FrameSkip: Learning from Fewer but More Informative Frames in VLA Training

Anonymous ACL submission

Abstract

Vision-Language-Action (VLA) policies are commonly trained from dense robot demonstration trajectories, often collected through teleoperation, by sampling every recorded frame as if it provided equally useful supervision. We argue that this convention creates a temporal supervision imbalance: long low-change segments dominate the training stream, while manipulation-critical transitions such as alignment, contact, grasping, and release appear only sparsely. We introduce **FrameSkip**, a data-layer frame selection framework that scores trajectory frames using action variation, visual-action coherence, task-progress priors, and gripper-transition preservation, then remaps training samples toward high-importance frames under a target retention ratio. Because FrameSkip operates only in the dataloader, it leaves the VLA architecture, action head, training objective, and inference procedure unchanged. Across RoboCasa-GR1, SimplerEnv, and LIBERO, FrameSkip improves the success-retention trade-off over full-frame training and simpler frame selection variants, achieving a macro-average success rate of 76.15% across the three benchmarks compared with 66.50% for full-frame training while using a compressed trajectory view that retains 20% of unique frames in the main setting. On a real-world dual-arm snack-cleanup task with $\pi_{0.5}$, FrameSkip improves the success rate from 18% to 26%. Code is available in an anonymous GitHub repository at anonymous.4open.science/r/FrameSkip-anonymization-CDDC/.

1 Introduction

Vision-Language-Action (VLA) models have recently emerged as a promising paradigm for robotic manipulation by combining visual grounding, language conditioning, and action prediction within a unified policy model (Team et al., 2024; Kim et al., 2024; Black et al., 2024; Zhou et al., 2025).

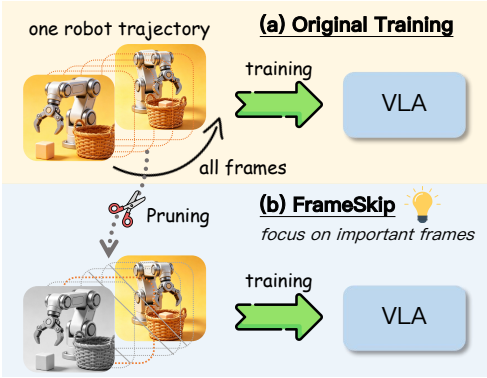

Figure 1: FrameSkip reframes training-time frame pruning as temporal supervision allocation: it reduces exposure to redundant low-change trajectory segments and increases exposure to manipulation-critical transitions.

As these systems scale to broader data mixtures, more tasks, and stronger vision-language backbones, they are increasingly trained on large embodied datasets such as Open X-Embodiment (O'Neill et al., 2024). These datasets are typically composed of dense robot demonstration trajectories, often collected through teleoperation, where each trajectory records a sequence of observations and actions produced while completing a task. This scaling trend has improved task coverage and generalization, but it also exposes a basic training convention that remains largely unquestioned: dense demonstrations are sampled as if every trajectory frame provided equally useful supervision.

This convention is mismatched with the temporal structure of robot demonstrations. Manipulation trajectories often contain long low-change segments, such as approaching an object, maintaining a grasp, or transporting an object steadily toward a target. In contrast, the moments that define the task outcome are sparse: alignment, contact, grasp closure, release, and abrupt changes in end-effector behavior may occupy only a small fraction

of the recorded trajectory. Uniform frame sampling therefore creates a temporal supervision imbalance. Under a fixed optimization budget, rare decision-critical transitions can be diluted by abundant but weakly informative observations.

Existing VLA research has largely addressed scaling through model architecture, action representation, data mixture design, and optimization strategy (Kim et al., 2024, 2025; Pertsch et al., 2025; Intelligence et al., 2025; NVIDIA et al., 2025a). Much less attention has been paid to how supervision is distributed across the frames within each demonstration. This raises a simple question: can VLA training benefit from reallocating supervision toward the frames that carry the most policy-relevant information?

We therefore view frame selection not merely as a way to reduce data volume, but as a mechanism for reallocating temporal supervision under a fixed optimization budget. Figure 1 illustrates this high-level idea. In this paper, we present FRAMESKIP, a data-layer frame selection framework for VLA training. FRAMESKIP assigns each frame an importance score from lightweight trajectory cues, including action variation, visual-action coherence, task-progress priors, and gripper-transition preservation. It then constructs compressed trajectory views under target retention ratios and remaps training samples toward retained high-importance frames. Importantly, FRAMESKIP does not modify the VLA architecture, action head, loss function, or inference procedure. This makes FRAMESKIP a direct way to study frame importance as a training principle rather than as a model-specific architectural change.

We evaluate FRAMESKIP as a question about the success-retention trade-off of VLA training rather than as a generic frame dropping heuristic. Under matched settings, we compare full-frame training, random frame selection, action-variation-only selection, and progressively stronger importance metrics on RoboCasa-GR1 (Nasiriany et al., 2024), SimplerEnv (Li et al., 2024c), and LIBERO (Liu et al., 2023). In the main setting, FRAMESKIP uses a compressed trajectory view that retains 20% of unique frames and improves the macro-average success rate across the three benchmarks from 66.50% with full-frame training to 76.15%, with consistent gains on all three benchmarks.

Our main contributions are as follows:

- To our knowledge, we present the first VLA training approach that optimizes supervision at the frame level, identifying temporal supervision imbalance as a practical and underexplored issue in VLA training.
- We introduce FRAMESKIP, an architecture-agnostic data-layer framework that selects more informative training frames using lightweight trajectory cues and gripper-transition preservation.
- We provide a systematic empirical study of importance-guided frame retention, including matched-ratio baselines and ablations over retention ratios, importance metrics, and warmup schedules.

2 Related Work

Vision-language-action models. VLA models combine visual grounding, language conditioning, and action prediction in a unified policy interface (Kim et al., 2024; Black et al., 2024; Zhou et al., 2025). Recent work improves these systems through stronger VLM initialization, action tokenization, diffusion or flow-matching action heads, and large-scale cross-embodiment data (Pertsch et al., 2025; Intelligence et al., 2025; NVIDIA et al., 2025a; O'Neill et al., 2024). These advances generally assume that the training set is consumed at its original temporal density. FRAMESKIP is complementary: it asks whether the same VLA families can be trained with fewer but more informative frames.

