LEARNING WHEN AND WHAT DATA TO USE FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) policies use heterogeneous inputs whose relevance can change during task execution. We study which already encoded source representations should continue to participate in action prediction. We propose **DataRouter**, a lightweight source-routing module conditioned on the VLA's current decision state. It scores native source representations against intermediate VLA states and retains a subset using an adaptive utility-coverage rule. Only the routing components are trained from demonstrated actions; pretrained VLA parameters remain frozen. Continuous gates support training, and hard selection omits unselected source representations from subsequent prediction. Across five evaluated VLA configurations, mean LIBERO-Plus success increases from 61.52% to 66.80%, and mean CALVIN chain length increases from 3.316 to 3.686. Mean source-routing ratios are 0.45 and 0.44, respectively. In the $\pi_{0.5}$ efficiency evaluation, relative FLOPs decrease to 0.58 and latency falls from 42.6 to 28.7 ms, including encoding and routing overhead. These results support decision-conditioned source utilization in the evaluated VLA configurations.

1 INTRODUCTION

Vision-language-action (VLA) models ground visual observations and language instructions in robot actions (Brohan et al., 2022; 2023a; Kim et al., 2025). Generalist policies increasingly combine multiple sensors, robot states, and interaction history (Mees et al., 2024; Black et al., 2024; 2025; Shukor et al., 2025; Li et al., 2024). However, access to more information does not imply that every source is equally useful at every step. Visual observations may be critical for localization, while state and history may become more informative during contact or motion execution. This motivates studying source relevance as a function of the current decision context.

We first empirically examine whether the same data source remains equally useful throughout VLA inference. Using a fixed $\pi_{0.5}$ policy on LIBERO-Plus (Black et al., 2025; Fei et al., 2025), we selectively remove image, text, or both inputs only within an early, middle, or late portion of task execution, while keeping the remaining inference process unchanged. We evaluate the resulting change in rollout success rate $\Delta\mathrm{SR} = \mathrm{SR}_{\mathrm{drop}} - \mathrm{SR}_{\mathrm{full}}$, where negative values indicate that removing the corresponding input harms task completion, while positive values indicate that its removal improves performance.

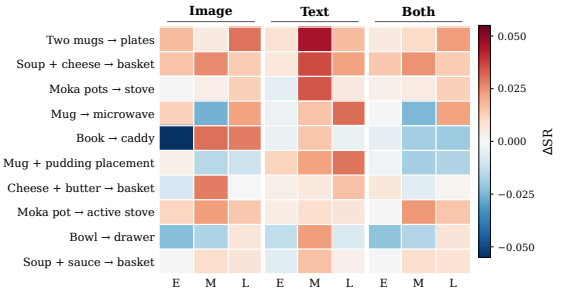
Figure 1 shows stage- and task-dependent effects. Removing text changes success by $+0.0002$, $+0.0223$, and $+0.0118$ in the early, middle, and late stages, respectively. Image removal slightly hurts early execution but improves the aggregate score in later stages. Task-level effects also differ in sign. These observations motivate conditioning source selection on the current task context. They describe the effects of input removal in this evaluation.

Our central question is whether a VLA can learn *when already encoded data remain useful and which sources should continue to participate in action prediction*. Multimodal fusion and conditional feature modulation already allow input-dependent weighting (Baltrušaitis et al., 2018; Perez et al., 2018; Arevalo et al., 2017). We condition source selection on the context already formed by the policy. Source-level granularity makes information types the objects of selection; decision-state conditioning makes their relevance depend on the current task stage. Routing acts after the representations needed for this assessment have been computed.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Removed	Early	Middle	Late
Image	−0.0029	+0.0067	+0.0124
Text	+0.0002	**+0.0223**	+0.0118
Both	−0.0001	−0.0011	+0.0068

(a) Overall ΔSR



(b) Task-level ΔSR

Figure 1: **Source-removal effects vary across stages and tasks.** Image, text, or both inputs are removed during early, middle, or late $\pi_{0.5}$ execution on LIBERO-Plus. Positive ΔSR indicates improved task success after removal; values are changes in success probability, not percentages.

To address this question, we propose **DataRouter**. A lightweight scoring module matches each native source representation to the current VLA decision state. Adaptive-Budget routing retains the highest-scoring sources until their normalized weights cover a fixed fraction of the total. The selected representations then continue through the frozen action-prediction computation. Only the routing components are trained from demonstrated actions.

Our contribution has two parts. First, DataRouter provides decision-conditioned source selection within an existing VLA forward execution, reusing the model's representations and preserving its pretrained weights. Second, evaluation on LIBERO-Plus and CALVIN across five VLA configurations shows improved task performance with reduced source participation. Existing ablations assess the conditioning and selection components, and a $\pi_{0.5}$ study measures the resulting inference cost. Learned-score diagnostics complement this evaluation; alternative selection rules and their parameter sweeps are provided in the appendix.

2 Related Work

Vision-language-action policies and robot data. RT-1 and RT-2 scale transformer-based robot control and transfer visual-language knowledge to actions (Brohan et al., 2022; 2023a). OpenVLA, Octo, π_0, $\pi_{0.5}$, and SmolVLA extend generalist policy learning across data sources and deployment settings (Kim et al., 2025; Mees et al., 2024; Black et al., 2024; 2025; Shukor et al., 2025). Diverse demonstrations are available from Open X-Embodiment, BridgeData V2, and DROID (Vuong et al., 2023; Walke et al., 2023; Khazatsky et al., 2024), with ALOHA 2 supporting bimanual data collection (Aldaco et al., 2024). Multimodal prompting, embodied reasoning, affordance grounding, action hierarchies, and history modeling provide complementary policy context (Jiang et al., 2023; Driess et al., 2023; Brohan et al., 2023b; Belkhale et al., 2024; Li et al., 2024). Action chunking and diffusion policies address action-sequence prediction (Zhao et al., 2023; Chi et al., 2025); flow matching provides another generative modeling foundation (Lipman et al., 2022). DataRouter studies which encoded source representations to retain for subsequent action prediction at each decision step.

Adaptive computation and routing. Adaptive computation time varies processing depth (Graves, 2016), while sparse mixture-of-experts models select parameterized modules (Shazeer et al., 2017; Fedus et al., 2022). Dynamic token pruning, adaptive halting, and token merging reduce visual-token computation (Rao et al., 2021; Yin et al., 2022; Bolya et al., 2022); ATP-LLaVA and LLaVA-PruMerge extend token reduction to multimodal models (Ye et al., 2025; Shang et al., 2025). In robot policies, LightVLA and VLA-Pruner prune visual tokens (Jiang et al., 2025; Liu et al., 2025), AC^2-VLA adapts computation using action context (Yu et al., 2026), and SP-VLA combines model scheduling with pruning (Li et al., 2025). DREAM performs context-dependent expert routing (Sheng et al., 2026). These methods already employ dynamic, context-dependent selection. DataRouter instead asks which already encoded information sources should continue to participate in action prediction given the VLA's existing context. Source-level grouping makes redundancy,

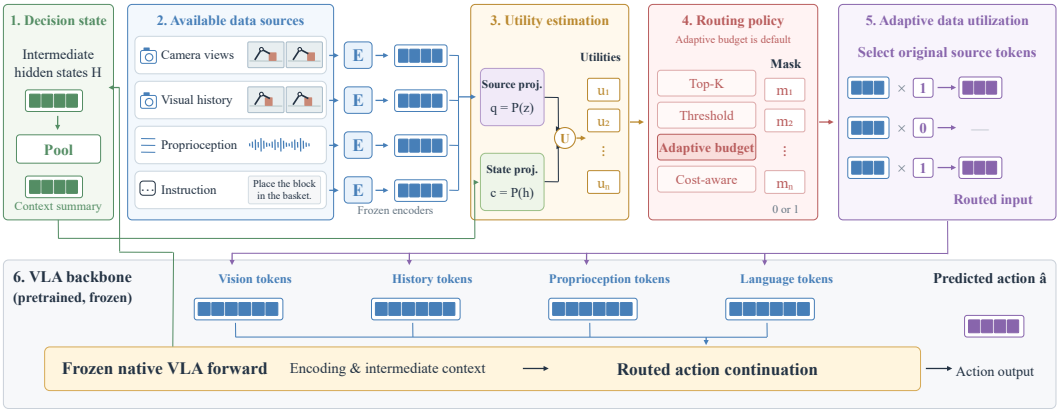


Figure 2: Overview of DataRouter. Encoded sources and the current VLA decision state are mapped into a shared routing space to estimate source utilities. Adaptive-Budget is the default source-selection rule. The alternative rules shown in the policy block are detailed in Appendix C.1. Selected representations continue through the same frozen VLA forward.

complementarity, and changing demand across vision, language, state, and history explicit selection considerations. DataRouter applies this distinction through a lightweight scoring module and adaptive source selection.

