Isaac 0.5: Percepts Scale Control

Perceptron

Abstract

We introduce Isaac 0.5, a 36B-parameter open-weight embodied foundation model that unifies video perception, embodied visual reasoning, and robot control in one sparse backbone. Every output reads from one shared representation: the model emits language and grounding outputs, discrete FAST action tokens, and continuous actions through a dedicated Flow expert conditioned on backbone states. We also introduce null-expert routing, which lets each token use 0 to 8 of the model's 256 routed experts as needed.

The scale and composition of its cotraining distinguish Isaac 0.5. Teleoperation provides direct control supervision, but every accepted hour occupies a robot, an operator, and task-specific infrastructure. Video is cheaper and more abundant but lacks robot actions and proprioception. To learn from it, Isaac 0.5 uses a proprietary self-supervised video objective that predicts the semantic content of future observations from preceding frames. We call the predicted task-relevant visible states and changes *percepts*. Its targets are constructed automatically from video, without human annotation or action labels; the target construction and loss implementation remain proprietary. Isaac 0.5 trains future-percept prediction, video perception, embodied visual reasoning, and control jointly. Video and autoregressive action supervision update the shared representation, while continuous actions train a Flow expert conditioned on its states. To our knowledge, this is the first study to scale general action-free video to one million hours alongside egocentric, handheld-gripper (UMI), and robot data.

Increasing general video from 1,000 to one million hours at a fixed 80:30:30 ratio of general, egocentric, and UMI video moves the within-grid teleoperation crossing for a well-calibrated offline action-loss threshold from 5,884 hours to 28, a 210.3× reduction. The adjacent measured teleoperation rungs bound this ratio at 83 to 300×. Additional video reduces action loss more when paired with more teleoperation data. We release the weights, inference and adaptation code, evaluation settings, and robot-specific deployment configurations for more than 35 embodiments represented in training.

1 Introduction

Isaac 0.5 is an open-weight embodied foundation model built on a single 36B-parameter sparse backbone. The model emits language, grounding, and FAST-tokenized discrete actions directly from its hidden states, and continuous actions through a dedicated Flow expert and diffusion transformer (DiT) conditioned on those same states. Perception, embodied reasoning, and control all train through that shared representation rather than through a policy hanging off a frozen encoder, so structure learned from visual data can reach the control outputs.

Our central question is whether inexpensive, action-free video can reduce reliance on teleoperated robot data. General video samples the visual world broadly but usually carries no explicit robot actions, proprioception, or dynamics. We test it as a primary scaling axis alongside egocentric, handheld-gripper (UMI), and robot data, scaling it to one million hours.

We learn from that video with a proprietary self-supervised objective that predicts the semantic content of future observations from preceding frames. We call a predicted task-relevant visible state or change a *percept*: an object is now grasped, a drawer is open, or a contact is about to occur. The term describes what the objective predicts, not how its targets are constructed. The targets come

from future video observations without human annotation or action labels; their construction and the loss implementation remain proprietary. General video takes the largest share of the *video* mixture because these percepts appear beyond robot data. Since every interface reads from the shared representation, the structure learned from video is available to the action interfaces.

We hold the general:egocentric:UMI video composition at 80:30:30 and scale all three streams against teleoperation. At the primary offline action-loss threshold, increasing general video from 1,000 to one million hours moves the within-grid teleoperation crossing from 5,884 to 28 hours, a factor of 210.3×. The adjacent measured teleoperation rungs bound this ratio at 83× to 300×. The exchange is conditional, not uniform: the empirical loss-versus-video slope is nearly flat at one teleoperation hour and stabilizes near -0.21 loss for each 10× increase in video from roughly 100 teleoperation hours onward.

The paper makes four contributions:

- **A unified, open-weight, multi-embodiment foundation model spanning perception, embodied reasoning, and control.** We introduce Isaac 0.5 and release its weights, inference and adaptation stack, evaluation suites and settings, and robot-specific deployment configurations for 35+ embodiments represented in training.
- **An embodied-first sparse architecture.** We extend zero-compute null-expert routing (Kilian et al., 2026) through the shared backbone that serves perception, embodied reasoning, and control, so the text, image, state, and action tokens of one embodied sequence draw different amounts of routed work. We complement the architecture with an interpretability study of learned expert allocation, showing where adaptive routed computation is most useful across tasks and phases.
- **New data scaling laws for embodied foundation models.** We establish how action-free video and teleoperation scale together. Within the measured range, scaling the fixed-ratio general, egocentric, and UMI video mixture reduces the teleoperation required to reach a well-calibrated offline action-loss threshold ($\tau = 2.50$) by a factor of 210.3×, with an endpoint bracket of 83× to 300×. Per-rung slopes show that video becomes more effective once the recipe includes sufficient action grounding.
- **A stall-resistant, high-MFU system for million-hour video training.** Predictive shard streaming, node-shared downloads, process-isolated decoding, best-fit multimodal packing, vision-token balancing, and compacted sparse execution keep the million-hour mixture flowing without optimizer stalls. End-to-end model FLOPs utilization (MFU) reaches 24% in the released hyper-sparse setting, where null routing leaves 2.5B of 36B active per token, and 48% in a dense setting whose recipe and weights fall outside this release (Chowdhery et al., 2023).

2 Data

The data recipe has to solve two different coverage problems. The first is visual: the model must recognize objects, geometry, contact, state change, and task progress across far more settings than can be recreated around a robot. The second is operational: it must connect those observations to actions expressed through a particular control interface. No single source provides both kinds of coverage at scale. General video is broad but action-free. Egocentric recordings bring the camera close to hands and contact, while UMI-style handheld grippers add device motion and timing without requiring a deployed robot (Grauman et al., 2022; Chi et al., 2024). Robot trajectories provide the strongest action grounding, but they are tied to an embodiment and are costly to collect.

We therefore treat the sources as complementary supervision rather than interchangeable examples. Broad visual data teaches recurring states and changes; interaction-centered video narrows the gap between observing an event and understanding how it unfolds; robot data supplies the state and action interface needed for control. The shared representation and typed training path in Section 3 let these sources train together without pretending that an absent action field is a negative action label. Section 4 later isolates one controlled tradeoff within this recipe by varying the linked general, egocentric, and UMI video mixture against teleoperation. The present section describes the larger 529-stream mixture, not just that experimental grid.

2.1 Mixture composition and exposure

Our mixture has 529 source streams. Direct draw mass is split between two top-level components: perception and embodied reasoning, and robotics-oriented data. The perception component spans general video, visual question answering, spatial grounding, documents, captioning, and other instruction data. The robotics component spans high-quality teleoperation, broader robot interaction, and interactive or simulated experience. This division is an accounting boundary, not a claim that perception and control are separate in the model. Both components update the shared training system described in Section 3.

Mixture size can be reported in several valid but non-equivalent units. Let q_s be the probability mass assigned to source stream s, and let N_s be the number of packed tokens attributed to that stream in the accepted recipe ledger. For category c,

$$q_c = \sum_{s \in c} q_s, \qquad e_c = \frac{\sum_{s \in c} N_s}{\sum_j N_j}. \tag{1}$$

The first quantity is *scheduler mass*: how often a component is selected before examples are constructed and packed. The second is *packed-token exposure*: the fraction of realized model positions contributed after that construction. Source-hours and native tokens answer different questions. Source-hours measure temporal coverage in recorded data, while native tokens describe the scale of an already-tokenized component.

The distinction is substantial in this recipe. Perception and embodied reasoning receive 69.7% of direct scheduler mass, while robotics receives 30.3%. Robotics examples expand into more packed positions per draw, so the audited exposure reverses the apparent balance: 79.6% of packed tokens come from the robotics component and 20.4% from perception and embodied reasoning. Reporting only the sampler weights would therefore understate how much of the optimizer's realized input is action-centered; reporting only packed exposure would hide the scheduler policy that produced it.

We use SkillRater (Sahi et al., 2026) during data-mixture discovery. Its capability-specific raters surface data for distinct target skills rather than compressing every sample into one global quality score. This matters for a mixture that asks for different forms of competence: a useful grounding example need not resemble a useful temporal-reasoning example, and neither is selected by the same evidence as a robot trajectory. SkillRater helps identify candidate data for these capabilities. It does not set the numbers in Figure 1; the reported masses are the final scheduler configuration, not rater scores.

Figure 1 partitions the same direct scheduler mass by modality and primary supervision type. Because one stream can carry several task labels, a chart that counted every label would double-count the mixture. We instead assign each stream once, in this order: robotics component membership; video modality; pointing; question answering or counting; OCR or structured extraction; captioning; and other instruction data. The non-video perception categories preserve their relative weights within the remaining scheduler mass.

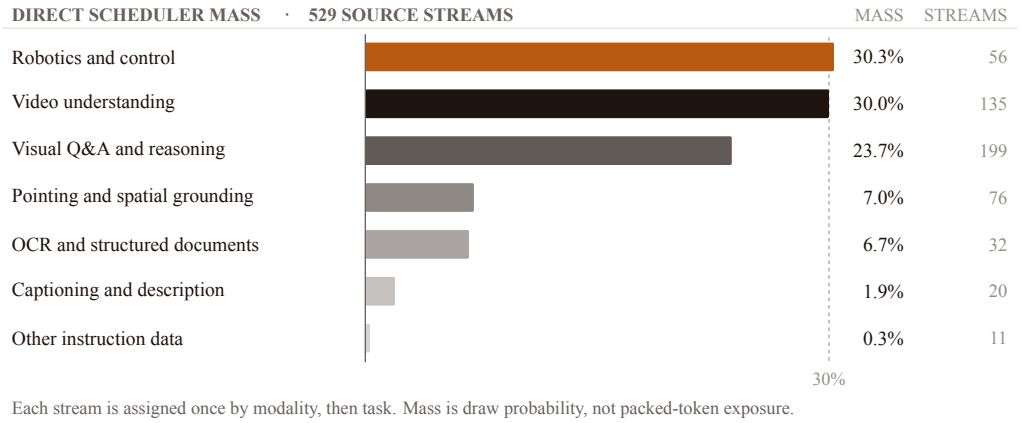


Figure 1: Video and robotics each comprise about 30% of the direct data mixture. Video understanding receives 30.0% and robotics and control receive 30.3% of scheduler mass; visual Q&A and reasoning contribute 23.7%. The 529 source streams are assigned once using the modality-then-task priority described in the text. Percentages report direct scheduler mass, not packed-token exposure.

The resulting view is deliberately more specific than the two-component split. Video understanding accounts for 30.0% of scheduler mass across 135 streams, nearly matching the 30.3% assigned to 56 robotics and control streams. Visual question answering and reasoning form the next largest category at 23.7% across 199 streams. Pointing and spatial grounding contribute 7.0% across 76 streams; OCR and structured documents contribute 6.7% across 32; captioning and description contribute 1.9% across 20; and the remaining 11 instruction streams account for 0.3%. The counts show why mass and catalog breadth are both useful. Robotics is concentrated in fewer, heavier streams, while question answering and grounding spread their mass across a wider source inventory.

Each category supports a different part of the embodied interface. Video supplies temporal context and visible state change. Question answering and reasoning attach language to what is present and what happens. Pointing trains explicit spatial outputs, while document and OCR data preserve competence on text embedded in the visual world. Robotics connects observations to embodiment-specific state and action. These roles overlap in individual examples, which is why the chart uses a declared priority rule rather than presenting the categories as disjoint scientific concepts.

Table 1: Packing shifts realized exposure toward robotics-oriented data. Component totals and child-level values report the final audited data mixture.

Source block	Streams	Scheduler mass	Packed-token exposure	Exposure / draw	Native scale and training role
Perception and embodied reasoning	**473**	**69.7%**	**20.4%**	**0.29×**	3T tokens; image, video, and instruction supervision
Video understanding	135	30.0%	8.0%	0.27×	Future-percept prediction and video understanding
General action-free video	105	17.1%	4.1%	0.24×	1,000,000 h; broad visual experience
Egocentric video	18	6.4%	2.0%	0.31×	375,000 h; first-person manipulation
UMI handheld-gripper video	12	6.5%	1.9%	0.29×	375,000 h; device motion and timing
Visual Q&A and reasoning	199	23.7%	7.5%	0.32×	Multimodal Q&A and counting
Pointing and spatial grounding	76	7.0%	2.2%	0.31×	Localization and coordinate supervision
OCR and structured documents	32	6.7%	1.9%	0.28×	Text recognition and schema extraction
Captioning and description	20	1.9%	0.7%	0.37×	Image and scene description
Other instruction data	11	0.3%	0.1%	0.33×	Remaining instruction-following streams
Robotics-oriented data	**56**	**30.3%**	**79.6%**	**2.63×**	100,000 h; 35+ embodiment configurations
High-quality teleoperation	18	10.0%	32.0%	3.20×	10,000 h; curated demonstrations
Broad robot interaction	30	14.0%	35.0%	2.50×	40,000 h; heterogeneous trajectories
Video games and simulation	8	6.3%	12.6%	2.00×	50,000 h; interactive and simulated episodes

Table 1 adds native scale behind those percentages. The perception and embodied-reasoning component contains 3T native tokens. The robotics component contains 100,000 source-hours across 35+ embodiment configurations: 10,000 hours of high-quality demonstrations, 40,000 hours of broader robot trajectories, and 50,000 hours from games and simulation.

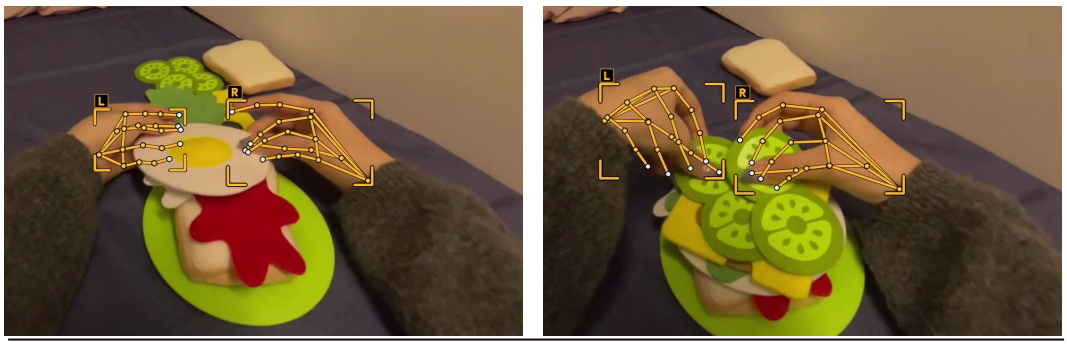
2.2 Egocentric annotation

Egocentric video occupies a useful middle ground between watching the world and commanding a robot. The recording is still action-free in the robot-control sense, but the first-person view makes hands, contact, and local task progress much more legible than they are in general video. We turn that visual evidence into structured supervision at three nested levels. A recording provides the full task context. Positive activity segments identify intervals with visible task progress. Within those intervals, frame-level instances localize the left and right hands using boxes and anatomical keypoints, while time-aligned labels describe each hand's visible action, manipulated object, visibility, contact, and annotation confidence.

The hierarchy matters because manipulation labels are not naturally one label per frame. A task can persist across several seconds, two hands can play different roles at the same instant, and a hand can remain important while briefly stationary. We preserve the source timestamp and native pixel coordinates for every supported field, keeping task progression, hand geometry, and hand-object interaction aligned to the underlying video. A positive activity segment says that visible progress occurs in that interval. It does not turn every unsupported hand field, or every frame outside the interval, into a negative action example. When visual evidence is insufficient, the field is omitted.

We generate these annotations with our proprietary Perceptron Mk1 model. We selected it for the production annotation pass because its measured cost and throughput were better suited to this scale

than the Gemini configuration we evaluated. This was an operational choice for constructing the stream, not an evaluation result about the trained Isaac 0.5 checkpoint. Figure 2 shows how the resulting labels retain the asymmetry of a bimanual task.



	Visible action	Action	Object	Contact	Visibility
00:08, "add the fried egg layer", subtask 00:06–00:09; wrist z 0.30 m / 0.29 m					
L	Lifting and lowering the fried egg onto the sandwich	put	fried egg	yes	visible
R	Guiding the edge of the fried egg into position	hold	fried egg	yes	visible
00:20, "arrange the cucumber slices on the cheese", subtask 00:17–00:21; wrist z 0.24 m / 0.24 m					
L	Picking up and arranging cucumber slices	put	cucumber slices	yes	visible
R	Spacing the cucumber slices across the cheese	put	cucumber slices	yes	visible

Figure 2: Egocentric supervision combines raw first-person video with time-aligned task and hand–object labels. Two frames from a sandwich sequence at 8 and 20 seconds, cropped to the annotated region; the keypoint and box overlays are the annotation tool's own, with no paper-side graphics added. The table gives the time-aligned labels carried by the two frames, one row per hand: the L and R markers in each frame identify the hands the rows describe. Confidence is recorded per label and omitted where the visual evidence does not support the field.

At 8 seconds, both hands contact the fried egg, but their roles differ: the left hand lowers it onto the sandwich while the right hand guides and holds its edge. At 20 seconds, both hands are labeled put, yet their descriptions still distinguish picking and arranging from spacing the slices. Collapsing either frame to one action label would lose this division of labor. The per-hand record keeps it visible while remaining tied to the shared task segment and source time.

2.3 Idle-span filtering

Source-hours are only useful if they retain their connection to training signal. In the open-source robotics datasets we process, a recorded trajectory can contain long periods of waiting, camera-only motion, or repeated frames. Counting those periods at face value inflates temporal coverage and spends decode, packing, and optimizer capacity on little additional task supervision.

We compact sustained idle runs only for robot sources with a validated idle signal. The rule removes the interior of a sufficiently long idle interval while retaining short pauses and the transition boundaries around it. Sources without a validated signal pass through unchanged. We do not equate visual stillness with idleness: a held pose, mechanical settling, or contact-rich motion can remain informative even when the end effector moves very little. Validation therefore includes these cases and is performed per source rather than through one global motion threshold.

Finally, the data record keeps recorded robot hours separate from optimizer-consumed robot hours. The former describes what was captured; the latter describes what remains after source-specific compaction and reaches training. This makes the 100,000-hour source inventory auditable without claiming that every recorded second receives equal optimization weight. Section 6.3 gives the corresponding implementation details in the streaming system.

3 Model and training

Isaac 0.5 is built so that video supervision changes the representation that generates actions. Semantic and autoregressive token objectives update the shared backbone, while the continuous-control objective trains a Flow expert conditioned on that representation. The scaling measurement in Section 4 depends on this connection: without a shared representation there is no route by which video hours could displace teleoperation hours. The released checkpoint is trained at the highest video budget in that measurement, one million general-video hours. We describe the architecture and its action interfaces first, then the training objectives, the trajectory compiler that feeds them, and the recipe used for the released checkpoint.

3.1 Shared architecture

Isaac 0.5 builds on an open Qwen-family vision-language recipe. Semantic and autoregressive token objectives train a shared sparse backbone, while a dedicated Flow/DiT route produces continuous actions from its states.

Visual inputs use patch size 16 and project into a 2,048-wide trunk of 40 sparse blocks, 30 with GDN linear attention and 10 with full attention (Yang et al., 2025b). Every block has 256 real routed MLPs, one learned null router row expanded into 256 null copies, and an always-on shared expert.

The 36B total covers every parameter in the checkpoint: the visual encoder, the 40 backbone blocks with their attention, router rows, 256 routed MLPs and always-on shared expert, and the continuous route's Flow expert and DiT. Active parameters per token are what one token actually executes, 2.5B at the 50% null reference where 4 of its 8 routes are real, and approximately 0.1B–3B as that number varies between 0 and 8. Weight memory is a third quantity: it holds all 256 experts whether a token executes them or not, while the visual encoder, attention, shared expert, and continuous-control path run regardless of how many routes are real.

3.2 Null experts

Standard mixture-of-experts layers route every token through a fixed number of feed-forward experts. Null experts add learned no-op choices to the router: when a null expert occupies a top-k slot, the layer skips the corresponding routed MLP. The number of real experts executed can therefore vary by token while all 256 experts and the maximum per-token budget remain available (Kilian et al., 2026). Embodied sequences interleave text, image, state, and action tokens that need not draw the same amount of routed work.

How variable routed work arises. The router has 257 learned rows: one for each of the 256 real experts and one for the null option. For each token, the null row produces one score $u_\varnothing$, which is duplicated into 256 null copies copies and concatenated with the 256 real-expert scores. This gives 512 candidate scores for top-8 selection and partitions each token's budget (Kilian et al., 2026): with j the number of real-expert scores strictly above $u_\varnothing$, the block selects

$$k_{\text{real}} = \min(8, j), \qquad k_{\text{null}} = 8 - k_{\text{real}}, \tag{2}$$

ignoring exact ties, which are broken deterministically by index. The shared null score is a learned threshold on real-expert scores, so k_{real} varies token by token between 0 and 8 while the expert library stays at full size. When $k_{\text{real}} > 0$, gate weights are renormalized over the selected real routes after the null routes are dropped; when $k_{\text{real}} = 0$, the routed contribution is zero. The shared expert and the residual path are unaffected either way.

Why 256 null copies rather than 8. For top-k selection alone the multiplicity barely matters: any number of null copies at or above 8 gives the same k_{real} in Equation 2, so 8 copies would reproduce the mechanism. Multiplicity matters through the router distribution. In a softmax over all 512 scores, 256 identical null logits contribute $256\, e^{u_\varnothing}$ to the partition function, equivalent to one null score with logit $u_\varnothing + \log 256$. Multiplicity therefore sets the null path's initial probability mass and shifts router gradients and load-balancing statistics without adding parameters.