Data Curation for Robot Learning. Coarse-grained approaches reweight datasets (Hejna et al., 2024) or filter trajectories (Hejna et al., 2025) but treat intra-trajectory frames uniformly. Scizor (Zhang et al., 2026) curates transitions via a learned task-progress predictor, aiming to remove low-quality and redundant data. FRAMESKIP differs in objective and mechanism: it does not learn an auxiliary transition-quality model or frame deletion policy, but reallocates training supervision within each trajectory using lightweight cues, including action variation, visual-action coherence, task-progress priors, and gripper-transition preservation, under a controllable retention ratio. TGM-VLA (Pu et al., 2026) addresses keyframe oversampling in 3D manipulation, but is specific to keyframe-based architectures. FRAMESKIP operates on raw frames without keyframe structure.

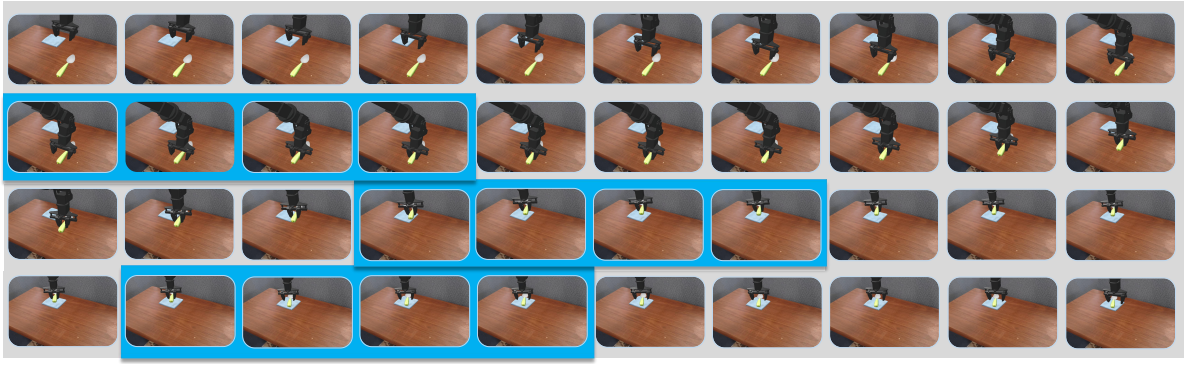

Figure 2: A flattened view of one dense 40-frame robot trajectory. Most recorded frames correspond to continuous, smooth, and low-change motion, while the highlighted segments cover sparse manipulation-critical transitions such as alignment, contact, grasping, and placement. This temporal imbalance motivates reallocating supervision toward frames with higher marginal value under a fixed training budget.

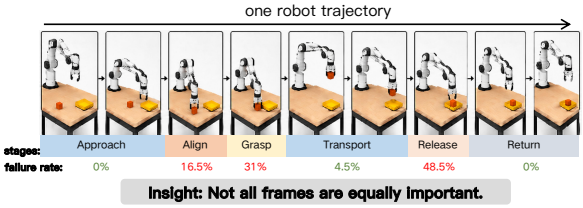

Figure 3: Stage-wise failures in a robot trajectory are concentrated around sparse manipulation-critical moments such as alignment, grasping, and release, rather than uniformly distributed across smooth approach, transport, and return stages.

3 Method

3.1 Motivation: Temporal Supervision Imbalance

Dense robot demonstrations are temporally imbalanced. As shown in Figure 2, a 40-frame trajectory is dominated by smooth, low-change motion, while alignment, contact, grasp closure, release, and other outcome-defining transitions occupy only short windows. Uniform frame sampling therefore spends much of the training budget on redundant supervision and gives rare decision-critical frames comparatively few updates.

This imbalance matters because failures are concentrated in the same sparse regions. Figure 3 shows that routine stages such as approach, transport, and return are often reliable, whereas alignment, grasping, and release have much higher failure rates. These observations motivate treating frame selection as temporal supervision allocation: under a fixed optimization budget, training should emphasize frames with higher marginal value rather than preserve the original temporal density.

3.2 FrameSkip Overview

FrameSkip is a training-time data-layer framework for reducing temporal redundancy in VLA demonstrations, with the overall pipeline illustrated in Figure 4. Given a robot demonstration trajectory, it first computes frame-level importance scores from lightweight trajectory statistics, then precomputes retained frame indices for a set of retention ratios, and finally uses these cached indices to remap dataset queries during training. The VLA model, action head, loss function, and inference procedure are left unchanged. This section formalizes the frame selection problem, describes the importance estimator, presents the ratio-aware pruning rule, and explains how the cached compressed views are integrated into minibatch training.

3.3 Problem Formulation

We consider a VLA training set composed of robot demonstration trajectories $\tau = \{(o_t, a_t, l)\}_{t=1}^{T}$, where o_t denotes the observation at step t, a_t denotes the action, and l denotes the language instruction associated with the trajectory. Standard training uses all frames in τ, implicitly assuming that each timestep contributes equally to learning. FrameSkip challenges this assumption by selecting a subset of frames that is intended to preserve the most informative supervision.

Given a target retention ratio $r \in (0, 1]$, our goal is to construct a subset of timestep indices $S_r \subseteq \{1, \dots, T\}$ such that $|S_r| \approx rT$ while preserving the frames that are most useful for learning the policy. The ratio r denotes the fraction of frames retained, rather than the fraction removed. Importantly, FrameSkip is a training-time data transformation: it does not change the VLA model