3 METHOD

3.1 PROBLEM FORMULATION

At decision step t, a pretrained VLA receives sources $\mathcal{D}_t = \{d_t^i\}_{i=1}^{N_t}$, where N_t is the number of available source groups. Each source is an identifiable input group, such as vision, instruction, state, or history; a group may contain multiple camera views. We learn which encoded groups should continue to participate in action prediction, conditioned on the VLA's decision state (Figure 2). Grouping tokens by source exposes heterogeneous information types as selection units, allowing their relevance to change with context. Routing takes place after native encoding and decision-state extraction. It retains their earlier influence and computation cost. Utility denotes a learned score for subsequent action prediction.

3.2 SOURCE AND DECISION-STATE REPRESENTATIONS

Available data. We reuse the pretrained VLA encoders to represent each source and project its representation into a shared routing space:

$$z_t^i = E_\theta^i(d_t^i), \qquad q_t^i = P_\psi(z_t^i) \in \mathbb{R}^{d_r}, \qquad i = 1, \ldots, N_t, \tag{1}$$

where E_θ^i is the corresponding frozen encoder, P_ψ denotes lightweight source pooling and projection, and d_r is the routing dimension. Projection input dimensions follow the native representation interface; all outputs share the routing dimension. We retain original representations $\{z_t^i\}$ for action prediction; projected representations $\{q_t^i\}$ serve only routing. Appendix A.1 lists the native interfaces used for the evaluated configurations.

Current decision state. Source representations describe the available information, whereas the decision state summarizes the context already formed by the VLA. We extract intermediate hidden states before final action prediction:

$$\mathcal{H}_t = \mathrm{VLA}_\theta^{\mathrm{int}}(x_t) \in \mathbb{R}^{L_t \times d_h}, \qquad \tilde{c}_t = P_{\mathrm{state}}\left(\mathrm{Pool}(\mathcal{H}_t)\right) \in \mathbb{R}^{d_r}, \tag{2}$$

where x_t is the current VLA input, L_t is the hidden-state sequence length, and d_h is the hidden dimension. The pooling operator produces a fixed-dimensional summary, which P_{state} maps into the routing space. Both are reused from the same native VLA forward execution.

State-conditioned utility. We estimate each source's utility from its routing representation and the current decision state:

$$u_{t,i} = U_\phi(q_t^i, \tilde{c}_t), \qquad \mathbf{u}_t = [u_{t,1}, \ldots, u_{t,N_t}], \qquad \mathbf{m}_t = \mathcal{G}(\mathbf{u}_t), \tag{3}$$

where U_ϕ denotes the learned utility matching function and $\mathcal{G}$ is a routing policy. Each binary entry $m_{t,i} \in \{0, 1\}$ determines whether source i is selected.

Scoring implementation. For a token-sequence source $z_t^i \in \mathbb{R}^{L_i \times d_i}$, the projection first applies layer normalization and mean pooling:

$$s_t^i = \frac{1}{L_i} \sum_{\ell=1}^{L_i} \mathrm{LN}(z_{t,\ell}^i), \qquad q_t^i = W_{\mathrm{src}} s_t^i + b_{\mathrm{src}}. \tag{4}$$

A source already represented by a vector is used directly as s_t^i. Decision states use the same normalization and pooling, followed by a separate linear projection. Input projection dimensions follow each native interface, with interface indices suppressed in the notation. All projected representations have dimension $d_r = 128$.

The utility estimator is a normalized bilinear interaction:

$$u_{t,i} = \bar{q}_t^{i\top} W_r \bar{c}_t, \qquad \bar{q}_t^i = \frac{q_t^i}{\|q_t^i\|_2}, \qquad \bar{c}_t = \frac{\tilde{c}_t}{\|\tilde{c}_t\|_2}, \tag{5}$$

where $W_r \in \mathbb{R}^{d_r \times d_r}$ is trainable. Pooling is used for scoring; selected sources retain their original token representations for action prediction. The cosine-utility ablation sets $W_r = I$.

3.3 Adaptive-Budget Source Selection

DataRouter uses cumulative utility coverage to adapt the number of retained sources at each decision step. This rule requires a coverage parameter rather than predefined per-source costs.

We normalize utilities as $w_{t,i} = \mathrm{softmax}(\mathbf{u}_t/\tau_u)_i$ with utility temperature $\tau_u > 0$ and sort them as $w_{t,(1)} \geq \cdots \geq w_{t,(N_t)}$. The selected budget is the smallest number of sources covering a target fraction $\rho \in (0, 1)$ of the normalized utility mass:

$$K_t = \min \left\{ k \in \{1, \ldots, N_t\} : \sum_{j=1}^{k} w_{t,(j)} \geq \rho \right\}, \qquad m_{t,i} = \mathbb{I}\left[i \in \mathrm{TopK}(\mathbf{u}_t, K_t)\right]. \tag{6}$$

We use $\tau_u = 1.0$ and fix $\rho = 0.70$ before evaluation. Concentrated weights can reach the coverage target with fewer sources, whereas more distributed weights require a larger subset. Coverage refers to normalized learned scores. It is not a guarantee about retained task information or success probability.

The same scoring module also supports fixed-budget Top-K, Threshold, and Cost-Aware rules. We treat these as alternative inference choices, described and evaluated in Appendix C.

3.4 Adaptive Data Utilization

At inference, the routing mask determines which original source representations enter subsequent action prediction:

$$Z_t^{\mathrm{route}} = \mathrm{Concat}\left(\{z_t^i \mid m_{t,i} = 1\}\right), \qquad \hat{a}_t = \pi_\theta\left(Z_t^{\mathrm{route}}\right), \tag{7}$$

where concatenation follows the original source order along the token dimension. Here, π_θ denotes the remaining frozen action-prediction computation within the same forward execution. Unselected source tokens are omitted from this subsequent computation.

Following continuous relaxations of discrete selection (Maddison et al., 2016; Jang et al., 2016; Xie et al., 2020; Louizos et al., 2017), we train with normalized gates $\hat{m}_{t,i} = \mathrm{softmax}(\mathbf{u}_t/\tau_g)_i$ using $\tau_g = 0.5$:

$$\hat{\mathbf{m}}_t = \mathcal{G}^{\mathrm{soft}}(\mathbf{u}_t) \in [0, 1]^{N_t}, \qquad Z_{t,\mathrm{train}}^{\mathrm{route}} = \left[\hat{m}_{t,1} z_t^1; \ldots; \hat{m}_{t,N_t} z_t^{N_t}\right]. \tag{8}$$

3.5 END-TO-END OPTIMIZATION

We optimize the routing modules using demonstrated actions while keeping all pretrained VLA parameters θ fixed. The trainable parameters Θ_{route} comprise P_ψ, P_{state}, and U_ϕ. For a trajectory $\{(\mathcal{D}_t, a_t^*)\}_{t=1}^T$, the action-supervision objective is written in architecture-independent form as

$$\mathcal{L}_{\text{BC}} = \frac{1}{T}\sum_{t=1}^{T} \ell_\theta\big(a_t^*, Z_{t,\text{train}}^{\text{route}}\big)\,. \tag{9}$$

Here ℓ_θ denotes the backbone's action-training loss. Negative log-likelihood applies to discrete action-token models; flow-based policies require their native flow-matching objective (Kim et al., 2025; Black et al., 2024; Shukor et al., 2025; Lipman et al., 2022). Although the backbone is frozen, gradients propagate through its action-prediction computation to the routing modules. Utility estimates are thus trained to support action prediction, without direct supervision of source importance.

