

1 A Data Collection System

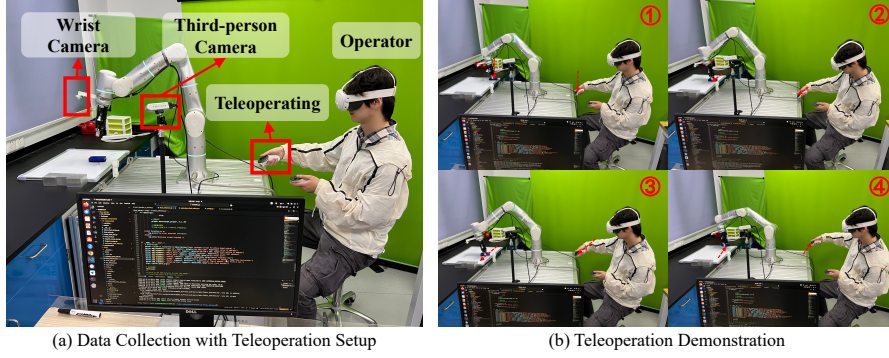


Figure 1: Our data collection system setup.

Our data collection system is depicted in Figure 1(a). The setup features a robotic arm equipped with both a wrist-mounted camera and a static third-person view camera to capture diverse visual perspectives. An operator, wearing a Quest 3 headset and using hand-held controllers, teleoperates the robot arm. A nearby computer runs the necessary software for data acquisition, which includes programs for interfacing with the robot, synchronizing sensor streams, and managing communication with the VR teleoperation hardware. Figure 1(b) illustrates the operator teleoperating the robotic arm to collect demonstration data for the *wipe board* task.

9 B Training Details

The models were mainly trained on compute nodes equipped with $8 \times$ NVIDIA RTX 4090 GPUs (24 GB VRAM each), 64 physical CPU cores, and 251 GB system RAM, using Adam optimization ($\beta_1 = 0.9$, $\beta_2 = 0.95$) with a peak learning rate of 2.5×10^{-5} decaying to 2.5×10^{-6} over 30,000 steps. Multi-task training utilized data parallelism across 2 GPUs (global batch size 16, effective 2048 via gradient accumulation) as additional GPUs provided diminishing returns due to communication overhead, completing 30,000 steps in ~ 12 hours, while single-task training used 1 GPU for 10,000 steps (~ 9 hours), both employing bfloat16 precision with gradient clipping ($\|\nabla\| = 1.0$).

The dimensionalities and key parameters of ForceVLA’s core processing modules: Input Projections, the FVLMoE block, and the Action Output Head are detailed in Table 1.

Table 1: Focused view of ForceVLA’s key architectural components: Input Projections, FVLMoE, and Action Output. Dimensions are indicative (e.g., D_{VLM} , $D_{\text{act_e}}$ for VLM and Action Expert).

Layer	Key Parameters / Dimensions
Input Projections	
Force Projection	Linear; Input: 6 (F/T), Output: $D_{\text{VLM}} = 2048$
State Projection	Linear; Input: $D_{\text{state}} = 32$, Output: $D_{\text{act_e}} = 1024$
Action Projection	Linear; Input: $D_{\text{action}} = 32$, Output: $D_{\text{act_e}} = 1024$
Action-Time MLP	2-layer MLP; Input: $2 \times D_{\text{act_e}}$, Hidden/Output: $D_{\text{act_e}}$; Swish activation
FVLMoE Module	
Input	Concatenation: $N_{\text{VL}} \times D_{\text{VLM}}$ (V-L features) & $1 \times D_{\text{VLM}}$ (Force token)
Pre-MoE Encoder	Transformer Encoder Block; $D_{\text{model}} = 2048$, $N_{\text{H}} = 8$, $D_{\text{h}} = 256$; MLP (expansion factor 1)
MoE Layer	Sparse MoE; $E = 4$ experts (MLPs: $D_{\text{model}} \rightarrow D_{\text{model}}$), Top- $k = 1$; Router: $D_{\text{model}} \rightarrow E$
Output Projection	Linear; Input: $D_{\text{model}} = 2048$, Output: $D_{\text{act_e}} = 1024$
Action Output Head	
Action Output Projection	Linear; Input: $D_{\text{act_e}} = 1024$, Output: $D_{\text{action}} = 32$