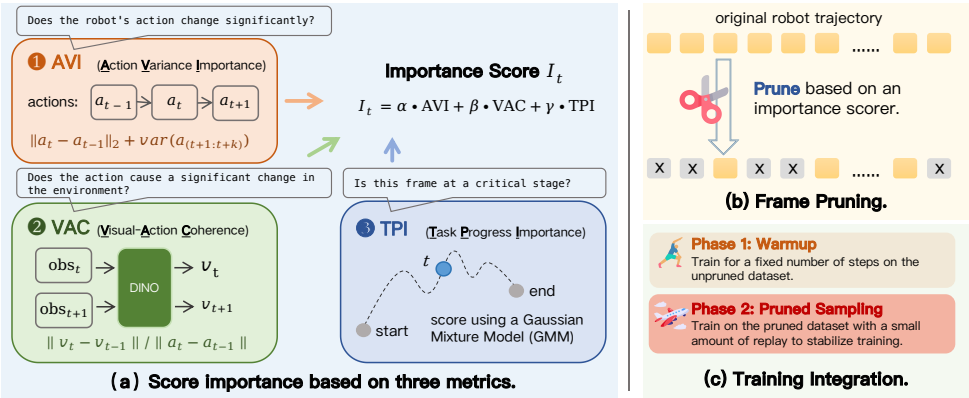

Figure 4: FRAMESKIP pipeline. FRAMESKIP scores frames in each demonstration trajectory, retains high-importance frames under a target ratio, and injects the compressed trajectory view into VLA training through dataloader index remapping while leaving the model and inference procedure unchanged.

architecture, the action representation, or the inference procedure. Instead, it changes which frames are exposed to the model during training.

3.4 Frame Importance Estimation

The core idea of FRAMESKIP is that trajectory frames should not be treated uniformly. We therefore assign each frame an importance score that combines multiple complementary signals. Intuitively, a frame should receive a higher score if it corresponds to a substantial action change, a visually grounded transition, or a stage of the trajectory where critical interaction is likely to happen. All component scores are min-max normalized within each trajectory before being combined; if a component is constant, it is mapped to a uniform score so that it does not introduce spurious preference.

Action Variation Importance. Our first signal captures local action dynamics. Let a_t denote the action at step t. We define Action Variation Importance (AVI) as

$$\text{AVI}(t) = \|a_t - a_{t-1}\|_2 + \lambda \, \text{MeanVar}(a_{t+1:t+k}), \tag{1}$$

where the first term measures the change relative to the previous action and the second term captures short-range action variation in the next k steps. In our implementation, $k = 3$ and $\lambda = 0.1$. Near trajectory boundaries, the look-ahead window is truncated to the available timesteps, and the score for the first frame is padded with the first available action-difference value. Frames with large AVI values typically correspond to abrupt motion changes, contact events, grasping, release, or other behavior transitions that are likely to be informative for policy learning.

Visual-Action Coherence. Action changes do not always imply meaningful interaction with the environment. To capture visually grounded transitions, FRAMESKIP incorporates Visual-Action Coherence (VAC):

$$\text{VAC}(t) = \frac{\|v_t - v_{t-1}\|_2}{\|a_t - a_{t-1}\|_2 + \epsilon}, \tag{2}$$

where v_t is a visual feature extracted from observation o_t by a DINOv2 visual encoder. This term gives higher weight to frames where visual change is large relative to the local action change, which is useful for identifying contact or object-motion stages that are not fully captured by action magnitude alone. In all reported FRAMESKIP experiments, VAC is enabled throughout frame-score preprocessing. To make the offline computation robust and affordable, we compute VAC on sparsely sampled video frames, interpolate the resulting scores back to the action sequence length, and clip extreme VAC values at the 95th percentile before normalization.

Task Progress Importance. Some interaction events are sparse but tend to occur in characteristic regions of a task trajectory. To encode this weak structural prior, we define Task Progress Importance (TPI) over the normalized progress $p_t = (t-1)/(T-1)$. In the main experiments, we use a dataset-adaptive progress prior. Specifically, for each benchmark, we fit a one-dimensional Gaussian mixture model (GMM) to the normalized progress locations of manipulation-critical stage centers annotated from a small subset of training

trajectories:

$$q(p) = \sum_{m=1}^{M} \pi_m \mathcal{N}(p; \mu_m, \sigma_m^2), \tag{3}$$

and define

$$\mathrm{TPI}(t) = \frac{q(p_t)}{\max_{s\in\{1,\dots,T\}} q(p_s)}. \tag{4}$$

This dataset-adaptive prior captures task-specific stage structure while keeping frame scoring independent of the VLA model and policy objective. The stage annotations are used only to estimate the offline progress prior during preprocessing and are not provided to the policy during training or evaluation.

When such annotations are unavailable, FRAMESKIP can use a simpler dataset-agnostic Gaussian prior:

$$\mathrm{TPI}(t) = \exp\left(-\frac{(p_t - 0.5)^2}{\sigma^2}\right), \quad p_t = \frac{t-1}{T-1}. \tag{5}$$

This fallback assumes that manipulation-critical stages are more likely to occur near the middle of a trajectory and requires no stage annotations; we use $\sigma^2 = 0.2$ for this Gaussian variant.

Combined score and gripper-transition preservation. We combine the signals into a single frame score:

$$I(t) = \alpha\, \widehat{\mathrm{AVI}}(t) + \beta\, \widehat{\mathrm{VAC}}(t) + \gamma\, \widehat{\mathrm{TPI}}(t), \tag{6}$$

where $\hat{\cdot}$ denotes min-max normalized scores and α, β, γ are scalar weights. In our default setting, AVI provides the dominant signal, while VAC and TPI act as auxiliary cues; we use $\alpha = 0.6$, $\beta = 0.2$, and $\gamma = 0.2$ unless otherwise specified.

For manipulation tasks, some of the most important moments coincide with gripper or end-effector state transitions. The gripper-aware variant therefore multiplies the combined score by a factor determined by the absolute change in the gripper or end-effector state dimensions specified by each benchmark action schema. When such dimensions are unavailable, this factor falls back to the action-variation signal already captured by AVI. This design does not introduce a new model component; it simply injects a task-relevant event prior into the scoring function so that contact-related stages are less likely to be removed during pruning.

3.5 Ratio-Aware Frame Pruning

Once importance scores are computed, FRAMESKIP prunes frames according to a target retention ratio r. For a trajectory of length T, the target number of retained frames is

$$K_r = \max(K_{\min}, \lfloor rT \rfloor), \tag{7}$$

where $K_{\min}$ prevents very short compressed trajectories. We first compute a threshold based on the empirical $(1-r)$-quantile of the importance scores and retain frames whose score exceeds that threshold:

$$S_r = \{t \mid I(t) \geq \theta_r\}, \tag{8}$$

where $\theta_r = \mathrm{Quantile}(I, 1-r)$, so the candidate set approximately contains the top rT frames.