We train the routing module separately for each evaluated VLA configuration. Appendix A.2 gives implementation details, and Appendix A.3 lists training hyperparameters.

The routing parameters are optimized solely with the action-supervision loss:

$$\Theta_{\text{route}}^* = \arg\min_{\Theta_{\text{route}}} \mathcal{L}_{\text{BC}}. \tag{10}$$

The coverage parameter controls hard selection at inference; it is not part of the training loss.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate our method on two robot manipulation benchmarks, **LIBERO-Plus** (Fei et al., 2025; Liu et al., 2023) and **CALVIN** (Mees et al., 2022), covering diverse manipulation tasks and long-horizon embodied decision making. LIBERO-Plus evaluation covers four suites, all seven perturbation types, and difficulty Levels 1–3, with five fixed-seed episodes per perturbed task. CALVIN uses 1,000 five-instruction chains under ABC→D. For LIBERO-Plus, we average five fixed-seed rollouts per perturbed task, macro-average within each suite, and give the four suites equal weight. Methods use the same initial conditions and seed schedule. CALVIN reports cumulative completion rates for instruction chains in the held-out environment D. Appendix B.1 provides the full evaluation protocols.

VLA Backbones. We evaluate our framework with five pretrained VLA configurations: $\pi_{0.5}$, OpenVLA, SmolVLA, AC^2-VLA, and LightVLA (Black et al., 2025; Kim et al., 2025; Shukor et al., 2025; Yu et al., 2026; Jiang et al., 2025). The last two are adaptive-computation configurations. All experiments retain the pretrained VLA weights and train only DataRouter. Implementation details for adapting the routing module to each VLA are provided in Appendix A.1.

Baselines. We compare DataRouter with three groups of methods. These comprise original all-data VLA inference; uniform fusion, attention fusion, soft gating, and random routing; and VLA-Pruner and SP-VLA (Liu et al., 2025; Li et al., 2025), where applicable. DataRouter uses Adaptive-Budget routing unless otherwise specified. Appendix C compares alternative inference rules using the same trained scoring module.

Evaluation Metrics. We report **success rate (SR, %)** on LIBERO-Plus and mean completed instruction-chain length (Len., maximum 5) on CALVIN. To quantify data utilization, we additionally report the **Routing Ratio (RR)**, defined as $\text{RR} = \frac{1}{T}\sum_{t=1}^{T} \frac{\sum_{i=1}^{N_t} m_{t,i}}{N_t}$, where $m_{t,i}$ indicates whether the i-th data source is selected at decision step t. A lower RR indicates fewer source groups retained for subsequent prediction, not a proportional reduction in tokens or FLOPs. We report source RR as "–" for VLA-Pruner and SP-VLA, whose token pruning does not define source-level routing.

4.2 MAIN RESULTS

LIBERO-Plus. We first evaluate whether decision-conditioned source selection improves task success across the five frozen VLA configurations.

Table 1: Main results on LIBERO-Plus across different pretrained VLA backbones. SR denotes success rate (%) and RR denotes the routing ratio, i.e., the fraction of available source groups retained for subsequent action prediction; "–" denotes methods without source-level routing. ↑ indicates higher is better and ↓ indicates lower is better.

	$\pi_{0.5}$		OpenVLA		SmolVLA		AC^2-VLA		LightVLA	
Method	SR↑	RR↓	SR↑	RR↓	SR↑	RR↓	SR↑	RR↓	SR↑	RR↓
Standard VLA Inference										
Original VLA	84.4	1.00	15.6	1.00	62.8	1.00	71.3	1.00	73.5	1.00
Adaptive Fusion / Routing										
Uniform Fusion	83.1	1.00	14.9	1.00	61.4	1.00	70.1	1.00	72.2	1.00
Attention Fusion	84.9	1.00	16.8	1.00	63.5	1.00	72.0	1.00	74.1	1.00
Soft Gating	85.6	1.00	17.4	1.00	64.2	1.00	72.6	1.00	74.8	1.00
Random Routing	79.2	0.62	12.1	0.65	57.3	0.61	66.8	0.63	68.4	0.60
VLA-Specific Adaptive Computation										
VLA-Pruner	85.1	–	16.2	–	63.9	–	72.4	–	75.6	–
SP-VLA	85.8	–	17.0	–	64.7	–	73.1	–	76.2	–
Ours										
DataRouter	**88.7**	**0.47**	**21.3**	**0.49**	**68.4**	**0.45**	**76.5**	**0.43**	**79.1**	**0.41**

Task performance improves across configurations. Table 1 shows that mean SR increases from 61.52% for the original policies to 66.80% with DataRouter, a gain of 5.28 percentage points. The individual gains are 4.3 points for $\pi_{0.5}$, 5.7 for OpenVLA, 5.6 for SmolVLA, 5.2 for AC^2-VLA, and 5.6 for LightVLA. Compared with SP-VLA, the strongest aggregate baseline, the mean gain is 3.44 points. All five configurations improve, although their absolute success rates remain different.

Improvements extend across task suites. The per-suite results in Appendix B show gains in Spatial, Object, Goal, and Long tasks. For $\pi_{0.5}$, these gains are 3.6, 3.1, 4.2, and 6.3 percentage points, respectively. Long-suite SR rises from 70.8% to 77.1%. Similar gains occur for SmolVLA (48.1% to 54.3%), AC^2-VLA (56.8% to 62.8%), and LightVLA (59.1% to 65.3%). The improvement therefore extends beyond the suites with higher baseline success. OpenVLA improves in every suite, but retains lower overall SR than the other evaluated configurations.

Learned selection retains fewer sources. Mean RR is 0.45, corresponding to 55% fewer selected groups than all-data inference. Random routing instead lowers SR in every configuration despite retaining more groups than DataRouter. For example, $\pi_{0.5}$ random routing gives 79.2% SR at RR$=0.62$, versus 88.7% at RR$=0.47$ for DataRouter. This motivates learning which representations to retain; the comparison is not budget-matched.

Comparison with soft fusion. Soft gating improves over the original VLA in all five configurations while retaining every source. DataRouter further improves SR by 3.1–4.3 percentage points over soft gating, with RR ranging from 0.41 to 0.49. Thus, in these evaluations, the improved scores accompany hard source selection rather than requiring every encoded group to remain active. This comparison describes the complete methods; Section 4.3 examines soft routing within the DataRouter implementation.

CALVIN. We next examine long-horizon instruction following under ABC→D. Table 2 reports mean completed chain length and source participation.

Table 2: Main results on CALVIN under the standard ABC→D protocol with 1,000 five-instruction chains. We report mean completed chain length (Len.) and source RR; "–" denotes token-pruning methods without source-level routing. ↑ indicates higher is better and ↓ indicates lower is better.

