CAUSALCACHE: ALIGNING COMPUTATION WITH ACTION IN DIFFUSION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Accelerating diffusion robot policies requires identifying computation that can be reused without compromising control. Feature similarity offers a convenient signal, but overlooks how approximation errors are transformed before they reach the robot: the remaining denoising network can absorb or amplify a local perturbation, while robot kinematics can turn similar joint-command errors into different end-effector displacements and rotations. We therefore view cache allocation as preserving task-space fidelity rather than matching intermediate features. We introduce CAUSALCACHE, a training-free framework that compiles cache policies from these propagated task-space consequences. Single-block interventions recompute the full network suffix, and an end-effector pose (EE-Pose) objective measures the resulting position and orientation errors. A fixed-budget compiler minimizes the tail risk of intervention costs, while conditional pair refinement evaluates interacting changes in the context of the compiled policy. Calibration requires neither policy retraining nor task-success labels; deployment executes a frozen step–layer matrix with no online selection overhead. On RoboTwin2, CAUSALCACHE retains near-full task success with 76.7% block reuse, delivering 3.94× action-denoising and 2.83× complete-policy inference speedups. Eight calibration episodes perform comparably to forty, while SmolVLA experiments on a physical SO-101 demonstrate the advantage of task-space scoring across three manipulation tasks. These results establish propagated action consequences as a practical basis for allocating computation in diffusion robot policies.

1 INTRODUCTION

Diffusion policies generate continuous robot actions through iterative denoising (Chi et al., 2023; Lipman et al., 2023). Each action chunk requires repeated network evaluations along two axes: denoising steps and transformer depth. A ten-step, thirty-layer action diffusion transformer (Action-DiT) evaluates 300 blocks per chunk; we call this uncached policy Full. Reusing computation without changing policy weights or the sampling schedule offers a direct route to faster deployment.

Feature caching exploits the similarity of intermediate representations across denoising steps. Work on U-Nets, diffusion transformers, and robot policies has established substantial reuse opportunities (Ma et al., 2024b;a; Zou et al., 2025; Ji et al., 2026). The central question, however, is not simply whether two features resemble one another. It is *whether replacing one computation changes the action that matters to the robot*. Similarity is measured where an approximation is introduced; control depends on what remains after the approximation has propagated.

We identify two transformations between a cache decision and its physical consequence (Figure 1(a,b)). First, subsequent layers and denoising steps can absorb or amplify the local residual error. Second, forward kinematics (FK) maps joint-command error into end-effector displacement and rotation. These transformations make cacheability both *suffix-dependent* and *geometry-dependent*. Two equally sized feature perturbations can produce different final actions, and two equally sized joint perturbations can produce different gripper poses. Measuring final action error is therefore not enough: allocation must preserve end-effector geometry.

Matched-budget experiments show stronger robustness from suffix-aware allocation than local cosine selection. Geometry matters even at a fixed temporal schedule: near the compression boundary, EE-

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

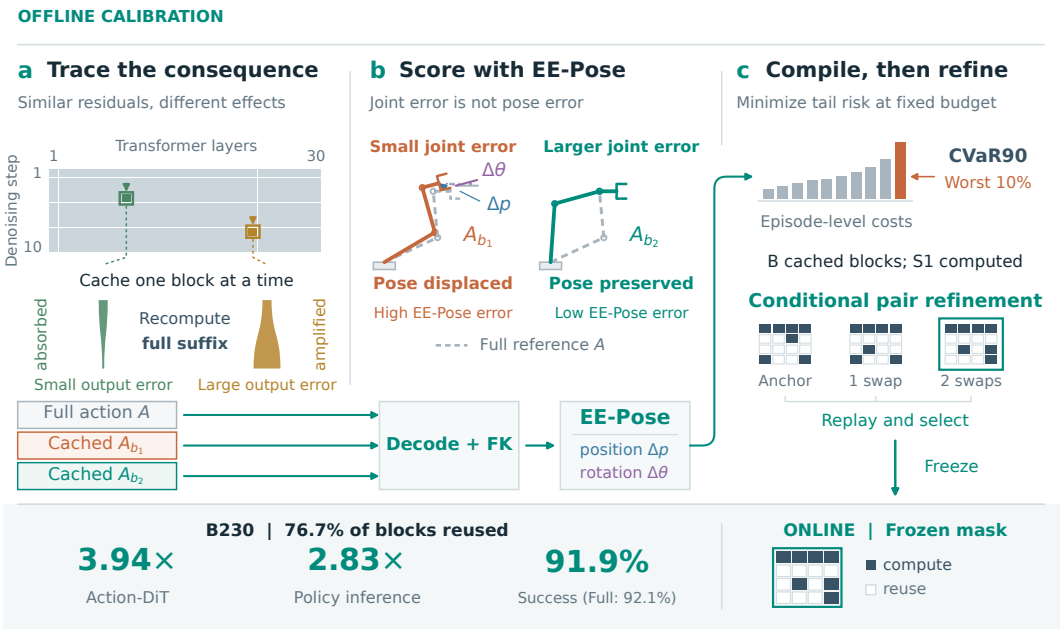

Figure 1: **CAUSALCACHE overview.** **(a)** Single-block suffix intervention. **(b)** EE-Pose scores position (blue) and rotation (violet) errors relative to Full. **(c)** CVaR90 compilation and pair refinement yield a frozen mask. Schematic workflow; performance: Table 1.

Pose improves full-benchmark success by 5.6 points over action mean squared error (Action-MSE) allocation through a single layer exchange (Appendix C). Broad temporal redundancy alone does not determine which computations to retain; their *propagated, task-space consequences* do.