The pruning procedure additionally enforces several practical constraints. First, when gripper-transition preservation is enabled, the pruner explicitly retains the first frame, the last frame, gripper or end-effector transition frames, and frames whose action changes fall in the top decile of the trajectory. Second, if the quantile rule keeps too many or too few frames relative to K_r, the pruner selects or adds frames by descending importance until the target count is met. Third, we optionally apply a temporal consistency constraint that fills unusually large gaps between consecutive retained frames. This avoids pathological cases in which a trajectory becomes too temporally discontinuous after pruning, at the cost of a slightly higher actual retention ratio.

In practice, FRAMESKIP supports multiple retention ratios for the same trajectory. We therefore precompute and cache pruning results for a configured superset of ratios. Each trajectory cache stores the retained indices and the actual achieved ratio for each configured setting, allowing the training pipeline to switch between compressed views without recomputing frame scores. The cache is keyed by the importance and pruning configuration; a separate list of training ratios can be chosen as a subset of the cached ratios to reuse the same cache across multiple schedules.

3.6 Sampling Strategy

FRAMESKIP uses compressed trajectories as the main source of supervision after an initial full-frame warmup. The motivation is to make the policy learn primarily from high-importance frames, while still preserving occasional access to the

original temporal density. This gives the training process two complementary signals: compressed mini-batches emphasize decision-relevant moments, whereas full-frame mini-batches act as an anchor that refreshes the broader trajectory context and reduces the risk of overfitting to overly sparse transitions.

Warmup. During the first N_{warm} optimization steps, FRAMESKIP uses the identity view with $r = 1.0$, which is equivalent to standard full-frame training. This stage gives the policy a stable initialization from dense temporal supervision before the frame-pruned views are introduced.

Pruned Sampling with Full-Frame Anchors. After warmup, most mini-batches are drawn from a frame-pruned view with a target retention ratio $r < 1.0$, so the effective training distribution is biased toward frames selected by the importance estimator. A small fraction of mini-batches are instead drawn from the full-frame view $r = 1.0$. We use this mixture to preserve global trajectory coverage while still concentrating supervision on high-importance frames. Under a fixed number of optimization steps, this schedule changes which timesteps dominate the gradient signal rather than changing the policy objective. In our main setting, FRAMESKIP uses a compressed view with $r = 0.2$, retaining 20% of unique frames from each trajectory and pruning the remaining 80% within that view. For every five pruned mini-batches, we insert one full-frame mini-batch as a context anchor. This schedule treats full-frame samples not as the default training signal, but as periodic context refreshes that stabilize learning under aggressive temporal compression.

3.7 Training Integration

FRAMESKIP integrates frame selection as a sample-space preserving data transformation. Instead of rewriting the dataset or changing the policy objective, it keeps the original trajectory index space unchanged and maps training queries to retained high-importance frames according to the active retention ratio. As a result, full-frame and compressed views share the same model, optimizer, loss, dataset mixture weights, and training schedule; only the temporal locations exposed to the learner are changed.

This design makes the comparison with full-frame training controlled and keeps FRAMESKIP architecture-agnostic. The same remapping mechanism can be applied to different VLA backbones and action heads because it operates before model-specific preprocessing. It also preserves compatibility with existing embodied-data pipelines, where dataset length, trajectory sampling weights, and distributed sampling behavior are often assumed by the trainer. Implementation details of the dataloader remapping are provided in Appendix A.4.

4 Experiments

4.1 Experimental Setup

Policy Architecture. Our VLA model follows a classic two-system design. The VLM is initialized from Qwen3-VL-4B-Instruct (Bai et al., 2025) to encode text instructions and visual observations, while the action expert is a randomly initialized Diffusion Transformer (DiT) (Peebles and Xie, 2023) that generates continuous robot action chunks with a flow-matching objective. Additional architecture details are provided in Appendix A.1.

Training Details. For each benchmark, we train the VLA policy on the corresponding benchmark-specific training set for a fixed number of optimization steps. The number of training steps is adjusted according to the size of each benchmark dataset, while the global batch size is kept fixed at 128 across all runs. All experiments are conducted on 8 NVIDIA H100 GPUs with DeepSpeed ZeRO-2 distributed training (Rajbhandari et al., 2020). Unless otherwise specified, FRAMESKIP uses a retention ratio of $r = 0.2$ and a 5:1 schedule between pruned mini-batches and full-frame anchor mini-batches. The same model architecture, optimizer setting, and remaining training configuration are used across compared methods so that differences can be attributed to the frame selection strategy rather than to changes in the underlying VLA training recipe. To facilitate reproducibility and future work on frame-level VLA training data optimization, we will publicly release the training code, frame-selection pipeline, and model checkpoints. Additional implementation and hyperparameter details are provided in Appendix A.

Benchmarks. We evaluate FRAMESKIP on three simulation benchmarks: RoboCasa-GR1, SimplerEnv (Li et al., 2024c), and LIBERO (Liu et al., 2023). These benchmarks cover different robot embodiments, manipulation settings, and evaluation protocols. This setting tests whether FRAMESKIP can be applied as a data-layer frame pruning method across multiple embodied data

Table 1: RoboCasa-GR1 simulation results on four representative pick-and-place tasks. The omitted columns indicate additional RoboCasa-GR1 tasks; Avg. is computed over all 24 tasks rather than only the shown tasks.

Method	PnP Bottle	PnP Can	PnP Cup	PnP Milk	···	Avg.
GR00T N1.5 (NVIDIA et al., 2025a)	54.0	50.0	38.0	60.0	···	48.2
GR00T N1.6 (Team et al., 2025)	51.5	13.0	8.5	14.0	···	47.6
TwinBrainVLA (Yu et al., 2026)	74.0	72.0	52.0	60.0	···	54.6
PhysBrain (Lin et al., 2025)	74.0	68.0	42.0	54.0	···	50.0
LangForce (Lian et al., 2026)	72.0	78.0	46.0	56.0	···	52.6
ABot-M0 (Yang et al., 2026)	86.0	74.0	48.0	46.0	···	58.3
Full-Frame Training	46.0	80.0	54.0	48.0	···	47.8
FrameSkip (ours)	74.0	80.0	46.0	60.0	···	**59.5**

Table 2: SimplerEnv simulation results on four held-out WidowX manipulation tasks. We report success rates (%) for each task and the average across the four tasks.