	$\pi_{0.5}$		OpenVLA		SmolVLA		AC^2-VLA		LightVLA	
Method	Len.↑	RR↓	Len.↑	RR↓	Len.↑	RR↓	Len.↑	RR↓	Len.↑	RR↓
Standard VLA Inference										
Original VLA	4.02	1.00	2.31	1.00	3.18	1.00	3.45	1.00	3.62	1.00
Adaptive Fusion / Routing										
Uniform Fusion	3.91	1.00	2.18	1.00	3.05	1.00	3.32	1.00	3.48	1.00
Attention Fusion	4.08	1.00	2.42	1.00	3.26	1.00	3.51	1.00	3.69	1.00
Soft Gating	4.11	1.00	2.47	1.00	3.31	1.00	3.56	1.00	3.74	1.00
Random Routing	3.64	0.61	1.95	0.64	2.87	0.60	3.12	0.62	3.28	0.59
VLA-Specific Adaptive Computation										
VLA-Pruner	4.09	–	2.39	–	3.24	–	3.53	–	3.78	–
SP-VLA	4.13	–	2.45	–	3.29	–	3.58	–	3.82	–
Ours										
DataRouter	**4.28**	**0.46**	**2.76**	**0.48**	**3.57**	**0.44**	**3.81**	**0.42**	**4.01**	**0.40**

Long-horizon performance improves across configurations. Mean completed chain length increases from 3.316 to 3.686, with gains in all five evaluated configurations. Relative to SP-VLA, the mean improvement is 0.232. For AC^2-VLA, DataRouter reaches 3.81 compared with 3.58 for SP-VLA; for LightVLA, the corresponding values are 4.01 and 3.82. DataRouter also exceeds the evaluated fusion and random-routing alternatives in every configuration.

Gains persist through the full instruction chain. The complete results in Appendix B complement these averages with cumulative success rates. For $\pi_{0.5}$, one-instruction success rises from 91.9% to 94.2%, while five-instruction success rises from 70.6% to 77.6%. Five-instruction gains are 12.0 percentage points for OpenVLA, 13.0 for SmolVLA, 12.5 for AC^2-VLA, and 13.5 for LightVLA. These rates show improvement in completing the full sequence, beyond the first instruction. Since later success requires earlier completion, they do not isolate an individual stage's contribution.

Source participation remains below all-data inference. Mean RR is 0.44 on CALVIN, corresponding to 56% fewer selected groups. The mean chain length summarizes all five cumulative completion rates, whereas RR summarizes which encoded groups continue through subsequent prediction. Together, these measurements describe improved task completion with reduced source participation. Section 4.4 separately measures whether this reduction translates into lower FLOPs and latency for the evaluated $\pi_{0.5}$ configuration.

The mean and final-chain rate provide complementary views. For $\pi_{0.5}$, the improvements at chain lengths one through five are 2.3, 4.3, 6.0, 6.4, and 7.0 percentage points. Their sum, divided by 100, gives the 0.26 increase in mean chain length. The gain therefore includes improvements across the cumulative completion curve, rather than only a change at its endpoint. The appendix reports the full curve for each configuration, allowing the aggregate scores to be interpreted alongside complete-chain success.

4.3 Ablation Studies

We vary one component at a time using $\pi_{0.5}$ on LIBERO-Plus. All variants use Adaptive-Budget routing unless otherwise noted.

Table 3: Component ablations on LIBERO-Plus with $\pi_{0.5}$.

Variant	SR ↑	RR ↓
DataRouter	**88.7**	**0.47**
w/o Decision State	85.9	0.58
w/ Cosine Utility	86.8	0.51
w/ Soft Routing	86.2	1.00

Conditioning and learned matching. Removing decision state lowers SR from 88.7% to 85.9% and increases RR from 0.47 to 0.58. Replacing bilinear matching with cosine similarity gives 86.8% SR at RR$=0.51$. Table 3 supports these components within the complete routing system, including their effects on the resulting budget.

Hard selection improves over the evaluated soft variant. Soft routing retains all sources (RR$=1.00$) but reaches 86.2% SR, compared with 88.7% at RR$=0.47$ for DataRouter. The full system has both the highest SR and the lowest RR among these ablations. This comparison concerns the implemented training and inference choices.

4.4 Efficiency Analysis

We measure full inference cost for $\pi_{0.5}$ on LIBERO-Plus, including encoding, routing, and subsequent prediction. Table 4 reports FLOPs relative to the original VLA and milliseconds per decision step on an A100; "–" denotes no source-level RR.

Table 4: Inference efficiency on LIBERO-Plus with $\pi_{0.5}$.

Method	SR ↑	RR ↓	FLOPs ↓	ms ↓
Original VLA	84.4	1.00	1.00	42.6
Attention Fusion	84.9	1.00	1.03	44.1
Soft Gating	85.6	1.00	1.05	44.8
VLA-Pruner	85.1	–	0.71	33.4
SP-VLA	85.8	–	0.64	30.9
DataRouter	**88.7**	**0.47**	**0.58**	**28.7**

Lower measured inference cost. DataRouter reduces relative FLOPs to 0.58 and latency from 42.6 to 28.7 ms while improving SR to 88.7%. These correspond to reductions of 42% and 32.6%, respectively. Compared with SP-VLA, it improves SR from 85.8% to 88.7%, with lower FLOPs (0.58 versus 0.64) and latency (28.7 versus 30.9 ms).

Source count and computation measure different effects. Retaining 47% of groups uses 58% of the original FLOPs because groups have different token lengths and pre-routing computation is retained. The wall-clock reduction is smaller than the FLOPs reduction. Attention fusion and soft gating slightly increase cost because all sources remain active. Appendix A.4 details the protocol. These efficiency measurements apply to this $\pi_{0.5}$ setup; the other configurations are assessed through task performance and source RR.

Measurement conditions. All compared methods use the same checkpoint, input configuration, action horizon, image resolution, and camera views. Timing uses batch size one, native mixed precision, 100 warm-up iterations, and 1,000 measured decision steps. CUDA synchronization brackets timing, with evaluation mode and gradients disabled. FLOPs include native encoding and all subsequent computation. No extra batching or caching is introduced beyond the original baseline implementation. This shared protocol makes the reported latency comparison specific to the same inference workload.

4.5 Behavior of the Default Router

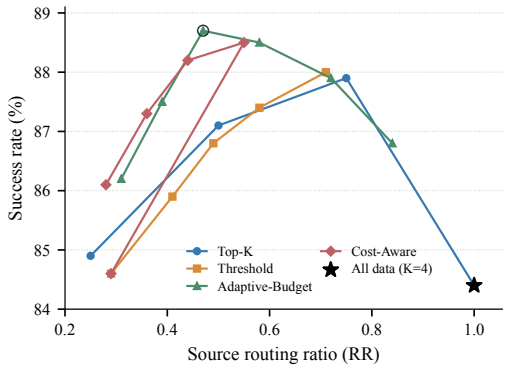


Figure 3: Performance–routing trade-off on LIBERO-Plus with $\pi_{0.5}$. The four rules share a trained scoring module.

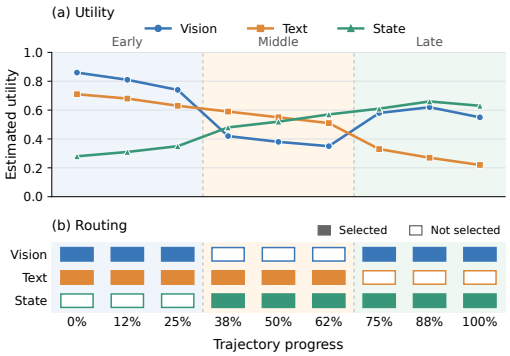


Figure 4: Learned utilities and routing masks for the displayed source groups along one $\pi_{0.5}$ trajectory.

Scores and selections change during execution. In Figure 4, vision and text have larger weights early; state becomes prominent during grasp and transport, and vision gains weight near placement. This describes one rollout, rather than a universal stage-wise ordering.

Coverage controls retained source count. The $\pi_{0.5}$ sweep uses Vision, Text, State, and History. The predefined $\rho = 0.70$ retains 1.88 groups on average (RR$= 0.47$), with 88.7% SR. Coverage 0.60 gives 1.56 groups and 87.5% SR; coverage 0.80 gives 2.32 groups and 88.5% SR. At 0.95, SR falls to 86.8% despite retaining 3.36 groups. Figure 3 summarizes the operating points; Appendix C details the alternative rules and sweeps.