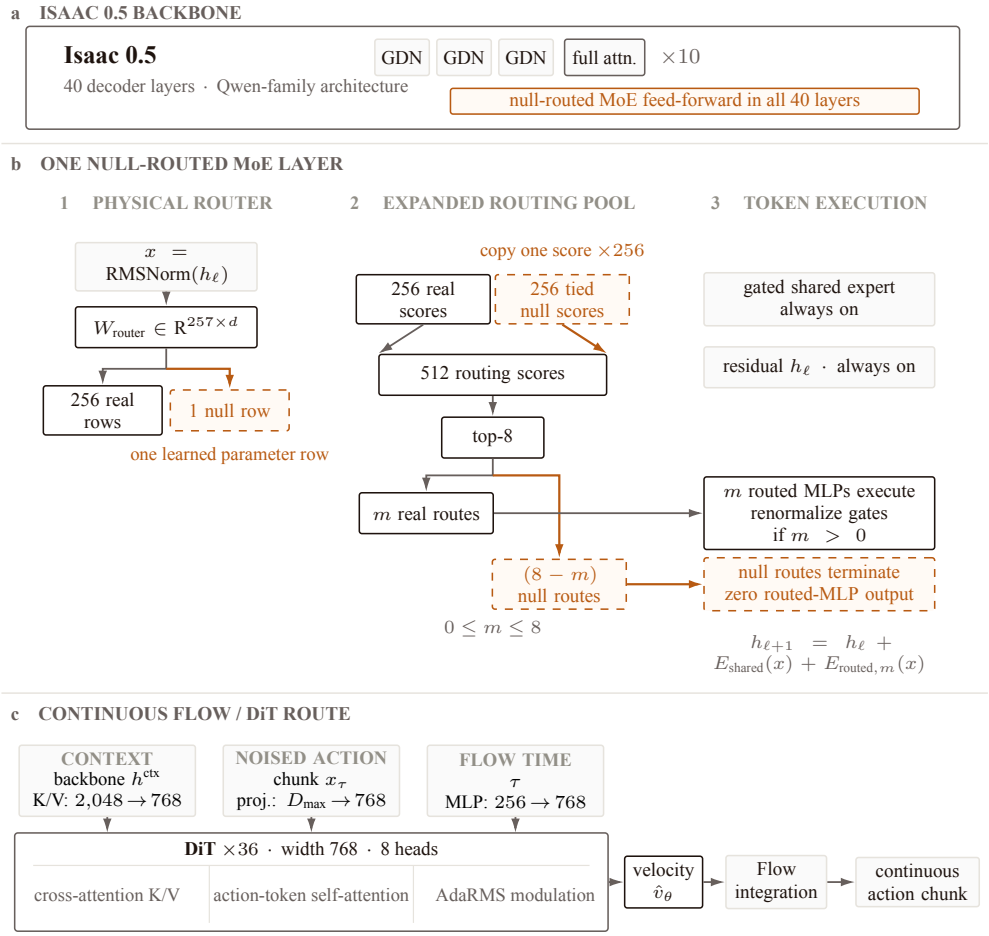
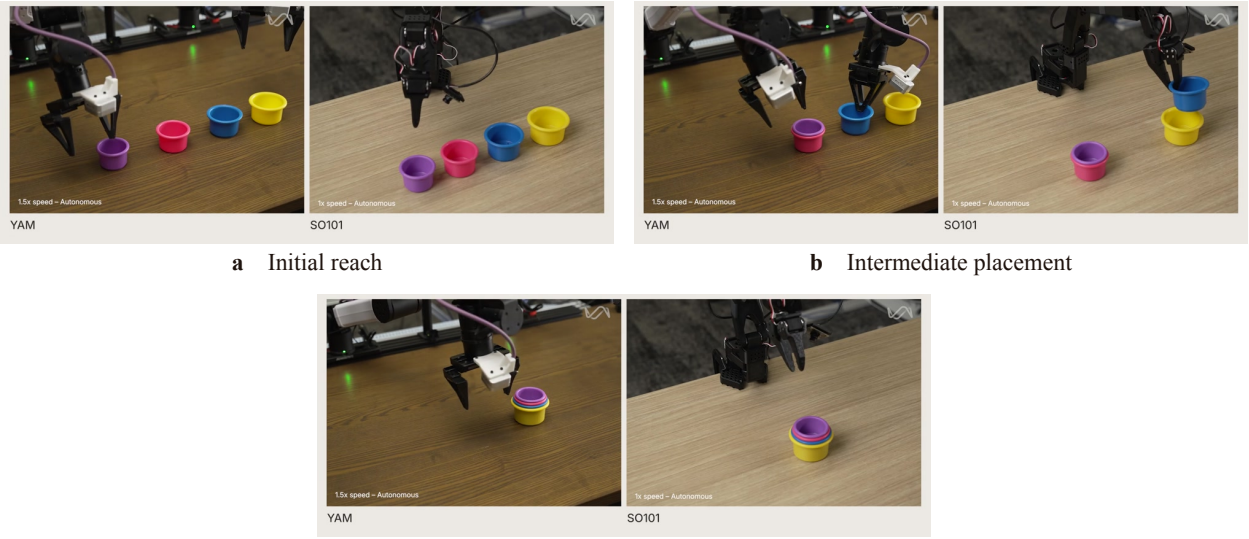


Figure 3: Token-adaptive null routing and the continuous-control DiT. Every backbone layer uses the same sparse feed-forward block. Selecting a null route skips the corresponding routed MLP, so each token executes a variable number of experts. The continuous-control DiT cross-attends to masked pre-action backbone context and predicts a velocity field from the noised action chunk and Flow time; integrating that field yields the action chunk.

3.3 Shared action interfaces

The autoregressive route uses the same hidden states for language, coordinates, grounding, embodied reasoning, and a disjoint 2,048-token FAST action vocabulary (Pertsch et al., 2025). Section 3.5 defines its normalization, tokenization, and training loss. The continuous route projects backbone states from 2,048 to 768 dimensions as cross-attention keys and values for a 36-block DiT (Peebles & Xie, 2022; Fang et al., 2026). The DiT combines that context with a noised action chunk and Flow time, then predicts the velocity used to recover continuous actions (Lipman et al., 2023; Black et al., 2024). Training draws $S = 4$ independent noise and Flow-time samples per action chunk; Section 6.6 describes how the system evaluates them together without replicating the long backbone context. Figure 3 shows both routes: the shared sparse block with its null-expert router, and the continuous-control DiT that cross-attends to backbone context.

The released checkpoint is trained on data from 35+ robot interface configurations—31 of which are real robot hardware or a hardware family, with the remainder comprising simulated settings, a capture rig, a gamepad layout, and one aggregated multi-platform dataset—with robot-specific observation, action, normalization, timing, and safety configurations included for each. Figure 4 shows the same checkpoint execute the same requested cup stack on YAM and SO-101 by changing only the embodiment string in the configuration prompt. Section 7 defines the adaptation regimes, and Appendix C gives the platform-by-platform configuration inventory.



a Initial reach **b** Intermediate placement

c Completed stack

Figure 4: One checkpoint executes the same task on two physical embodiments by changing only the embodiment string in the configuration prompt. a–c, paired YAM and SO-101 rollouts for the same requested cup stack at the initial reach, an intermediate placement, and task completion. The source video displays YAM at $1.5\times$ speed and SO-101 at $1\times$ and aligns the two rollouts by task phase.

3.4 Self-supervised future-percept prediction

We use future-percept prediction to refer to a proprietary self-supervised video objective. Given an observation history, it predicts semantic information derived from a future observation rather than reconstructing future pixels. The objective requires no human annotations or action labels. We call the predicted task-relevant visible state or change a percept. This may be an object state, spatial relation, affordance, task phase, visible contact change, or likely near-future task state. The term describes the prediction target but not how that target is constructed.

At the disclosed boundary,

$$\mathcal{L}_{\text{sem}} = \mathbb{E}_{(o_{\le t}, z_{t+\Delta})} \left[\ell(g_\theta(o_{\le t}), z_{t+\Delta}) \right], \tag{3}$$

where $o_{\le t}$ is the observation history through time t, $\Delta > 0$ the prediction offset, $z_{t+\Delta}$ the future-percept target, g_θ the semantic predictor, and ℓ its loss. We do not disclose the generator that produces z or the implementation of ℓ. The joint objective below includes $\mathcal{L}_{\text{sem}}$ explicitly; any internal weighting of its targets is part of the undisclosed ℓ.

3.5 Discrete and continuous control

Continuous-feature normalization. Each policy-state dataset, normalization scope, and objective has its own action and proprioception statistics. For a scalar feature x_d, the compiler maps the first percentile to -1 and the ninety-ninth percentile to approximately $+1$:

$$\widetilde{x}_d = 2\frac{x_d - q_{0.01,d}}{q_{0.99,d} - q_{0.01,d} + 10^{-6}} - 1. \tag{4}$$

If the two quantiles coincide for a sparse binary or contact channel but its observed minimum and maximum differ, those extrema replace the collapsed bounds for that dimension. For example, this occurs when a gripper is closed throughout nearly all of a dataset but opens in a small number of frames. Proprioceptive features are not clipped after normalization, so valid tails can remain outside $[-1, 1]$. For FAST and Flow action targets, the compiler drops a chunk when $\max_{t,d} |\widetilde{a}_{t,d}| > 20$; otherwise it clips the normalized actions to $[-10, 10]$. Serving applies the same clipping bound before mapping predicted actions back to robot units.

For a normalized action chunk a, FAST produces a token sequence $q = \mathrm{FAST}(a)$ and trains it with the same autoregressive objective as the model's other discrete outputs (Pertsch et al., 2025):

$$\mathcal{L}_{\mathrm{FAST}} = -\sum_{j} \log p_\theta(q_j \mid c, q_{<j}) \,, \tag{5}$$

where c is the shared-backbone context. The tokens are decoded and unnormalized into robot actions at inference.

The continuous route uses linear conditional Flow Matching in the clean-at-one convention (Lipman et al., 2023; Fang et al., 2026). For each of $S = 4$ independent draws per action chunk,

$$\epsilon_s \sim \mathcal{N}(0, I), \qquad z_s \sim \mathrm{Beta}(1.5, 1), \qquad \tau_s = 0.001 + 0.999 z_s, \qquad x_s = (1-\tau_s)a + \tau_s \epsilon_s. \tag{6}$$

Here τ_s is the noise level and the DiT receives Flow time $t_s = 1 - \tau_s$. The straight path has constant target velocity $a - \epsilon_s$, giving

$$\mathcal{L}_{\mathrm{Flow}} = \frac{1}{S} \sum_{s=1}^{S} \frac{\| M \odot [\hat{v}_\theta(x_s, t_s; c) - (a - \epsilon_s)] \|_2^2}{\|M\|_1}, \tag{7}$$

where M masks padded time steps and unused action dimensions. This is the time-reversed, sign-flipped form of the clean-at-zero convention used by π_0 (Black et al., 2024). At inference, we start from Gaussian noise and take $N = 10$ explicit Euler steps,

$$x^{(0)} \sim \mathcal{N}(0, I), \qquad x^{(k+1)} = x^{(k)} + \frac{1}{N} \hat{v}_\theta\left(x^{(k)}, \frac{k}{N}; c\right), \quad k = 0, \ldots, N-1, \tag{8}$$

then unnormalize $x^{(N)}$ into continuous robot actions. Because the Flow expert is conditioned on the shared backbone, perception and reasoning losses can change the context used for control while the FAST and Flow decoders remain distinct.

3.6 Joint training objective

The complete objective has three supervised parts, an LM softmax regularizer, and two router auxiliaries. Let Ω be the eligible autoregressive target positions in a packed batch. It includes perception and reasoning responses, coordinates and control tokens, and FAST action tokens; prompt and padded positions are excluded. The token-normalized next-token loss and its softmax z-loss are

$$\mathcal{L}_{\mathrm{NTP}} = -\frac{1}{|\Omega|} \sum_{i \in \Omega} \log p_\theta(y_i \mid y_{<i}, c), \qquad \mathcal{L}_z^{\mathrm{LM}} = \frac{1}{|\Omega|} \sum_{i \in \Omega} \left(\log \sum_v e^{s_{i,v}} \right)^2, \tag{9}$$

where $s_{i,v}$ is the vocabulary logit for token v. Equation 5 is the FAST subset of $\mathcal{L}_{\mathrm{NTP}}$, not another term added to it. The future-percept term $\mathcal{L}_{\mathrm{sem}}$ remains separate because its target construction and loss are defined at the boundary in Equation 3.

The Flow term is independently normalized by action chunk. Combining the supervised objectives with the sparse-router regularizers gives the scalar used for backpropagation:

$$\begin{aligned} \mathcal{L}_{\mathrm{train}} = {} & \mathcal{L}_{\mathrm{NTP}} + \mathcal{L}_{\mathrm{sem}} + 10^{-4}\, \mathcal{L}_z^{\mathrm{LM}} + \mathcal{L}_{\mathrm{Flow}} \\ & + 0.02\, \mathcal{L}_{\text{MoE-LB}} + 10^{-3}\, \mathcal{L}_z^{\mathrm{router}}. \end{aligned} \tag{10}$$

Here $\mathcal{L}_{\text{MoE-LB}}$ is the mean globally counted load-balancing loss over the sparse backbone layers, and $\mathcal{L}_z^{\mathrm{router}}$ is the mean squared router log-partition over those layers. The NTP objective is normalized by its valid-token count, while Flow is normalized by valid action chunks. The retained recipe uses no weight-decay loss. Unlike the knowledge-insulated $\pi_{0.5}$+KI recipe (Driess et al., 2025), it does not detach the backbone activations; Section 3.9 gives the retained optimization settings. Consequently, $\mathcal{L}_{\mathrm{Flow}}$ updates both the Flow path and the shared backbone, while the semantic, token, and router terms also update the backbone.

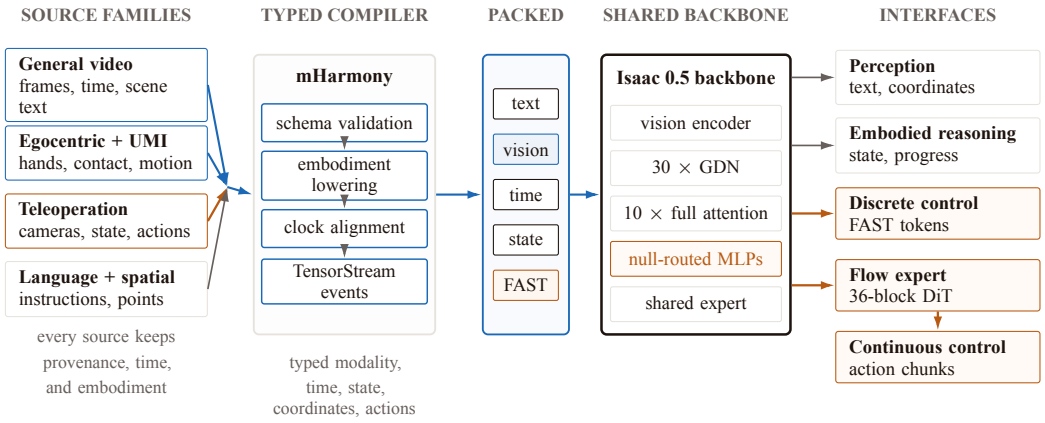


Figure 5: From heterogeneous experience to one model. mHarmony validates and lowers source schemas into typed TensorStream events before packing. The visual encoder and sparse backbone are shared; the Flow expert and DiT are interface-specific.

3.7 Typed multimodal training

Source datasets make incompatible assumptions. Video has frame time but no robot state; a pointing example has coordinates but no action; a teleoperated trajectory has cameras, proprioception, and embodiment-specific controls. Continuous action chunks also cannot be flattened safely into a text sequence. Multimodal Harmony (mHarmony) resolves these differences before packing: its strongly typed compiler validates source schemas, lowers embodiment-specific coordinates and state, aligns clocks, and emits TensorStream events (Perceptron Team, 2026).

Each event carries its modality, timestamp, provenance, and relevant coordinate or embodiment frame. Figure 5 traces this path from heterogeneous sources to one packed sequence. Text, image and video blocks, spatial targets, robot state, and FAST actions enter a 16,384 tokens packed sequence. Continuous Flow targets stay keyed outside the token stream and share the same observation identifiers, and causal action history enters as typed history.

Typing lets unlike sources share a representation without erasing the distinctions control needs. Before packing, the system interprets each state vector under its robot schema, retains the coordinate frame, preserves causal order, and selects the correct normalization. The same contract covers all 35+ released robot interfaces.

3.7.1 Trajectory lowering

Lowering turns a robot episode into one causal training sample. The action anchor $t^\star$ is the step where the target action chunk begins; observations and completed history through $t^\star$ are input context. The compiler then resolves the robot configuration from dataset provenance and trajectory metadata and renders a Configuration block with the robot family, objective form, control rate, relative-action semantics, action layout, and available camera views.

The prompt-facing block contains only the resolved fields the model uses. For example:

```
Configuration:
robot_type: agilex
form: Flow
fps: 10
mistake: false
action_representation: relative
control_mode: ee
action_layout: [ee_x, gripper]
camera_views: [external_high, left_wrist, right_wrist]
```

The form field selects the action decoder: Flow invokes the continuous controller used at serving, whereas separate FAST examples train the autoregressive action vocabulary. The fps field is the robot control rate from the trajectory or its normalization scope (the embodiment- or task-specific statistics entry). It is unrelated to target_fps = 2, which samples general video at two frames per second for video understanding.

The recipe independently samples three binary conditioning masks for each lowered training window: C_{scene} gates an available scene description, C_{FAST} gates available causal FAST-action history, and C_{mistake} gates the episode-level mistake label. In particular, C_{mistake} is not the label itself; it decides whether an available Boolean has_mistake value is exposed to the model:

$$C_{\text{scene}} \sim \text{Bernoulli}(0.5), \qquad C_{\text{FAST}} \sim \text{Bernoulli}(0.5), \qquad C_{\text{mistake}} \sim \text{Bernoulli}(0.8075). \tag{11}$$

Figure 6 shows the operation of these draws: a value of one keeps the corresponding optional component in the model-visible prefix, while zero omits it without changing the observations, policy state, or target action.

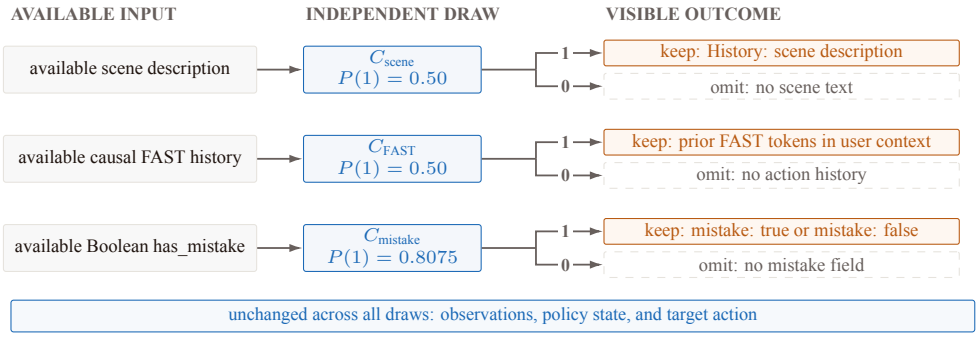


Figure 6: Conditioning masks independently hide optional context. Each lowered training window draws the three gates once; a one keeps the available component, while a zero omits it. Observations, policy state, and the action target remain fixed.

If $C_{\text{mistake}} = 1$, the compiler writes lowercase mistake: true or mistake: false in the prompt; if $C_{\text{mistake}} = 0$, it omits the field. This gate affects only sources with an explicit Boolean has_mistake annotation; sources without one always omit the field, and the compiler rejects non-Boolean or conflicting annotations. The retention probability $0.8075 = 0.85 \times 0.95$ combines the episode-metadata and component-retention rates.

The mistake flag describes the episode and has no effect on scene history. An available scene or subtask event can enter user-side History if it ends by the action anchor; a later event stays hidden and becomes an assistant-side Prediction only when that target is enabled. At serving time, any nonzero training support makes available action history deterministic and defaults the mistake value to false, giving training and inference the same authenticated prefix format. Figure 7 summarizes the lowering path and the contract it preserves across training and serving.

3.8 Embodiment-agnostic episodes

Section 3.7.1 states the lowering contract in terms of robot episodes, but nothing in it is specific to robots. An episode is an observation stream in causal order, with a clock, an anchored action at each decision point, and an outcome. Teleoperation supplies episodes of that shape. So does a recorded session in which an agent operates a graphical interface, where the screen is the observation and the outcome is whether the session satisfied the stated task.

Isaac 0.5 represents both with the same episode object and lowers both through the same pass. The action interface is the only point of variation. A physical embodiment emits continuous targets in a declared coordinate frame and action layout; a digital embodiment emits discrete named calls whose arguments are checked against the interface it declares. Neither is inferred from the data; the compiler rejects a digital action it cannot match against the declared interface, just as it rejects a physical action whose width disagrees with its layout.

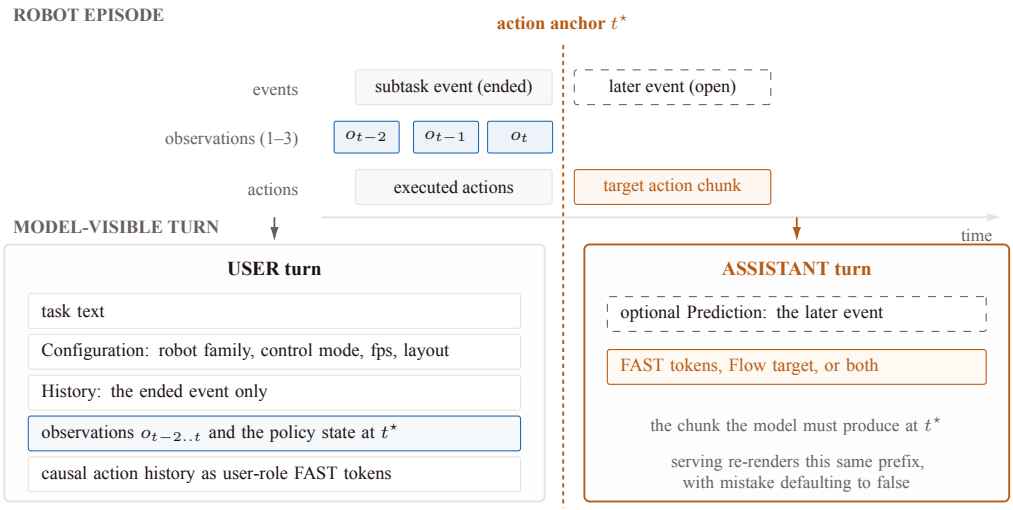


Figure 7: Trajectory lowering preserves the robot contract across training and serving. An episode is anchored at a target action time and paired with a bounded observation window and a FAST or Flow target. Ended semantic events are appended as History: after the Configuration: block; future annotations stay target-side.

The two cases share more than a container. When a digital action refers to a location on the screen, that location uses the same normalized coordinate representation the model applies to pointing and detection, rather than raw pixels or an interface-specific element identifier. A click target and a detection target are the same kind of object to the model, produced through the same output pathway. That is the channel through which perception training reaches control.

3.8.1 Bounded observation context

For robot episodes, the model conditions on one to three observations. We sample the count from a Poisson distribution with rate 2, truncated to this interval:

$$N_{\text{obs}} \sim \text{Poisson}(2) \mid 1 \le N_{\text{obs}} \le 3. \tag{12}$$

Equivalently,

$$\Pr(N_{\text{obs}} = n) = \begin{cases} 0.375, & n \in \{1, 2\}, \\ 0.250, & n = 3. \end{cases} \tag{13}$$

These bounds limit sequence-length variation and match the serving interface.