Method	Put Spoon on Towel	Put Carrot on Plate	Stack Green Block on Yellow Block	Put Eggplant in Yellow Basket	Avg.
OpenVLA (Kim et al., 2024)	4.2	0.0	0.0	12.5	4.2
RoboVLM (Li et al., 2024b)	50.0	37.5	0.0	83.3	42.7
ThinkAct (Huang et al., 2025)	58.3	37.5	8.7	70.8	43.8
SpatialVLA (Qu et al., 2025)	20.8	20.8	25.0	70.8	34.4
CogACT (Li et al., 2024a)	71.7	50.8	15.0	67.5	51.3
VideoVLA (Shen et al., 2025)	75.0	20.8	45.8	70.8	53.1
π_0 (Black et al., 2024)	29.1	0.0	16.6	62.5	27.1
$\pi_{0.5}$ (Intelligence et al., 2025)	49.3	64.7	44.7	69.7	57.1
GR00T N1.6 (Team et al., 2025)	64.5	65.5	5.5	93.0	57.1
VLA-JEPA (Sun et al., 2026)	75.0	70.8	12.5	70.8	57.3
TwinBrainVLA (Yu et al., 2026)	87.5	58.3	33.3	79.1	64.5
LangForce (Lian et al., 2026)	89.6	63.8	33.3	79.2	66.5
Full-Frame Training	87.5	50.0	29.2	54.2	55.2
FrameSkip (ours)	90.63	54.17	45.59	95.83	**71.6**

regimes rather than only within a single robot platform.

4.2 Simulation Benchmarks

RoboCasa-GR1. RoboCasa-GR1 is a tabletop manipulation benchmark built on RoboCasa (Nasiriany et al., 2024), where a GR1 robot performs bimanual manipulation with two dexterous hands. We evaluate on 24 tabletop tasks and train with the 24K GR1 teleoperation simulation demonstrations released by NVIDIA. This benchmark tests multi-task VLA learning and dexterous-hand control. The main results are shown in Table 1, and the full 24-task results are provided in Appendix B.

SimplerEnv. SimplerEnv evaluates WidowX manipulation policies in simulation (Li et al., 2024c). We use four evaluation tasks whose scenes and instructions are held out from training, making the benchmark a test of out-of-domain generalization. Following the standard setting, we train on the BridgeV2 real-robot dataset and evaluate in SimplerEnv simulation. The results are shown in Table 2.

LIBERO. LIBERO is a Franka-based simulation benchmark for language-conditioned manipulation (Liu et al., 2023). We evaluate on four task suites and train with the official expert demonstrations provided by the benchmark. LIBERO

Table 3: LIBERO simulation results on four task suites.

Method	L-Spatial	L-Object	L-Goal	L-Long	Avg.
Diffusion Policy (Chi et al., 2023)	78.5	87.5	73.5	64.8	76.1
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.5
SpatialVLA (Qu et al., 2025)	88.2	89.9	78.6	55.5	78.1
CoT-VLA (Zhao et al., 2025)	87.5	91.6	87.6	69.0	83.9
GR00T N1 (NVIDIA et al., 2025a)	94.4	97.6	93.0	90.6	93.9
F1 (Lv et al., 2025)	98.2	97.8	95.4	91.3	95.7
InternVLA-M1 (Chen et al., 2025)	98.0	99.0	93.8	92.6	95.9
π_0 (Black et al., 2024)	98.0	96.8	94.4	88.4	94.4
$\pi_{0.5}$ (Intelligence et al., 2025)	98.8	98.2	98.0	92.4	96.9
GR00T N1.6 (Team et al., 2025)	97.7	98.5	97.5	94.4	97.0
Full-Frame Training	97.8	98.8	97.4	92.0	96.5
FrameSkip (ours)	98.6	99.0	98.2	93.8	**97.4**

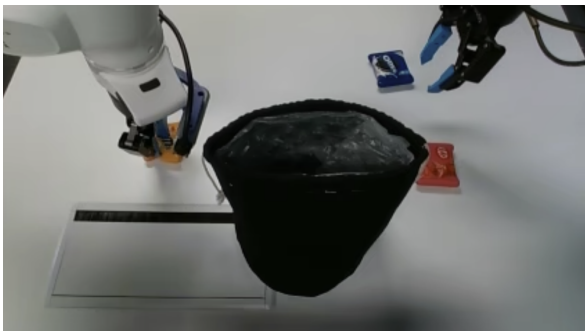

Figure 5: Real-world dual-arm snack-cleanup setup. The robot uses two Franka Research 3 arms with Robotiq 85 grippers and Intel RealSense D435 cameras to place three snacks into a bag and tie the bag.

complements RoboCasa-GR1 and SimplerEnv by testing FrameSkip on a standardized single-arm embodiment with expert trajectories. The results are shown in Table 3.

Results. Across the three simulation benchmarks, FrameSkip consistently improves over full-frame training under the same VLA architecture and training recipe. It improves the macro-average success rate across RoboCasa-GR1, SimplerEnv, and LIBERO from 66.50% to 76.15% while using a compressed trajectory view that retains 20% of unique frames in the main setting, suggesting that reallocating supervision toward informative frames is a useful training signal rather than merely a data reduction heuristic.

4.3 Real-World Experiment

We further evaluate FrameSkip on a real-world snack-cleanup task using two Franka Research 3 arms with Robotiq 85 grippers and Intel RealSense D435 head/hand cameras, as shown in Figure 5. The robot must use both arms to place three snacks from the table into a bag and tie the bag. We train from 500 expert trajectories.

We use $\pi_{0.5}$ as the policy backbone and compare full-frame training with FrameSkip under

Table 4: Ablation on the retention ratio r. We report the average success rate (%) across the 24 RoboCasa-GR1 tasks.

Retention r	10%	20%	30%	40%	50%	60%	100%
RoboCasa-GR1 Avg.	55.00	59.50	59.50	56.75	**59.75**	55.92	47.80

Table 5: Ablation on the frame importance metric. All variants use the same retention ratio and training schedule; only the frame scoring rule is changed.