Table 5: Spearman correlation with leave-one-out source sensitivity, computed for evaluation only. Sensitivity labels are not used in training.

Relevance Signal	$\pi_{0.5}$	OpenVLA	SmolVLA	AC^2-VLA	LightVLA	Average
Feature Similarity	0.31	0.27	0.29	0.33	0.30	0.30
Attention Score	0.42	0.38	0.40	0.44	0.41	0.41
Decision-State Similarity	0.48	0.43	0.46	0.51	0.47	0.47
Learned Utility (Ours)	**0.67**	**0.61**	**0.64**	**0.69**	**0.66**	**0.65**

Learned scores agree with source-removal sensitivity. Table 5 reports mean Spearman correlation 0.65 for learned utility, compared with 0.47 for decision-state similarity and 0.41 for attention scores. Learned-score correlations range from 0.61 to 0.69 across configurations. This provides a diagnostic alongside task performance; redundancy and complementarity make removal sensitivity distinct from causal source importance. Appendix D defines the diagnostic, and Appendix C gives alternative rules and complete parameter sweeps.

5 Conclusion

DataRouter selects which encoded source representations continue to participate in action prediction, conditioned on the current VLA decision state. Its lightweight scoring module and Adaptive-Budget rule reuse native representations while preserving pretrained VLA weights. On LIBERO-Plus and CALVIN, the evaluated configurations show improved task performance with fewer retained source groups. Existing ablations support the complete routing design, and the $\pi_{0.5}$ efficiency study reports lower inference cost. These results establish an empirical case for decision-conditioned source utilization in the evaluated settings.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing, improving the clarity and organization of the manuscript.

REPRODUCIBILITY STATEMENT

Code for DataRouter will be made available at `https://anonymous.4open.science/r/datarouter-BD84`. Sections 3.1–3.5 describe the routing formulation and training objective. Appendix A specifies VLA interfaces, training, and efficiency measurement. Appendix B gives benchmark protocols and complete results; Appendices C and D contain alternative routing rules and auxiliary diagnostics. We retain the original pretrained VLA weights and train only DataRouter. Source representations and intermediate decision states are reused within the same native forward execution, without a second VLA pass.

REFERENCES

Jorge Aldaco, Travis Armstrong, Robert Baruch, Jeff Bingham, Sanky Chan, Kenneth Draper, Debidatta Dwibedi, Chelsea Finn, Pete Florence, Spencer Goodrich, et al. Aloha 2: An enhanced low-cost hardware for bimanual teleoperation. *arXiv preprint arXiv:2405.02292*, 2024.

John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González. Gated multimodal units for information fusion. *arXiv preprint arXiv:1702.01992*, 2017.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2): 423–443, 2018.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

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

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning*, pp. 17–40, 2025.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023a.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pp. 287–318, 2023b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, pp. 8469–8488, 2023.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, pp. 2679–2713, 2025.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, volume 2024, pp. 26703–26721, 2024.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration. In *The Fourteenth International Conference on Learning Representations*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the semantic-action gap in visual token pruning for efficient vla inference. *arXiv preprint arXiv:2511.16449*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Oier Mees, Dibya Ghosh, Karl Pertsch, Kevin Black, Homer Rich Walke, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *First workshop on vision-language models for navigation and manipulation at ICRA 2024*, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, 2024.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763, 2021.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *IEEE/CVF International Conference on Computer Vision*, pp. 22857–22867, 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Kai Sheng, Liuyi Wang, Zongtao He, Xiao Lin, Chengju Liu, and Qijun Chen. Dream: Dynamic routing of experts via attention-based mixture for vision-language-action modeling. *Knowledge-Based Systems*, pp. 115585, 2026.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on robot learning*, pp. 1723–1736, 2023.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. *Advances in neural information processing systems*, 33:20520–20531, 2020.

Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. Atp-llava: Adaptive token pruning for large vision language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24972–24982, 2025.

Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10799–10808, 2022.

Wenda Yu, Tianshi Wang, Fengling Li, Jingjing Li, and Lei Zhu. Acˆ 2-vla: Action-context-aware adaptive computation in vision-language-action models for efficient robotic manipulation. *arXiv preprint arXiv:2601.19634*, 2026.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *International Conference on Computer Vision*, pp. 11941–11952, 2023.

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

A Implementation and Training

This appendix specifies the native VLA interfaces, routing implementation, training settings, and efficiency measurements.

A.1 VLA-Specific Implementation

Our framework directly reuses the native data representations and intermediate hidden states produced by each pretrained VLA. No additional modality-specific encoder is introduced. Visual-language pretraining such as CLIP and SigLIP provides transferable representations (Radford et al., 2021; Zhai et al., 2023); the interfaces below refer to the representations available from each selected checkpoint. Since different VLA architectures expose different representations and input configurations, we adapt only the lightweight representation interface while keeping the utility estimation and routing formulation unchanged.

Table 6: Native data sources and source representations used by DataRouter across different VLA backbones. All representations are directly reused from the corresponding frozen VLA.

VLA Backbone	Available Data Sources	Source Representation
$\pi_{0.5}$	Multi-view RGB (base and wrist cameras), language instruction, proprioceptive state, and optional short visual history	Vision tokens from SigLIP, language tokens from Gemma, and projected proprioceptive/state representations
OpenVLA	RGB observation and language instruction; additional views or proprioception require a checkpoint-specific interface	Fused DINOv2 (Oquab et al., 2024) and SigLIP (Zhai et al., 2023) features, with Llama 2 (Touvron et al., 2023) language representations
SmolVLA	Multi-view RGB observations, language instruction, and sensorimotor/proprioceptive state	Native visual-language representations from SmolVLA and projected state features
AC^2-VLA	Current RGB observation, language instruction, and previous-action or action-context information	Multimodal input representations together with action-context features used by the original model
LightVLA	RGB observations and language instruction under an OpenVLA-style input interface	Visual representations after the model's native token-processing module together with language representations

Implementation notes. We use the original pretrained VLA weights without updating them. DataRouter reuses source representations and intermediate hidden states obtained within the same native VLA forward execution, then routes representations before action prediction continues. No second VLA forward pass is required for routing. Only the source projection, decision-state projection, and utility-estimation modules are trained. Tables 6 and 7 summarize the representation interfaces and routing positions; the original VLA architectures are described in their respective references.

Table 7: Decision-state extraction and routing positions used by DataRouter across different VLA backbones. The pretrained VLA parameters remain frozen in all cases.

VLA Backbone	Decision-State Representation	Routing Position
$\pi_{0.5}$	Intermediate residual-stream representation from the multimodal backbone, extracted before final action generation	At the interface between the multimodal backbone and the action-generation module
OpenVLA	Intermediate hidden representation from the language-model backbone before discrete action-token prediction	Immediately before the action-prediction stage
SmolVLA	Intermediate VLM representation produced before flow-matching action generation	At the interface between the VLM representation and the flow-matching action expert
AC^2-VLA	Action-context-conditioned intermediate representation produced before adaptive computation decisions	Before subsequent adaptive token/computation processing and action prediction
LightVLA	Intermediate language-model representation after the model's native visual-token processing stage	After native visual-token processing and before final action decoding

A.2 Routing Module Implementation Details

Source and decision-state pooling. Different VLA backbones expose source representations with different token lengths and hidden dimensions. Here Z_t^i denotes the original token-sequence representation z_t^i in the main text; pooling is used only to score sources, while action prediction retains the original tokens. For a token-sequence source $Z_t^i \in \mathbb{R}^{L_i \times d_i}$, we first apply layer normalization and mean pooling:

$$s_t^i = \frac{1}{L_i} \sum_{\ell=1}^{L_i} \mathrm{LN}(Z_{t,\ell}^i). \tag{11}$$

If a source is already represented by a single vector, we directly use that vector as s_t^i. The source representation is then projected into the routing space:

$$q_t^i = W_{\mathrm{src}} s_t^i + b_{\mathrm{src}} \in \mathbb{R}^{d_r}. \tag{12}$$

For the VLA decision state, given intermediate hidden states $\mathcal{H}_t \in \mathbb{R}^{L_t \times d_h}$, we use the same lightweight aggregation strategy:

$$h_t = \frac{1}{L_t} \sum_{\ell=1}^{L_t} \mathrm{LN}(\mathcal{H}_{t,\ell}), \qquad \tilde{c}_t = W_{\mathrm{state}} h_t + b_{\mathrm{state}} \in \mathbb{R}^{d_r}. \tag{13}$$

We use $d_r = 128$ for all backbones. The input dimension of W_{src} follows the native source interface; the notation suppresses interface indices. Source and state projections are trained with the utility module, without changing native VLA weights.