3.8.2 Causal action history

Causal action history means previously executed actions serialized before the current prediction target. This gives the model a closed-loop self-calibration signal: before producing a new action, it can cross-check recent commands against the observations that followed them, estimate execution drift, and compensate in its next prediction. The same comparison can calibrate the model in context to a new physical instance of a familiar embodiment, including differences in actuator response, latency, and control scaling, without updating its weights. For two selected observations o_i and o_j, the compiler inserts every executed action in the half-open interval $[i, j)$ before o_j:

$$o_i \;\longrightarrow\; \text{FAST}(a_i, \ldots, a_{j-1}) \;\longrightarrow\; o_j \;\longrightarrow\; \text{target}(a_j, \ldots). \tag{14}$$

Figure 8 shows which action rows are visible at each anchor. The compiler draws C_{FAST} once per packed window, separately from the observation-count and scene-dropout draws. FAST tokens from preceding actions enter as input context, receive no action-token loss, and stay distinct from FAST targets. At the start of an episode, padding does not create action history. Because the interval stops at j, current and future actions cannot enter the context. The target can carry FAST supervision, Flow supervision, or both (Pertsch et al., 2025). Causal action history is an explicit input field, not an internal recurrent state.

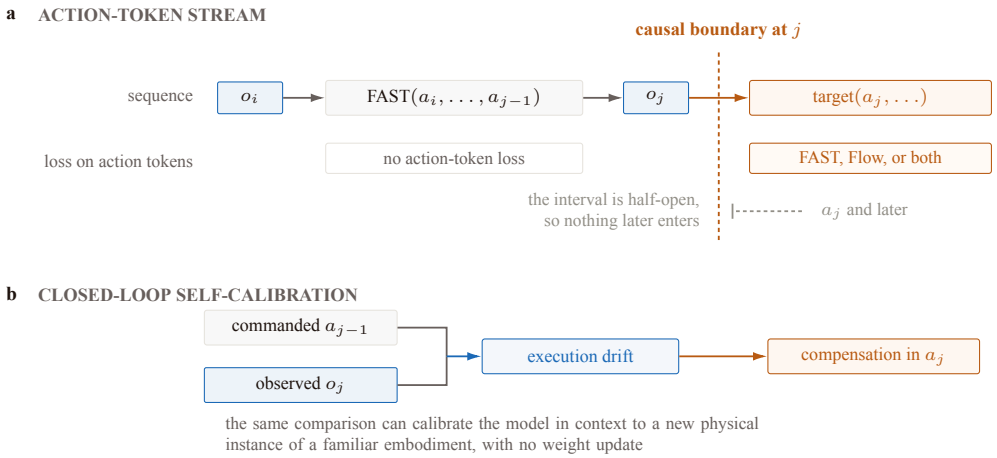


Figure 8: Causal action history. a, executed actions re-enter the sequence as user-role FAST tokens that receive no action-token loss, so the same vocabulary appears on both sides of the boundary with opposite loss roles. The interval is half-open at j, so current and future actions cannot enter the context. **b**, comparing the commanded a_{j-1} against the observed o_j estimates execution drift, which the model can compensate for in a_j.

3.9 Training recipe details

3.9.1 Optimization

We train with AdamW (Loshchilov & Hutter, 2019), using $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-6}$, and weight decay 0. The base learning rate is 10^{-5}, with 5×10^{-6} for vision parameters and 5×10^{-5} for the Flow expert. After 200 warm-up steps, each rate follows cosine decay to 10% of its peak value. We clip the global gradient norm at 1.0.

Figure 9 tests the numerical robustness of the joint training stack across losses with different scales and normalization rules. In the analyzed 100k-step run, which used a 500-step warm-up, the global, full discrete, FAST, Flow, router z-loss, and CCE z-loss curves decline, while the load-balancing loss remains tightly centered near 8.0. Together, these trajectories provide run-level evidence of robust joint optimization through step 100,000. Section 6 describes the systems that execute this mixed-objective recipe.

3.9.2 Latency-conditioned real-time chunking

Training adds real-time chunking (RTC) so a new continuous-action chunk can continue from commands that will have executed by the time inference returns (Black et al., 2025a;b). Let $H \geq 2$ be the action-chunk horizon, d the committed prefix length, and $D = \min(12, H - 1)$ the largest simulated delay. Let $p \in [0, 1]$ be the probability of a nonzero delay and $\lambda > 0$ the Poisson rate. With probability $1 - p$, $d = 0$; otherwise d follows a Poisson distribution truncated to $1, \ldots, D$:

$$\Pr(d = j) = \begin{cases} 1 - p, & j = 0, \\ p \dfrac{\mathrm{Pois}_\lambda(j)}{\sum_{u=1}^{D} \mathrm{Pois}_\lambda(u)}, & 1 \leq j \leq D. \end{cases} \tag{15}$$

Here $\mathrm{Pois}_\lambda(j) = e^{-\lambda}\lambda^j / j!$. The recipe uses $H = 50$, $p = 0.5$, and $\lambda = 5$, with 4 independent Flow training draws per chunk. Prefix rows keep their action timestamps and enter the Flow/DiT expert as clean conditioning at Flow time $t = 1$ (noise level $\tau = 0$) under the clean-at-one convention above, excluded from the velocity loss. We set the training delay distribution with p and λ and do not fit it to measured serving latency.

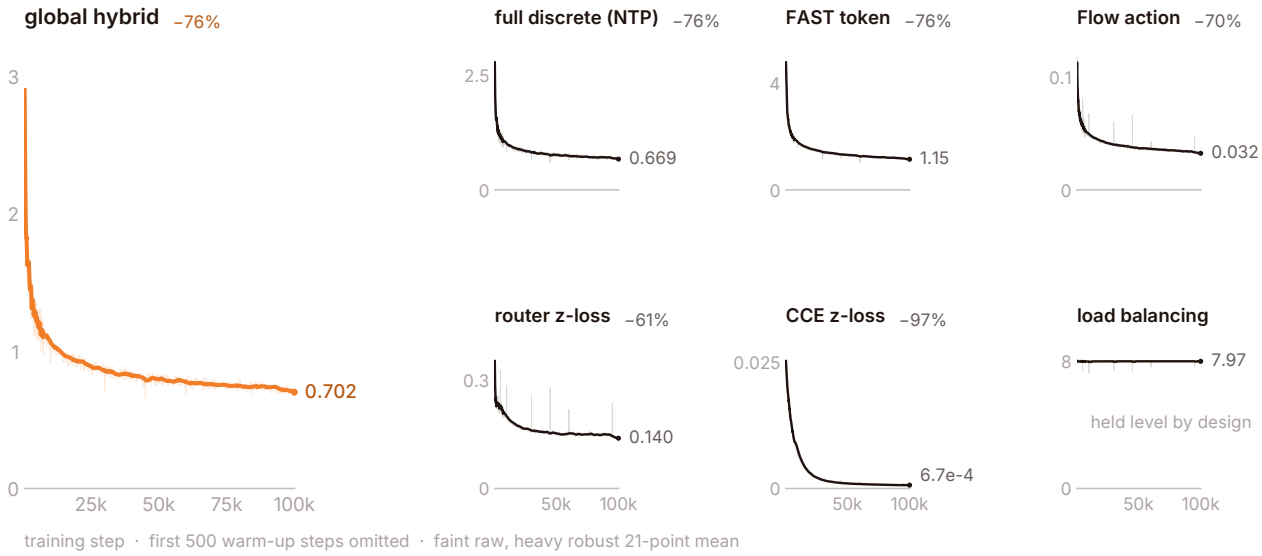


Figure 9: Training-stack robustness across primary and auxiliary losses. Training-loss traces for the release pretraining run through checkpoint 100,000, each on its native scale. The large panel is the global hybrid objective the optimizer descends; the small panels are its full discrete NTP, FAST token, and Flow action components and the router z-loss, CCE z-loss, and load-balancing auxiliaries. Faint traces are the raw logged observations, heavy lines are robust trailing 21-point means, the printed value is the robust mean at step 100,000, and the percentage is its change from the post-warm-up level. The first 500 optimizer warm-up steps are omitted. The six optimized losses decline through step 100,000, while the load-balancing regularizer holds its level near 8.0 by design.

4 Percepts scale control: how video substitutes for teleoperation

Having described the model and what it is trained on, we now measure what the video side of that cotraining is worth. We ask how much video cotraining changes the teleoperation needed to reach a well-calibrated offline action-loss threshold. The measurement is a two-factor grid: seven general-video budgets from 1,000 to one million hours, crossed with teleoperation budgets over a common 2–10,000-hour support, one run per cell and nested video subsets. Tasks and embodiments are held to the fixed evaluation setting of Section 3 rather than varied. Every run in the grid uses the architecture and objectives of Section 3, and the released Isaac 0.5 checkpoint is the grid's highest video-budget point, so the measurement below describes the recipe we shipped rather than a separate study.

4.1 Types of training data

We use four kinds of training data, with corpus composition and filtering described in Section 2. *General video* shows appearance, motion, and semantic change, but not the actions that produced them. *Egocentric video* adds a first-person view of hands, contact, and task progress (Grauman et al., 2022), including hand-pose data that gives approximate human hand motion. *Universal Manipulation Interface (UMI)-style handheld grippers* record gripper motion and timing without a robot, but only for actions the device can perform (Chi et al., 2024). *Task-specific teleoperation* is the most expensive and records robot commands, robot state, timing, and outcomes directly.

The grid varies general video against teleoperation. Every video budget carries passive-egocentric and UMI companion streams, each contributing $3V/8$ hours for a point labeled V; we call this the *linked* video mixture, since all three streams scale together. Teleoperation varies independently. The purchasing analysis in Section 4.4 uses teleoperation:general video:egocentric:UMI $=$ 1:10:3.75:3.75. This composition preserves the measured 80:30:30 video proportions and allocates ten general-video hours per teleoperation hour. The grid supplies the loss response; Section 4.4 combines it with prices to obtain the cost-aware mixture.

4.2 Measurement design

Let V be general-video hours under the 80:30:30 mix, h the accepted task-specific teleoperation hours, and $L(V, h)$ the offline action loss, lower being better. Both V and h are in hours. L is a single held-out action objective evaluated under fixed settings and held identical across every cell of the grid, so losses are comparable within the grid; τ is quoted in the units of that objective. The comparison relies only on that invariance, not on the choice of objective. For a loss threshold τ, the teleoperation requirement is

$$H_\tau(V) = \inf\{h : L(V, h) \leq \tau\}. \tag{16}$$

Where no measured teleoperation level reaches τ, we report the target as not reached and do not estimate the shortfall. For two video-hour levels $V_a < V_b$, the substitution factor is

$$S_\tau(V_a \to V_b) = \frac{H_\tau(V_a)}{H_\tau(V_b)}. \tag{17}$$

Within the measured range we fit $H_\tau(V) = c_\tau V^{-\beta_\tau}$, the power-law form used in neural scaling-law studies (Kaplan et al., 2020; Hoffmann et al., 2022), with c_τ a fitted scale constant. The exponent β_τ is the elasticity: on log scales it says how fast the required teleoperation falls as video hours rise. Across two video levels we compute the same quantity as the negative change in $\log H_\tau$ over the change in $\log V$. A positive elasticity means more video pretraining goes with fewer teleoperation hours at the same threshold.

We form a monotone envelope separately at each fixed video budget V: at every teleoperation rung, we use the lowest loss observed at that rung or any lower-hour rung. Threshold crossings use linear interpolation in $\log_{10} h$ between adjacent measured settings, and Figure 10A applies the same interpolation in $\log_{10} V$ over the common 2–10,000-hour teleoperation support. Every crossing and surface value therefore stays inside the measured grid. Separately, at every teleoperation rung represented by at least five video budgets, we regress raw loss on $\log_{10} V$. These per-rung slopes in Figure 10B describe the observed grid. The grid has one run per cell and the video subsets are nested, so we attach no standard errors to the slopes and do not read them as causal effects.

4.3 Results

We use $\tau = 2.50$ as the primary offline threshold. We also tested it in a calibrated real-world setting and found it reliable. Within the offline grid, it is the strictest contour both the 1k and 1M video budgets reach within their measured teleoperation support, so H_τ is defined at both budgets without extrapolation. The runs at 1k general-video hours do not reach the stricter $\tau = 2.45$. We report the neighboring thresholds as sensitivity analyses.

At the primary threshold, the within-grid crossings are 5,884 hours at 1k general-video hours and 28 hours at 1M. Their ratio is 210.3×, with fitted elasticity 0.79. At the looser threshold the same 1k-to-1M ratio is 27.6× at elasticity 0.49.

The crossings lie between measured teleoperation rungs: 2,500 and 6,000 hours at 1k video, and 20 and 30 hours at 1M. These endpoints bound the 1k-to-1M substitution factor at 83× to 300×, equivalent to elasticities between 0.64 and 0.83.

The raw grid explains when that substitution appears. At one teleoperation hour, the six supported video budgets span only 0.050 loss, and 10× more video is associated with a change of −0.006. That spread is comparable to individual non-monotonic reversals elsewhere in the grid. From roughly 100 teleoperation hours onward, the per-rung changes cluster near −0.21 loss for each 10× increase in video. The exchange rate is therefore conditional rather than uniform. At the smallest teleoperation budgets the two inputs behave as complements: video moves loss very little until the recipe also contains action-grounded experience. The substitution measured by Equation 17 is a property of that grounded regime, not a constant conversion between the two inputs.

The 1k-to-1M ratio also depends on the loss threshold. Across $\tau = 3.00, 2.75, 2.50$, and 2.45, the ratios are respectively 5.7×, 27.6×, 210.3×, and undefined because no run at 1k general-video hours reaches the last threshold. Appendix Table 14 reports the crossings.

Figure 10C shows a bounded recipe contrast: seven *non-perceptive* runs, one per video budget, trained without the semantic future-percept objective of Section 3.4. None reaches $\tau = 2.75$ within

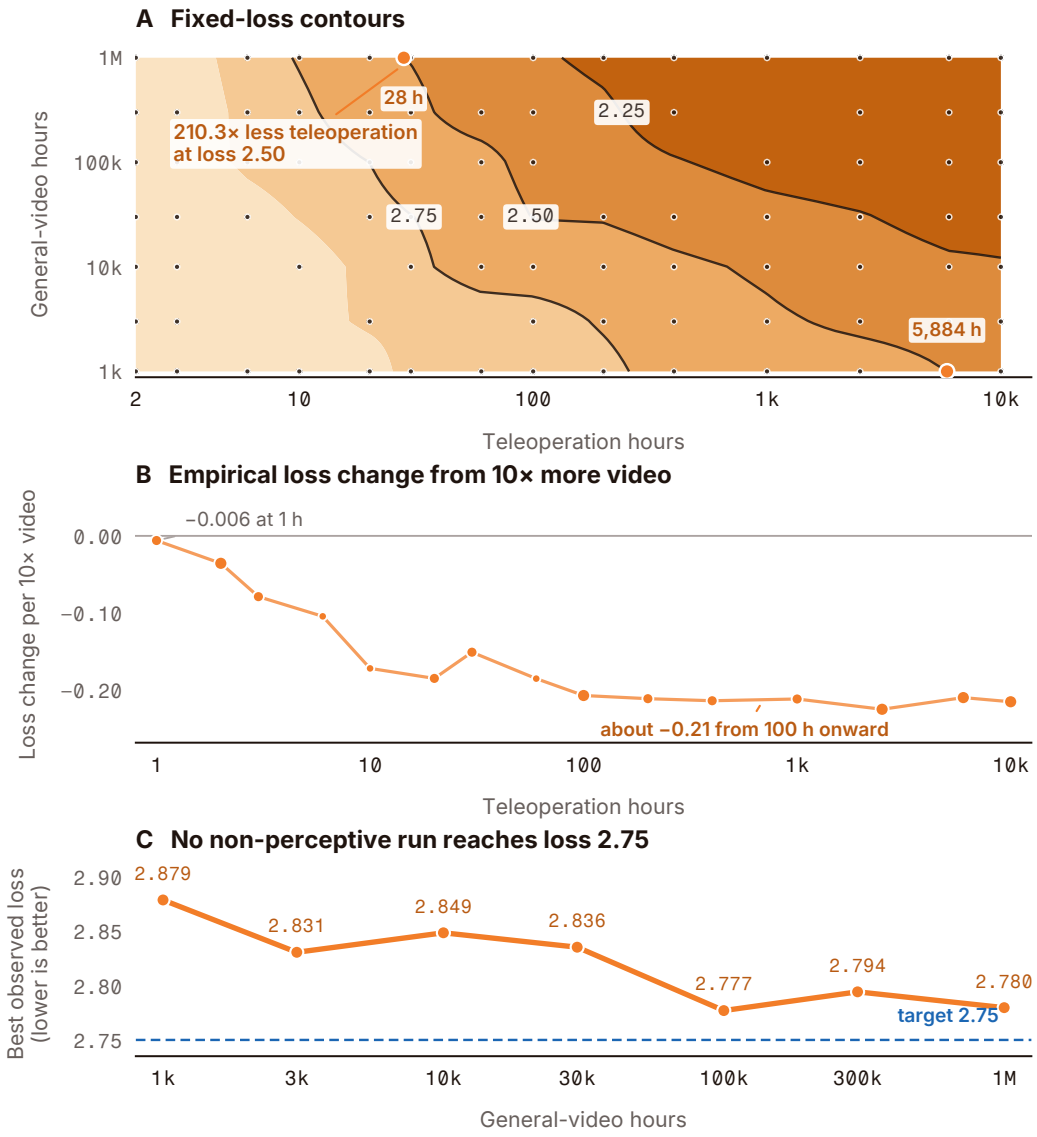


Figure 10: Video substitutes for teleoperation only in the action-grounded regime. **A**, offline action loss over the measured scaling grid, with log-space interpolation between adjacent settings. Small dark points mark measured cells; orange points mark the 1k and 1M threshold crossings behind the 210.3× ratio. **B**, empirical raw-loss change for each 10× increase in video at teleoperation rungs supported by at least five video budgets; point size encodes the number of supported budgets. **C**, best observed non-perceptive loss at each video budget. All seven remain above 2.75.

its measured teleoperation support. These runs do not span the same teleoperation rungs as the main grid, and are not documented as differing from it only in the objective—the 300k non-perceptive run, for example, stops at 1,000 teleoperation hours—so we do not turn this censoring result into an objective-effect coefficient.

Video elasticity is positive at every supported threshold. Its magnitude varies with the threshold and available support.

4.4 Cost implications for a fixed acquisition plan

We price the fixed 1:10:3.75:3.75 data mix using Table 2 and architecture-derived video-training compute.

Figure 11 prices the fixed composition and reads the loss surface of Figure 10 along that path as the budget grows. The rest of this section derives the two panels and reports the corresponding plan at $1M.

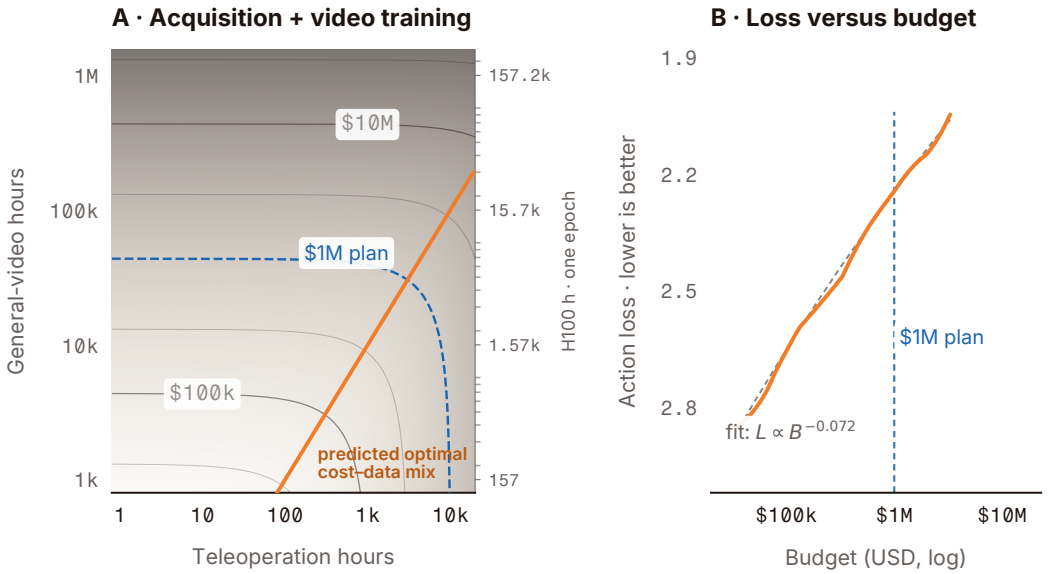


Figure 11: Cost and loss along a fixed acquisition plan. A, acquisition plus one video-training epoch for general, egocentric, and UMI video with $E = U = 3V/8$; the right axis gives the corresponding H100 GPU-hours and the orange diagonal is the teleoperation:general-video:egocentric:UMI composition 1:10:3.75:3.75. Its intersection with the dashed $1M contour gives Table 3. **B**, the monotone loss surface of Figure 10 evaluated along that path (solid), with the power-law fit of Equation 23 (dashed).

4.4.1 Planning prices

Table 2 assigns planning prices to the data types in Section 4.1 and to NVIDIA H100 GPU time. The values 3.75 in the mix are source-hours per teleoperation hour, and the price is $30 per accepted source-hour. Video-side training compute follows from the architecture calculation below.

Table 2: Planning prices and purchasing composition. Acquisition rows are priced per accepted source-hour. The quantity column gives each resource per one teleoperation hour, q. The H100 row covers one epoch over all 17.5 linked video hours.

Cost item	Planning price	Quantity per q	Treatment
Task-specific teleoperation	$100	1	robot, operator, resets
General video	$0.10	10	licensing, filtering, storage
Passive egocentric	$30	3.75	wearable capture and quality assurance
UMI-style handheld	$30	3.75	rig and collector time
H100 video-side compute	$2.10 / GPU h	1.572 GPU h	one video epoch; 30% MFU

Let h, V, E, U denote accepted hours of teleoperation, general video, passive egocentric video, and UMI data, and let c_h, c_V, c_E, c_U be their acquisition prices per accepted hour. Let F_v be the training

FLOPs per video source-hour, P_G the accelerator's dense-BF16 peak, and ρ the utilization. A max-length source-hour at 2 frames per second saturates the configured patch cap and produces 460,800 emitted visual tokens. The repository FLOP formulas give a full-position classifier calculation using four classifier passes per packed position under the 50% null calculation reference, including forward and backward training passes. The emitted visual positions account for 19.185 PFLOP per source-hour. Applying the $5\times$ multiplier for captions, timestamps, control scaffolding, and other learned document tokens gives $F_v = 95.925$ PFLOP per source-hour. The one-epoch intensity in GPU-hours per video source-hour is

$$g = \frac{F_v}{3{,}600\,\rho P_G}. \tag{18}$$

With $\rho = 30\%$ and $P_G = 989\,\mathrm{TFLOP\,s^{-1}}$, this gives $g = 0.08981$ H100 GPU-hours per video source-hour. We apply the same intensity to general, egocentric, and UMI video. Acquisition plus one video epoch then costs

$$C_{\text{plan}}(h, V, E, U) = c_h h + c_V V + c_E E + c_U U + c_G g(V + E + U), \tag{19}$$

where c_G is the price per GPU-hour. Let r_h, r_V, r_E, r_U be the four quantities in Table 2, and define the cost of one unit of that recipe as

$$\kappa = r_h c_h + r_V c_V + r_E c_E + r_U c_U + (r_V + r_E + r_U) c_G g. \tag{20}$$

The recipe cost is therefore $C_{\text{mix}}(q) = \kappa q$, the orange diagonal in Figure 11A. Substituting the values in Table 2 gives $\kappa = \$329.30$ per accepted teleoperation hour. This total covers acquisition and one video epoch; teleoperation-side training and additional video epochs are outside the plan.