Metric Variant	RoboCasa-GR1	SimplerEnv	LIBERO	Avg.
Random	47.67	56.51	96.3	66.83
AVI	54.25	57.29	97.05	69.53
AVI+TPI	57.42	59.90	97.00	71.44
AVI+VAC	58.75	65.08	97.15	73.66
AVI+VAC+TPI	59.00	67.33	97.2	74.51
FRAMESKIP Full	59.50	71.55	97.4	76.15

the same training budget of 10K optimization steps and batch size 64. Over 100 real-world trials per method, FRAMESKIP improves the success rate from 18% to 26%, indicating that frame-level supervision allocation also benefits a real dual-arm manipulation setting.

4.4 Ablation Studies

We ablate four design choices in FRAMESKIP: the retention ratio, the frame-importance metric, the length of full-frame warmup, and the number of optimization steps. Additional ablation protocol details are provided in Appendix A.6.

Effect of retention ratio. On RoboCasa-GR1, we vary the retention ratio $r \in \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 100\%\}$ under the same model and training budget. As shown in Table 4, all pruned settings outperform full-frame training, with the best result at $r = 50\%$ and strong performance already at $r = 20\%$–30%. This supports our claim that reallocating supervision can be more effective than exposing the model to every temporally redundant frame.

Effect of importance metric. We compare random pruning, action-only scoring, progressively richer importance metrics, and the full FRAMESKIP scorer under the same retention ratio. Table 5 shows that each added cue improves the average success rate, and the full scorer performs best across the three-benchmark average. The gains over random pruning indicate that the benefit comes from where supervision is allocated, not simply from seeing fewer frames.

Effect of warmup steps. Finally, we vary the full-frame warmup length on RoboCasa-GR1. Table 6 shows only modest variation from 2500 to 15000 steps, with the best result at 5000 steps, suggesting that FRAMESKIP is not highly sensitive to this hyperparameter.

Table 6: Ablation on the number of full-frame warmup steps. After warmup, all variants use the same retention ratio and pruned/full-frame mini-batch schedule. We report the average success rate (%) across the 24 RoboCasa-GR1 tasks.

Warmup Steps	2500	5000	7500	10000	12500	15000
RoboCasa-GR1 Avg.	58.42	**59.50**	59.08	58.75	58.33	58.25

Table 7: Step-efficiency ablation on RoboCasa-GR1 and SimplerEnv. We compare full-frame training and FRAMESKIP under matched fractions of the standard benchmark budget. Standard budgets are 100K steps for RoboCasa-GR1 and 60K steps for SimplerEnv; the warmup length is fixed across step budgets.

Method	Step Budget	RoboCasa-GR1 Avg.	SimplerEnv Avg.
Full-Frame Training	25%	39.3	45.8
FRAMESKIP	25%	51.2	56.51
Full-Frame Training	50%	41.6	47.4
FRAMESKIP	50%	54.3	68.75
Full-Frame Training	75%	44.4	53.1
FRAMESKIP	75%	57.6	69.01
Full-Frame Training	100%	47.8	55.2
FRAMESKIP	100%	59.5	71.6

Effect of training budget. We further evaluate whether FRAMESKIP improves optimization-step efficiency on RoboCasa-GR1 and SimplerEnv by comparing it with full-frame training under matched reduced step budgets. This controls for the possibility that fewer optimization steps alone improve generalization. As shown in Table 7, FRAMESKIP outperforms full-frame training at matched budgets and can reach or exceed the standard full-frame baseline with fewer steps.

5 Conclusion

We presented FRAMESKIP, a data-layer frame selection framework that reallocates VLA training supervision toward informative trajectory moments without changing the model or inference procedure. Across RoboCasa-GR1, SimplerEnv, and LIBERO, FRAMESKIP improves the macro-average success rate from 66.50% to 76.15% while retaining 20% of unique frames in the main compressed view. These results suggest that frame-level supervision allocation is a practical lever for improving VLA training under fixed optimization budgets.

Limitations

FRAMESKIP is evaluated primarily on simulated manipulation benchmarks, with one real-world dual-arm snack-cleanup task. Its behavior on broader real-robot deployments may differ due to sensing noise, actuator delays, domain shift, and safety constraints that are not fully captured by our current evaluation. The current frame-importance estimator is intentionally lightweight, but it still relies on offline preprocessing and, in the main TPI setting, a small set of stage annotations to fit a dataset-level progress prior. Although we provide an annotation-free fallback, the best configuration may require modest benchmark-specific calibration. Finally, FRAMESKIP reallocates training exposure under a fixed optimization budget; it does not guarantee wall-clock speedups unless preprocessing, caching, and training-system overheads are accounted for in a specific deployment.

Ethics Statement

This work studies data selection for robot policy training in simulation and a limited real-world manipulation setting, and does not introduce new human-subject data or personally identifiable information. The main ethical risks are those associated with embodied manipulation systems: improved training efficiency can make capable robot policies easier to develop, but unsafe deployment may still cause physical damage or unintended behavior in real environments. We therefore view FRAMESKIP as a training-time tool that should be paired with standard robot safety practices, including constrained evaluation, failure monitoring, and task-specific safety checks before broader deployment. We also release code anonymously for review to support reproducibility while preserving the double-blind submission process.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, and 1 others. 2024. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*.

Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, Yang Tian, Bin Wang, Bolun Wang, Fangjing Wang, Hanqing Wang, Tai Wang, Ziqin Wang, Xueyuan Wei, Chao Wu, and 10 others. 2025. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *Preprint*, arXiv:2510.13778.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. 2023. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*.

Joey Hejna, Chethan Anand Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. 2024. Remix: Optimizing data mixtures for large scale imitation learning. In *8th Annual Conference on Robot Learning*.

Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. 2025. Robot data curation with mutual information estimators. *Preprint*, arXiv:2502.08623.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. 2025. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *Preprint*, arXiv:2507.16815.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, and 17 others. 2025. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *Preprint*, arXiv:2504.16054.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024. Openvla: An open-source vision-language-action model. In *Annual Conference on Robot Learning (CoRL)*.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. 2024a. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *Preprint*, arXiv:2411.19650.

Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong,

Hanbo Zhang, and Huaping Liu. 2024b. Towards generalist robot policies: What matters in building vision-language-action models. *Preprint*, arXiv:2412.14058.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. 2024c. Evaluating real-world robot manipulation policies in simulation. In *Annual Conference on Robot Learning (CoRL)*.

Shijie Lian, Bin Yu, Xiaopeng Lin, Laurence T. Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Cong Huang, and Kai Chen. 2026. Langforce: Bayesian decomposition of vision language action models via latent action queries. *Preprint*, arXiv:2601.15197.

Xiaopeng Lin, Shijie Lian, Bin Yu, Ruoqi Yang, Changti Wu, Yuzhuo Miao, Yurun Jin, Yukun Shi, Cong Huang, Bojun Cheng, and Kai Chen. 2025. Physbrain: Human egocentric data as a bridge from vision language models to physical intelligence. *Preprint*, arXiv:2512.16793.

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

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. 2025. F1: A vision-language-action model bridging understanding and generation to actions. *Preprint*, arXiv:2509.06951.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. 2024. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*.

NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, and 24 others. 2025a. Gr00t n1: An open foundation model for generalist humanoid robots. *Preprint*, arXiv:2503.14734.

NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, and 24 others. 2025b. Gr00t n1: An open foundation model for generalist humanoid robots. *Preprint*, arXiv:2503.14734.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, and 1 others. 2024. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. 2025. Fast: Efficient action tokenization for vision-language-action models. *Preprint*, arXiv:2501.09747.

Fanqi Pu, Lei Jiang, and Wenming Yang. 2026. Tgm-vla: Task-guided mixup for sampling-efficient and robust robotic manipulation. *Preprint*, arXiv:2603.00615.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. 2025. Spatialvla: Exploring spatial representations for visual-language-action model. *Preprint*, arXiv:2501.15830.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. 2025. Videovla: Video generators can be generalizable robot manipulators. *Preprint*, arXiv:2512.06963.

starVLA. 2025. Starvla: A lego-like codebase for vision-language-action model developing. GitHub repository.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. 2026. Vla-jepa: Enhancing vision-language-action model with latent world model. *Preprint*, arXiv:2602.10098.

GEAR Team, Allison Azzolini, Johan Bjorck, Valts Blukis, Fernando Castañeda, Rahul Chand, and 1 others. 2025. Gr00t n1.6: An improved open foundation model for generalist humanoid robots. `https://research.nvidia.com/labs/gear/gr00t-n1_6/`.

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. 2024. Octo: An open-source generalist robot policy. *Preprint*, arXiv:2405.12213.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, Feng Xiong, Xing Wei, Zhiheng Ma, and Mu Xu. 2026. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *Preprint*, arXiv:2602.11236.

Bin Yu, Shijie Lian, Xiaopeng Lin, Yuliang Wei, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Xinming Wang, Bailing Wang, Cong Huang, and Kai Chen. 2026. Twinbrainvla: Unleashing the potential of generalist vlms for embodied tasks via asymmetric mixture-of-transformers. *Preprint*, arXiv:2601.14133.

Yu Zhang, Yuqi Xie, Huihan Liu, Rutav Shah, Michael Wan, Linxi Fan, and Yuke Zhu. 2026. Scizor: Self-supervised data curation for large-scale imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. 2025. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. *Preprint*, arXiv:2503.22020.

Zhongyi Zhou, Yichen Zhu, Minjie Zhu, Junjie Wen, Ning Liu, Zhiyuan Xu, Weibin Meng, Ran Cheng, Yaxin Peng, Chaomin Shen, and Feifei Feng. 2025. Chatvla: Unified multimodal understanding and robot control with vision-language-action model. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

A Additional Implementation Details

A.1 Policy Architecture Details

For the controlled full-frame and FRAMESKIP runs, we use the same two-expert VLA architecture implemented with the QwenGR00T module in StarVLA (starVLA, 2025). Given a minibatch, the model receives image observations, a language instruction, an action sequence, and optionally robot state. The VLM interface first formats the images and instruction into Qwen-VL inputs and runs the VLM with hidden-state output enabled. We use the final VLM hidden states as the multimodal conditioning sequence for the action expert. The action head cross-attention dimension is set from the VLM hidden size, so the action expert can attend directly to the language-visual token sequence without an additional task-specific policy encoder.

The action expert is a flow-matching action head built on a cross-attention DiT. During training, the target action chunk is the current-and-future action window of length $H = future_action_window_size + 1$. The head samples Gaussian noise and a continuous flow-matching time, interpolates between noise and the target action chunk, and trains the DiT to predict the velocity from the noised action trajectory to the target action trajectory. The noised actions are embedded with an action encoder and sinusoidal time encoding; optional robot state is projected by an MLP; learned future tokens and action tokens are then processed by the DiT while cross-attending to the VLM hidden states. The decoded velocity is supervised with mean-squared error. At inference time, the action head initializes an action chunk from Gaussian noise and performs Euler integration for the configured number of flow-matching steps to produce normalized continuous actions. In our main configurations, the action expert uses a DiT-B backbone with 16 DiT layers, 32 target future tokens, 1000 timestep buckets, 4 inference steps, and an action horizon of 16 for RoboCasa-GR1/SimplerEnv and 8 for LIBERO; action and state dimensions follow the benchmark action schema.

A.2 Pruning Cache

FRAMESKIP stores trajectory-level pruning results in a cache containing the original importance scores and the retained indices for each configured ratio. The cache supports reuse across experiments as long as the importance and pruning configurations

remain compatible. During distributed training, cache construction can be restricted to rank zero and loaded by other workers after synchronization.

A.3 Frame-Score Preprocessing

For VAC, we use a DINOv2 visual encoder and extract visual features from at most 16 sparsely sampled video frames per trajectory before interpolating VAC scores back to the original trajectory length. The implementation records frame extraction failures and trajectories without usable visual features so that unreliable preprocessing runs can be identified before training.

For GMM-TPI, we fit the progress prior separately for each benchmark using 5% of the corresponding training trajectories. The annotation records only the normalized progress locations of manipulation-critical stage centers, such as alignment, grasping, and release; it does not provide action labels, success labels, or per-frame supervision to the policy. We fit a three-component one-dimensional GMM over these progress values and normalize the resulting density within each trajectory before adding it to the frame-importance score. This prior is used only during offline frame-score preprocessing. The VLA policy, training loss, and evaluation protocol do not access these annotations. When such annotations are unavailable, we use the dataset-agnostic Gaussian prior described in Section 3.