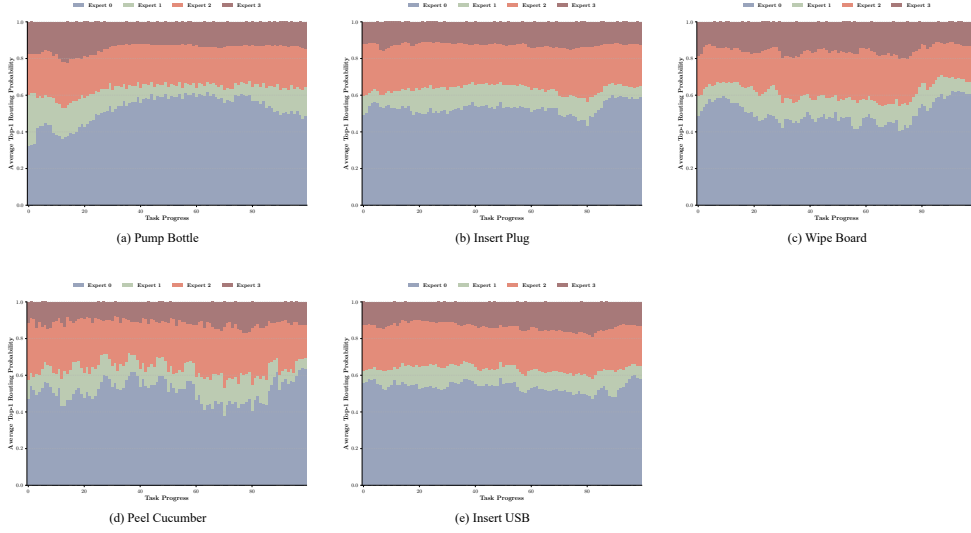


Figure 2: Open-loop evaluation of expert load across different task completion percentages for various tasks: (a) Pump Bottle, (b) Insert Plug, (c) Wipe Board, (d) Peel Cucumber, and (e) Insert USB. Each subplot represents the average expert load (vertical axis) as a function of the task completion percentage (horizontal axis) over the episodes in the test dataset.

To analyze routing dynamics, we first measured the probability distribution over expert selections for each token as it was processed by the router in the MoE module. For variable-length episodes, we applied percentile-based normalization: each task (~ 10 episodes) was processed by segmenting every episode’s token sequence into 100 temporally equidistant intervals, computing the mean top-1 probability per segment, and then averaging these means across episodes. This ensured cross-episode comparability while preserving temporal routing dynamics.

As shown in Figure 2, different tasks exhibit distinct expert utilization patterns. Some tasks (e.g., *insert plug*, *peel cucumber*) show clear temporal specialization, where certain experts dominate specific phases of the task. Others (e.g., *wipe board*) demonstrate a more consistent preference for a single expert throughout execution. These trends suggest that the router learns to allocate computation dynamically across experts based on task-specific semantics and temporal structure.

What’s more, we found that Expert 0 dominates nearly half of the tokens across multiple tasks. This persistent activation suggests that Expert 0 may function as a general-purpose expert, responsible for the fusion of multiple modalities or routine control primitives that are shared across tasks. Its broad involvement contrasts with the more selective, phase-specific activation of Expert 1 or Expert 3, reinforcing the hypothesis of functional specialization among experts. Such asymmetry in routing frequency reflects not only temporal semantic variance within tasks but also architectural bias toward certain experts, potentially shaped during training.

D Multi-task Evaluation

Table 2: Multi-task joint training success rates (%). ForceVLA (Ours) demonstrates superior average performance and excels or matches the best performance in all individual tasks. Best performance(s) in each column are in **bold**.