4.4.2 Turning a dollar budget into a build plan

A budget B determines one point on the purchasing-composition path. Let q_B be the number of recipe units that the budget purchases. Then

$$q_B = \frac{B}{\kappa}, \qquad (h_B, V_B, E_B, U_B) = q_B(r_h, r_V, r_E, r_U). \tag{21}$$

Let $\widetilde{L}(V, h)$ be the monotone surface in Figure 10: linear in $\log_{10} h$ between adjacent teleoperation rungs and in $\log_{10} V$ between adjacent video levels, and defined over the measured support. The conditional loss–budget response is

$$L_{\text{mix}}(B) = \widetilde{L}(V_B, h_B). \tag{22}$$

For a compact summary, we fit a two-parameter power law to the five measured-video levels whose composition-path teleoperation value lies within support:

$$\widehat{L}_{\text{mix}}(B) = 2.246 \left(\frac{B}{\$1M}\right)^{-0.07224}. \tag{23}$$

The fit is ordinary least squares in log loss versus log budget over \$32.9k–\$3.29M, and every knot interpolates between teleoperation rungs. In raw loss its root mean squared error is 0.0180 and $R^2 = 0.9961$. Doubling the budget multiplies fitted loss by $2^{-0.07224}$, a 4.9% reduction.

At \$1M, Equation 21 gives 3,037 teleoperation hours, 30,367 general-video hours, and 11,388 hours each of passive egocentric and UMI data. One epoch over the 53,143 total video hours requires 5.098×10^{21} FLOP, or 4,773 H100 GPU-hours at the stated MFU.

Table 3: The \$1M data-and-training plan. Quantities obey teleoperation:general video:passive egocentric:UMI = 1:10:3.75:3.75. The budget covers acquisition and one epoch over all three video streams.

Resource	Mix	Quantity	Spend
Task-specific teleoperation	1	3,037 h	\$303.7k
General video	10	30,367 h	\$3.0k
Passive egocentric	3.75	11,388 h	\$341.6k
UMI-style handheld	3.75	11,388 h	\$341.6k
H100 video-side compute	—	4,773 GPU h	\$10.0k
Total			\$1.0M

On the declared 16-H100 topology, 4,773 aggregate GPU-hours correspond to 298 wall-clock hours at 30% MFU. At the top of Panel A, one million general-video hours imply 1.75 million linked video hours and 157.2k H100 GPU-hours for one video epoch under the same architecture calculation.

At one million general-video hours, the fixed composition sets 100,000 teleoperation hours and 375,000 hours each of passive egocentric and UMI data.

5 Evaluation

We evaluate Isaac 0.5 on perception, embodied reasoning, and closed-loop control, because no one of them is sufficient. A policy with weak general perception is a specialist; a visual model without closed-loop results has not shown it can act as a robot model.

5.1 Broad benchmark results

Tables 4–7 report the broad image scorecard. Every local comparison uses the same named metric and reports the raw benchmark score before random-baseline adjustment. The local sweep contains Qwen3-VL at 2B, 4B, 8B, and 32B. Where a public value exists for the same Qwen model, it is printed in gray directly beneath the score from our run of that model; the two frequently differ, which is the protocol sensitivity described below. These gray entries are Qwen-authored release numbers except for evaluations sourced from NVIDIA, Liquid AI, and SCOPE (Qwen Team, 2025; 2026a;d; NVIDIA, 2025; Liquid AI, 2026; Cao et al., 2026).

Table 4: Grounding and counting. Models are rows and benchmarks are columns. Scores use the benchmark-native CountBench, RefCOCO, and ScreenSpot-Pro protocols.

Model	Cost multiple of Isaac	CARPK Counting	LVIS Count	ScreenSpot Pro
Isaac 0.5 36B-A2.5B	1.0	**19.07**	**32.78**	**62.60**
Qwen3-VL-32B	12.8	1.60	27.06	54.80 57.9
Qwen3-VL-8B	3.2	1.87	28.72	54.6
Qwen3-VL-4B	1.6	6.00	25.41	46.20 59.5
Qwen3-VL-2B	0.8	1.33	28.34	36.00 48.5
Qwen3.5 35B-A3B	1.2	1.67	28.27	68.60
Qwen3.6 35B-A3B	1.2	1.72	29.16	70.80
Qwen3.5-4B	1.6	6.81	28.82	60.40
Cosmos 3 Super	12.8	1.52	25.76	57.20
Cosmos-Reason2-32B	12.8	1.50	25.30	45.20
Cosmos 3 Nano	3.2	1.77	27.21	52.80
Cosmos-Reason2-8B	3.2	1.67	25.64	50.80
Cosmos 3 Edge	0.8	1.37	29.12	46.00
Cosmos-Reason2-2B	0.8	1.21	25.82	46.40

The nominal cost multiple is the advertised active backbone size divided by Isaac's advertised 2.5B active size (Isaac = 1.0); it uses rounded release labels rather than exact tensor counts and is a parameter-based inference-cost proxy, not measured end-to-end FLOPs or latency. Black values are benchmark scores; gray Qwen3-VL values are public reports. (Qwen Team, 2025; 2026b;c;d; NVIDIA, 2026a;b; Li et al., 2025; Paiss et al., 2023).

Table 5: Structured visual understanding: documents and OCR.

Model	Cost multiple of Isaac	AI2D	DocVQA	M3Exam English	OCRBench	OCRBench v2
Isaac 0.5 36B-A2.5B	1.0	**87.24**	**96.54**	**81.40**	**92.61**	66.12
Qwen3-VL-32B	12.8	83.39 89.5	95.70 96.9	78.20	89.29 89.5	66.03 67.4/59.2
Qwen3-VL-8B	3.2	77.95 85.7	95.95 96.1	67.20	89.77 89.6	**67.70** 65.4/61.2
Qwen3-VL-4B	1.6	76.55 84.1	94.90 95.3	57.80	88.89 88.1	65.68 63.7/57.6
Qwen3-VL-2B	0.8	66.16 76.9	92.40 93.3	52.60	86.17 85.8	62.76 56.3/53.0
Qwen3.5 35B-A3B	1.2	86.60 92.6	96.10	80.20	91.50 91.0	65.40
Qwen3.6 35B-A3B	1.2	86.90 92.7	96.30	80.80	92.00 90.0	65.80
Qwen3.5-4B	1.6	81.90 89.6	94.40 94.8	64.80	85.40 85.0	60.70 58.7

The nominal cost multiple is the advertised active backbone size divided by Isaac's advertised 2.5B active size (Isaac = 1.0); it uses rounded release labels rather than exact tensor counts and is a parameter-based inference-cost proxy, not measured end-to-end FLOPs or latency. Black values are scores from our runs; gray values are publicly reported Qwen results under the cited source protocols. When both are available, the local value is stacked above the public report. The best score from our runs is bold when at least two such results are present. n/r = not reported. Slash-separated OCRBench v2 values preserve Qwen's two published settings (Qwen Team, 2025; 2026a;d; NVIDIA, 2025; Liquid AI, 2026; Cao et al., 2026).

5.1.1 Prompt sensitivity and reproducibility

We use a slight variant of the public benchmark prompt by design. In our evaluations, we have observed that benchmark-specific prompt and harness optimization can materially change scores without any change to the underlying model. For local comparisons, we therefore prioritize a consistent protocol across models over exact reproduction of each release recipe. All of our local runs disable thinking and use greedy decoding. By contrast, the concrete inference settings behind many author-reported values are not disclosed, including whether thinking or chain-of-thought prompting was enabled and which decoding parameters, generation budgets, image preprocessing, and answer-extraction rules were used. The paired local and author-reported entries demonstrate this protocol sensitivity directly. We use the runs from our evaluation as the primary comparison and show author-reported values as protocol-specific context rather than as prompt-matched measurements.

Table 6: Structured visual understanding: charts, tables, and text.

Model	Cost multiple of Isaac	ChartQA	Reducto TableBench	TextVQA	TreeBench
Isaac 0.5 36B-A2.5B	1.0	**86.24**	**76.52**	**83.58**	**48.40**
Qwen3-VL-32B	12.8	82.08 88.5	73.66	79.90	46.42
Qwen3-VL-8B	3.2	77.92 89.6	72.11 82.4	81.81 82.9	**48.40** 26.9
Qwen3-VL-4B	1.6	75.04 84.6	66.78	81.55	40.25 45.2
Qwen3-VL-2B	0.8	67.36 79.1	58.34	80.35	32.35
Qwen3.5 35B-A3B	1.2	83.10	74.80	82.70	46.80
Qwen3.6 35B-A3B	1.2	84.00	75.40	83.10	47.40
Qwen3.5-4B	1.6	77.40 84.2	68.50	80.20 81.2	40.80

The nominal cost multiple is the advertised active backbone size divided by Isaac's advertised 2.5B active size (Isaac = 1.0); it uses rounded release labels rather than exact tensor counts and is a parameter-based inference-cost proxy, not measured end-to-end FLOPs or latency. Black values are scores from our runs; gray values are publicly reported Qwen results under the cited source protocols. When both are available, the local value is stacked above the public report. The best score from our runs is bold when at least two such results are present. n/r = not reported (Qwen Team, 2025; 2026a;d; NVIDIA, 2025; Liquid AI, 2026; Cao et al., 2026).

Table 7: General visual intelligence. Models are rows and benchmarks are columns. Where both are available, each cell stacks our run over the public report value in gray.

Model	Cost multiple of Isaac	A-OK VQA	BLINK	CV Bench	Emb. Spatial	MME yes/no	Real World QA	SEED Bench	VLMs Blind VQA v2	
Isaac 0.5 36B-A2.5B	1.0	89.80	**66.80**	**73.93**	**84.80**	**91.58**	79.19	**79.60**	**70.40**	83.76
Qwen3-VL-32B	12.8	89.20	62.60	73.22	81.5	90.14	77.62 79.0	77.40	64.90	82.62
Qwen3-VL-8B	3.2	87.20	58.00	73.08	78.5	89.17	68.85 71.5	72.80	63.60	81.64
Qwen3-VL-4B	1.6	83.40	54.80 65.8	73.50	79.6	86.90	71.34 70.9	72.60	62.40	81.44
Qwen3-VL-2B	0.8	79.80	45.80 53.8	69.94	69.2	82.14	64.79 63.9	61.40	49.00 56.0	78.89
Qwen3.5 35B-A3B	1.2	90.0	65.6	73.5	83.1	90.9	79.5	79.0	69.0	83.5
Qwen3.6 35B-A3B	1.2	**90.4**	66.2	73.8	84.3	91.3	**80.1**	79.3	69.8	**84.1**
Qwen3.5-4B	1.6	84.8	58.7	72.0	81.3	87.4	72.0	76.1	61.5	81.0

The nominal cost multiple is the advertised active backbone size divided by Isaac's advertised 2.5B active size (Isaac = 1.0); it uses rounded release labels rather than exact tensor counts and is a parameter-based inference-cost proxy, not measured end-to-end FLOPs or latency. Black values are scores from our runs; gray values are publicly reported Qwen results under the cited source protocols. The column maximum is bold. (Qwen Team, 2025; 2026a;d; NVIDIA, 2025; Liquid AI, 2026; Cao et al., 2026).

Tables 8–9 then compare physical-AI benchmarks without repeating them across model-size panels. Isaac 0.5 has 36B total parameters but only 2.5B active parameters per token in expectation. Each table reports a nominal cost multiple: the model's advertised active reasoner or backbone size divided by Isaac's advertised 2.5B active size. The proxy uses rounded release labels rather than exact tensor counts; dense models use their advertised backbone size, sparse models use their advertised active size, and Cosmos 3 uses the advertised size of one active reasoner tower rather than the dual-tower checkpoint total. This parameter-based quantity is a transparent inference-cost proxy, not a measurement of end-to-end FLOPs or latency (Qwen Team, 2025; 2026a;d; Google DeepMind, 2026; NVIDIA, 2026a;b). Models appear as rows, braced into matched sparse-and-compact, large, mid-scale, and small open-model tiers. Native benchmark scales remain separate.

5.2 Perception

The perception evaluation covers recognition, grounding, tracking, spatial localization, physical state, temporal order, and video understanding. The primary physical-AI scorecard includes HallusionBench, VideoPhy2, CausalVQA, MVPBench, MVBench, BLINK depth and spatial subsets, and VSI-Bench rather than academic science or mathematics question answering (Li et al., 2024; Yang et al., 2025a; Fu et al., 2024). On the physical and temporal scorecard (Table 8), Isaac 0.5 records the best available score on six of the seven benchmarks: HallusionBench 77.4 against 69.8 for the strongest comparator, VideoPhy2 64.8 against 60.8, CausalVQA 83.0 against 81.0, MVPBench 76.6 against 71.0, and BLINK Spatial 92.3 against 90.9, with BLINK Depth tied at 92.7. It trails on MVBench, where Cosmos 3 Super reaches 74.5 against 72.8 at 12.8 times the active-parameter cost. Comparator values keep their native protocols and are author-reported, so these are matched-metric rather than matched-protocol comparisons.

Table 8: Physical and temporal reasoning. Each benchmark appears once; models are rows. The matched, large, mid-scale, and small blocks use the available benchmark results.

	Model	Cost multiple of Isaac	Hallusion Bench	Video Phy2	Causal VQA	MVP Bench	MV Bench	BLINK Depth	BLINK Spatial
Matched	**Isaac 0.5 36B-A2.5B**	1.0	**77.4**	**64.8**	**83.0**	**76.6**	72.8	**92.7**	**92.3**
	Qwen3.6 35B-A3B	1.2	69.8	60.8	78.0	71.0	70.8	89.6	88.4
	Qwen3.5 35B-A3B	1.2	67.9	59.6	76.0	69.0	70.4	88.8	87.2
	Qwen3.5-4B	1.6	65.0	54.2	71.0	63.0	68.6	84.3	82.7
	Gemma 4-26B-A4B	1.6	66.1	56.4	73.0	66.0	69.2	86.8	85.6
Large	Cosmos 3 Super	12.8	49.0	47.4	77.0	70.3	**74.5**	91.9	88.8
	Qwen3-VL-32B	12.8	52.8	36.8	81.0	58.6	72.6	82.3	88.1
	Cosmos-Reason2-32B	12.8	50.8	43.3	74.5	62.2	72.5	85.5	87.4
	Gemma 4-31B	12.4	57.8	33.1	76.5	27.0	64.4	87.1	90.9
Mid-scale	Cosmos 3 Nano	3.2	45.3	45.6	70.0	66.9	73.2	**92.7**	81.8
	Qwen3-VL-8B	3.2	50.5	28.2	72.0	51.6	69.1	87.1	87.4
	Cosmos-Reason2-8B	3.2	42.0	37.1	71.5	54.2	70.1	87.9	83.9
	Gemma 4-E4B	1.6	42.0	13.8	38.0	32.8	48.9	77.4	76.9
Small	Cosmos 3 Edge	0.8	40.7	40.3	37.0	53.3	58.2	79.8	72.0
	Qwen3-VL-2B	0.8	42.6	7.9	57.0	43.6	60.3	74.2	77.6
	Cosmos-Reason2-2B	0.8	28.6	12.8	53.5	43.7	60.5	83.1	75.5
	Gemma 4-E2B	0.8	36.0	8.4	29.5	31.9	41.2	70.2	62.9

Braces group the matched sparse-and-compact models and the large, mid-scale, and small open models. The nominal cost multiple is the advertised active reasoner/backbone size divided by Isaac's advertised 2.5B active size (Isaac = 1.0); it uses rounded release labels rather than exact tensor counts and is a parameter-based inference-cost proxy, not measured end-to-end FLOPs or latency. For Cosmos 3, the proxy uses the active reasoner tower rather than the dual-tower checkpoint total. Isaac VideoPhy2 is reported using the benchmark composite score. Comparator values are publicly reported in Table 10 of the Cosmos 3 technical report or the exact Qwen3.5/Qwen3.6 model cards (NVIDIA, 2026a; Qwen Team, 2026b;c;d); protocols may differ from the Isaac run. The column maximum is bold; higher is better.

5.3 Embodied reasoning

Embodied reasoning is evaluated through judgments about task progress, spatial relations, affordances, temporal change, physical state, likely consequences, and recovery. The reported benchmarks are ERQA, EmbSpatial-Bench, RefSpatialBench, RoboSpatial, BLINK, and the Cosmos-Reason1 evaluation sets (Gemini Robotics Team, 2025; Du et al., 2024; Song et al., 2025; Fu et al., 2024; Azzolini et al., 2025). The scorecard pairs raw benchmark scores with a task-balanced macro and fixes prompts, decoding, visual sampling, and benchmark adapters across the primary comparison. On the seven-benchmark spatial and embodied scorecard (Table 9), Isaac 0.5 leads on five benchmarks, including VSI-Bench at 64.99 against 61.7 and RefSpatialBench at 65.00 against 64.3. It trails on ERQA at 52.5 against 66.1 and on Where2Place at 65.1 against 71.0.

Table 9: Spatial and embodied reasoning. Each benchmark appears once; models are rows. The matched, large, mid-scale, and small blocks use the available benchmark results.

	Model	Cost multiple of Isaac	VSI-Bench	ERQA	Ref Spatial	Cosmos ER	Cosmos CS	Where2 Place	RoboSp. Home
Matched	**Isaac 0.5 36B-A2.5B**	1.0	**64.99**	52.5	**65.00**	**76.2**	**68.1**	65.1	**72.3**
	Qwen3.6 35B-A3B	1.2	61.7	**66.1**	64.3	72.5	64.8	66.5	68.2
	Qwen3.5 35B-A3B	1.2	61.0	64.8	63.5	71.2	63.7	65.2	67.4
	Qwen3.5-4B	1.6	57.2	54.0	54.6	60.8	57.6	55.8	62.1
	Gemma 4-26B-A4B	1.6	58.4	57.6	59.1	63.5	60.2	61.0	64.0
Large	Cosmos 3 Super	12.8	60.9	51.2	57.0	74.1	66.4	**71.0**	70.0
	Qwen3-VL-32B	12.8	59.5	46.62	52.7	61.3	63.4	56.0	65.1
	Cosmos-Reason2-32B	12.8	58.0	42.8	48.0	74.9	65.2	59.0	64.3
	Gemma 4-31B	12.4	47.6	47.8	46.6	54.3	61.1	52.0	63.0
Mid-scale	Cosmos 3 Nano	3.2	54.9	46.0	53.1	69.7	63.9	64.0	66.3
	Qwen3-VL-8B	3.2	55.1	42.61	47.6	56.9	58.4	53.0	64.8
	Cosmos-Reason2-8B	3.2	52.0	44.2	41.1	71.2	63.1	50.0	64.5
	Gemma 4-E4B	1.6	28.4	30.2	24.6	44.3	43.2	17.0	42.0
Small	Cosmos 3 Edge	0.8	59.2	42.0	48.4	56.6	51.5	55.0	63.4
	Qwen3-VL-2B	0.8	49.8	39.60	27.1	48.9	49.7	32.0	44.6
	Cosmos-Reason2-2B	0.8	45.0	37.2	30.7	59.0	54.8	33.0	52.0
	Gemma 4-E2B	0.8	27.3	32.5	16.2	38.0	35.3	15.0	36.3

Braces group the matched sparse-and-compact models and the large, mid-scale, and small open models. The nominal cost multiple is the advertised active reasoner/backbone size divided by Isaac's advertised 2.5B active size (Isaac = 1.0); it uses rounded release labels rather than exact tensor counts and is a parameter-based inference-cost proxy, not measured end-to-end FLOPs or latency. For Cosmos 3, the proxy uses the active reasoner tower rather than the dual-tower checkpoint total. The local Isaac ERQA score uses raw multiple-choice accuracy over 399 unique sample IDs from 400 rows; one exact duplicate is removed. Qwen3-VL-2B, 8B, and 32B use the same local ERQA metric and deduplicated sample set. Isaac RefSpatial is a 277-sample local centroid score. Comparator values are publicly reported in Table 10 of the Cosmos 3 technical report or the exact Qwen3.5/Qwen3.6 model cards (NVIDIA, 2026a; Qwen Team, 2026b;c;d); protocols may differ from the Isaac run. The column maximum is bold; higher is better.

5.4 Closed-loop control

The scored closed-loop comparison centers on LIBERO's deterministic spatial, object, goal, and long-horizon suites (Liu et al., 2023). Table 10 compares Isaac 0.5 with the author-reported baselines under each source protocol. The direct NVIDIA rows use GR00T N1.7 and the Cosmos3 Nano and Edge DROID policy checkpoints (NVIDIA, 2026c;a). The physical rollouts in Figures 12–14 are qualitative demonstrations and remain separate from this scored comparison.

Figure 12 shows perception-control coupling across relational classification, game-state reasoning, and symbolic sequencing on physical hardware.

5.4.1 Adaptation

The adaptation protocol measures acquisition of a new task or embodiment from benchmark-specific demonstrations. Table 10 reports LIBERO spatial, object, goal, and long-horizon transfer after adaptation on 500 demonstrations per suite (Liu et al., 2023).

a Judge symmetry; sort

b Read board; place piece

c Read letters; spell

Figure 12: Physical rollouts couple relational and symbolic perception to closed-loop control. **a**, visual symmetry determines the labeled bin for object placement; **b**, the evolving board state determines the next game-piece placement; **c**, letter identity and the partial word guide ordered block placement.

Table 10: LIBERO success after benchmark-specific adaptation, in percent. Each suite contains ten tasks and 500 demonstrations; MolmoAct2 reports 2,000 evaluation rollouts across the four suites. Most numeric rows are transcribed as author-reported context from MolmoAct2 (Fang et al., 2026); the GR00T N1.7 row follows NVIDIA's official 200-episode-per-suite report (NVIDIA, 2026c). Cost multiple is the advertised active reasoner/backbone size divided by Isaac's 2.5B active size; it is a parameter proxy, not measured FLOPs or latency.

