this study has not investigated how EF-VLA scales with larger datasets or more complex tasks. Future works can investigate its performance and generalization in more challenging scenes.

In summary, we present EF-VLA, a vision-language-action model that implements early fusion between vision and language features. This is achieved by utilizing a pre-trained vision-language model and an early fusion method to extract task-relevant semantic information. The experimental results demonstrate that this early fusion approach enables effective multi-task learning with few demonstrations and facilitates extrapolation to unseen objects and environment configurations. The results further suggest that EF-VLA has a higher task success rate in handling unseen scenes with distractor objects than the existing state-of-the-art VLAs.

Acknowledgement

Dataset usage and model training work are solely conducted at UC Berkeley.

References

[1] E. Collaboration, A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2024.

[2] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2024.

[3] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model, 2024. URL `https://arxiv.org/abs/2406.09246`.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[6] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[7] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.

[8] Y. Rao, W. Zhao, G. Chen, Y. Tang, Z. Zhu, G. Huang, J. Zhou, and J. Lu. Denseclip: Language-guided dense prediction with context-aware prompting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[9] M. Lan, C. Chen, Y. Ke, X. Wang, L. Feng, and W. Zhang. Clearclip: Decomposing clip representations for dense vision-language inference. In *ECCV*, 2024.

[10] X. Dong, J. Bao, Y. Zheng, T. Zhang, D. Chen, H. Yang, M. Zeng, W. Zhang, L. Yuan, D. Chen, et al. Maskclip: Masked self-distillation advances contrastive language-image pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10995–11005, 2023.

[11] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[12] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023.

[13] A. Rashid, S. Sharma, C. M. Kim, J. Kerr, L. Y. Chen, A. Kanazawa, and K. Goldberg. Language embedded radiance fields for zero-shot task-oriented grasping. In *7th Annual Conference on Robot Learning*, 2023. URL `https://openreview.net/forum?id=k-Fg8JDQmc`.

[14] J.-B. Alayrac, A. Recasens, R. Schneider, R. Arandjelović, J. Ramapuram, J. De Fauw, L. Smaira, S. Dieleman, and A. Zisserman. Self-supervised multimodal versatile networks. *Advances in neural information processing systems*, 33:25–37, 2020.

[15] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, C. Schuhmann, L. Schmidt, and J. Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829, 2023.

[16] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021.

[17] L. Yao, R. Huang, L. Hou, G. Lu, M. Niu, H. Xu, X. Liang, Z. Li, X. Jiang, and C. Xu. Filip: Fine-grained interactive language-image pre-training. *arXiv preprint arXiv:2111.07783*, 2021.

[18] L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

[19] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

[20] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[22] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

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

[24] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[25] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[26] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[27] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv:2212.06817*, 2022.

[28] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2022.

[29] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. VIMA: General robot manipulation with multimodal prompts. *International Conference on Machine Learning (ICML)*, 2023.

[30] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[31] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. ViNT: A Foundation Model for Visual Navigation. In *7th Annual Conference on Robot Learning (CoRL)*, 2023.

[32] L. Fu, H. Huang, G. Datta, L. Y. Chen, W. C.-H. Panitch, F. Liu, H. Li, and K. Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024.

[33] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[34] H. Laurençon, L. Tronchon, M. Cord, and V. Sanh. What matters when building vision-language models? *arXiv preprint arXiv:2405.02246*, 2024.

[35] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024.

[36] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[37] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[38] J. L. Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[39] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[40] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[41] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] OpenAI. Gpt-4o system card. `https://cdn.openai.com/gpt-4o-system-card.pdf`, 2024. Accessed: 2024-09-14.

A Environment Setup

A.1 Simulation Tasks

For the training tasks, we use the original tasks in LIBERO-Goal, LIBERO-Spatial, and LIBERO-Object. We also build unseen evaluation tasks based on 10 original LIBERO-90 tasks, by changing language instructions and target object color and type in the task bddl files. The 10 unseen tasks are listed in Table 3.

Changes	Unseen
object type	Put the moka pot in the bottom drawer of the cabinet
object type	Put the moka pot on the wine rack
object type	Pick up the ketchup and put it in the basket
object type	Pick up the ketchup on the plate
object type	Pick up the bottle and put it in the tray
object color	Put the black bowl on top of the cabinet
object color	Put the black bowl on the plate
object color	Put the red mug to the right of the plate
object color	Put the yellow and white mug in the front of the red mug
object color	Put the red mug to the front of the moka

Table 3: The 10 in-distribution tasks and 7 unseen tasks we used in our real-world setting.

A.2 Real-world Tasks

The full list of tasks for our real-world evaluation is provided in Table 4.