Utility estimator. We implement the learned utility matching function using a normalized bilinear interaction:

$$u_{t,i} = \bar{q}_t^{i\top} W_r \bar{c}_t, \qquad \bar{q}_t^i = \frac{q_t^i}{\|q_t^i\|_2}, \qquad \bar{c}_t = \frac{\tilde{c}_t}{\|\tilde{c}_t\|_2}, \tag{14}$$

where $W_r \in \mathbb{R}^{d_r \times d_r}$ is learnable. Utility normalization uses $\tau_u = 1.0$ at adaptive-budget inference, whereas the normalized training gates use $\tau_g = 0.5$. Threshold routing reuses τ_g in its sigmoid; there is no third temperature. A positive utility temperature changes distribution concentration, not source ranking. The cosine-utility ablation in Table 3 removes the learnable bilinear transformation by setting $W_r = I$.

Shared utility estimator across routing policies. For each evaluated VLA configuration, the utility estimator is trained once and reused by the alternative inference rules. The four routing policies therefore differ only in how the same learned utility scores are converted into hard source-selection masks at inference. The policy comparison therefore changes the inference rule without retraining the utility estimator. We do not use separate soft Top-K or cost-aware training surrogates in these experiments.

Training surrogate. The default DataRouter is trained using the Adaptive-Budget configuration. During training, we use normalized utility weights as the differentiable surrogate:

$$\hat{m}_{t,i} = \frac{\exp(u_{t,i}/\tau_g)}{\sum_{j=1}^{N_t} \exp(u_{t,j}/\tau_g)}, \tag{15}$$

and construct the differentiable routed representation as

$$Z_{t,\text{train}}^{\text{route}} = \left[\hat{m}_{t,1} z_t^1; \ldots; \hat{m}_{t,N_t} z_t^{N_t}\right]. \tag{16}$$

The resulting action loss is back-propagated through the frozen downstream VLA computation into the routing parameters. The coverage parameter ρ is used only when converting learned utilities into hard Adaptive-Budget routing decisions at inference.

Training minimizes only the action-supervision loss. Hard source selection is controlled by the inference policy.

Inference rules and empty-set handling. At inference, Top-K, Threshold, Adaptive-Budget, and Cost-Aware routing all operate on the same learned utility vector $\mathbf{u}_t$ using the hard rules defined in Appendix C.1. For Threshold and Cost-Aware Routing, if no source satisfies the selection criterion, we retain the source with the highest learned utility:

$$i^* = \arg\max_i u_{t,i}, \qquad m_{t,i^*} = 1. \tag{17}$$

This guarantees that at least one source representation is passed to subsequent action prediction.

A.3 Training and Hyperparameters

All pretrained VLA parameters are frozen throughout training. Only the data projection, decision-state projection, and utility-estimation parameters are optimized. One normalized-gate surrogate trains the utility estimator; the four hard policies reuse it at inference.

Table 8: Training and routing hyperparameters.

Hyperparameter	Value
Routing dimension d_r	128
Learning rate	1×10^{-4}
Batch size	64
Training steps	20,000
Utility temperature τ_u	1.0
Gate / threshold temperature τ_g	0.5
Adaptive coverage ρ (fixed before evaluation)	0.70
Threshold δ (default)	0.35
Top-K budget K (default)	2
Cost coefficient λ_{cost} (default)	0.20
Optimizer	AdamW (Loshchilov & Hutter, 2017)
Weight decay	1×10^{-4}

A.4 Efficiency Measurement Details

The $\pi_{0.5}$ efficiency measurements in Table 4 are conducted under the same hardware and software environment using a single NVIDIA A100 GPU. Unless otherwise specified, inference uses batch

size 1 with mixed-precision computation (FP16/BF16, depending on the native VLA implementation). For each method, we first run 100 warm-up iterations and then measure latency over 1,000 decision steps. We report the average wall-clock latency per decision step. CUDA synchronization is performed before and after timing to avoid asynchronous execution effects.

FLOPs are estimated for the complete forward computation of each evaluated method under the same input configuration and are normalized by the original VLA inference cost. The reported FLOPs include the native source encoding, DataRouter computation, and subsequent VLA action-prediction computation. Because DataRouter performs source selection only after the native source representations and decision state have been obtained, the source-encoding cost is retained in all measurements. The reduction in FLOPs therefore comes from omitting unselected source representations from subsequent VLA computation.

For fair comparison, all methods in this efficiency comparison use the same pretrained checkpoint, observation configuration, action horizon, image resolution, and number of camera views. We disable gradient computation and use evaluation mode for all latency measurements. No additional batching or caching is introduced unless it is part of the original baseline implementation.

B Benchmark Protocols and Complete Results

B.1 Benchmark Details

LIBERO-Plus. LIBERO-Plus (Fei et al., 2025) extends LIBERO (Liu et al., 2023) with perturbations to object layout, camera viewpoints, robot initial states, language instructions, lighting, background textures, and sensor noise. We evaluate all seven types at difficulty Levels 1–3 on Spatial, Object, Goal, and Long (LIBERO-10), each with ten base tasks. This setting assesses source utilization under these perturbations, rather than robustness at the most severe difficulty levels.

Each perturbed task is evaluated for five episodes using seeds $\{0, 1, 2, 3, 4\}$. Every method and backbone uses the same initial conditions and seed schedule. We first average the five binary rollout outcomes for each perturbed task, then macro-average tasks within each suite. The overall SR is the equally weighted mean of the four suite scores. Success uses the official task-specific oracle; maximum episode lengths follow suite defaults (approximately 220–520 steps).

Uncertainty-reporting protocol. The specified protocol for 95% confidence intervals is task-level bootstrap with 1,000 resamples. Resampling is performed within suites, retaining each sampled task's five rollout outcomes together and recomputing the equally weighted overall score. Percentile endpoints are the 2.5th and 97.5th percentiles. The result tables report point estimates; numerical intervals require task-level rollout outcomes and are not inferred from aggregate success rates.

CALVIN. CALVIN (Mees et al., 2022) is a long-horizon, language-conditioned manipulation benchmark built on a Franka Emika Panda arm in simulation. It contains four visually distinct environments (A, B, C, D) and 34 atomic skills that are composed into multi-step sequences. We adopt the standard zero-shot generalization split **ABC→D**: the policy is trained on environments A, B and C and evaluated exclusively in the unseen environment D.

Following the official long-horizon multi-task language control (LH-MTLC) protocol, each evaluation rollout consists of a chain of 5 consecutive language instructions. We evaluate 1,000 such sequences. A sequence is considered successful only if all five subtasks are completed in order; the primary metrics are (i) the success rate of completing at least k consecutive instructions ($k = 1 \dots 5$) and (ii) the average successful chain length (maximum 5). Each subtask is allotted a maximum of 360 control steps. Success of individual subtasks is judged by the official CALVIN task oracle.

B.2 Complete Results

We provide per-suite LIBERO-Plus results and updated per-chain CALVIN results to complement the aggregate comparisons in Section 4.2. Per-suite results for VLA-Pruner and SP-VLA are not included in the available manuscript.