Model	Cost multiple of Isaac	Spatial	Object	Goal	Long	Average
Isaac 0.5	1.0	98.0	99.0	98.8	93.0	**97.2**
TraceVLA (Zheng et al., 2025)	2.8	84.6	85.2	75.1	54.1	74.8
OpenVLA	2.8	84.7	88.4	79.2	53.7	76.5
SpatialVLA (Qu et al., 2025)	1.2	88.2	89.9	78.6	55.5	78.1
CoT-VLA (Zhao et al., 2025)	2.8	87.5	91.6	87.6	69.0	83.9
π_0	1.2	96.8	98.8	95.8	85.2	94.2
ThinkAct (Huang et al., 2025)	2.8	88.3	91.4	87.1	70.9	84.4
MolmoAct-7B-D	2.8	87.0	95.4	87.6	77.2	86.8
GR00T N1.7 (NVIDIA, 2026c)	1.2	97.7	98.5	97.5	94.4	97.0
Cosmos3-Nano	3.2	86.8	87.6	86.4	82.0	85.7
Cosmos3-Edge	0.8	51.2	51.7	51.0	48.4	50.6
$\pi_{0.5}$	1.2	98.8	98.2	98.0	92.4	96.9
NORA-1.5 (Hung et al., 2025)	1.2	97.3	96.4	94.5	89.6	94.5
MolmoAct2	2.0	97.8	100.0	97.8	93.2	**97.2**

5.4.2 Trajectory quality and control rate

Success is paired with completion time, end-effector and joint path length, Cartesian and joint jerk, self-collisions, slip count, controller saturation, and safety interventions whenever the platform exposes them. Table 11 combines published deployment measurements with the Isaac request-shape calculation. Each latency is converted to a reciprocal policy rate under the request protocol named in the table caption.

Table 11: Deployment footprint and latency under each model's stated request protocol. Isaac uses an architecture calculation on one H100 SXM at 40% of dense-BF16 peak for three 1024×1024 images and ten Flow steps. GR00T N1.7 uses NVIDIA's full TensorRT result, Cosmos3-Nano uses the published single-Hopper PhyAI result, and Cosmos3-Edge uses NVIDIA's native-PyTorch PBR (NVIDIA, 2026c; Wang et al., 2026; NVIDIA, 2026a). Policy rate is reciprocal latency. Checkpoint size is the released repository footprint, except Isaac's FP8 parameter footprint.

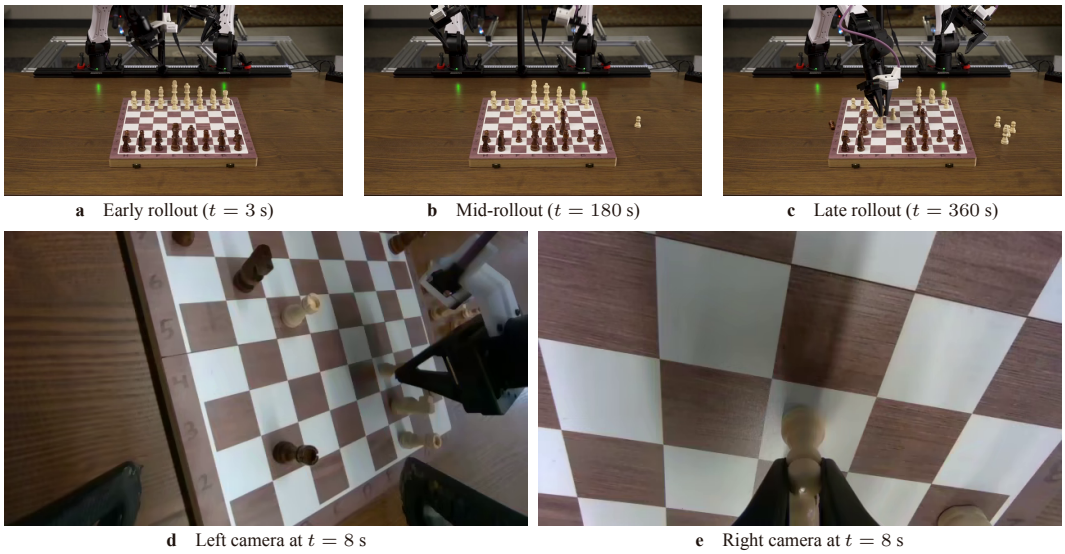
Model	Active parameters	Total parameters	GPU	Latency (ms)	Policy rate (Hz)	Checkpoint (GB)
Isaac 0.5	2.5B	36B	H100 SXM 80GB	70	14.3	36.0
GR00T N1.7 (NVIDIA, 2026c)	3B	3B	H100 80GB	27.9	35.9	6.93
Cosmos3-Nano-Policy-DROID (Wang et al., 2026; NVIDIA, 2026a)	16B	16B	Hopper GPU	1132	0.88	32.9
Cosmos3-Edge-Policy-DROID (NVIDIA, 2026a)	4B	4B	H100 SXM 80GB	1250	0.80	9.17

Across all control evaluations, the primary record contains task-balanced success or progress, task- and episode-level uncertainty, control frequency, and serving latency under fixed deployment settings. Individual success proportions use Wilson intervals (Wilson, 1927). Prediction failures, serving timeouts, invalid actions, dropped observations, controller faults, and safety interventions remain in the denominator, and held-out embodiments stay separate in Section 7.1.

5.5 End-to-end chess demonstration

The physical chess rollout shown in Figure 14 composes board perception, move selection, and physical execution in one loop. Figure 13 makes the orchestration explicit: embodied reasoning and VLA control are two routed modes of the same Isaac 0.5 checkpoint, not separately deployed high- and low-level models. At the start of each turn, Isaac 0.5 reads the board from the robot's camera views and reconstructs the current position. Move selection then runs in one of two declared modes. In the closed-book mode, the model selects a move from its learned chess knowledge. In the tool-assisted mode, Isaac 0.5 calls Stockfish through its tool interface and consumes the returned move. Both modes terminate at the same typed boundary: one exact source-to-destination command, such as move the pawn from e2 to e4.

The selected command is passed without paraphrase to the VLA control request together with the current camera observations and robot state. The control path grounds the named squares and piece in the scene, generates action chunks, and executes the pick-and-place motion. A new observation then starts the next turn. This division keeps the strategy boundary explicit: Stockfish, when enabled, chooses a move but never issues robot commands; the VLA always receives the exact move and performs its physical realization. None of the attempted baseline policies completed a successful trial after training on 500 hand-collected trajectories, leaving each at 0% success. Figure 14 documents the resulting full rollout.



a Early rollout ($t = 3$ s) **b** Mid-rollout ($t = 180$ s) **c** Late rollout ($t = 360$ s)

d Left camera at $t = 8$ s **e** Right camera at $t = 8$ s

Figure 14: The uncut YAM chess video shows multi-turn physical execution from early to late in the rollout. a–c, frames at 3, 180, and 360 seconds from the 6:13 end-to-end demonstration; **d–e**, synchronized left and right model-input camera views during manipulation in episode 366. Together they show board progression and the fixed three-camera observation interface.

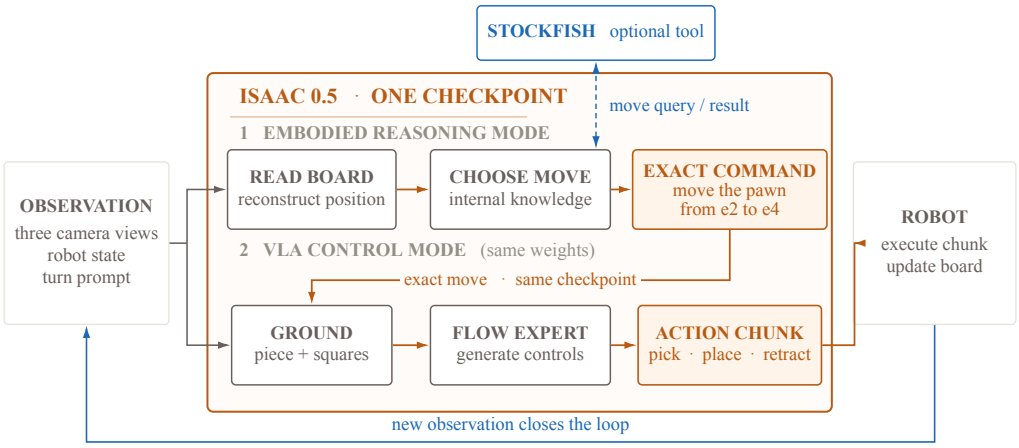


Figure 13: Isaac 0.5 closes the chess loop with one checkpoint. Camera views, robot state, and the turn prompt enter the shared model; its embodied-reasoning mode reconstructs the board and chooses a move from internal knowledge or an optional Stockfish call. The exact natural-language move becomes the typed input to the same checkpoint's VLA control mode, which grounds the board geometry and generates the action chunk.

5.6 Model finetuning elasticity

We measure how readily an open-weight policy acquires task-specific control from one episode and transfers that update to a nearby setting. We instantiate the benchmark on a physical chess-manipulation task with a fixed robot, board, camera arrangement, and action interface. The chess setting provides a controlled progression from repeated motor behavior to instruction-conditioned pick-and-place and then to multi-move execution, while keeping the objects and workspace recognizable across levels. Figure 14 shows the shared workspace and synchronized camera views.

The benchmark has three levels. In the *fixed-move* level, the board is reset and the robot repeatedly picks up the same piece and executes the same source-to-target move; this isolates grasp and motion adaptation. In the *notation-conditioned* level, the instruction specifies an arbitrary source and destination in coordinate chess notation, such as e2-e4, and the robot must identify and move the

corresponding piece. In the *offensive-line* level, the robot executes one prescribed offensive opening line together with a large yet finite set of branch variations selected by the preceding moves. This final level tests whether single-episode adaptation transfers across related multi-move sequences. It does not claim unrestricted chess play or independent strategy selection: the line and legal branch are supplied, and the evaluated behavior is their physical execution.

For each level we collect two complete expert episodes, e_A and e_B. The pair preserves the robot and action semantics while introducing a controlled shift: a perturbed piece pose or execution path for the fixed move, a different notation-specified move or board state for arbitrary pick-and-place, or a different variation within the same offensive line. We evaluate both directions: finetune on e_A and test on e_B, then restore the released checkpoint, finetune on e_B, and test on e_A. The comparison includes $\pi_{0.5}$, SmolVLA, MolmoAct2, GR00T N1.7, and Isaac 0.5 (Black et al., 2025c; SmolVLA Team, 2025; Fang et al., 2026; NVIDIA, 2026c).

Training implementations. We finetune each checkpoint in its native training stack rather than reimplementing every model in one framework. Isaac 0.5 uses the same internal training codebase and action interfaces used for its pretraining and release finetuning. For $\pi_{0.5}$ we use Physical Intelligence's official OpenPI training pipeline (Physical Intelligence, 2025); for SmolVLA we use the official Hugging Face LeRobot implementation (SmolVLA Team, 2025); for MolmoAct2 we use the released Allen Institute training repository and its pinned LeRobot integration (Fang et al., 2026); and for GR00T N1.7 we use NVIDIA's official Isaac-GR00T finetuning pipeline (NVIDIA, 2026c). We pin the code revision, base-checkpoint identifier, software environment, trainable-parameter set, optimizer, and learning rate for every run.

A shared converter materializes the same 10-Hz chess episodes in each framework's required dataset schema and verifies the observation timestamps, action ordering, units, and number of valid anchors after loading. Each native trainer retains its released action objective and normalization path, but one epoch always means one pass over those verified anchors with no replacement sampling or repeated short episodes. Predictions are decoded from the native model representation into the common benchmark action space before Equation 24 is evaluated. The primary ranking uses each release's recommended adaptation mode; full-parameter and parameter-efficient results are compared separately only in the matched-interface sensitivity analysis.

The native starting recipes are reported in Appendix A.2 (Table 16). The recipes are intentionally model-specific: the native-recipe comparison treats the released adaptation interface as part of the system being evaluated. We retain the recommended trainable-parameter mask, optimizer family, peak learning rate, weight decay, gradient clipping, regularization, and effective batch size, with the batch capped at $|\mathcal{I}_e|$ so an episode is never filled by duplicate samples.

We resample observations, state, and actions onto a common 10-Hz control grid. Every valid control time is an action anchor, and one epoch is one randomized pass over all valid windows in the finetuning episode. Adjacent windows overlap, so shuffling changes optimization order but does not make the examples statistically independent. Each policy keeps its native action objective and generated chunk length. Evaluation uses the first $H_{\text{eval}} = 10$ future action rows from every chunk, corresponding to one second; longer chunks are cropped only for scoring. Iterative policies use their released inference settings, with 10 Flow steps for $\pi_{0.5}$ and SmolVLA and 8 for MolmoAct2. We report the setting used by every other baseline rather than equating Flow steps with control rate.

Let $\mathcal{I}_e$ be the valid anchors in episode e, let $K_m = 10$ for a stochastic policy and $K_m = 1$ for a deterministic policy, and let M_t mask padded rows, unused action dimensions, and discrete channels at anchor t, as in Equation 7. For policy m with parameters θ_m, held-out continuous-action loss is

$$L_m(e;\theta_m) = \frac{1}{K_m|\mathcal{I}_e|} \sum_{t\in\mathcal{I}_e} \sum_{k=1}^{K_m} \frac{\left\| M_t \odot \left[\hat{a}^{(k)}_{\theta_m,t:t+H_{\text{eval}}-1} - a_{t:t+H_{\text{eval}}-1} \right] \right\|_2^2}{\|M_t\|_1}. \tag{24}$$

Actions use the benchmark's fixed normalization and coordinate convention; held-out episodes do not contribute normalization statistics. We report discrete gripper error separately, together with loss by action dimension and prediction offset.

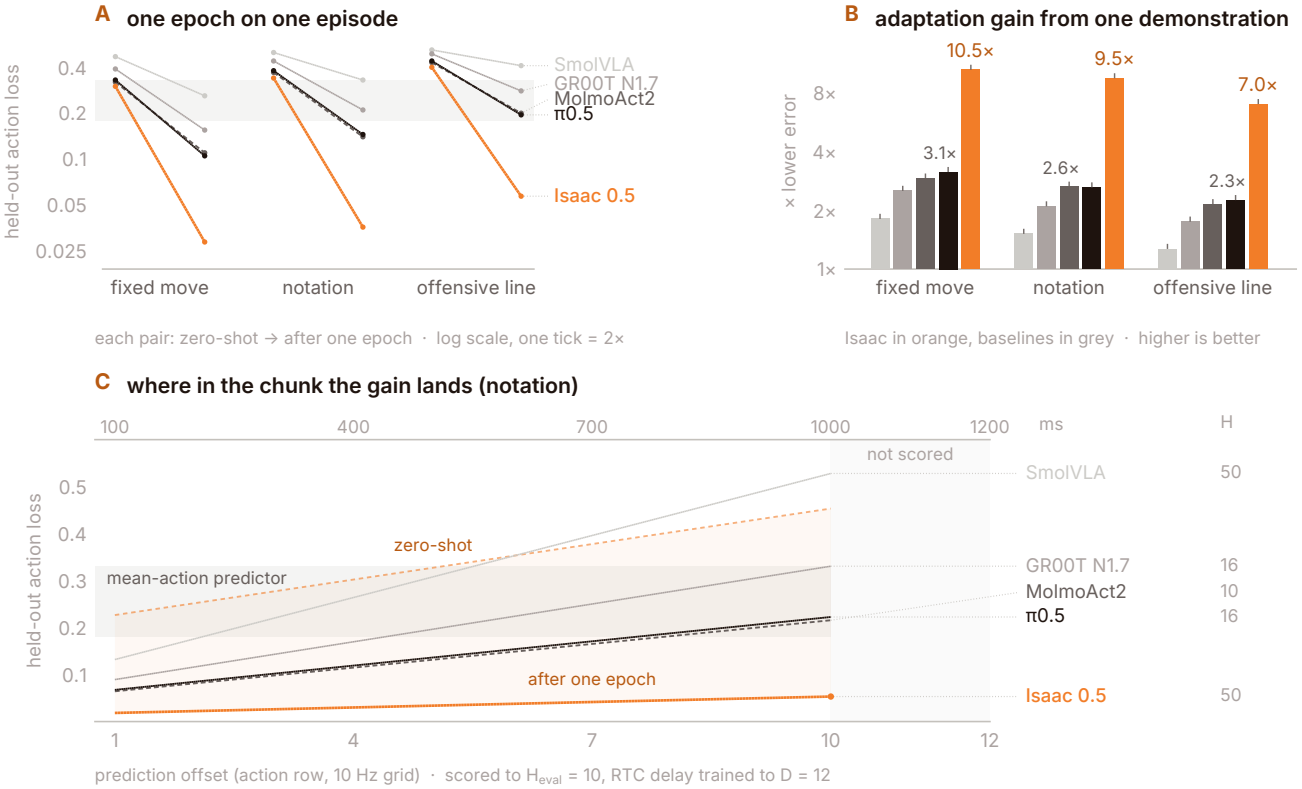


Figure 15: One-episode finetuning-elasticity diagnostic. **A**, held-out action loss on the same held-out episodes before and after one epoch, at each conditioning level; the grey band is the loss a mean-action predictor would score. **B**, the resulting loss-reduction factor $e^{\varepsilon^{\mathrm{FT}}}$ from Equation 26, with the standard error across optimization seeds; the orange figure is Isaac 0.5's factor, the grey one the best baseline's. **C**, post-finetuning loss by prediction offset for notation-conditioned transfer; the shaded wedge is Isaac 0.5's zero-shot-to-finetuned gap, the right-hand column gives each policy's native action-chunk length H, and rows 11–12 are trained against the simulated RTC delay $D = 12$ but are not scored.

Let $\theta_m^{(0)}$ be the released parameters and $\theta_m^{(A)}$ the parameters after one epoch on e_A. The directional adaptation gain is the log loss reduction

$$G_m(A \to B) = \log \frac{L_m(e_B; \theta_m^{(0)})}{L_m(e_B; \theta_m^{(A)})}, \tag{25}$$

and the model finetuning elasticity score for the pair is

$$\varepsilon_m^{\mathrm{FT}}(A, B) = \frac{1}{2} \left[G_m(A \to B) + G_m(B \to A) \right]. \tag{26}$$

This finite log response matches the sign and log-scale convention of the data elasticity in Section 4: positive values mean that one-episode finetuning lowers held-out loss, zero means no change, and negative values mean that the update hurts transfer. Because the adaptation budget is fixed, $\varepsilon_m^{\mathrm{FT}}$ is a one-episode score, not a derivative with respect to data volume.

We report zero-shot loss, post-finetuning loss, G_m in both directions, and $\varepsilon_m^{\mathrm{FT}}$ across task pairs and optimization seeds. Aggregation and uncertainty use task pairs, not overlapping action windows, as the resampling unit. Post-finetuning loss measures final action accuracy, whereas elasticity measures the improvement produced by finetuning. Offline action loss can also penalize a valid action that differs from the recorded expert, so this diagnostic complements rather than replaces closed-loop adaptation.

Figure 15 reports this diagnostic. One epoch on a single episode lowers held-out action loss for every policy at every conditioning level, so $\varepsilon_m^{\mathrm{FT}}$ is positive throughout. Isaac 0.5 records the largest

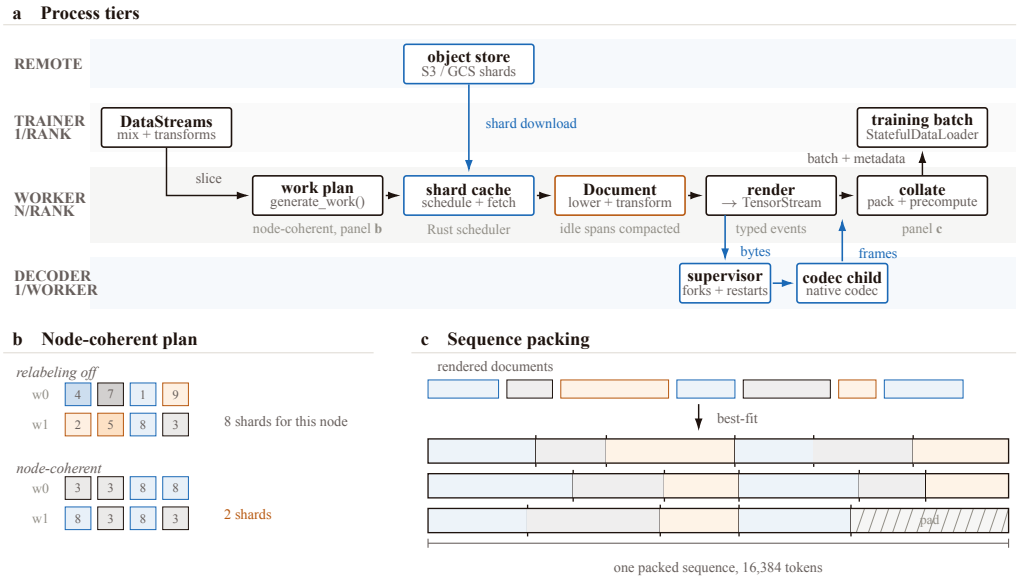


Figure 16: The training data plane. a, work moves from the trainer through isolated data-processing and codec tiers. **b**, node-coherent relabeling aligns worker shard demand so each shard is downloaded once per node. **c**, the collator best-fits variable-length documents into fixed-width sequences while preserving document boundaries.

loss-reduction factor at each level by a wide margin—10.5× on the fixed move, 9.5× under notation conditioning, and 7.0× on the offensive line—with $\pi_{0.5}$ next at 3.1×, 2.6×, and 2.3×. The corresponding elasticity gaps are 1.21, 1.29, and 1.13 against a standard error of 0.06 across optimization seeds, so the separation is decisive at every level rather than only in aggregate. Elasticity falls for every policy as the level requires the update to generalize further from the finetuning episode.

6 Training and inference systems

Training a percept-to-control model depends on a steady supply of useful tokens. General video varies in length and is expensive to decode. Robot trajectories are smaller and more structured, but their collection cost makes optimizer steps spent on idle spans especially wasteful. Sparse layers add another imbalance, since different token types request different amounts of routed work. We handle each problem at its source: typed compilation before training, predictive streaming and isolated decoding in the input pipeline, vision balancing before encoding, router balancing inside the model, and compacted execution after routing. Figure 16 shows where each of these runs. The worker also returns more than a batch: it precomputes the model-side tensors that do not depend on the forward pass, including multimodal rotary positions, modality and role masks, the per-action Flow context mask, and the per-modality token counts the MFU accounting needs. The training step therefore does not pay for them between forward and backward.