Model	Pump Bottle	Insert Plug	Insert USB	Wipe Board	Average SR
π_0 -fast w/o F	0.0%	0.0%	0.0%	0.0%	0.0%
π_0 -fast w/ F	0.0%	0.0%	0.0%	0.0%	0.0%
π_0 -base w/o F	20.0%	0.0%	0.0%	0.0%	5.0%
π_0 -base w/ F	50.0%	100.0%	10.0%	10.0%	42.5%
ForceVLA (Ours)	80.0%	100.0%	10.0%	80.0%	67.5%

39 The results of joint multi-task learning are detailed in Table 2. Notably, both π_0 -fast variants (0%
 40 average success rate) failed to acquire skills in this setting, indicating their limited capacity for diverse,
 41 concurrent learning. The π_0 -base w/o F model also performed poorly (5% average success rate),
 42 managing only 20% success on a single task. Adding direct force input (π_0 -base w/ F) improved the
 43 average performance to 42.5%, primarily due to its success in the *insert plug* task. Our **ForceVLA**
 44 model demonstrated superior multi-task capabilities, achieving the highest average success rate
 45 of 67.5%. It obtained 80% success in both *pump bottle* and *wipe board* tasks, and matched top
 46 performance in *insert plug* (100%) and *insert usb* (10%). This robust performance across multiple
 47 distinct tasks indicates ForceVLA’s capacity for concurrent skill learning, proficient instruction
 48 following for varied goals, and highlights the role of its FVLMoE architecture in utilizing multimodal
 49 cues, particularly force, within a shared policy.

50 **E Real-world Experiments Visualization**

51 In this section, we present key frames from real-world experiment videos. Each visualization contrasts
 52 failure cases of baseline models with successful task completions by our ForceVLA model under
 53 similar conditions.