We introduce CAUSALCACHE, a training-free framework that aligns computation with action. It intervenes on one block, recomputes the complete suffix, and scores end-effector pose discrepancy over the executed action prefix. An exact-budget compiler minimizes conditional value at risk (CVaR90: the worst-10% episode average). Calibration uses Full-policy outputs on saved rollout or demonstration observations, without success labels or retraining. Only the initialization step is protected; later refreshes are allocated from data.

Joint reuse changes the context in which each block operates. We therefore refine the compiled anchor through budget-neutral conditional swaps and exact pair measurements (Figure 1(c)), testing compensation or interference before selecting a new mask under the same risk objective. This combines global causal initialization with local, context-dependent refinement at fixed deployment cost. Deployment reads a frozen mask without an online router.

Across five runs on RoboTwin2's 50 tasks, CAUSALCACHE reuses 76.7% of blocks at 91.9% mean success, versus 92.1% for Full, with 3.94× Action-DiT and 2.83× complete-policy inference speedups. Eight calibration episodes perform comparably to forty, with singleton measurements reusable across budgets. Physical SO-101 experiments with three SmolVLA policies show that EE-Pose outperforms action-MSE allocation across all tested tasks and budgets.

Our contributions are threefold:

- **A consequence-based view of cacheability.** We connect network suffix propagation and robot kinematics, and use finite interventions to score what a cache decision changes at the policy output.
- **Task-space cache compilation.** An executed-prefix EE-Pose objective, an exact-budget CVaR compiler, and conditional pair refinement turn unlabeled calibration data into frozen step–layer masks with no online selection overhead.
- **Control and systems evidence.** Fast-WAM and SmolVLA demonstrate efficient control in simulation and on hardware. Matched-budget ablations isolate suffix scoring, geometry, and refinement; calibration studies establish data efficiency and cross-task transfer.

2 Related Work

Diffusion policies and inference acceleration. Diffusion Policy couples action denoising with receding-horizon control (Chi et al., 2023). Shortcut Models learn few-step generation (Frans et al., 2025), while Fast-WAM removes test-time future-video generation (Yuan et al., 2026). We instead reuse computation inside a frozen action denoiser, preserving its conditioning, action horizon, and denoising schedule.

Feature caching and error propagation. DeepCache reuses high-level U-Net features (Ma et al., 2024b). Learning-to-Cache learns input-invariant, timestep-dependent layer routes, while MD-DiT searches full, cached, and skipped block configurations (Ma et al., 2024a; Shen et al., 2024). ToCa considers token-level error propagation through attention-derived scores; TeaCache predicts output changes from timestep-modulated inputs (Zou et al., 2025; Liu et al., 2025a). The closest robotic precedent is Block-wise Adaptive Caching (BAC) (Ji et al., 2026): its Adaptive Caching Scheduler (ACS) optimizes feature similarity, and its Bubbling Union Algorithm (BUA) coordinates upstream updates with downstream feed-forward networks (FFNs) to mitigate caching-error propagation. We score complete-suffix action consequences and use anchor-centered pair interventions to measure coupled cache decisions.

Task-space fidelity and tail-risk allocation. Operational-space control describes robot behavior in end-effector coordinates (Khatib, 1987). EE-Pose brings this perspective to cache selection through position and orientation errors after forward kinematics. Critical Interval MSE emphasizes temporal criticality through interval selection and action alignment (Huang et al., 2026). We instead score same-input, executed-prefix geometry. CVaR (Rockafellar & Uryasev, 2000) aggregates calibration episode costs for exact-budget compilation, separating the choice of action metric from tail-risk protection.

3 CausalCache

3.1 Overview and cache allocation

CausalCache compiles a static cache matrix in four offline stages (Figure 1): *intervene* on individual blocks, *score* their final end-effector errors, *allocate* a fixed cache budget, and *refine* the matrix through budget-preserving swaps. Calibration uses saved observations from rollouts or demonstrations, with Full-policy outputs as references and no task-success labels or policy training. The resulting matrix is shared across deployment inputs; online inference only looks up whether to compute a block or reuse its residual.

An Action-DiT has N denoising steps and L layers, indexed by $b = (s, l)$ with $s = 1, \dots, N$ in execution order and $l = 0, \dots, L-1$. Given hidden state $h_{s,l}$, denoising time t_s, and conditioning c, the full layer map F_l produces residual $r_{s,l} = F_l(h_{s,l}, t_s, c) - h_{s,l}$. A binary mask $x \in \{0,1\}^{N \times L}$ chooses computation (0) or reuse (1):

$$h_{s,l+1} = h_{s,l} + \begin{cases} r_{s,l}, & x_{s,l} = 0, \\ r_{\rho_l(s),l}, & x_{s,l} = 1, \end{cases} \tag{1}$$

where $\rho_l(s) < s$ is the last real evaluation of layer l. Only real evaluation refreshes the stored residual. All scheduler steps remain unchanged. For an exact budget of B cached blocks, the feasible set is

$$\mathcal{X}_B = \left\{ x \in \{0,1\}^{N \times L} : \sum_{s,l} x_{s,l} = B, \quad x_{1,l} = 0 \; \forall l \right\}. \tag{2}$$

S1 initializes the caches; no later refresh interval is prescribed. The reuse fraction is $B/(NL)$. We write $x_b = x_{s,l}$ and use $S = \{b : x_b = 1\}$ interchangeably with its indicator mask.

3.2 Measuring suffix consequences

A *singleton intervention* caches one block. For saved input z (observation, conditioning, and initial action noise), let $A^x(z)$ be the model-normalized action chunk under mask x, and $A(z) = A^0(z)$

the Full reference. At each eligible block $b \in \mathcal{B} = \{(s,l) : s > 1\}$, the prefix remains Full, so $\rho_l(s) = s - 1$. We reuse that step's residual and recompute the entire suffix: subsequent layers, scheduler updates, and denoising steps. The resulting action is $A_b(z) = A^{\{b\}}(z)$, giving

$$e_{z,b} = r_{s-1,l} - r_{s,l}, \qquad \delta_b(z) = A_b(z) - A(z). \tag{3}$$

The first arrow in Figure 2 follows the local residual error $e_{z,b}$ through the Full suffix to its final action effect $\delta_b(z)$. Both residuals come from the Full trace; inputs and noise remain identical across interventions. With $N = 10$ and $L = 30$, this yields $(N-1)L = 270$ measurements per input.