A.4 Dataloader Remapping

During training, each sampled index is first resolved to its trajectory and original timestep using the standard dataset index. Given the active retention ratio, the dataloader retrieves the cached retained indices for that trajectory and maps the requested timestep to the nearest retained timestep that is not earlier than the request, falling back to the final retained timestep at the end of the trajectory. The selected frame is then loaded through the original data access, transform, and collation pipeline. For analysis, each returned sample records the active ratio, original timestep, and remapped timestep.

A.5 Main Training Schedule

For the main experiments, the active compressed view uses a retention ratio of $r = 0.2$, corresponding to an 80% frame pruning ratio per trajectory. Training alternates between five mini-batches from this pruned view and one mini-batch from the full-frame view with $r = 1.0$. The full-frame mini-batch is used only as a periodic context anchor; evaluation is performed with the standard policy inference procedure and does not require frame pruning.

Because the three benchmarks use different training datasets and established evaluation recipes, we follow the commonly used training budgets for each benchmark rather than forcing a single step count across all settings. For RoboCasa-GR1, we train on the corresponding expert demonstration data for 100K optimization steps. For SimplerEnv, we train on the BridgeV2 dataset for 60K optimization steps. For LIBERO, we train on expert teleoperation demonstrations for 30K optimization steps.

A.6 Additional Ablation Details

The retention-ratio ablation studies the trade-off between denser trajectory context and stronger concentration on high-importance frames, using $r = 100\%$ as the full-frame reference. The importance-metric ablation uses random pruning as a pruning-only control, AVI as an action-only baseline, AVI+TPI and AVI+VAC to isolate auxiliary cues, AVI+VAC+TPI to test their combination, and FRAMESKIP Full to include gripper-transition preservation. The warmup ablation varies only the number of initial full-frame optimization steps; after warmup, all variants use the same retention ratio and pruned/full-frame mini-batch schedule. The training-budget ablation keeps the warmup length fixed and compares full-frame training and FRAMESKIP under matched fractions of the standard benchmark budget, varying only the total number of optimization steps.

B Full RoboCasa-GR1 Results

Table 8: **Results of evaluating the VLA models with the GR1 robot in the RoboCasa-GR1 Tabletop simulation environment**. The results for Isaac-GR00T N1.5 and Isaac-GR00T N1.6 are sourced from the official Isaac-GR00T GitHub repository (NVIDIA et al., 2025b). We highlight the best results in **bold** and the second-best results with underline.

Task	GR00T N1.5	GR00T N1.6	VP-VLA	TwinBrainVLA	PhysBrain	LangForce	FrameSkip
PnP Bottle To Cabinet Close	54.0	51.5	54.0	74.0	74.0	72.0	74.0
PnP Can To Drawer Close	50.0	13.0	72.0	72.0	68.0	78.0	82.0
PnP Cup To Drawer Close	38.0	8.5	44.0	52.0	42.0	46.0	46.0
PnP Milk To Microwave Close	60.0	14.0	74.0	60.0	54.0	56.0	64.0
PnP Potato To Microwave Close	32.0	41.5	34.0	36.0	24.0	36.0	46.0
PnP Wine To Cabinet Close	38.0	16.5	48.0	46.0	54.0	46.0	76.0
PnP * to * Close (Avg)	45.3	24.2	54.3	56.7	52.7	55.7	**63.7**
PnP Novel From Cuttingboard To Basket	38.0	58.0	66.0	62.0	62.0	66.0	58.0
PnP Novel From Cuttingboard To Cardboardbox	46.0	46.5	54.0	46.0	44.0	40.0	58.0
PnP Novel From Cuttingboard To Pan	58.0	68.5	74.0	70.0	56.0	68.0	70.0
PnP Novel From Cuttingboard To Pot	62.0	65.0	54.0	66.0	58.0	48.0	66.0
PnP Novel From Cuttingboard To Tieredbasket	28.0	46.5	56.0	52.0	40.0	44.0	54.0
PnP Novel From Cuttingboard To * (Avg)	46.4	56.9	60.8	59.2	52.0	53.2	**61.2**
PnP Novel From Placemat To Basket	30.0	58.5	48.0	30.0	42.0	54.0	52.0
PnP Novel From Placemat To Bowl	60.0	57.5	74.0	54.0	56.0	62.0	66.0
PnP Novel From Placemat To Plate	56.0	63.0	70.0	64.0	80.0	52.0	66.0
PnP Novel From Placemat To Tieredshelf	36.0	28.5	26.0	38.0	14.0	24.0	30.0
PnP Novel From Placemat To * (Avg)	45.5	51.9	**54.5**	46.5	48.0	48.0	53.5
PnP Novel From Tray To Cardboardbox	52.0	51.5	44.0	46.0	40.0	50.0	54.0
PnP Novel From Tray To Plate	48.0	71.0	66.0	72.0	66.0	58.0	62.0
PnP Novel From Tray To Pot	60.0	64.5	38.0	56.0	52.0	62.0	66.0
PnP Novel From Tray To Tieredbasket	52.0	57.0	58.0	46.0	50.0	44.0	66.0
PnP Novel From Tray To Tieredshelf	32.0	31.5	24.0	28.0	22.0	22.0	40.0
PnP Novel From Tray To * (Avg)	48.8	55.1	46.0	49.6	46.0	47.2	**57.6**
PnP Novel From Plate To Bowl	58.0	57.0	52.0	60.0	54.0	54.0	58.0
PnP Novel From Plate To Cardboardbox	44.0	43.5	44.0	46.0	50.0	48.0	46.0
PnP Novel From Plate To Pan	60.0	51.0	56.0	56.0	68.0	54.0	62.0
PnP Novel From Plate To Plate	64.0	78.7	62.0	66.0	78.0	78.0	72.0
PnP Novel From Plate To * (Avg)	56.5	57.6	53.5	57.0	**62.5**	58.5	59.5
Average	48.2	47.6	53.8	54.6	50.0	52.6	**59.5**