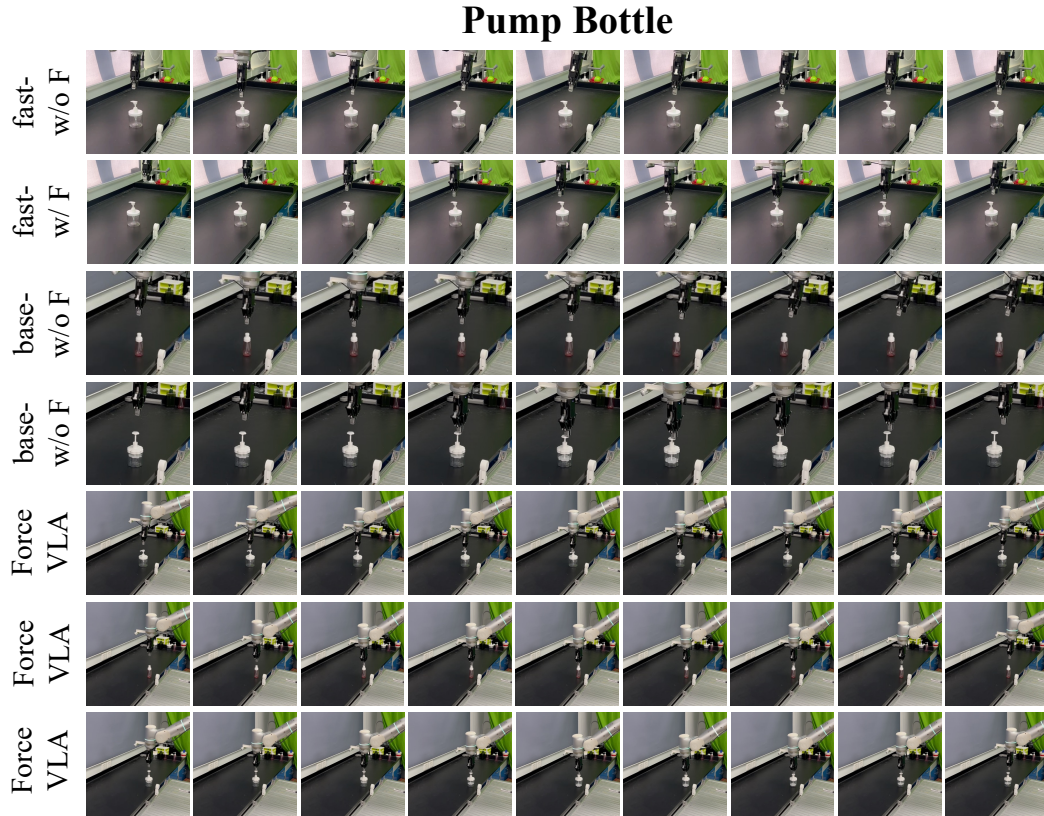


Figure 3: Key frames from Pump Bottle task videos.

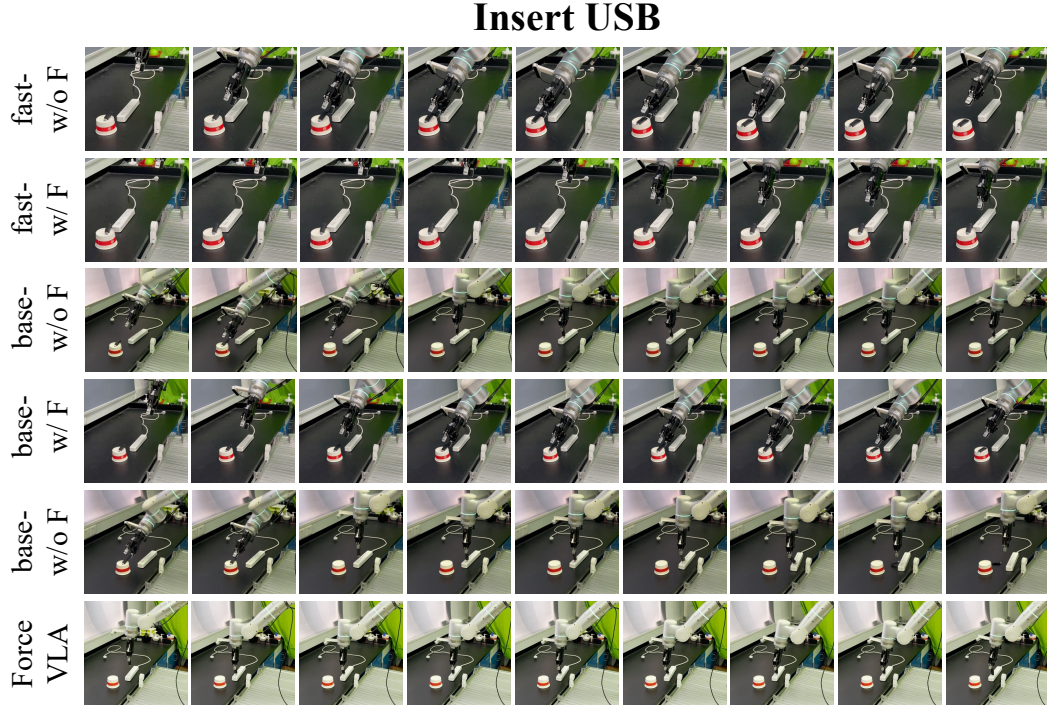


Figure 4: Key frames from Insert USB task videos.