6.1 mHarmony

mHarmony is our strongly typed compiler for model inputs. It extends OpenAI's open-source Harmony renderer and parser rather than introducing another data format (OpenAI, 2025). mHarmony lowers video, text, images, state, task descriptions, and actions into TensorStream events with explicit modality, time, coordinates, embodiment, and loss semantics (Perceptron Team, 2026). Observed actions stay distinct from actions the model must predict, which lets the shared backbone train on general video, egocentric interaction, robot percepts, and action-supervised trajectories without treating an absent control field as a negative action label.

Typed prompting controls. We separate the semantic instruction x from a set of generation controls. mHarmony represents the latter as a typed system field

$$h \subseteq \{\text{point}, \text{box}, \text{polygon}, \text{clip}, \text{think}, \text{focus}, \text{tools}\}, \qquad p_\theta(y \mid x, h), \tag{27}$$

rather than relying on a natural-language description of the output format. This follows the control-code view of conditional generation introduced by CTRL (Keskar et al., 2019), but makes the control surface typed and multimodal. For supervised examples, the renderer infers spatial hints only from the assistant's target-side structured values—never from coordinates in the user prompt—and infers think from a target-side reasoning channel; a dataset may instead provide an explicit typed default. At inference, the requested hints are supplied explicitly. The spatial members select the answer grammar, think enables a reasoning channel, and focus and tools govern tool use. With no spatial member, the answer remains ordinary text; think may coexist with a spatial member. The same autoregressive model can therefore receive the same semantic query under different generation controls without changing the task instruction itself.

One pointing grammar. Points, boxes, and polygons share an integer coordinate grid. For an image of width W and height H, pixel coordinates are compiled as

$$(\widetilde{x}, \widetilde{y}) = (\text{round}(1000x/W)\,, \text{round}(1000y/H))\,, \qquad \widetilde{x}, \widetilde{y} \in \{0, \ldots, 1000\}. \tag{28}$$

A point contains one coordinate pair; a box contains ordered top-left and bottom-right pairs; and a polygon contains a variable-length hull. Collections group several targets, tracks bind one geometry type across timestamps, and clips represent a start time with an optional end time. During rendering, every spatial integer becomes one token from a reserved 1,001-entry coordinate group, enclosed by point-boundary sentinels and the geometry tag. Thus the hint controls the response family, while the shared grammar fixes its serialization and coordinate semantics across detection, grounding, video tracking, and screen-space actions.

One contract for training and serving. Training and inference both use mHarmony to render roles, channels, system controls, modality placeholders, coordinate and FAST token groups, the assistant-generation prefix, and action stop tokens. The contract also fixes image and video preprocessing. A versioned model-I/O manifest fingerprints the encoding and model-visible special tokens, then checks the vision processor and generation stop set against the checkpoint. A serialization change alters the sequence the model sees even when the user-facing example looks the same. Drift in role boundaries, patch counts, sampled frames, coordinate mappings, reasoning controls, loss masks, or terminal tokens pushes inference off the training distribution, and can truncate an action or run generation past the intended return. These failures usually present as poor model quality or high latency rather than as formatting bugs, which is why we bind the format to the checkpoint.

6.2 Sequence packing

mHarmony emits one event stream per document. The collator assigns document i a token length ℓ_i, a number of continuous-action chunks a_i, and a skip count s_i, then adds it to a working queue $\mathcal{Q}$. Packing starts when the queue reaches its document limit M, when it contains at least one context window of tokens, or when an old document reaches the repacking threshold S:

$$|\mathcal{Q}| \geq M \quad \vee \quad \sum_{i \in \mathcal{Q}} \ell_i \geq L \quad \vee \quad \max_{i \in \mathcal{Q}} s_i \geq S. \tag{29}$$

Here L is the context length, M is the document limit of the working queue, S is the repacking skip threshold, ρ is the minimum fill ratio, and A is the maximum number of continuous-action chunks in a sequence.

Each packing pass snapshots the queue and sorts documents by decreasing ℓ_i. An overlength document is split into an L-token prefix and a suffix; the prefix is packed immediately and the suffix returns to the queue. The pass then visits the sorted documents once. A partial bin b records its used tokens u_b and action chunks c_b. For document i, the feasible bins are

$$\mathcal{F}_i = \{b : u_b + \ell_i \leq L,\ c_b + a_i \leq A\}, \qquad b^* = \arg\min_{b \in \mathcal{F}_i} \big[L - (u_b + \ell_i)\big]. \tag{30}$$

The collator places i in b^*, the feasible bin that leaves the least unused token capacity. If $\mathcal{F}_i$ is empty, it opens a new bin. Sorting first and minimizing the residual gives best-fit decreasing packing, while the second feasibility constraint prevents a token-efficient bin from creating an oversized Flow batch.

After the pass, bins with $u_b/L \geq \rho$ are emitted. Documents in an underfilled bin have their skip counts incremented and return to $\mathcal{Q}$, where later arrivals may complete the bin. Once $s_i \geq S$, each subsequent arrival triggers another packing pass, but S does not force the underfilled bin to emit. A bin that cannot accept an otherwise token-compatible document solely because of the action-chunk limit is emitted rather than requeued. At end of input, a forced pass emits the residual bins. Padding is added only after a bin is selected for emission. Document order within each bin is unchanged, and complete documents retain their end-of-document markers; these markers define variable-length causal-attention segments, so packed documents cannot attend to one another. On the complete training mixture, the algorithm leaves 2% of tokens as padding.

Packing guarantee. For an IID admitted stream, bin assignment is semantically conditionally independent: conditioned on the metadata visible to the packer, it introduces no dependence in the remaining content. All departure from IID therefore reduces to the joint law of that low-dimensional metadata and can be measured from packing traces. For a fixed emitted multiset and matched stochastic draws, packed and unpacked execution produce the same hidden states, masked losses, and parameter gradients, including through null-route removal, grouped-expert compaction, and the global router objective. The forced flush preserves the surviving one-document marginal exactly, but neither the skip threshold nor best-fit assignment guarantees a small joint IID distance. Appendix B gives the proofs, counterexample, and padding bound.

6.3 Reliable video input

A Rust scheduler uses each worker's future sample order to plan shard downloads, cache births, and evictions in node-wide shared memory. Node-coherent relabeling preserves the plan's slot structure while rewriting cloud samples into node-local chunks, so workers on the same node request the same shards and fetch each one once. Repeated accesses separated by a long gap get their own cache lifetimes. This keeps a video-heavy mixture from turning remote storage latency into optimizer idle time. Cache hit rate alone does not show that, so we measure end-to-end data wait under the full training mixture.

Video decoding runs in a process separate from its DataLoader worker. Timeouts, codec errors, and fatal decoder failures, including segmentation faults we observed in FFmpeg, terminate only the decoder child; a supervisor starts a clean child and the worker resumes. Spatial transforms and TensorStream assembly sit outside that failure boundary. Long soak tests count decoded clips, replacements, and unrecovered failures, and measure latency by codec and the share of optimizer time spent waiting for input.

We compact long stationary spans only for robot sources with a validated idle signal. A source-specific threshold and minimum duration remove the interior of sustained idle runs while keeping short pauses and transition boundaries. The validation set includes holds, settling, and contact-rich motion, because a visually stationary end effector can still carry control information. The data record separates recorded robot hours from optimizer-consumed robot hours.

6.4 Distributed vision balancing

After temporal sampling and patchification, visual examples contain different numbers of tokens. At each step, local vision balancing recomputes a regularized optimal-transport plan from the current per-rank block sizes. Its transport cost favors assignments within the local NVSwitch domain, limiting slower cross-domain traffic. The resulting plan moves whole visual blocks across data-parallel ranks before encoding, reducing stragglers without changing an example's contents. Global router balancing acts later, after token routing, to regularize aggregate expert use, and exchanges only assignment statistics across ranks; token routes stay local. Figure 17 contrasts the two. We measure them separately: vision-token imbalance before encoding, expert-assignment dispersion after routing.

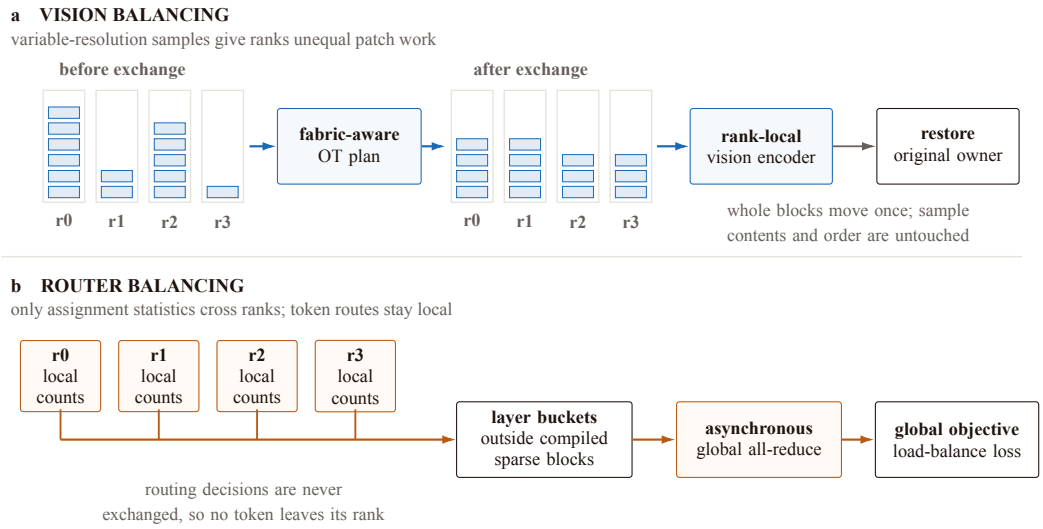


Figure 17: The two balancing mechanisms act at different points and on different things. **a**, vision balancing moves whole visual blocks across ranks before encoding; the exchange conserves patches, so ranks finish together without any example changing. **b**, router balancing leaves every token on its own rank and exchanges only per-layer assignment counts, which an asynchronous all-reduce turns into one global load-balance objective.

6.5 Sparse execution that realizes null routing

Null-expert routing changes which MLPs execute even though the router still returns a fixed top-8 decision. The kernel compacts real assignments into a contiguous prefix and leaves the null suffix out of the grouped matrix multiplication, so compile shapes stay static while the number of real expert calls varies by token. The continuous Flow/DiT path has a different execution graph and is compiled separately. In the linear-attention blocks, MoE-compatible GDN fusion combines normalization with the input projection without changing router scores or expert assignments (Yang et al., 2025b). A bounded fused token loss avoids creating a vocabulary-sized intermediate for unusually large packed batches.

Observed allocation. Null routing is strongly task- and phase-specific (Figure 18). Prompt allocation varies sharply across tasks, while decode tokens consistently use nearly all eight real expert slots. A separate video evaluation shows the same prompt/decode split. These counts measure routed-MLP invocations rather than end-to-end FLOPs or latency.

Across three review rounds, we inspected 142 candidate renderings and selected the 16 maps with the clearest localized contrast (Figure 19). In these examples, highly nulled context surrounds recognizable low-null structure such as airplanes, people, a skier, geometric and scientific diagrams, and answer-bearing document regions. The maps show routed compute allocation: a patch with fewer real routes receives fewer expert-MLP updates, while its residual-stream representation continues through the layer stack.

In a preregistered 512-sample paired intervention, forcing the two top-corner image tokens to take only null routes in decoder layers 3–39 changed the task-balanced official score by -0.227 percentage points relative to sham, with a fixed-cohort 95% bootstrap interval of [-2.205, +1.742] percentage points. The result did not pass the deployment safety gates, so the clamp was not expanded or deployed.

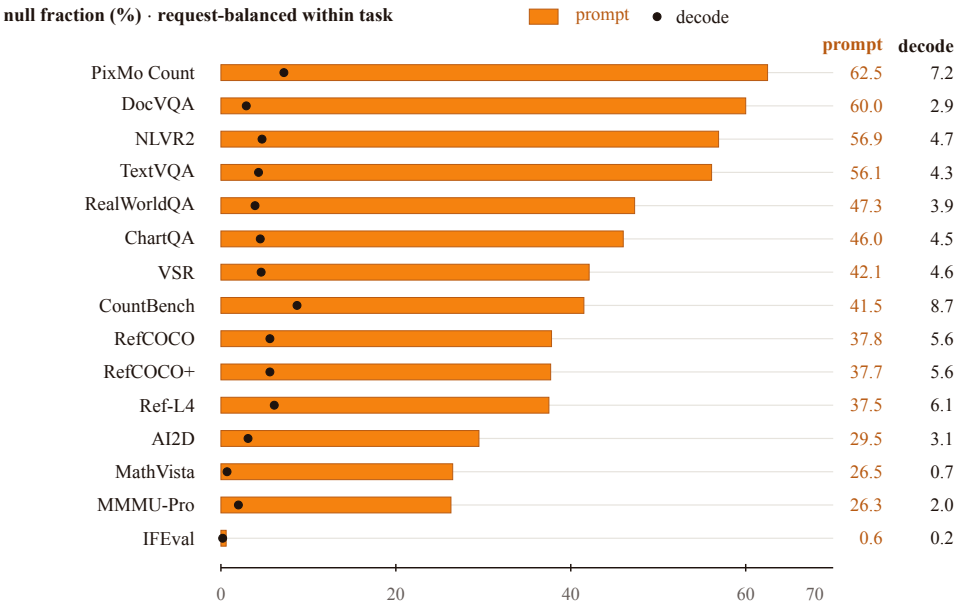


Figure 18: **Null routing varies sharply by task and is concentrated in prompt processing.**

6.6 Batched Flow training with grouped-query attention

For each continuous-action chunk, training draws $S = 4$ independent noise and Flow-time samples. Running the 36-block DiT once per sample would repeat the whole action-expert stack, while tiling the backbone context would copy its keys and values S times. We instead flatten the sample and chunk axes on the action side and fold the sample index into the cross-attention query heads. With B action chunks, action horizon N, context length C, N_{kv} key/value heads, and head dimension d_h, grouped-query attention (Ainslie et al., 2023) receives

$$Q \in \mathbb{R}^{B \times N \times (SN_{\text{kv}}) \times d_h}, \qquad K, V \in \mathbb{R}^{B \times C \times N_{\text{kv}} \times d_h}. \tag{31}$$

Each context head is shared by the S corresponding sample-specific query heads, and the output is then restored to the sample and chunk axes. Action-side activations scale with S, but the projected context keys and values, the context mask, and the packed-context indices stay at batch size B. For masked contexts, the system derives variable-length metadata once and packs the shared keys and values once for the full DiT evaluation. Each DiT block is compiled with a dynamic leading dimension so that different numbers of action chunks do not force a new static graph.

Figure 20 shows the resulting layout. This batching applies to independent training samples, not to the Flow integration steps that produce an action. Inference performs 10 sequential Euler updates, since each depends on the state from the previous one. The invariant backbone context is projected once and reused across them.

6.7 Optimized two-stage inference

We serve the policy as two co-located vLLM-Omni stages on a single accelerator. The first stage performs one multimodal prefill through the backbone and returns the post-final-normalization hidden states. The second stage runs the 0.58B DiT, conditions on those states, applies the 10 Euler updates, and returns an action chunk. The transition carries the full prefix activations and the action-context visibility mask. Prefix caching is therefore part of the execution contract: without the merged prefix, the action expert would receive only the final hidden-state row rather than the observation context.

The robot process owns the control clock, action queue, and random-number stream. Each request contains raw uint8 camera frames, proprioception in robot units, the task prompt, an anchor timestamp, explicit Flow noise, and the action rows already executed since the previous request. Those

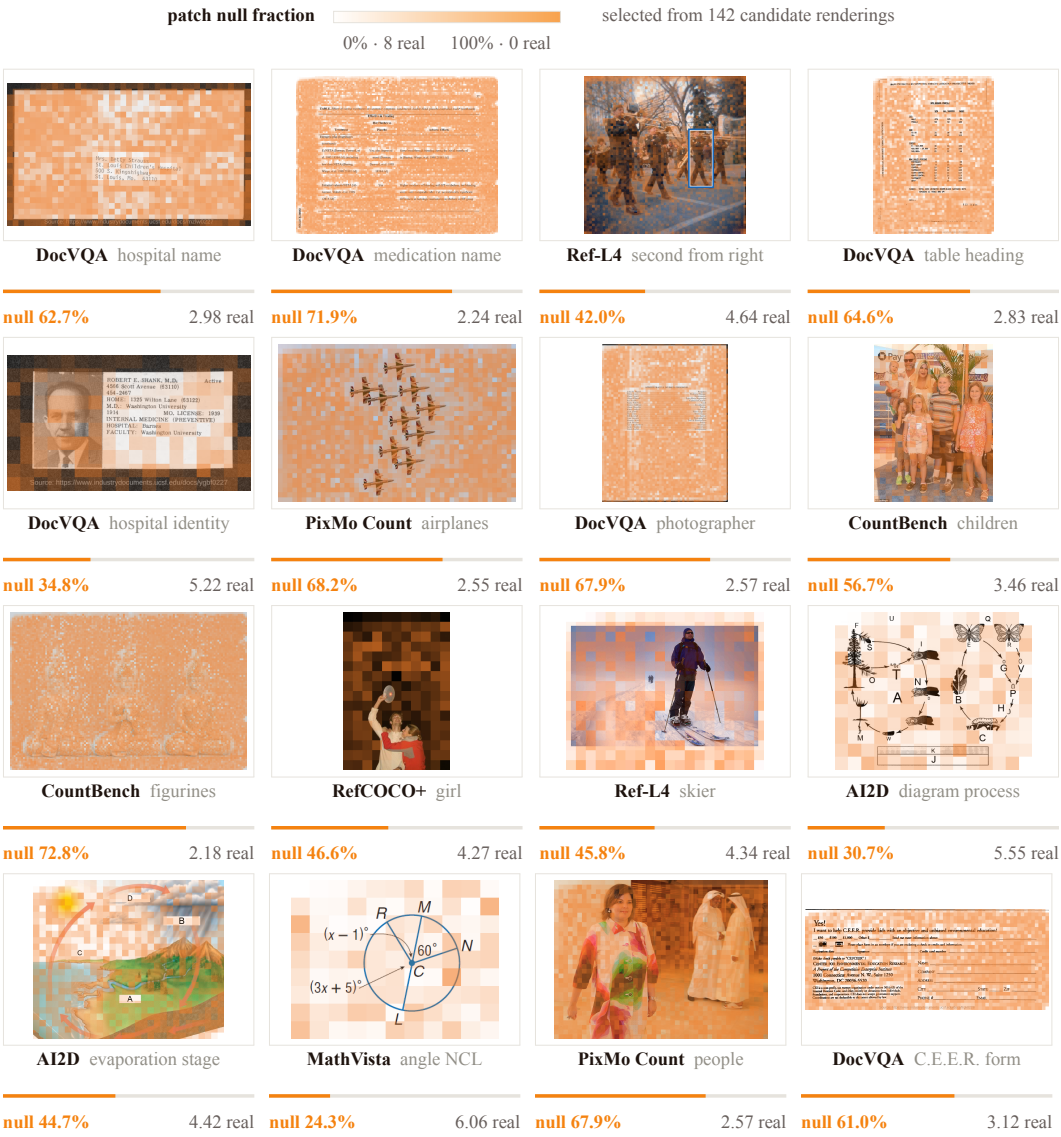


Figure 19: Real routes remain concentrated on recognizable visual structure in high-contrast examples. Orange opacity gives each image patch's null-selection fraction across 40 decoder layers and eight route slots: transparent patches take all eight real routes, while the darkest orange patches take none. People, countable objects, geometric and scientific diagrams, and answer-bearing document regions remain visible as lower-null structure against more strongly nulled context. The DocVQA panels include localized hospital names, medication text, a table heading, a photographer credit, and a C.E.E.R. form. Labels report the image-level mean null fraction and the corresponding mean number of real routes per patch and layer. The maps use recorded patch grids with nearest-neighbor expansion and no smoothing. The 16 panels were selected from 142 renderings across three review rounds and span seven tasks.

executed rows serve two purposes: they pin the real-time-chunking prefix, and they supply the causal action history of Equation 14. The server verifies the payload digest, renders the request through mHarmony, normalizes the state, and runs both stages. It returns a normalized action chunk, which the client converts to robot units once. Keeping rendering and normalization on one side of the process boundary makes the served input auditable against the training contract and avoids sending internal TensorStream objects over the wire. Figure 21 shows the resulting serving path and where it sits in the robot control loop.

We test this path at the boundaries where a plausible action can hide a serving error: event-for-event rendered-stream equality, exact multimodal rotary positions, vision-tower parity in fp32, explicit-

a COMPUTE AND CONTEXT SHAPES

METHOD	ACTION BATCH	CONTEXT K/V BATCH	DiT EVALUATIONS
Serial	B	B	S
Tiled context	SB	SB	1
Isaac 0.5 + GQA	SB	B	1

$S = 4$ independent noise and Flow-time draws per action chunk; only the tiled row pays for S copies of the context

b GROUPED-QUERY ATTENTION INSIDE THE DiT

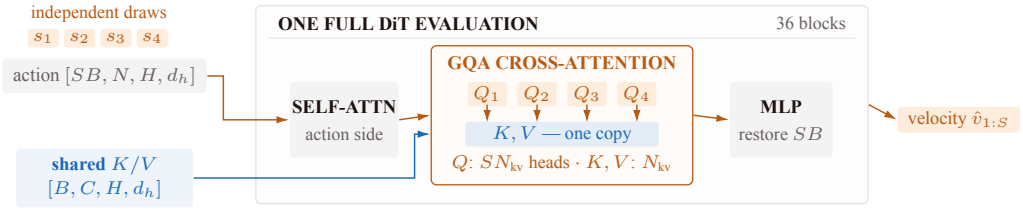
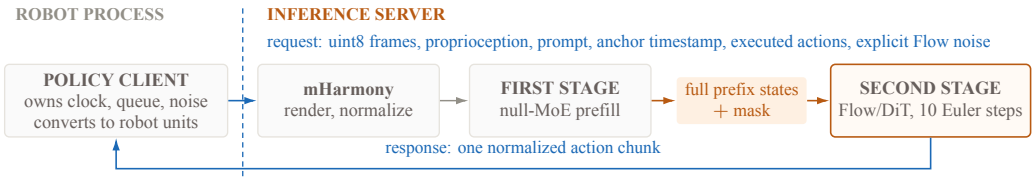


Figure 20: Batched Flow training through grouped-query attention. The $S = 4$ independent action-side samples share one batch-B context, so S full DiT evaluations collapse into one without tiling context: the queries carry SH heads while K and V keep H and stay at batch B. The 10 dependent Euler evaluations at inference stay sequential.

a REQUEST PATH



b CONTROL TIMELINE

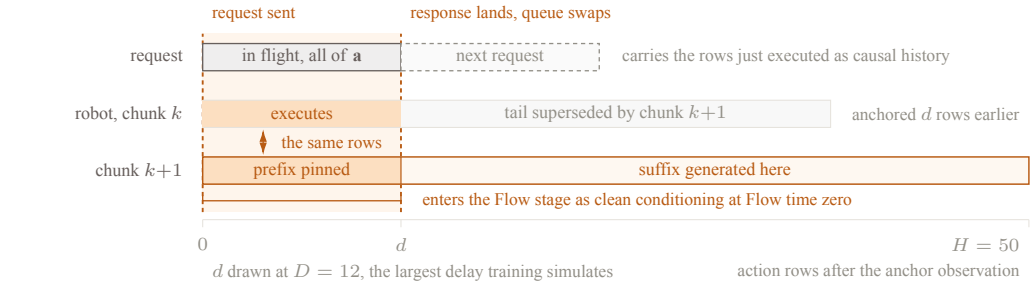


Figure 21: The serving path hides one inference inside the robot control loop. a, the serving contract. The robot process owns the control clock, the action queue, and the noise stream, and its request carries raw frames, proprioception, the prompt, an anchor timestamp, the executed actions, and explicit Flow noise. Rendering and normalization run on the server, so the served input stays auditable against the training contract, and only the client converts back to robot units. The two stages are co-located on a single accelerator, so the transition is an in-process handoff of the full prefix activations and the action-context visibility mask rather than a second hop. **b**, why the loop closes. The in-flight window spans everything drawn in **a**, not the two stages alone. The robot keeps executing chunk k through the d action rows that elapse while the request is in flight, so RTC pins the first d rows of chunk $k+1$ to exactly those commands and the 10 sequential Euler updates fill only the suffix; the same executed rows then re-enter the next request as causal action history. The axis is in action rows, which each robot's control rate converts to milliseconds. Training draws d from Equation 15 with $D = 12$ and $H = 50$; we do not fit this distribution to measured serving latency.

noise action parity, and closed-loop LIBERO evaluation. The visibility mask receives its own fail-closed check because a permissive mask lets the DiT attend to the trailing action marker that training excludes.