LIBERO-Plus. Table 9 reports per-suite results for the evaluated LIBERO-Plus setting. **DataRouter improves performance consistently across LIBERO-Plus task suites.** The gains are observed not only on Spatial, Object, and Goal tasks, but also on the more challenging Long-Horizon split. For example, with $\pi_{0.5}$, DataRouter improves Long-Horizon SR from 70.8% to 77.1%, while with LightVLA it increases from 59.1% to 65.3%. Similar improvements are observed for SmolVLA and AC^2-VLA. In contrast, Random Routing consistently reduces performance across all task suites, supporting learned source selection over this random-routing baseline in the evaluated setting.

Table 9: Complete results on LIBERO-Plus across different VLA backbones and task suites. SR (%) is reported for Spatial, Object, Goal, and Long-Horizon tasks together with the reported average.

VLA	Method	Spatial	Object	Goal	Long	Avg.
$\pi_{0.5}$	Original VLA	89.2	91.5	86.1	70.8	84.4
	Uniform Fusion	88.0	90.3	84.7	69.4	83.1
	Attention Fusion	89.6	91.9	86.8	71.3	84.9
	Soft Gating	90.1	92.4	87.5	72.4	85.6
	Random Routing	84.5	86.8	81.2	64.3	79.2
	DataRouter	**92.8**	**94.6**	**90.3**	**77.1**	**88.7**
OpenVLA	Original VLA	19.2	13.9	15.1	14.2	15.6
	Uniform Fusion	18.6	13.4	14.5	13.1	14.9
	Attention Fusion	20.8	15.2	16.3	14.9	16.8
	Soft Gating	21.5	15.8	16.9	15.4	17.4
	Random Routing	15.2	11.0	12.1	10.1	12.1
	DataRouter	**25.6**	**19.8**	**20.7**	**19.1**	**21.3**
SmolVLA	Original VLA	68.4	71.2	63.5	48.1	62.8
	Uniform Fusion	66.9	69.8	62.0	46.9	61.4
	Attention Fusion	69.1	72.0	64.2	48.7	63.5
	Soft Gating	69.8	72.7	64.9	49.4	64.2
	Random Routing	62.5	65.3	58.1	43.3	57.3
	DataRouter	**73.6**	**76.5**	**69.2**	**54.3**	**68.4**
AC^2-VLA	Original VLA	76.8	79.5	72.1	56.8	71.3
	Uniform Fusion	75.5	78.2	70.8	55.9	70.1
	Attention Fusion	77.6	80.3	72.9	57.2	72.0
	Soft Gating	78.2	80.9	73.5	57.8	72.6
	Random Routing	72.1	74.8	67.4	52.9	66.8
	DataRouter	**81.4**	**84.2**	**77.6**	**62.8**	**76.5**
LightVLA	Original VLA	79.1	81.6	74.2	59.1	73.5
	Uniform Fusion	77.8	80.3	72.9	57.8	72.2
	Attention Fusion	79.8	82.3	74.9	59.4	74.1
	Soft Gating	80.4	82.9	75.6	60.3	74.8
	Random Routing	73.6	76.2	69.1	54.7	68.4
	DataRouter	**84.2**	**86.8**	**80.1**	**65.3**	**79.1**

CALVIN. Table 10 reports success rates at each chain length. For a common set of five-instruction chains, $\mathbb{E}[L] = \sum_{k=1}^{5} \Pr(L \geq k)$, connecting these rates to the mean lengths in Table 2.

Across the five VLA configurations, five-instruction success increases from 70.6% to 77.6% for $\pi_{0.5}$, from 29.0% to 41.0% for OpenVLA, from 45.0% to 58.0% for SmolVLA, from 54.0% to 66.5% for AC^2-VLA, and from 58.5% to 72.0% for LightVLA.

Table 10: Complete results on CALVIN under the ABC→D protocol. Columns 1–5 report success rates (%) for completing at least that many consecutive instructions. Avg. Len. is the mean completed chain length; RR is the source-routing ratio.

VLA	Method	1 ↑	2 ↑	3 ↑	4 ↑	5 ↑	Avg. Len. ↑	RR ↓
$\pi_{0.5}$	Original VLA	91.9	84.6	79.4	75.5	70.6	4.02	1.00
	Soft Gating	93.2	86.4	81.5	77.6	72.3	4.11	1.00
	Random Routing	87.4	78.2	71.5	65.8	61.1	3.64	0.61
	DataRouter	**94.2**	**88.9**	**85.4**	**81.9**	**77.6**	**4.28**	**0.46**
OpenVLA	Original VLA	72.0	55.0	42.0	33.0	29.0	2.31	1.00
	Soft Gating	74.5	58.0	45.5	36.5	32.5	2.47	1.00
	Random Routing	65.0	48.0	35.0	26.0	21.0	1.95	0.64
	DataRouter	**78.0**	**62.0**	**51.0**	**44.0**	**41.0**	**2.76**	**0.48**
SmolVLA	Original VLA	83.5	72.0	63.5	54.0	45.0	3.18	1.00
	Soft Gating	85.0	74.0	65.5	56.5	50.0	3.31	1.00
	Random Routing	78.5	66.0	56.5	47.0	39.0	2.87	0.60
	DataRouter	**87.5**	**78.0**	**70.5**	**63.0**	**58.0**	**3.57**	**0.44**
AC^2-VLA	Original VLA	86.0	76.5	68.5	60.0	54.0	3.45	1.00
	Soft Gating	87.5	78.0	70.5	62.5	57.5	3.56	1.00
	Random Routing	81.0	70.5	61.5	53.0	46.0	3.12	0.62
	DataRouter	**89.5**	**82.0**	**75.0**	**68.0**	**66.5**	**3.81**	**0.42**
LightVLA	Original VLA	88.0	79.5	72.0	64.0	58.5	3.62	1.00
	Soft Gating	89.5	81.0	74.0	66.5	63.0	3.74	1.00
	Random Routing	83.0	73.0	65.0	56.5	50.5	3.28	0.59
	DataRouter	**91.5**	**84.5**	**78.5**	**74.5**	**72.0**	**4.01**	**0.40**

C Alternative Routing Rules and Sensitivity

C.1 Utility-Aware Routing Policy Details

All four routing policies share one trained utility estimator within each evaluated VLA configuration. They differ only in their inference-time conversion of the same utility scores into source-selection masks. Adaptive-budget routing is the default; estimator weights are not shared across different VLA configurations.

Fixed-budget Top-K. The highest-scoring sources are retained, with the budget capped by the number of available sources:

$$m_{t,i} = \mathbb{I}[i \in \mathrm{TopK}(\mathbf{u}_t, \min(K, N_t))]\,. \tag{18}$$

Threshold routing. With gate temperature $\tau_g > 0$ and threshold $\delta \in (0, 1)$,

$$p_{t,i} = \sigma(u_{t,i}/\tau_g), \qquad m_{t,i} = \mathbb{I}[p_{t,i} > \delta]. \tag{19}$$

Adaptive-budget routing. Normalize the utilities and order them so that $w_{t,(1)} \geq \cdots \geq w_{t,(N_t)}$. The smallest prefix covering mass ρ determines the budget:

$$w_{t,i} = \frac{\exp(u_{t,i}/\tau_u)}{\sum_j \exp(u_{t,j}/\tau_u)}, \qquad K_t = \min\left\{ k \in \{1, \ldots, N_t\} : \sum_{j=1}^{k} w_{t,(j)} \geq \rho \right\}. \tag{20}$$

The mask retains the Top-K_t sources in their original source order.

Cost-aware routing. For relative source costs c_i, the hard mask solves

$$\mathbf{m}_t \in \operatorname*{arg\,max}_{\mathbf{m} \in \{0,1\}^{N_t}} \sum_i m_i(u_{t,i} - \lambda_{\mathrm{cost}} c_i). \tag{21}$$

The separable rule selects source i when $u_{t,i} > \lambda_{\text{cost}} c_i$. Costs refer to avoidable downstream processing; native representations used for routing have already been computed. For Threshold and Cost-Aware routing, an empty selection is replaced by the highest-utility source.