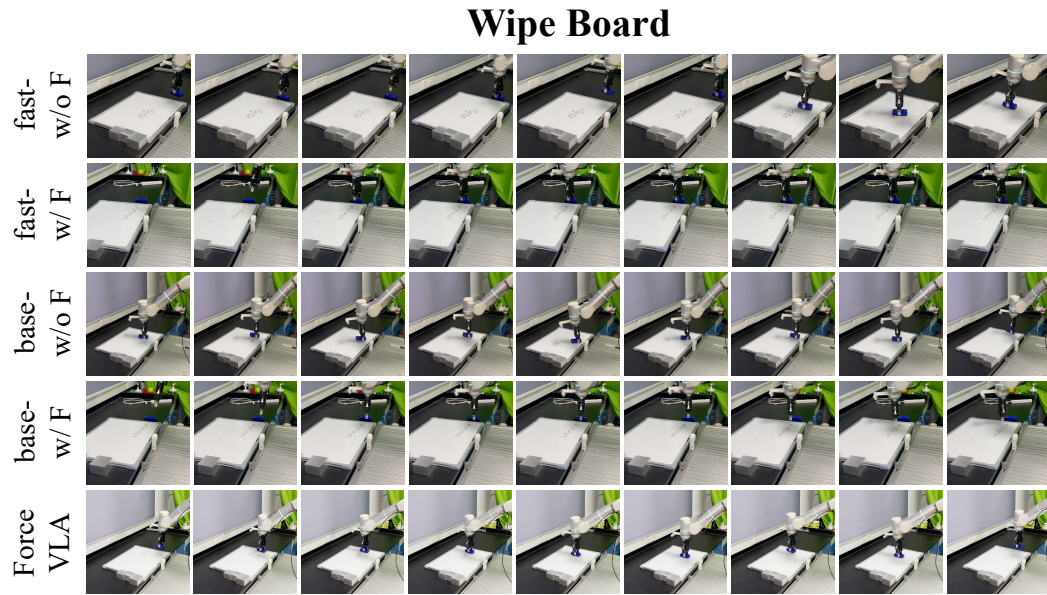


Figure 5: Key frames from Wipe Board task videos.

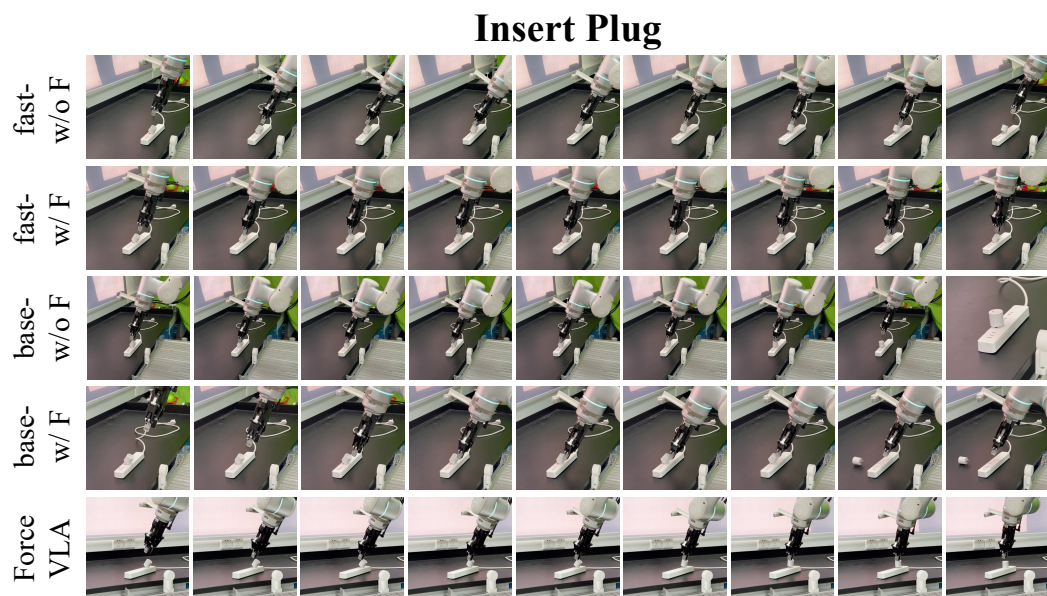


Figure 6: Key frames from Insert Plug task videos.