The latency path is short-prefill and launch-bound rather than throughput-bound. We compile the DiT step and batch independent Flow samples along the sample axis while sharing the backbone context keys and values. The backbone is then the largest remaining term. Null routing reduces routed-MLP work, but it does not reduce the vision tower, attention, shared-expert, or stage-transition costs, and the current serving configuration uses one tensor-parallel and one expert-parallel rank.

6.8 What we measure end to end

End-to-end training measurements use the complete model, production data mixture, and training topology, with timing started only after compilation, allocator growth, and input warm-up stabilize. The measurement window contains wall-clock throughput, optimizer-consumed packed tokens, source-native units, MFU, peak active and reserved memory, step-time quantiles, input wait, decode replacements, and failed samples. Component microbenchmarks are kept separate. End-to-end training reaches 24% MFU in the released hyper-sparse setting. A dense setting reaches 48% MFU, but its recipe and weights fall outside this open-source release.

Serving measurements bind the checkpoint to a complete request shape: hardware and precision, runtime revision, batch size, observation count and resolution, context length, generated FAST tokens or Flow horizon, and control frequency. The measurement record contains median and tail latency, weight memory, observed real routes per token, routed-expert FLOPs, and total executed FLOPs. These quantities separate the work skipped by null routing from the vision, attention, shared-expert, and continuous-control computation that remains.

7 Adaptation

Isaac 0.5 supports a staged adaptation path from a shared foundation checkpoint to an embodiment checkpoint and then to a task policy. Reasoning, discrete action, and continuous action all use the same backbone, while each robot configuration preserves its own observations, action semantics, normalization, timing, and safety limits.

7.1 Three adaptation regimes

The scope of an adaptation stage is distinct from the mechanism used to update the model. Zero-shot evaluation uses a supported interface with no task-specific weight updates. Parameter-efficient adaptation updates a declared parameter subset under matched training compute. Full-model adaptation may update the whole model when sensor semantics, action dimensions, or control timing differ from the supported interface. Embodiment and task adaptation can each use either parameter-efficient or full-model updates; the choice is reported with the resulting checkpoint.

During development, we explored several more structured update strategies, including expert-only training, detaching the action loss from the shared backbone, and freezing subsets of the vision–language model. In the task-adaptation settings we tested, these restrictions added recipe complexity without producing a better final policy. We therefore retain conventional end-to-end finetuning as the default: the shared backbone and action expert are updated jointly using the same trajectory representation and objectives used during training.

The retained Flow adaptation recipe predicts $H = 50$ action steps and draws $S = 4$ independent noise and Flow-time targets per action chunk. It uses AdamW with a peak learning rate of 5×10^{-5}, 300 warm-up steps, cosine decay, global-norm clipping at 1.0, and masks padded action rows from the Flow loss. For each run, we report the number of trajectories and accepted robot hours, control frequency, valid action anchors, optimizer steps, global batch size, optimizer-consumed action chunks, and equivalent dataset traversals; steps alone do not determine data exposure when variable numbers of trajectory examples are packed together.

Table 12 shows a sharp gap between zero-shot control and every adapted policy on LIBERO-Long. Zero-shot succeeds on 6% of trials. Updating only the Flow/DiT action expert raises success to 91%, and additionally updating the interface projections reaches 92.5%, just 0.5 percentage points below full finetuning at 93%. Rank-64 LoRA reaches 85.5%, below both narrower update scopes.

Table 12: LIBERO-Long adaptation strategies. The comparison varies update scope while holding the intended initialization, demonstrations, and evaluation protocol fixed.

Strategy	Trainable scope	LIBERO-Long success (%)
Zero-shot	None	6
Expert-only	Flow/DiT action expert	91
Frozen backbone	Action expert and interface projections	92.5
Rank-64 LoRA	Backbone adapters and full action expert	85.5
Full finetuning	Shared backbone, vision path, and action expert	93.0

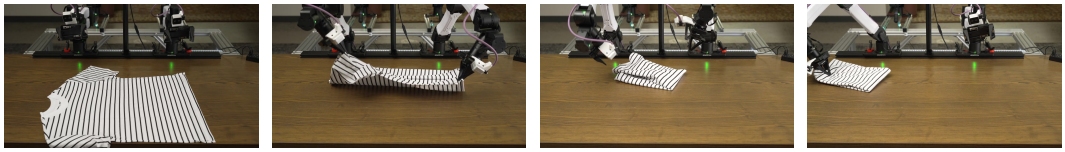
Most of the adaptation benefit in this comparison comes from the control-facing parameters. Full finetuning gives the highest success rate, but its margin over a frozen backbone is small. We retain full finetuning as the default recipe because it performs best here; the frozen-backbone result is a strong alternative that updates fewer parameters.

The adaptation record contains task-balanced capability, task-specific robot hours, training compute, and the number of updated parameters. Seen and held-out embodiments use the same robot-data budgets and scoring rules as a specialist trained from scratch and a foundation-model baseline.

7.2 New tasks

A task example may contain language, one or more observations, optional proprioception, and either FAST action tokens or continuous Flow targets. Video pretraining supplies visual and semantic structure; task-specific robot data supplies the action semantics and dynamics that video lacks. Adaptation curves use accepted robot hours and optimizer-consumed action tokens rather than endpoint scores alone.

As a small-data case study, we adapt the YAM-midtrained Isaac 0.5 checkpoint to T-shirt folding from five task demonstrations. Figure 22 shows the resulting policy moving the same garment through distinct intermediate states to completion.



a Initial layout **b** First fold **c** Compacting fold **d** Completed fold

Figure 22: A physical T-shirt-folding rollout passes through visibly distinct intermediate states. a–d, the initial garment layout, first bimanual fold, compacting fold, and completed state from the YAM deployment.

During adaptation, recomputing $q_{0.01}$ and $q_{0.99}$ separately for every task, as in MolmoAct2 (Fang et al., 2026), can be problematic: the $q_{0.99}$ bound places the largest 1% of actions outside the nominal range, where clipping saturates them. Those tail actions may be rare but task-critical, so discarding their magnitude can be especially detrimental for tasks that require occasional large commands.

Each of the 35+ released interfaces declares observations, proprioceptive fields, action representation, normalization statistics, controller timing, and safety limits. Every listed robot appears in the reported training mixture, and Appendix C gives its per-platform configuration inventory.

The paired YAM and SO-101 rollout in Figure 4 makes the consequence of this contract visible: the checkpoint and requested task stay fixed, while changing the embodiment string in the configuration prompt selects the embodiment-specific camera geometry, kinematics, timing, and action coordinates.

Discrete control maps supported action traces into the shared FAST vocabulary (Pertsch et al., 2025). Continuous control keeps embodiment-specific action dimensions and timing in the Flow/DiT interface (Lipman et al., 2023; Peebles & Xie, 2022).

8 Related work

Our work draws on three threads: experience that scales beyond robot trajectories, embodied models that turn such experience into action, and the conditional computation and systems used to train them together.

8.1 Video and egocentric pretraining for robotics

RT-1 trains a transformer policy on heterogeneous real-robot demonstrations, while RT-2 transfers vision-language representations into action prediction (Brohan et al., 2022; 2023). Open X-Embodiment and DROID broaden robot supervision across embodiments, sites, and tasks (Open X-Embodiment Collaboration, 2023; Khazatsky et al., 2024). Their action alignment is precise, but acquiring executable robot trajectories remains expensive.

Human interaction data offers greater scale at the cost of weaker action alignment. UMI records handheld-gripper trajectories without deploying the target robot (Chi et al., 2024). EgoScale, ACE-Ego-0, and EgoWAM study transfer from egocentric human activity to manipulation or world-action learning (Zheng et al., 2026; Li et al., 2026b;a); Ego4D supplies broad egocentric activity without a robot-control interface (Grauman et al., 2022). These projects establish that transfer is possible. We ask a different question: by how much does video change the teleoperation needed to reach a fixed threshold? We keep general video, passive egocentric video, pseudo-action supervision, handheld-gripper data, and synchronized robot trajectories as distinct regimes.

Generic video pretraining can improve a representation through ordinary visual or language objectives. The Isaac 0.5 objective instead predicts task-relevant semantic change. One training mixture cannot separate scale under the full recipe from the objective's incremental effect and its proposed mechanism, so we measure the three separately.

8.2 Generalist and open-weight robot models

Octo and OpenVLA make broad robot training and downstream adaptation accessible in open-weight models (Octo Model Team et al., 2024; Kim et al., 2024); SmolVLA emphasizes running on modest hardware (SmolVLA Team, 2025). MolmoAct and MolmoAct2 place spatial and action reasoning near control, and FAST compresses action trajectories into discrete tokens (Lee et al., 2025; Fang et al., 2026; Pertsch et al., 2025). Flow-based policies offer a complementary continuous parameterization: π_0 couples a vision-language backbone to a Flow action expert, $\pi_{0.5}$ studies open-world generalization, and $\pi_{0.7}$ trains a steerable generalist policy from demonstrations, autonomous experience, failures, and non-robot context (Black et al., 2024; 2025c; Physical Intelligence et al., 2026).

Recent proprietary systems push human-interaction pretraining further. GEN-1 is trained from scratch on more than half a million hours of human physical-interaction data and reports adapting to a new task and embodiment with about one hour of robot data; the same base model was later extended across substantially different end effectors (Generalist Team, 2026a;b). Dyna-2 trains a world-action model on more than one million hours of egocentric human video using future-video and future-action prediction, and reports that increasing human-video scale improves both held-out robot prediction and post-trained robot performance (Dyna Robotics, 2026). These results are author-reported under proprietary training and evaluation protocols, so their absolute values are not directly comparable to ours.

Table 13 places Isaac 0.5 against these systems on the axes that distinguish them: backbone sharing, discrete and continuous action interfaces, the supervision regimes each is trained on, and whether perception, embodied reasoning, and control are evaluated from one set of weights.

Isaac 0.5 combines a shared backbone with a FAST token interface and a dedicated Flow/DiT expert, and the same model is evaluated on perception, embodied reasoning, and closed-loop control rather than specialist vision-language-action performance alone. Primary comparisons match metrics, adaptation budgets, robot interfaces, and evaluation settings. Author-reported scores retain their native protocols.

Table 13: Control-recipe comparison. Isaac 0.5 and $\pi_{0.7}$ combine observation history, training-time RTC, and mistake conditioning; Isaac 0.5 additionally conditions on executed prior actions. Qwen-VLA and $\pi_{0.7}$ do not provide public weights and runnable code; the other listed models do (Octo Model Team et al., 2024; Kim et al., 2024; SmolVLA Team, 2025; Fang et al., 2026; Black et al., 2025c; Physical Intelligence et al., 2026; Physical Intelligence, 2025; Qwen Team, 2026e; Wu et al., 2026).

Model	Robot training scope	Obs. steps	RTC-trained	Prev. actions	Mistake modeling	Non-robot video	Flow expert	Open source
Isaac 0.5	35+ embodiments	1–3	✓	✓	✓	✓	✓	✓
$\pi_{0.7}$	multiple robots	$\leq$6 / camera	✓	×	✓	✓	✓	×
$\pi_{0.5}$	∼7 robots	1	×	×	×	×	✓	✓
Qwen-VLA	∼10 robots	1+	×	×	×	✓	✓	×
LingBot-VLA	9 robots	1	×	×	×	×	✓	✓
MolmoAct2	∼5 emb.	1	×	×	×	✓	✓	✓
SmolVLA	1 emb.	1	×	×	×	×	✓	✓
Octo	25 datasets	2	×	×	×	×	×	✓
OpenVLA	970k traj.	1	×	×	×	×	×	✓

The reported evaluation includes EmbSpatial-Bench and ERQA for embodied visual reasoning, and LIBERO for benchmark-specific robot adaptation (Du et al., 2024; Gemini Robotics Team, 2025; Liu et al., 2023). Simulation supplies breadth and repeatability; it does not replace physical rollouts.

8.3 Conditional computation and scalable multimodal systems

Chameleon shows that visual and textual tokens can share a sequence model (Chameleon Team, 2024). Sparse mixture-of-experts models increase total parameters while routing a fixed number of experts per token (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021; Zoph et al., 2022). DeepSeek-V3 demonstrates how routing, load balancing, kernels, and end-to-end systems evidence can be reported together at large scale (DeepSeek-AI, 2024). Zero-compute null experts let copies of one learned null logit occupy top-k router slots without invoking a routed MLP, so the number of real experts executed becomes input-dependent (Kilian et al., 2026). This differs from choosing fixed top-4 or top-8, and it does not by itself reduce weight memory, always-on computation, or end-to-end latency. Section 6 reports the complete-stack measurements.

9 Conclusion

Isaac 0.5 runs perception, embodied reasoning, and control through one sparse backbone, with null routing leaving 2.5B of 36B parameters active per token. The design bets that when every interface reads the same hidden states, supervision from cheap, action-free video can reach control. The evaluation record supports the bet. Isaac 0.5 posts the best available score on six of seven physical and temporal benchmarks and on five of seven spatial and embodied benchmarks (Section 5). One epoch on a single expert episode cuts held-out action loss by 7–10.5×, roughly three times the gain of the strongest baseline (Section 5.6). The training system sustains 24% MFU while it streams the million-hour video mixture (Section 6).

The central measurement is an exchange rate between the two data axes. Video needs grounding before it pays: at one teleoperation hour the loss-versus-video curve is nearly flat, and the slope settles near -0.21 loss for each 10× increase in video only from roughly 100 teleoperation hours onward. Past that point the exchange compounds. Scaling general video from 1,000 to one million hours cuts the teleoperation needed to reach the primary action-loss threshold from 5,884 hours to 28, a factor of 210.3× with an endpoint bracket of 83× to 300×. Video scale is a lever on robot-data efficiency within this recipe, not a replacement for action-grounded data.

We release the weights, the inference and adaptation stack, the evaluation suites and settings, and robot-specific normalization and deployment artifacts for 35+ embodiments represented in training.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, Liang Feng, Francesco Ferroni, Rama Govindaraju, Jinwei Gu, Siddharth Gururani, Imad El Hanafi, Zekun Hao, Jacob Huffman, Jingyi Jin, Brendan Johnson, Rizwan Khan, George Kurian, Elena Lantz, Nayeon Lee, Zhaoshuo Li, Xuan Li, Maosheng Liao, Tsung-Yi Lin, Yen-Chen Lin, Ming-Yu Liu, Xiangyu Lu, Alice Luo, Andrew Mathau, Yun Ni, Lindsey Pavao, Wei Ping, David W. Romero, Misha Smelyanskiy, Shuran Song, Lyne Tchapmi, Andrew Z. Wang, Boxin Wang, Haoxiang Wang, Fangyin Wei, Jiashu Xu, Yao Xu, Dinghao Yang, Xiaodong Yang, Zhuolin Yang, Jingxu Zhang, Xiaohui Zeng, and Zhe Zhang. Cosmos-Reason1: From physical common sense to embodied reasoning, 2025.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, 2025a.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking, 2025b.

Kevin Black et al. π_0: A vision-language-action flow model for general robot control, 2024.

Kevin Black et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization, 2025c.

Anthony Brohan et al. RT-1: Robotics transformer for real-world control at scale, 2022.

Anthony Brohan et al. RT-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

Wentao Cao, Po-Hsun Chang, Sicheng Chen, Xuanming Liu, and Haotian Lu. SCOPE: Symbolic constraint-guided active perception. In *ICML 2026 AI for Math Workshop*, 2026. URL https://openreview.net/pdf/978cc35e94f3fb4555c7fd223b8aabfc3576d4dd.pdf.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models, 2024.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots, 2024.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 2023.

Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. RoboNet: Large-scale multi-robot learning, 2019.

DeepSeek-AI. DeepSeek-V3 technical report, 2024.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better, 2025.

Mengfei Du, Binhao Wu, Zejun Li, Xuanjing Huang, and Zhongyu Wei. EmbSpatial-Bench: Benchmarking spatial understanding for embodied tasks with large vision-language models, 2024.

Dyna Robotics. Dyna-2: A 1-million-hour scaling law for world-action models. *Dyna Research*, August 2026. URL https://dyna.co/dyna-2.

Haoquan Fang et al. MolmoAct2: Action reasoning models for real-world deployment, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. BLINK: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision (ECCV)*, 2024.

Gemini Robotics Team. Gemini robotics: Bringing AI into the physical world, 2025.

Generalist Team. GEN-1: Scaling embodied foundation models to mastery. *Generalist AI Blog*, 2026a. URL https://generalistai.com/blog/gen-1.

Generalist Team. Towards machines with a thousand hands. *Generalist AI Blog*, 2026b. URL https://generalistai.com/blog/towards-machines-with-a-thousand-hands.

Google DeepMind. Gemma 4 model card. Apache-2.0 open-weight model checkpoint, 2026. URL https://ai.google.dev/gemma/docs/core/model_card_4.

Kristen Grauman et al. Ego4d: Around the world in 3,000 hours of egocentric video, 2022.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. ThinkAct: Vision-language-action reasoning via reinforced visual latent planning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Chia-Yu Hung, Navonil Majumder, Haoyuan Deng, Liu Renhang, Yankang Ang, Amir Zadeh, Chuan Li, Dorien Herremans, Ziwei Wang, and Soujanya Poria. NORA-1.5: A vision-language-action model trained using world model- and action-based preference rewards, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. CTRL: A conditional transformer language model for controllable generation, 2019.

Alexander Khazatsky et al. DROID: A large-scale in-the-wild robot manipulation dataset, 2024.

Mac Kilian, Armen Mkrtchyan, Luke Zettlemoyer, Akshat Shrivastava, and Armen Aghajanyan. Improving MoE compute efficiency by composing weight and data sparsity, 2026.

Moo Jin Kim et al. OpenVLA: An open-source vision-language-action model, 2024.

Jason Lee et al. MolmoAct: Action reasoning models that can reason in space, 2025.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding, 2020.

Baoyu Li, Xinchen Yin, Mengying Lin, Yixin Zhang, and Danfei Xu. EgoWAM: World action models beyond pixels with in-the-wild egocentric human data, 2026a.

Hao Li et al. ACE-Ego-0: Unifying egocentric human and robotic data for VLA pretraining, 2026b.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. ScreenSpot-Pro: GUI grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia (ACM MM)*, 2025.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. MVBench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *International Conference on Learning Representations*, 2023.

Liquid AI. LFM2.5-VL-3B: A better and faster vision-language model for the edge. Model release report, August 2026. URL https://www.liquid.ai/blog/lfm2-5-vl-3b.

Bo Liu et al. LIBERO: Benchmarking knowledge transfer for lifelong robot learning, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Loïc Magne, Anas Awadalla, Guanzhi Wang, et al. NitroGen: An open foundation model for generalist gaming agents, 2026.

NVIDIA. NVIDIA Nemotron Nano V2 VL, 2025. URL https://arxiv.org/abs/2511.03929.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. OpenMDW-1.1 model collection and technical report, 2026a. URL https://huggingface.co/collections/nvidia/cosmos3.

NVIDIA. Cosmos-Reason2-32B model card. NVIDIA Open Model License checkpoint, 2026b. URL https://huggingface.co/nvidia/Cosmos-Reason2-32B.

NVIDIA. Isaac GR00T N1.7: A foundation model for generalist robots. Model checkpoints and reference implementation, 2026c. URL https://github.com/NVIDIA/Isaac-GR00T.

Octo Model Team et al. Octo: An open-source generalist robot policy, 2024.

Open X-Embodiment Collaboration. Open X-embodiment: Robotic learning datasets and RT-X models, 2023.

OpenAI. OpenAI Harmony. https://github.com/openai/harmony, 2025. Open-source renderer and parser for the Harmony response format, Apache-2.0 license.

Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. Teaching CLIP to count to ten. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2022.

Perceptron Team. Isaac 0.5: An open foundation model for robot learning, 2026. Launch blog draft.

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

Physical Intelligence. openpi: Open-source models and packages for robotics. GitHub repository, 2025. URL https://github.com/Physical-Intelligence/openpi.

Physical Intelligence, Bo Ai, et al. $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities, 2026.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual-language-action model. In *Robotics: Science and Systems (RSS)*, 2025.

Qwen Team. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Qwen Team. Qwen3.5: Towards native multimodal agents. Apache-2.0 model checkpoint, February 2026a. URL https://huggingface.co/Qwen/Qwen3.5-27B.

Qwen Team. Qwen3.5-35B-A3B model card. Apache-2.0 model checkpoint, 2026b. URL https://huggingface.co/Qwen/Qwen3.5-35B-A3B.

Qwen Team. Qwen3.5-4B model card. Apache-2.0 model checkpoint, 2026c. URL https://huggingface.co/Qwen/Qwen3.5-4B.