In-Distribution	Unseen
Put potato in pot to black bowl	Put yellow cube in black bowl
Pickup potato	Pick up radish and place it in grey bowl
Pick up and place deer in grey bowl	Put blue bear in pink bowl
Pick up green triangle	Put yellow cube in grey bowl
Put tiger to black bowl	Put apple with a green leaf in black bowl
Put red cube into black bowl	Pick up blue sponge and place it in steel pot
Put blue cube into grey bowl	Pick up black dog and place it in the pink bowl
Put the red ball in black bowl	
Put green triangle into pink bowl	
Put blue cube in pink bowl	

Table 4: The 10 in-distribution tasks and 7 unseen tasks we used in our real-world setting.

B Model and Training Details

B.1 Proprioception Parametrization

We parameterize the proprioception space using a 10-dimensional representation. This includes the absolute end effector translation (x, y, z), a 6DoF rotation vector, and a continuous end-effector gripper state. The 6DoF rotation vector is derived by flattening the first two rows of the $SO(3)$ rotation matrix.

B.2 Action Parametrization

We employ delta end effector pose as our action parameterization. At each prediction step, the model predicts t actions. Given a sequence of *absolute* end effector action transforms $T_1, T_2, \cdots, T_t$ in a trajectory and the current end-effector pose T_{ee}, we define the relative transforms that the model needs to predict as $T_{\text{ee}}^{-1}T_1, T_{\text{ee}}^{-1}T_2, \cdots T_{\text{ee}}^{-1}T_t$. We then append the continuous absolute gripper position to each delta action. Similar to the proprioception representation, we express the delta action using the relative end effector translation and a 6DoF rotation vector, resulting in a 10-dimensional action representation.

B.3 Model Architecture for EF-VLA and Baselines

The details of our model parameters can be found in Table 5. All the baselines share the same hyper-parameters with EF-VLA. For EF-VLA w.o. CLIP Vision, we use a ViT Encoder based on the implementation of `https://github.com/google-research/vision_transformer` with a ViT-Ti/16 configuration with half of the number of attention layers. For EF-VLA w.o. f_e and EF-VLA w.o. f_l', we use the same model configuration but only remove the corresponding attention pooling layers. We incorporate action chunking into OpenVLA by asking it to predict the next 16 actions, which performs better than vanilla OpenVLA which predicts only the next step. For Octo, we use the official Hugging Face Checkpoint at `hf://rail-berkeley/octo-small-1.5` which is in a comparable size with our model. During inference, we cache the CLIP feature outputs. This enables the ViT-L/14 EF-VLA model to perform inference at $> 15Hz$ on a single NVIDIA 3090Ti, allowing real-time control.

Hyperparameter	Value
CLIP Model	ViT-L/14
# Pooling Readouts	4
# Pooling Attention Heads	8
# Pooling Attention Blocks	2
# Text-Pooling Output Dimension	128
# Image-Pooling Output Dimension	512
# Proprio-Pooling Output Dimension	64
Causal Transformer Parameters:	
# Attention Blocks	4
# Attention Heads	8
# Latent Dimension	512
# Context Length	12
# Action Prediction Horizon	12

Table 5: Hyperparameters in model architecture.

B.4 Training Hyper-parameters

We use the AdamW optimizer with a cosine learning rate decay schedule and linear learning rate warm-up. We list training hyperparameters in Table 6. All these hyper-parameters are shared between real-world and simulation. All the models are trained on 4 NVIDIA A100 80GB GPUs.

Hyperparameter	Value
Learning Rate	3e-4
Warmup Steps	2000
Weight Decay	0.01
Learning Rate Scheduler	cosine
Gradient Clip Threshold	1
Batch Size	64
Total Gradient Steps	40000
Image Resolution	224×224
Random Resized Ratio	[0.9, 1.1]
Random Brightness	0.2
Random Contrast	[0.8, 1.2]
Random Saturation	[0.8, 1.2]
Random Hue	0.1

Table 6: Hyperparameters used for training.

C Vision-Language Attention Visualization

In Figure 6, we visualize the cosine similarity between the output of the CLIP ViT-L/16 encoder and the per-token text features in three different settings: (1) fine-tuning the encoder, (2) a frozen CLIP's output features (X_{out}), and (3) a frozen CLIP's last attention block's feature (X_{attn}) as described in Section 3.1.