3.3 Scoring task-space fidelity

EE-Pose measures end-effector position and orientation errors; Action-MSE averages squared differences over all normalized action channels, including grippers. Figure 2 contrasts small joint errors that displace the grasp (a) with larger compensating changes that preserve it (c), relative to Full (b). The Decode stage inverse-normalizes arm commands; $\Delta q_{z,b}$ denotes their intervened-minus-Full difference. We apply forward kinematics (FK) separately to Full and masked joint commands before comparing their poses. For command k and arm $a = 1, 2$, $p^x_{k,a} \in \mathbb{R}^3$ and $R^x_{k,a} \in \mathrm{SO}(3)$ denote tool center point (TCP) position and rotation matrix, with unsuperscripted Full references. We score only the H_z commands executed by Full, including termination truncation:

$$D^p_z(x) = \frac{1}{6H_z} \sum_{k=1}^{H_z} \sum_{a=1}^{2} \|p^x_{k,a} - p_{k,a}\|_2^2, \tag{4}$$

$$D^R_z(x) = \frac{1}{2H_z} \sum_{k=1}^{H_z} \sum_{a=1}^{2} \theta(R^\top_{k,a} R^x_{k,a})^2, \tag{5}$$

$$D^{\mathrm{pose}}_z(x) = \frac{1}{2}\left(D^p_z(x)/s_p + D^R_z(x)/s_R\right). \tag{6}$$

The factors $6H_z$ and $2H_z$ average three position coordinates and one squared geodesic angle θ (radians) per arm and command, respectively. Position and rotation scales $s_p, s_R > 0$ are mean episode-level singleton energies over calibration episodes and blocks, fixed across masks (Appendix A). TCP position uses $D^p_z(x)$ alone; EE-Pose adds orientation but not gripper aperture.

For episode e with chunks $\mathcal{Z}_e$, averaging over executed commands gives the measured mask loss and singleton cost:

$$\ell_e(x) = \frac{\sum_{z \in \mathcal{Z}_e} H_z D^{\mathrm{pose}}_z(x)}{\sum_{z \in \mathcal{Z}_e} H_z}, \qquad c_b(e) = \ell_e(\{b\}). \tag{7}$$

Why local similarity is insufficient. The top row of Figure 2 composes suffix propagation, decoding, and FK. Let $M_{z,b}$ be this map's Jacobian at Full and $\Delta\xi_{z,b}$ the stacked position and local rotation changes over the executed prefix. With W the positive diagonal weights implementing EE-Pose scaling and averaging, define $\|v\|^2_W = v^\top W v$ for a pose vector v. Locally,

$$\|\Delta\xi_{z,b}\|^2_W \approx e^\top_{z,b} G_{z,b} e_{z,b}, \qquad G_{z,b} = M^\top_{z,b} W M_{z,b} \succeq 0. \tag{8}$$

Thus $G_{z,b}$ is positive semidefinite, and its largest eigenvalue $\lambda_{\max}(G_{z,b})$ bounds the quadratic error:

$$0 \le e^\top_{z,b} G_{z,b} e_{z,b} \le \lambda_{\max}(G_{z,b}) \|e_{z,b}\|_2^2. \tag{9}$$

Equal local error norms can therefore yield different pose errors, depending on the gain and sensitive directions of $G_{z,b}$. Local cosine does not measure these suffix- and configuration-dependent effects. This analysis motivates our score; the compiler evaluates the nonlinear suffix and FK directly, without computing Jacobians. The expansion and proof appear in Appendix A.

3.4 Risk-aware mask compilation

With m equally weighted calibration episodes, the initial compiled mask $x^\star$ minimizes the tail risk of summed singleton costs:

$$C_e(x) = \sum_{b \in \mathcal{B}} c_b(e) x_b, \qquad x^\star = \arg\min_{x \in \mathcal{X}_B} \mathrm{CVaR}_\alpha\{C_e(x)\}_{e=1}^m. \tag{10}$$

Conditional value-at-risk (CVaR) at $\alpha = 0.9$ averages the worst 10% of episode costs, after summing blocks within each episode. HiGHS solves the resulting mixed-integer linear program; its standard CVaR formulation appears in Appendix A. Complete-mask replay instead applies the same CVaR to measured losses $\ell_e(x)$, replacing the singleton approximation $C_e(x)$.

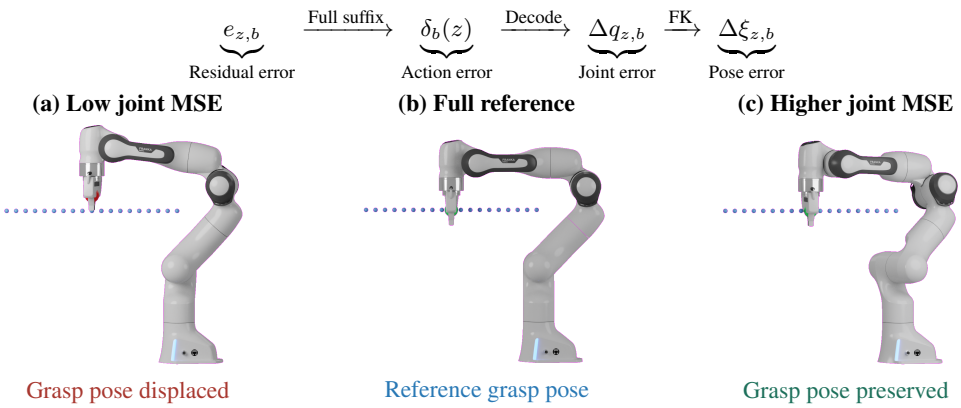

Figure 2: **Joint similarity versus grasp-pose fidelity.** Conceptual comparison: (a) small joint error, displaced grasp; (b) Full reference; (c) larger compensating error, preserved grasp.