Qwen Team. Qwen3.6-35B-A3B model card. Apache-2.0 model checkpoint, 2026d. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B.

Qwen Team. Qwen-VLA: Unifying vision-language-action modeling across tasks, environments, and robot embodiments, 2026e.

Naveen Sahi, Jeremy Dohmann, Armen Aghajanyan, and Akshat Shrivastava. SkillRater: Untangling capabilities in multimodal data, 2026.

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, 2017.

SmolVLA Team. SmolVLA: A vision-language-action model for affordable and efficient robotics, 2025.

Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. RoboSpatial: Teaching spatial understanding to 2D and 3D vision-language models for robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Chenghua Wang, Daliang Xu, Dongqi Cai, Duojin Sun, Hao Zhang, Haoze Qian, Huaiyuan Zhang, Jinshuo Cui, Kezhao Zhao, Longxi Gao, Mengwei Xu, Rongjie Yi, Tianyue Zhang, Weikai Xie, Xiyuan Tan, Xuanzhe Liu, Yingying Qin, Yiwen Lu, Yuan Yao, Yuezhi Zu, Yunhan Guo, and Ziqi Guo. PhyAI: Real-time physical AI at the edge, scalable rollouts in the cloud, 2026. URL https://arxiv.org/abs/2608.03682.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Shihan Wu et al. RoboCOIN: An open-sourced bimanual robotic data collection for integrated manipulation, 2025.

Wei Wu et al. A pragmatic VLA foundation model, 2026.

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule, 2025b.

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. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations (ICLR)*, 2025.

Ruijie Zheng et al. EgoScale: Scaling dexterous manipulation with diverse egocentric human data, 2026.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models, 2022.

A Elasticity definitions for other capability measures

Section 4 measures one elasticity: how fast the teleoperation required to reach a fixed offline action-loss threshold falls as general-video hours rise. The same definitions apply to other capability measures, weakly supervised control sources, and per-task or per-robot analyses.

Let P and W be accepted source-hours of purely perceptual and weakly supervised control pretraining, and let h be accepted task-specific teleoperation hours. For objective o, let $Q(P, W, h, o)$ be a higher-is-better capability measurement under fixed evaluation settings. The teleoperation required to reach threshold τ is

$$H_\tau(P, W, o) = \inf\{h : Q(P, W, h, o) \geq \tau\}. \tag{32}$$

We interpolate only between adjacent measured settings. If none reaches the threshold, we report it as not reached and do not extrapolate.

For two perceptual source-hour levels $P_a < P_b$ at fixed weak-control hours, the substitution factor is

$$S_\tau(P_a, P_b \mid W, o) = \frac{H_\tau(P_a, W, o)}{H_\tau(P_b, W, o)}, \tag{33}$$

and the local elasticity is

$$\varepsilon_\tau(P \mid W, o) = -\frac{\partial \log H_\tau(P, W, o)}{\partial \log P}. \tag{34}$$

A positive elasticity means the teleoperation requirement falls as perceptual pretraining grows. It says nothing about why.

To separate the semantic objective from the amount of source data, let o_{sem} denote training with the semantic future-percept objective and o_{ctrl} an otherwise matched run without it. The semantic-

objective ratio is

$$R_\tau^{\text{sem}}(P, W) = \frac{H_\tau(P, W, o_{\text{ctrl}})}{H_\tau(P, W, o_{\text{sem}})}. \tag{35}$$

This isolates the objective only when both runs share initialization, source examples, action losses, optimization schedule, and training compute.

Economic value depends on more than recorded hours. For source-specific accepted-hour prices c_P, c_W, c_h and training cost $C_{\text{train}}(P, W, h, o)$, the cost to reach τ is

$$C_\tau(P, W, o) = c_P P + c_W W + c_h H_\tau(P, W, o) + C_{\text{train}}(P, W, H_\tau(P, W, o), o)\,. \tag{36}$$

The least-cost mixture depends on collection, filtering, storage, and compute. Section 4.4 applies this definition to the fixed composition used in the planning analysis.

A.1 Grounded-scaling diagnostics

The primary 210× result is an endpoint ratio at $\tau = 2.50$. Table 14 exposes its dependence on the chosen loss threshold rather than treating one contour as a universal exchange rate.

Table 14: Endpoint sensitivity of the teleoperation substitution factor. Crossings use the monotone envelope and log-teleoperation interpolation in Section 4.2; no value is extrapolated.

τ	H_τ(1k)	H_τ(1M)	Ratio
3.00	25.1	4.4	5.7×
2.75	257.1	9.3	27.6×
2.50	5,884.3	28.0	210.3×
2.45	not reached	38.6	undefined

As a compact interaction diagnostic, we also fit

$$L = c + a(\log_{10} V - 4) + b(\log_{10} h - 2) + g(\log_{10} V - 4)(\log_{10} h - 2). \tag{37}$$

The centering matters: a and b are marginal slopes at $V = 10^4$ and $h = 10^2$, not uncentered main effects. On the perceptive grid, $a = -0.166$, $b = -0.312$, and $g = -0.044$. The residual calculation gives a naive cell-level standard error of 0.010 for g and $R^2 = 0.930$. These are conditional fit diagnostics, not inferential uncertainties: the grid has one run per cell and the video subsets are nested, violating the independent-error assumption. We therefore show the empirical per-rung slopes in the main figure and do not use the bilinear fit for low-teleoperation claims.

We compared this surface against the power-sum form

$$L = E + \left(AV^{-\alpha} + Bh^{-\beta}\right)^k. \tag{38}$$

Here $k = 1$ recovers an additive separable power law, so k is a direct non-separability diagnostic. Table 15 fits each model without the 1M-video column and evaluates on that column. The coupled model predicts that held-out column best, but its fitted floor reaches the nonnegative boundary and model choice was retrospective. We therefore use it only as a descriptive diagnostic; we do not interpret k as a structural elasticity or use the surface to replace empirical slopes where observations exist.

Table 15: Retrospective surface-model diagnostic. Models are fit without the 1M-video column and evaluated on that column. Model selection was not preregistered; these RMSEs are descriptive.

Model	Train RMSE	1M RMSE	k
Bilinear + interaction	0.116	0.164	–
Separable power sum	0.075	0.121	1.00
Coupled power sum	0.055	0.071	0.31

A.2 Model finetuning elasticity hyperparameters

Table 16 records the native starting recipe used for each open-policy baseline in the model finetuning elasticity evaluation of Section 5.6.

Table 16: Native finetuning recommendations used for the open-policy comparison. These are the official small-data or new-embodiment starting points, not a claim that one setting is optimal for chess. Published multi-thousand-step budgets are replaced by the common one-epoch exposure described in the text; the remaining settings come from the pinned native implementations (Physical Intelligence, 2025; SmolVLA Team, 2025; Fang et al., 2026; NVIDIA, 2026c).

Policy	Trainable parameters	Optimizer and learning rate	Other retained settings
$\pi_{0.5}$	Full model in OpenPI's small custom-DROID profile	AdamW; peak LR 2.5×10^{-5}; $\beta=(0.9,0.95)$; $\epsilon=10^{-8}$; weight decay 10^{-10}; global-norm clip 1	Batch 32; cosine schedule, originally 1k-step warmup to a 2.5×10^{-6} floor; EMA 0.99
SmolVLA	Action expert and state projection; vision encoder frozen	AdamW; LR 10^{-4}; $\beta=(0.9,0.95)$; $\epsilon=10^{-8}$; weight decay 10^{-10}; global-norm clip 10	Batch 64; cosine schedule, originally 1k-step warmup and 30k-step decay to 2.5×10^{-6}; 10 Flow steps
MolmoAct2	Rank-64 LoRA on the VLM path plus the full action expert, recommended for small or similar-embodiment data	VLM, vision, connector, and action-expert LRs all 5×10^{-5} under the released optimizer defaults	Global batch 64, device batch 2; dynamic sequence length; no packing; 8 Flow steps
GR00T N1.7	Multimodal projector and diffusion action head; language and visual backbones frozen	Released finetuning optimizer; LR 10^{-4}; weight decay 10^{-5}; 5% warmup	Global batch 64 by default (32 in the quick-start); state dropout 0.2; percentile normalization; default action chunk 16

B Sequence-packing guarantees

This appendix separates three properties that are easy to conflate: equivalence of the packed computation, preservation of the one-document marginal, and independence among documents assigned to the same bin. The first two hold under the stated contract; the third does not hold in general.

B.1 Packing invariance under null-expert sparse execution

The dense-attention part is standard; the additional claim is that null routing does not break packing equivalence even though it changes the executed sparse graph. Fix an emitted bin $P=(x_1,\dots,x_m)$ and condition on the same stochastic draws in the packed and unpacked evaluations. End-of-document markers give $M_P=\bigoplus_i M_i$. Let Π map the flattened token order of the unpacked logical batch to its packed order. Document-local embeddings, block-diagonal attention, and the cancellation of constant within-block rotary offsets give

$$H_P=\Pi H_{\text{unpacked}} \tag{39}$$

at the input to every sparse layer, provided the preceding layer has the same property.

Let R return the selected expert IDs and scores, let F_{null} be the complete null-expert MoE contribution after dispatch and scatter, and let A collect the auxiliary router objective and usage counts. Under the current contract—rowwise routing with stable tie-breaking, no capacity-based token dropping, and symmetric count reductions—the sparse layer satisfies

$$R(\Pi H)=\Pi R(H),\qquad F_{\text{null}}(\Pi H)=\Pi F_{\text{null}}(H),\qquad A(\Pi H)=A(H). \tag{40}$$

The first equality follows because router logits and stable top-k selection are tokenwise. The dispatcher sorts route rows by logical expert ID; compute experts precede the logical null slots, so

removing the null suffix retains the same multiset of real routes under Π. Real-route renormalization uses per-token indexed sums, each expert MLP acts independently on its assigned rows, and scatter restores the token map. The shared expert is tokenwise. Global load balancing, router z-loss, and expert-usage updates depend only on permutation-invariant sums and counts. A capacity rule that dropped routes by arrival order would invalidate the result.

Combining Equation 40 with the dense block argument proves Equation 39 by induction through the full backbone. The continuous-action expert separately restricts every action chunk to context with the same provenance. Therefore, for any additive masked loss head with document numerator $n_i(\theta)$ and denominator d_i,

$$\mathcal{L}_{\text{packed}}(\theta) = \frac{\sum_i n_i(\theta)}{\sum_i d_i} = \mathcal{L}_{\text{unpacked}}(\theta), \qquad \nabla_\theta \mathcal{L}_{\text{packed}} = \nabla_\theta \mathcal{L}_{\text{unpacked}}. \tag{41}$$

Thus the binning policy may determine which logical batch is formed, but concatenating a fixed batch changes neither its mathematical computation nor its gradient. Floating-point kernels may still differ at roundoff because a permutation can change reduction order.

B.2 Conditional independence and the metadata reduction

Let an admitted document be $X_i = (Z_i, Y_i)$, where Z_i contains every intrinsic field consulted by admission and binning—token length, action-chunk count, role and modality structure, and truncation class—and Y_i is the remaining content. Skip count is state derived only from the history of these fields; arrival order and randomized tie-breaking are exogenous. Assume the admitted documents are IID from P and the exogenous variables are independent of Y conditional on Z. Because the packing decision is measurable with respect to this information, the law Q_m of an emitted bin of arity m factorizes as

$$Q_m(dz_{1:m}, dy_{1:m}) = Q_m^Z(dz_{1:m}) \prod_{j=1}^{m} P(dy_j \mid z_j). \tag{42}$$

Hence the residual contents remain independent after conditioning on packer-visible metadata. Moreover, the packed law and the IID reference share the same conditional kernel, so integrating over Y gives

$$d_{\text{TV}}\big(Q_m, P^{\otimes m}\big) = d_{\text{TV}}\big(Q_m^Z, P_Z^{\otimes m}\big), \qquad D_{\text{KL}}\big(Q_m \,\|\, P^{\otimes m}\big) = D_{\text{KL}}\big(Q_m^Z \,\|\, P_Z^{\otimes m}\big). \tag{43}$$

The total departure from IID created by bin assignment is therefore exactly captured by the low-dimensional packing metadata. This makes an empirical ϵ auditable from packing traces. A bound relative to the pre-admission source distribution must additionally include filtering, truncation, and over-limit drops.

B.3 What the skip threshold does and does not prove

Once a queued document reaches $s_i \geq S$, every subsequent arrival triggers another packing pass. The current rule does not force its underfilled bin to emit, so s_i may continue to increase. The forced end-of-input pass, rather than S, supplies finite-input completion. Let $\widehat{P}_{\text{survive}}$ be the empirical law of documents in bins that pass all filters and limits, measured before concatenation, and let $\widehat{P}_{\text{packed}}$ be the law after the forced flush. Packing only permutes this multiset, and therefore

$$d_{\text{TV}}\Big(\widehat{P}_{\text{survive}}, \widehat{P}_{\text{packed}}\Big) = 0. \tag{44}$$

This is $\epsilon = 0$ for the surviving one-document marginal, not for the joint law within a bin. To see why no nontrivial distribution-free joint bound exists, take K equiprobable length classes and a capacity rule that admits exactly one document of each class. The packed K-tuple law Q_K is supported only on heterogeneous tuples, while the IID product assigns that support probability $K!/K^K$. Therefore

$$d_{\text{TV}}\big(Q_K, P^{\otimes K}\big) \geq 1 - \frac{K!}{K^K} \longrightarrow 1. \tag{45}$$

The skip threshold controls when packing is retried; it does not make the documents within a bin approximately independent.

B.4 Padding bound

Let $\mathcal{N}$ be the ordinary emitted bins, each of which satisfies $u_b/L \geq \rho$, and let $\mathcal{E}$ contain the action-limit and final-flush exceptions. With fixed capacity L, the overall padding fraction obeys

$$R_{\text{pad}} \leq \frac{(1-\rho)L|\mathcal{N}| + \sum_{b\in\mathcal{E}}(L-u_b)}{L\,(|\mathcal{N}| + |\mathcal{E}|)}. \tag{46}$$

C Robot support matrix

Table 17 summarizes the inventory. The support inventory contains 37 platform configurations, which resolve to 36 unique public platform identities after aliases are normalized. Of these identities, 31 map to real robot hardware or a hardware family, two are simulated settings, one is a gamepad layout, one is a physical capture rig, and one aggregates a multi-platform dataset (Wu et al., 2025; Liu et al., 2023; Magne et al., 2026; Chi et al., 2024; Dasari et al., 2019). Public names and kinds were checked against manufacturer, benchmark, or dataset documentation. This is a configuration inventory, not evidence of a released interface or physical checkpoint validation.

Table 17: Robot platform inventory. Counts summarize the platform types represented in the support matrix.

Contract kind	Count	Examples
Robot hardware or hardware family	31	AgiBot, Franka, Unitree, WidowX, and xArm
Simulated benchmark or setting	2	LIBERO; USC Cloth Simulation
Physical capture interface	1	UMI 4.0
Gamepad controller	1	Xbox / PlayStation layout from NitroGen
Multi-platform dataset	1	RoboNet

Table 18: Robot support matrix, part 1 of 3. One row per public platform identity; fields union all matching configuration contracts.

Platform	Kind	Config family	Action contract	Cameras	State	Hz
AgiBot G2	Hardware	agibot_g2	joint_gripper_40	3 views: head/left_wrist/right_wrist	71D (components)	variable
AgileX COBOT Magic	Hardware	agilex	joint_gripper_14; joint_gripper_eef_pose_26	1-3 views: external_high/front/head/left_wrist/right_wrist	14/26D (components)	variable
AgileX Split ALOHA	Hardware	agilex	joint_gripper_eef_pose_26	3 views: external_high/head/left_wrist/right_wrist	26D (components)	variable
AI2 AlphaBot 2	Hardware	alpha_bot	joint_gripper_eef_pose_28	3 views: head/left_wrist/right_wrist	28D (components)	variable
AIRBOT MMK2	Hardware	airbot	joint_hand_36	3 views: external_high/head/left_wrist/right_wrist	36D (components)	variable
Bimanual UFACTORY xArm	Hardware	xarm_bimanual	state_14 (ee)	1 view: primary	12D (components)	10
Bimanual YAM follower	Hardware	bi_yam	joint_gripper_14; joint_gripper_16	3-5 views: external_high/left_wrist/right_wrist/top/top_left/top_right	14/16D (components)	15/30/60
Everyday Robots helper robot	Hardware	everyday_robots	gripper_10; gripper_7; state_7	1 view: primary	7/8/9D (components)	3/10
Fourier GR-1	Hardware	gr1	joint_hand_29; joint_hand_padded_44	1 view: ego	29D (components)	variable
Franka arms (aggregate)	Hardware	franka	gripper_7; gripper_eef_pos_8; state_15; state_20; state_4; state_6; state_7; state_8 (ee/joint)	1-4 views: external_high/primary/secondary/unknown/wrist	2/7/8/13/14/21/23/24/25/28D (components/none)	5/10/20

Table 19: Robot support matrix, part 2 of 3. One row per public platform identity; fields union all matching configuration contracts.

Platform	Kind	Config family	Action contract	Cameras	State	Hz
Franka Panda	Hardware	franka	gripper_eef_pose_7 (ee); gripper_7; state_7 (ee)	2-3 views: primary/ secondary/wrist	8/14D (components)	15
Franka Research 3	Hardware	franka	joint_gripper_8 (joint)	6 RGB/depth streams: external_ gopro/external_ zed2_1/external_ zed2_2/shoulder/ wrist	8D (components)	variable
Galaxea R1 Lite	Hardware	galaxea	joint_gripper_14; joint_gripper_18	3-4 views: external_high/ head_left/head_ right/left_wrist/ right_wrist	14/16D (components)	variable
Galbot G1	Hardware	galbot	joint_gripper_16	3 views: external_ high/left_wrist/ right_wrist	35D (components)	variable
Hello Robot Stretch	Hardware	stretch	state_7 (ee)	1 view: primary	7D (components)	30
Kinova JACO	Hardware	jaco	gripper_4	2 views: primary/ wrist	21D (components)	10
KUKA LBR iiwa	Hardware	kuka	gripper_eef_delta_7; state_4 (ee)	1 view: primary	9/28D (components)	10/30
LEJU KUAVO 4 (RoboCOIN)	Hardware	leju	joint_hand_54	3 views: head/left_ wrist/right_wrist	54/90D (components)	variable
LeRobot SO-100 / SO-101	Hardware	so100_ so101	joint_gripper_6 (joint)	not recorded	6D (components)	variable
LIBERO (Panda simulation)	Simulation	libero	gripper_7	2 views: primary/ wrist	8D (components)	20
OpenArm follower	Hardware	openarms_ follower	joint_gripper_16	3 views: base/left_ wrist/right_wrist	16D (components)	variable
Rainbow Robotics RB-Y1	Hardware	rby1	mixed_delta_ absolute_joint_base_ gripper_torso_20 (joint)	5 RGB/depth streams: head/left_ wrist/right_wrist	22D (components)	variable
RealMan RMC-AIDA-L	Hardware	realman	joint_gripper_eef_ pose_28	3 views: external_ high/head/left_ wrist/right_wrist	28D (components)	variable
Rethink Robotics Sawyer	Hardware	sawyer	gripper_7; state_5 (ee)	1 view: primary	21D (components/ none)	10

Table 20: Robot support matrix, part 3 of 3. One row per public platform identity; fields union all matching configuration contracts.

Platform	Kind	Config family	Action contract	Cameras	State	Hz
RoboNet multi-platform collection	Multi-robot	various	state_5 (ee)	3 views: primary/ secondary	5D (components)	10
SoftStone Tianqing A2-D	Hardware	ruantong	joint_gripper_17; joint_gripper_34	3 views: external_ high/left_wrist/ right_wrist	17/41D (components)	variable
Trossen WidowX	Hardware	widowx	gripper_7	1 view: primary	7D (components)	5
UFACTORY xArm	Hardware	xarm	state_2; state_7; state_ 8 (ee/joint)	1-3 views: primary/ secondary/wrist	2/8/15/20D (components)	10
UMI 4.0 capture rig	Capture	umi	bimanual_ee_pose_ rotation6d_gripper_20 (ee)	1 view: head	20D (components)	variable
Unitree G1 (WBT setting)	Hardware	unitree_ g1_wbt	ee_hand_body_60	2-4 views: external_high/ external_high_2/ head_left/head_ right/left_wrist/ right_wrist	60D (components)	variable
Unitree G1 EDU U3	Hardware	g1	joint_hand_28; joint_ hand_30	1-3 views: external_high/ head/head_left/ left_wrist/right_ wrist	28D (components)	variable
Unitree Go1	Hardware	go1	state_12 (joint)	not recorded	68D (components)	10
Universal Robots UR5	Hardware	ur5	gripper_7	2 views: primary/ wrist	14D (components)	10
USC Cloth Simulation	Simulation	usc_cloth_ sim	state_4 (ee)	1 view: primary	none	10
Willow Garage PR2	Hardware	pr2	state_8 (ee)	1 view: primary	7D (components)	10
Xbox / PlayStation gamepad	Controller	generic	gamepad_binary17_ joystick4_21	1 view: screen_ capture	none	30/60

Tables 18–20 give one row per checked public platform identity. Duplicate Franka Panda entries are merged after reconciling their source schemas, camera layouts, state widths, and source rates.

D Null-routing trace and intervention protocol

The routing study used the step-30,000 learned-null checkpoint with 256 real experts, one learned physical null row expanded to 256 exchangeable logical null slots, and top-8 routing. Exact physical reconstruction compared each real router logit with the shared physical null-row logit; a tie counted as a real route. The 7,491-request multimodal trace contained 5,006 identity-eligible rows and 2,485 drifted rows. Drifted rows remained available for routing summaries after artifact validation but were excluded from quality, output, and timing joins. A separate video trace contained 2,144 identity-admitted requests. Request-balanced and token-weighted estimands were kept separate; Figure 18 reports the request-balanced values.

The image analysis reconstructed 7,554 processor-validated maps. Same-media examples had median spatial correlation near 0.972, compared with about 0.182 for unrelated images with the same grid shape. The gap indicates both stable spatial structure and content dependence. In two disjoint 64-request cohorts, the physical-router corner contrast was positive for every request at decoder layer

0 and retained the same sign through all 40 layers. Swapping corner embeddings with matched interior embeddings nearly reversed the layer-0 contrast; broadcasting one interior embedding to every image slot reduced the layer-0 routed-real residual to 0.005. At layer 3, the first full-attention layer, the residual returned to 0.646. Swapping the complete projected layer-3 attention output attenuated the heldout routed-real contrast by 77.4% and the physical router-margin contrast by 79.5% at the point estimates. Swapping decoder MRoPE labels left almost all of the contrast intact: final vision content seeds the early preference, and non-MRoPE full-attention slot/order dynamics regenerate it.