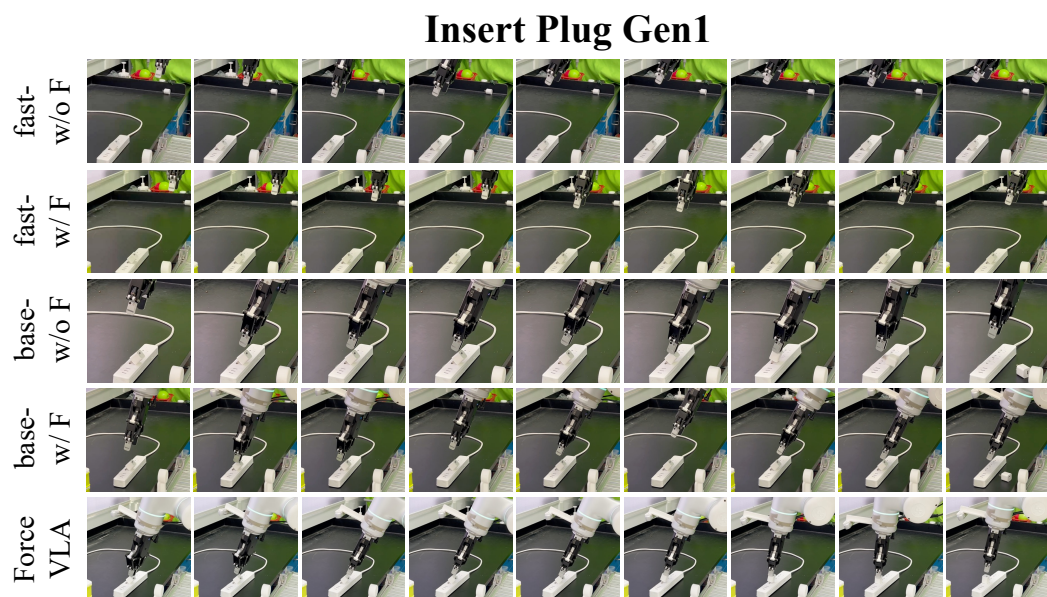


Figure 7: Key frames from Insert Plug Generalization task 1 videos.

Insert Plug Gen2

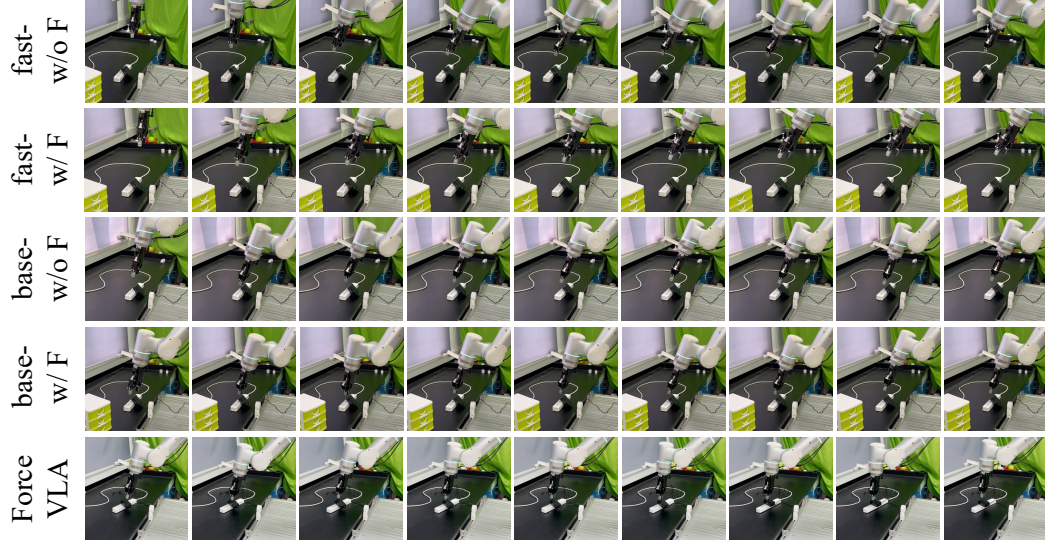


Figure 8: Key frames from Insert Plug Generalization task 2 videos.