3.5 Conditional interaction refinement

Refinement reallocates computation around the compiled mask while keeping its budget fixed. Let S be the cache set of $x^\star$, our *anchor*. A budget-neutral swap $i = (a_i \to b_i)$ restores $a_i \in S$ to real computation and caches $b_i \in \mathcal{B} \setminus S$, giving $S_i = (S \setminus \{a_i\}) \cup \{b_i\}$ with $|S_i| = B$. Compatible swaps have disjoint endpoints; S_{ij} applies both.

We propose swaps using singleton costs and replay the anchor, singles, and pairs with fixed calibration inputs, noise, and pose scales. *Conditional* selects the anchor or one measured swap; *Additive* also predicts two-swap actions by summing measured single-swap action changes:

$$\widehat{A}^{\mathrm{add}}_{ij}(z) = A^{S_i}(z) + A^{S_j}(z) - A^{S}(z). \tag{11}$$

Exact Pair replaces this prediction with the replayed action $A^{S_{ij}}(z)$, capturing compensation or interference between swaps. Both pair variants also admit the anchor and single swaps. Candidate actions pass through inverse normalization and nonlinear FK before episode-level EE-Pose CVaR90 scoring; each variant selects the lowest-risk mask. The complete method uses Exact Pair; Conditional and Additive are ablations. Appendix A.2 details interaction diagnostics and proposal enumeration.

Deployment uses the frozen mask without an online selector; Algorithm 1 summarizes the complete procedure.

4 Experiments

We first evaluate the control–latency trade-off in simulation and transfer to a physical robot, then isolate the contributions of suffix scoring, task-space geometry, and conditional refinement. Finally, we examine the compiled structure and the calibration required to obtain it.

4.1 Evaluation setup

Policies and benchmark. Released Fast-WAM checkpoints (Yuan et al., 2026) use ten steps and thirty Action-DiT layers. RoboTwin2 (Chen et al., 2025) comprises 50 tasks, two conditions (clean/randomized), and ten episodes per task-condition (1,000 per mask). SO-101 evaluation uses SmolVLA (Section 4.3); its action expert is the corresponding action-denoising module.

Calibration. All 270 RoboTwin2 singleton candidates share 777 chunks from 40 episodes: four tasks, two conditions, and five fresh-seed episodes per task-condition, without success filtering. Our primary masks protect only S1 and transfer to all 50 tasks, including 46 absent from calibration.

Baselines. Low NFE=6 sets the number of function evaluations (NFE) to six fully computed steps without cache reuse. Uniform shares a periodic refresh schedule across layers; LocalCos compiles

a static step–layer mask from adjacent-step residual similarity. BAC (Ji et al., 2026) optimizes feature-similarity schedules per submodule with ACS and coordinates attention–FFN refreshes with BUA. TeaCache (Liu et al., 2025a) accumulates predicted feature changes from timestep-modulated inputs to trigger online refreshes, with or without forced S10 computation (terminal protection). In contrast, CAUSALCACHE scores suffix-propagated action consequences offline. Appendix E details the configurations.

Controls and reporting. Comparisons fix checkpoints, conditioning, and execution horizons; paired rollouts share initial states, instructions, and random-number-generator (RNG) states. Caching retains ten denoising steps. We measure Action-DiT and complete-policy inference latency on a single NVIDIA RTX 4090, excluding simulation and rendering. Success averages tasks and scene conditions equally; core frontier and refinement results are five-run means. Appendix D details supplementary cohorts and timing.

4.2 SIMULATION: CONTROL–LATENCY TRADE-OFF

In the main frontier, CAUSALCACHE (Ours) denotes EE-Pose/CVaR90 compilation followed by Exact Pair refinement (Section 3.5). Across five runs, CAUSALCACHE reaches 92.0% success at B210 and 91.9% at B230, against 92.1% for Full (Table 1). B230 reuses 76.7% of Action-DiT blocks, reducing module latency from 330.9 to 83.9 ms (3.94×) and complete policy latency from 382.8 to 135.5 ms (2.83×), with a 0.2-point success difference from Full.

Table 1: **RoboTwin2 success (%) and inference latency.** DiT and Policy denote Action-DiT and complete policy inference.

Method	Clean	Random	Mean	Cache (%)	DiT (ms)	Policy (ms)
Full	93.0	91.2	92.1	0.00	330.9	382.8
LocalCos B180	77.4	75.4	76.4	60.00	151.7	202.0
Low NFE=6	90.0	89.6	89.8	0.00	198.8	249.0
Uniform	91.0	89.2	90.1	60.00	140.6	193.0
TeaCache (no protect)	90.2	90.4	90.3	70.24	101.0	152.9
LocalCos B150	90.8	91.2	91.0	50.00	189.2	239.4
TeaCache	91.4	91.0	91.2	69.79	101.6	153.5
BAC	91.8	91.0	91.4	60.00	140.8	192.8
LocalCos B120	92.2	91.0	91.6	40.00	222.3	271.9
CAUSALCACHE B245	78.6	76.8	77.7	81.67	66.2	117.7
CAUSALCACHE B242	90.2	89.0	89.6	80.67	69.8	121.3
CAUSALCACHE B240	92.2	89.6	90.9	80.00	72.1	123.6
CAUSALCACHE B230	92.8	91.0	**91.9**	76.67	83.9	135.5
CAUSALCACHE B210	92.8	91.2	**92.0**	70.00	107.8	159.5

Pareto advantage. At B230, CAUSALCACHE improves mean success over BAC by 0.5 points with 40.4% lower Action-DiT latency, and over TeaCache by 0.7 points with 17.4% lower latency. It also exceeds every other accelerated baseline shown in both success and speed (Table 1). Together with Full, CAUSALCACHE forms the empirical Pareto frontier (Figure 3). B240 extends the frontier toward faster inference; the steep B242–B245 segment marks the loss of control quality near the reuse limit. Consequence-based allocation thus reduces computation at high success without changing the denoising schedule or adding an online selector.