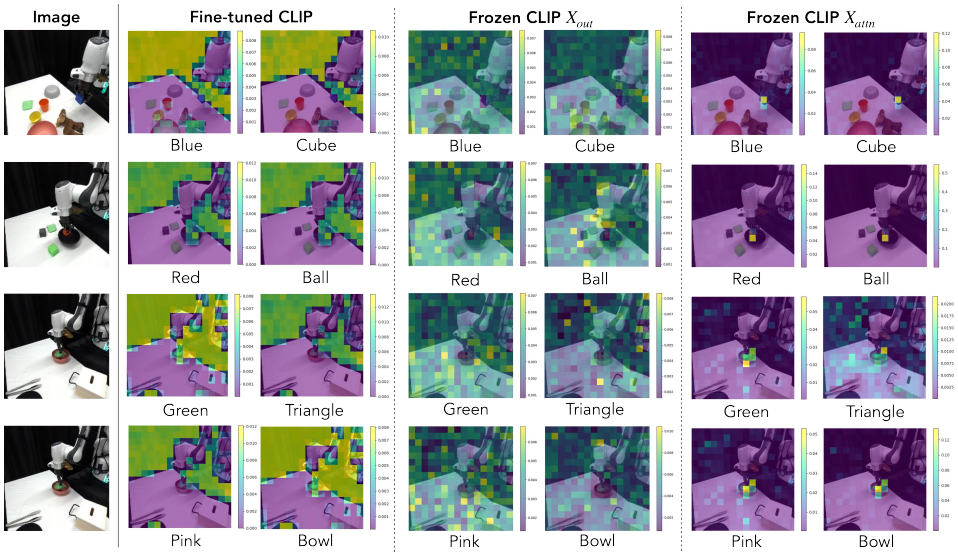

Figure 6: Examples of attention maps for CLIP fine-tuned with VLA (left) and frozen CLIP's output (X_{out}) (middle) and frozen CLIP's attention features (X_{attn}) (right). The first column shows the side view observation and the text query is below each attention map. Fine-tune CLIP pays attention to the background and the frozen CLIP's output (X_{out}) is noisy. In contrast, the frozen CLIP (X_{attn}) pays attention to the correct object associated with the text query. These examples indicate that fine-tuning CLIP on robotic datasets can degrade the performance of the pre-trained CLIP, especially when the robotics dataset is small. It also highlights the benefits of using X_{attn} for fused vision-language features.

In the finetuning v.s. frozen CLIP (X_{attn}) comparison, fine-tuning EF-VLA's CLIP results in overfitting to foreground-background separation, causing it to lose zero-shot object detection ability. This limits the model's ability to highlight the correct object, leading to a significant drop in task success rates (26% vs 68% for training tasks and 15 vs 62% for unseen tasks). Conversely, a frozen CLIP (X_{attn}) preserves object detection capabilities, providing better downstream performance.

In the Vanilla CLIP output (X_{out}) v.s. ClearCLIP output (X_{attn}) comparison, CLIP produces noisy features, degrading vision-language alignment and making object localization harder. By using the attention output (X_{attn}) as in ClearCLIP [9] instead of the final feature map, EF-VLA can localize objects more accurately without fine-tuning or additional parameters.

To provide further motivations for why using X_{out} (per [9]) instead of the output feature map of CLIP, we compare the cosine similarity for each of these options respectively. Similar to what ClearCLIP has noted, after adding residual connection and the final FFN, the features become noisy and worsen the alignment between language and visual features. The noisy attention map makes it challenging for the model to identify the correct features directly from the feature map, which makes it necessary for existing VLA (i.e. OpenVLA [42]) to fine-tune the CLIP vision encoder. In comparison, by using X_{attn}, object localization becomes an easier task in EF-VLA: we can extract the location of the object by getting the *softmax* across the attention map without using any parameters (see Figure 3).

It may initially seem unexpected that this type of visualization is reasonable. However, this can be explained by the fact that LayerNorm operates independently of the patch dimension, as it normalizes along the channel dimension. When combined with the vision-alignment weight matrix w_v, the operation $\hat{f}_v = \text{LN}_{\text{post}}(f_v)w_v$ remains linear. Therefore we can linearize the final attention block:

$$\hat{f}_v = \text{LN}_{\text{post}}(X_{out})w_v \tag{2}$$

$$= \text{LN}_{\text{post}}(X_{res} + X_{attn} + \text{FFN}(\text{LN}(X_{sum})))w_v \tag{3}$$

$$= \text{LN}_{\text{post}}(X_{res})w_v + \text{LN}_{\text{post}}(X_{attn})w_v + \text{LN}_{\text{post}}(\text{FFN}(\text{LN}(X_{sum})))w_v \tag{4}$$

For ClearCLIP, or Frozen CLIP X_{attn}, we are visualizing the $\text{LN}_{\text{post}}(X_{attn})w_v$ term.