C.2 Policy Comparison

All rules use the same utility estimator for $\pi_{0.5}$ on LIBERO-Plus. Table 11 reports their operating points. Adaptive-Budget is the default because it adapts the source count without predefined source costs. Cost-Aware is an alternative when such costs are available.

Table 11: Comparison of utility-aware routing policies on LIBERO-Plus with $\pi_{0.5}$.

Routing Policy	SR ↑	RR ↓
Top-K (K=2)	87.1	0.50
Threshold (δ=0.35)	86.8	0.49
Adaptive-Budget	**88.7**	0.47
Cost-Aware	88.2	**0.44**

Figure 3 in the main text visualizes the performance–routing trade-off; the numerical sweeps below specify each operating point.

C.3 Parameter Sensitivity

We provide additional analysis of the four utility-aware routing policies under different routing budgets and hyperparameter settings. All policies share the same trained utility-estimator weights and differ only in their hard inference rules. Unless otherwise specified, experiments are conducted with $\pi_{0.5}$ on LIBERO-Plus.

Source grouping and shared evaluation samples. In the $\pi_{0.5}$ LIBERO-Plus policy sweeps, the routable inputs are grouped into Vision, Text, State, and History. Multiple camera views belong to the Vision group. All four groups are available at every evaluated decision step, so $N_t = 4$. All sweep settings use the same evaluation trajectories and decision-step sample set for routing statistics. Token validity masks are not included in source selection counts.

Fixed-Budget Top-K Routing. For $K \leq 4$, Top-K selects exactly K source groups and has RR $= K/4$. Table 12 shows SR increasing from 84.9% at $K = 1$ to 87.9% at $K = 3$. The $K = 4$ mask retains all sources and is the all-data endpoint, with SR 84.4%; it is not a distinct routing configuration. Thus, the best observed Top-K operating point retains three of the four encoded groups for subsequent prediction.

Table 12: Fixed-budget Top-K routing on LIBERO-Plus with $\pi_{0.5}$. Four source groups are always available, so #Sources $= K$ and RR $= K/4$. $K = 4$ and all-data inference share one endpoint.

K	SR ↑	RR ↓	#Sources ↓
1	84.9	0.25	1.00
2	87.1	0.50	2.00
3	87.9	0.75	3.00
4 (All)	84.4	1.00	4.00

Threshold Routing. We next vary the selection threshold δ, where a larger threshold produces more selective routing. Table 13 shows that increasing δ progressively reduces RR from 0.71 to 0.29, but also decreases SR from 88.0% to 84.6%. The result indicates that a single global threshold provides effective control over source utilization, but becomes sensitive when aggressive sparsity is required.

Table 13: Effect of the selection threshold δ for Threshold Routing on LIBERO-Plus with $\pi_{0.5}$.

δ	SR ↑	RR ↓
0.20	88.0	0.71
0.30	87.4	0.58
0.35	86.8	0.49
0.40	85.9	0.41
0.50	84.6	0.29

Adaptive-Budget Routing. Adaptive-Budget Routing dynamically determines the number of selected sources according to the cumulative utility coverage ρ. As shown in Table 14, SR increases as ρ grows from 0.50 to 0.70, reaching the best result of 88.7% at RR$= 0.47$. Further increasing ρ selects more sources but gradually reduces performance, with SR decreasing to 86.8% at $\rho = 0.95$. We fixed the default coverage to $\rho = 0.70$ before evaluation.

Table 14: Effect of utility coverage ρ for Adaptive-Budget Routing on LIBERO-Plus with $\pi_{0.5}$.

ρ	SR ↑	RR ↓	#Sources ↓
0.50	86.2	0.31	1.24
0.60	87.5	0.39	1.56
0.70	**88.7**	0.47	1.88
0.80	88.5	0.58	2.32
0.90	87.9	0.72	2.88
0.95	86.8	0.84	3.36

Cost-Aware Routing. Finally, we vary the cost coefficient λ_{cost} to study how explicitly penalizing computation affects the performance–cost trade-off. For the reported positive coefficients ($\lambda_{\text{cost}} = 0.10$–$0.40$), Table 15 shows decreasing RR and relative cost as the coefficient increases. A moderate cost penalty maintains high task performance: $\lambda_{\text{cost}} = 0.10$ achieves 88.5% SR with a relative cost of 0.62, while $\lambda_{\text{cost}} = 0.20$ further reduces the relative cost to 0.53 with an SR of 88.2%. More aggressive cost penalties continue to reduce computation but lead to larger performance degradation.

Table 15: Effect of the cost coefficient λ_{cost} for Cost-Aware Routing on LIBERO-Plus with $\pi_{0.5}$.

λ_{cost}	SR ↑	RR ↓	Relative Cost ↓
0.00	84.6	0.29	0.71
0.10	**88.5**	0.55	0.62
0.20	88.2	0.44	0.53
0.30	87.3	0.36	0.45
0.40	86.1	0.28	0.38

Summary. Overall, the four routing policies provide different operating characteristics. Top-K offers direct control over the number of retained sources, Threshold Routing controls sparsity through a global cutoff, Adaptive-Budget dynamically adjusts the routing budget based on utility coverage, and Cost-Aware Routing explicitly incorporates computational cost. The results further show that the highest performance generally occurs at an intermediate routing level rather than when all available sources are retained.

Table 16: Reported relative source-cost weights c_i (Vision $= 1.0$). These are relative weights for avoidable post-routing work, not measured end-to-end runtimes.

Data Source	Relative Cost c_i
Vision (multi-view RGB)	1.00
Text (language instruction)	0.25
State (proprioception)	0.15
History (short visual history)	0.80
Other (model-specific)	0.35

D Auxiliary Utility Diagnostics

We provide additional analyses of the learned utility estimates across VLA backbones, tasks, and data sources.

D.1 Trajectory Illustration

Figure 4 in Section 4.5 shows the learned scores and resulting masks along one $\pi_{0.5}$ trajectory. Since masks are derived from these scores, their agreement is expected. The leave-one-out diagnostic below provides a separate comparison with empirical source sensitivity.

D.2 Leave-One-Out Source Sensitivity

For evaluation only, we measure the sensitivity of the VLA to each available data source using leave-one-out removal. Given the full set of available data $\mathcal{D}_t$, the sensitivity of source d_t^i is defined as

$$\Delta_{t,i}^{\mathrm{LOO}} = \mathcal{L}_{\mathrm{BC}}\left(\mathcal{D}_t \setminus \{d_t^i\}\right) - \mathcal{L}_{\mathrm{BC}}\left(\mathcal{D}_t\right). \tag{22}$$

A larger $\Delta_{t,i}^{\mathrm{LOO}}$ indicates that removing the corresponding source causes a larger increase in action-prediction loss. We use this quantity only as an empirical sensitivity measure for evaluation and never as supervision during training. Importantly, leave-one-out sensitivity should not be interpreted as the exact contribution of an individual source, since heterogeneous sources may contain redundant or complementary information.

Learned utility is more consistent with empirical source sensitivity than alternative relevance signals. As shown in Table 5, Learned Utility achieves the highest Spearman correlation for all five VLA backbones, with an average correlation of 0.65. In comparison, feature similarity, attention score, and decision-state similarity obtain average correlations of 0.30, 0.41, and 0.47, respectively. The improvement over decision-state similarity indicates that directly comparing source and decision-state representations is informative but insufficient; the learned matching function provides a stronger correspondence with the observed sensitivity to source removal.

The correlation trend is consistent across heterogeneous VLA backbones. Learned Utility achieves correlations ranging from 0.61 to 0.69 across the five models, whereas none of the alternative relevance signals exceeds 0.51. These results provide independent supporting evidence for the learned relevance scores in the evaluated configurations. Leave-one-out sensitivity is a diagnostic, not ground-truth causal contribution; redundancy and complementarity can affect its interpretation.