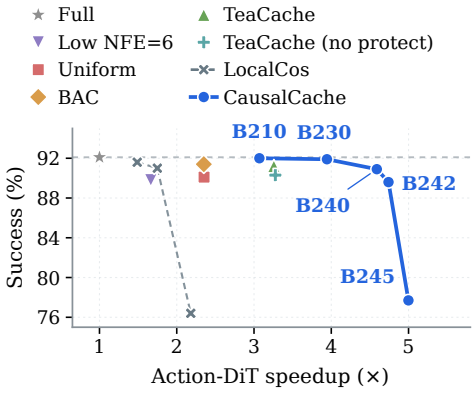

Figure 3: **RoboTwin2 Pareto frontier.**

4.3 REAL-WORLD TRANSFER



Figure 4: **SO-101 setup.**

Platform and protocol. We evaluate three task-specific SmolVLA (Shukor et al., 2025) policies on a three-camera SO-101 (Figure 4). Policies are fine-tuned for 20,000 steps with batch size 32, then frozen. Each 10×16 cache matrix uses eight demonstration episodes without success labels, protects only S1, and uses no refinement. Full, Action-MSE, and EE-Pose share the checkpoint, cameras, instruction, and execution horizon. Twenty trials per task and configuration give 540 physical trials; Appendix E.2 details the protocol.

Table 2: **SO-101 success (%) and action-expert speedup relative to Full.**

Task completion						
	**Pick-place** Grasp and place blue in the box, then repeat for red.					
	**T placement** Grasp and place the T in its prescribed orientation.					
	**Stacking** Stack blue on red; it must stay stable after release.					

Method	Reuse	Pick-place	T placement	Stacking	Mean	Expert speedup
Full	0%	80	80	80	80.0	1.00×
Action-MSE B40	25%	70	70	75	71.7	1.29×
EE-Pose B40	25%	**80**	**80**	**80**	**80.0**	1.28×
Action-MSE B60	37.5%	70	65	65	66.7	1.51×
EE-Pose B60	37.5%	**75**	**80**	**80**	**78.3**	1.49×
Action-MSE B80	50%	65	60	60	61.7	1.84×
EE-Pose B80	50%	**70**	**75**	**70**	**71.7**	1.83×
Action-MSE B90	56.25%	60	50	55	55.0	2.07×
EE-Pose B90	56.25%	**65**	**70**	**60**	**65.0**	2.07×

Physical task completion. EE-Pose matches Full's 80.0% mean success at 25% reuse and outperforms Action-MSE on every task at all four budgets (Table 2). The largest gain is on T placement at B90: 70% versus 50%, consistent with its orientation requirement. Appendix Figure 8 illustrates improved stacking alignment with EE-Pose compared with Action-MSE.

Inference efficiency. At B80, EE-Pose delivers 1.83× action-expert and 1.57× complete-model speedup on RTX 3090 (Appendix Table 10). Compared with Action-MSE's 1.84× expert speedup, it gains 10 success points at essentially the same inference cost. These matched-cost gains support geometry-aware compilation beyond Fast-WAM, on a different policy architecture and physical robot.

4.4 COMPONENT ABLATIONS

We isolate the contributions of suffix scoring, task-space geometry, and conditional refinement.

4.4.1 CONSEQUENCE OVER SIMILARITY

To isolate suffix scoring from the complete method (Table 1), we compare LocalCos and Causal Unary at matched budgets, without EE-Pose or refinement. LocalCos ranks adjacent-step residual similarity; Causal Unary caches one block, recomputes the full suffix, and scores final action discrepancy. Success improves from 76.4% to 91.1% at B180 and from 28.9% to 91.4% at B200.

The gap grows from 14.7 to 62.5 points as reuse increases, while Causal Unary maintains approximately 91% success. This pattern is consistent with the suffix-propagation component of Equations 8–9: local perturbations can be attenuated or amplified depending on their direction and the remaining computation. Measuring these propagated consequences provides a strong allocation principle before adding the task-space geometry examined next.

4.4.2 TASK-SPACE FIDELITY AND TAIL RISK

Position, orientation, and risk. With CVaR90 fixed, mean success rises from 63.33% with Action-MSE to 65.83% with TCP position and 74.17% with EE-Pose (Table 3), an 8.33-point gain over position alone. For EE-Pose, CVaR90 adds 2.33 points over Mean aggregation (71.83%). The gains reflect task geometry: on bottle disposal, adding orientation raises success from 41% to 68%. Turning a switch uses a closed gripper to push, so position and pose obtain similar success (68% and 69%).

Table 3: Success (%) on the six hardest RoboTwin2 tasks, averaged across five cache budgets. All objectives use CVaR90 except EE-Pose Mean.

Task	Action MSE	TCP position	EE-Pose Mean	EE-Pose CVaR90
Hanging mug	37	**52**	50	51
Turn switch	**78**	68	69	69
Place can in basket	68	77	83	**87**
Pick diverse bottles	72	83	88	**92**
Put bottles in dustbin	54	41	64	**68**
Place mouse pad	71	74	77	**78**
Mean	63.33	65.83	71.83	**74.17**

The EE-Pose advantage over Action-MSE also holds on SO-101 with SmolVLA (Section 4.3; Table 2), extending task-space scoring to physical control.