Insert Plug Gen3

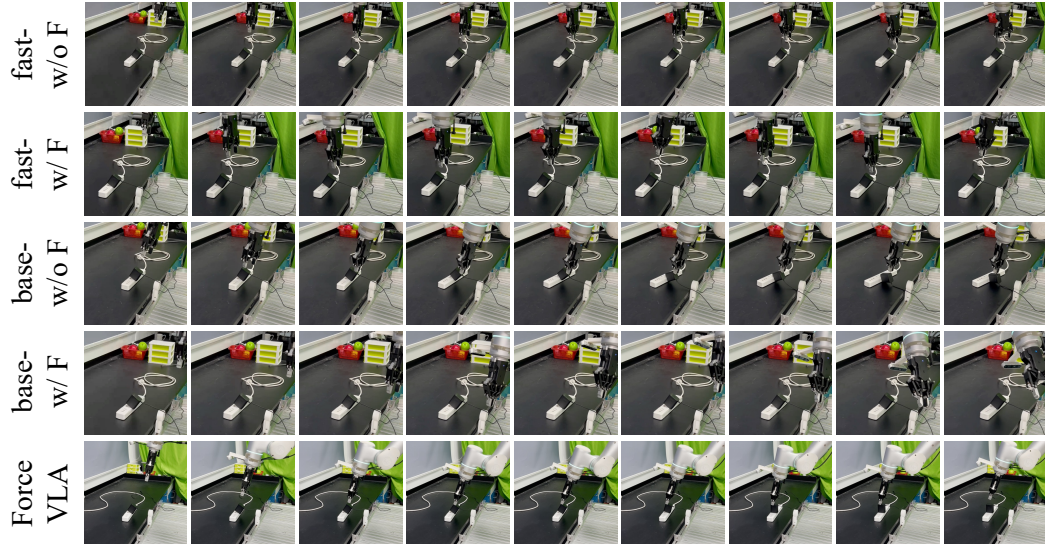


Figure 9: Key frames from Insert Plug Generalization task 3 videos.

Insert Plug Occ1



Figure 10: Key frames from Insert Plug Occlusion task 1 videos.

Insert Plug Occ2

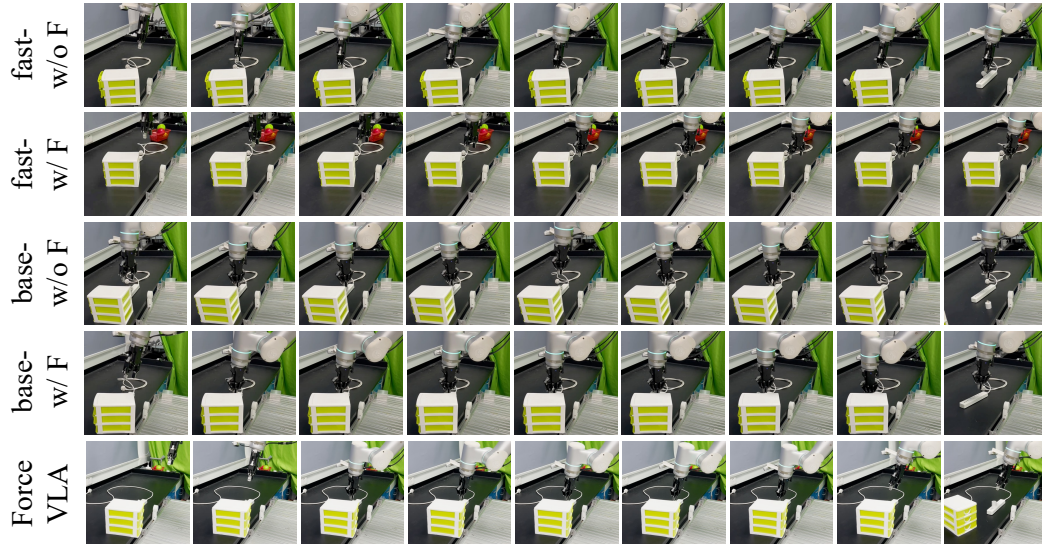


Figure 11: Key frames from Insert Plug Occlusion task 2 videos.



Figure 12: Key frames from Insert Plug Unstable task videos.