4.4.3 CONDITIONAL INTERACTION REFINEMENT

Refine in the deployed context. Across five runs on all 50 RoboTwin2 tasks, Conditional and Exact Pair both improve mean success from 91.3% to 92.0% at B210 and from 91.1% to 91.9% at B230 (Table 4), without additional deployment computation. At B240/B242/B245, both match the anchor's mean success. Conditional refinement thus improves practical operating points while maintaining success at the higher reuse levels reported here.

Table 4: **Conditional refinement.** Success (%) and mean loss interaction.

Budget	Anchor	Conditional	Exact Pair	$\mathbb{E}[h_{ij}]$
B210	91.3	**92.0**	**92.0**	-0.009
B230	91.1	**91.9**	**91.9**	-0.017
B240	90.9	90.9	90.9	$+1.556$
B242	89.6	89.6	89.6	$+1.534$
B245	77.7	77.7	77.7	$+1.158$

From compensation to interference. For two budget-neutral swaps i and j, h_{ij} measures how much more or less EE-Pose error they produce together than predicted by adding their separate changes to the anchor (Equation 19). Negative h_{ij} means less error than predicted (compensation); positive h_{ij} means more (interference). The small negative means at B210/B230 and positive means at the higher budgets in Table 4 motivate selective joint updates. Exact Pair selects by measured full-mask CVaR90, not the sign of h_{ij}, and retains the anchor as a candidate. This uses joint measurements to test promising changes before committing them to the deployed mask (Appendix B).

4.5 CALIBRATION EFFICIENCY AND TRANSFER

Budget-reusable calibration. Singleton measurements are reused across budgets; only allocation is re-solved.
Figure 5 shows 50-task means and min–max ranges across calibration draws at a fixed evaluation budget. Eight episodes achieve 91.70%, only 0.02 points below forty with 80% fewer trajectories. The early plateau supports identifying useful reuse structure from few episodes, without relearning the policy.

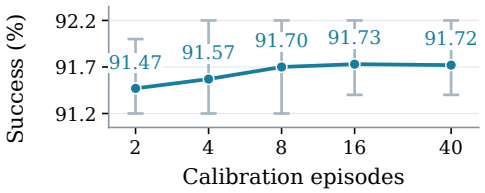

Figure 5: **Calibration data efficiency.**

Transfer beyond calibration tasks. On the 46 task identities absent from calibration, EE-Pose B245 achieves 79.78% versus 74.89% for action MSE, a 4.89-point gain with paired bootstrap 95% interval $[1.63, 8.26]$. One compiled matrix serves multiple tasks without task-specific recompilation; Appendix D.1 reports one-time setup costs and deployment amortization.

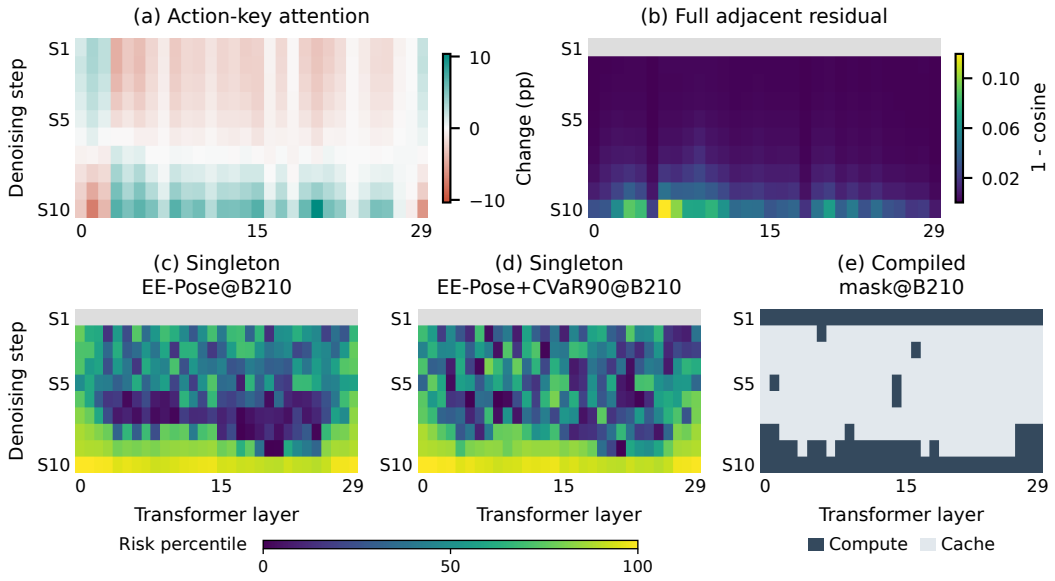

Figure 6: **Attention, sensitivity, and compiled allocation. (a)** Layer-centered action-key attention (percentage points). **(b)** Adjacent Full residual distance ($1 - \cos$). **(c,d)** Mean/CVaR90 singleton EE-Pose risk percentiles. **(e)** B210 mask: dark computes, light reuses. Gray: unmeasured S1 in (b–d).

5 Analysis and Discussion

From conditioning to action refinement. Figure 6(a) subtracts each layer's across-step mean attention mass. Video- and action-key attention masses sum to one, making their centered trends exact opposites. With fixed video keys, attention shifts toward action keys in later steps; stable early residuals and stronger late changes (b) support a conditioning–refinement transition, paralleling T-GATE (Liu et al., 2025b). Attention alone does not determine output impact (Kobayashi et al., 2020); suffix interventions resolve which blocks to preserve. The B210 mask (e) allocates 55 of its 60 computed blocks beyond S1 to S8–S10, retaining five earlier refreshes. Full S10 computation emerges without terminal protection or a reuse-streak limit: broad temporal redundancy coexists with selective layer protection.

Preserving consequences rather than features. Equal joint errors can induce different pose errors, while coordinated changes can compensate (Equation 8). Panels (c,d) rank mean and CVaR90 costs separately across the 270 eligible blocks; higher percentiles indicate larger costs. Both highlight late-stage sensitivity but differ in block priorities (rank correlation 0.777): CVaR emphasizes large worst-episode costs. *EE-Pose determines what to preserve; CVaR determines how strongly to protect the tail.* Beyond these per-block diagnostics, the compiler optimizes CVaR of episode-level cost sums (Equation 10), while conditional and pair interventions test refinements in context (Section 4.4.3). This aligns approximation with downstream use (Donti et al., 2017; Grimm et al., 2020): across robots, the transferable principle is consequence-based allocation, with contact forces and gripper timing as natural extensions of the protected quantities.

6 Conclusion

CausalCache aligns computation with robot action by tracing cache errors through the network suffix and robot kinematics. EE-Pose scoring, fixed-budget CVaR compilation, and conditional pair refinement turn these measurements into a static cache policy, requiring neither retraining nor online selection. Experiments on RoboTwin2 and physical SO-101 manipulation demonstrate strong speed–success trade-offs across Fast-WAM and SmolVLA, while calibration studies show that few episodes suffice to identify useful reuse structure. Efficient robot inference should preserve the geometry of the action, not every intermediate feature.

AI USE STATEMENT

Generative AI tools were used to assist with code debugging, literature organization, spelling checks, and language editing. The authors reviewed the AI-assisted materials and take responsibility for the manuscript, code, analyses, and reported results.

ETHICS STATEMENT

This work studies inference efficiency through RoboTwin2 and a tabletop SO-101 evaluation protocol. It does not use human-subject data. Faster policies may improve deployability, but caching can also perturb safety-critical actions; we therefore report tail-risk objectives, closed-loop evaluation, and limitations rather than treating action similarity as a safety certificate. Real-world deployment requires hardware-specific safety validation.

REPRODUCIBILITY STATEMENT

The appendix specifies residual reuse semantics, exact-budget constraints, calibration splits, and the pose data contract. Supplementary artifacts include fixed matrices, saved initial noise and replay inputs, official action-normalization statistics and robot-frame definitions, paired episode manifests, and machine-readable metric results. Objective and refinement comparisons are identified by their calibration bank, mask, and evaluation cohort.

REFERENCES

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

Priya L. Donti, Brandon Amos, and J. Zico Kolter. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3fc2c60b5782f641f76bcefc39fb2392-Abstract.html.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/559a0998fab1d19b80e7e43a5852401c-Abstract-Conference.html.

Christopher Grimm, André Barreto, Satinder P. Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3bb585ea00014b0e3ebe4c6dd165a358-Abstract.html.

Haoxu Huang, Tongsam Zheng, Yifan Chen, Jiacheng You, and Yang Gao. Critical interval MSE: Toward reliable offline validation for robot manipulation policies. *arXiv preprint arXiv:2606.29898*, 2026. URL https://arxiv.org/abs/2606.29898.

Kangye Ji, Yuan Meng, Hanyun Cui, Ye Li, Jianbo Zhou, Shengjia Hua, Lei Chen, and Zhi Wang. Block-wise adaptive caching for accelerating diffusion policy. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2712b17bb58ea5b2b65c45857b024744-Abstract-Conference.html.

Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. URL https://khatib.stanford.edu/publications/pdfs/Khatib_1987_RA.pdf.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7057–7075. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.574. URL https://aclanthology.org/2020.emnlp-main.574/.

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

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7353–7363, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liu_Timestep_Embedding_Tells_Its_Time_to_Cache_for_Video_Diffusion_CVPR_2025_paper.html.

Haozhe Liu, Wentian Zhang, Jinheng Xie, Francesco Faccio, Mengmeng Xu, Tao Xiang, Mike Zheng Shou, Juan-Manuel Perez-Rua, and Jürgen Schmidhuber. Faster diffusion via temporal attention decomposition. *Transactions on Machine Learning Research*, 2025b. URL https://openreview.net/forum?id=xXs2GKXPnH.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f0b1515be276f6ba82b4f2b25e50bef0-Abstract-Conference.html.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15762–15772, 2024b. URL https://openaccess.thecvf.com/content/CVPR2024/html/Ma_DeepCache_Accelerating_Diffusion_Models_for_Free_CVPR_2024_paper.html.

R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2(3):21–41, 2000. URL https://www.risk.net/journal-risk/2161159/optimization-conditional-value-risk.

Mingzhu Shen, Pengtao Chen, Peng Ye, Guoxuan Xia, Tao Chen, Christos-Savvas Bouganis, and Yiren Zhao. MD-DiT: Step-aware mixture-of-depths for efficient diffusion transformers. In *NeurIPS Workshop on Adaptive Foundation Models*, 2024. URL https://openreview.net/forum?id=1jWhiakK7N.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/2506.01844.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/bbe024e0517fe12ac3a8d388b19ff9fe-Abstract-Conference.html.

A GEOMETRY AND COMPILATION PROCEDURE

Local consequence metric. Fix the observation and initial noise, and let $f(v)$ denote the final normalized action after injecting a residual perturbation v at block b. Let $k(q)$ be the end-effector pose expressed locally using translation and a rotation-vector chart around the Full pose. Assume both maps are twice continuously differentiable in the neighborhood considered. Write $q = Uf(v) + q_0$, where U selects arm coordinates and applies inverse-normalization scales, and q_0 is the corresponding affine offset. Let $J_f = Df(0)$ and $J_k = Dk(Uf(0) + q_0)$ be the two Jacobians at Full. Taylor expansion gives

$$f(v) - f(0) = J_f v + O(\|v\|^2), \qquad k(Uf(v)+q_0) - k(Uf(0)+q_0) = J_k U J_f v + O(\|v\|^2). \tag{12}$$

Here $O(\|v\|^j)$ denotes a remainder of order j in perturbation magnitude. In the main-text notation, the first-order propagation is

$$\delta_b(z) \approx J_f e_{z,b}, \qquad \Delta\xi_{z,b} \approx M_{z,b} e_{z,b}, \qquad M_{z,b} = J_k U J_f. \tag{13}$$

The composed Jacobian $M_{z,b}$ therefore includes the denoising suffix, inverse action normalization, and robot kinematics. For a positive diagonal component-weight matrix W, define $\|w\|_W^2 = w^\top W w$ for any pose-change vector w. Squaring yields

$$D_{\text{pose}}(v) = v^\top G_{z,b} v + O(\|v\|^3), \qquad G_{z,b} = J_f^\top U^\top J_k^\top W J_k U J_f \succeq 0. \tag{14}$$

Here $D_{\text{pose}}(v)$ is the local pose energy caused by v; $\succeq 0$ denotes positive semidefiniteness. Indeed, $G_{z,b} = M^\top W M$ with $M = J_k U J_f$. This is the composed Jacobian $M_{z,b}$ from the main text. To prove Equation 9, expand $v = \sum_j a_j u_j$ in an orthonormal eigenbasis u_j of $G_{z,b}$, with eigenvalues $\lambda_j \geq 0$ and coefficients $a_j = u_j^\top v$. Then

$$0 \leq v^\top G_{z,b} v = \sum_j \lambda_j a_j^2 \leq \lambda_{\max}(G_{z,b}) \sum_j a_j^2 = \lambda_{\max}(G_{z,b}) \|v\|_2^2. \tag{15}$$

For unequal eigenvalues, equal-norm perturbations along the corresponding eigenvectors have different quadratic costs. This establishes directional sensitivity; the nonlinear pose energy additionally includes the Taylor remainder above. For a full action prefix, the Jacobians and weights are stacked over commands and arms; W absorbs command and arm averaging, position-coordinate normalization, and the fixed modality scales s_p, s_R. The result is local to a smooth rotation chart. The experimental implementation evaluates nonlinear FK and the geodesic rotation angle directly.

This derivation identifies two sources of anisotropy: the denoising suffix and the kinematic map. It does not require additive effects for a set of cached blocks. Independent singleton costs are used as a computationally tractable allocation surrogate; joint replay and anchor-centered corrections measure deviations from that surrogate.

Fixed modality normalization. For a three-dimensional rotation matrix Q, we compute its geodesic angle as

$$\theta(Q) = \arccos\left(\text{clip}\left(\frac{\text{tr}\, Q - 1}{2}, -1, 1\right)\right). \tag{16}$$

Here tr denotes matrix trace and clip bounds the argument to $[-1, 1]$ for numerical stability. The resulting angle is in radians. With m equally weighted calibration episodes and eligible block set $\mathcal{B}$, the fixed scale for each component $\nu \in \{p, R\}$ is

$$s_\nu = \frac{1}{m|\mathcal{B}|} \sum_{e=1}^{m} \sum_{b \in \mathcal{B}} \frac{\sum_{z \in \mathcal{Z}_e} H_z D_z^\nu(\{b\})}{\sum_{z \in \mathcal{Z}_e} H_z}. \tag{17}$$

The position denominator $6H_z$ averages three coordinates across two arms; the rotation denominator $2H_z$ averages one squared angle per arm. The policy predicts 32 commands and ordinarily executes 24; H_z additionally accounts for early termination. Unexecuted commands never enter the score. Both scales remain fixed during whole-mask and pair replay, and no task-success labels enter their calculation.

CVaR optimization. For episode cost $C_e(x)$ from Equation 10, the exact-budget compiler uses the standard empirical CVaR epigraph:

$$\min_{x\in\mathcal{X}_B,\eta,u} \eta + \frac{1}{(1-\alpha)m}\sum_{e=1}^{m} u_e, \qquad u_e \geq C_e(x) - \eta, \quad u_e \geq 0. \tag{18}$$

Here $\alpha = 0.9$ is the risk level, $\eta \in \mathbb{R}$ is the optimized tail threshold, and $u = (u_1, \ldots, u_m)$ collects nonnegative episode excess costs. HiGHS solves this mixed-integer linear program. Its feasible set $\mathcal{X}_B$ enforces exactly B cached blocks and S1 initialization. The objective averages the upper 10% of empirical probability mass, including fractional weight at the tail boundary when $(1-\alpha)m$ is not an integer.

A.1 Algorithm

Input: Frozen policy, inverse action normalizer, robot FK, unlabeled Full trajectories, budget B, risk level α, and refinement variant (default: Exact Pair).
Output: Frozen mask $x^\star \in \mathcal{X}_B$ for deployment.

Offline: measure, score, compile, and refine

Save inputs z, initial noise, Full actions $A(z)$, residual traces, and lengths H_z.
for each saved input z and eligible block $b = (s, l) \in \mathcal{B}$ **do**
  $A_b(z) \leftarrow$ replay the entire suffix after substituting $r_{s-1,l}$ for $r_{s,l}$.
  Store the signed action change $\delta_b(z) \leftarrow A_b(z) - A(z)$.
end for
Decode joint commands and apply FK to the first H_z commands of each action.
Compute D_z^p, D_z^R; freeze scales s_p, s_R from calibration singleton energies.
Aggregate normalized EE-Pose costs into $c_b(e)$ (Equation 7).
$x^\star \leftarrow$ solve the exact-budget CVaR program (Equation 18).
$S \leftarrow \{b : x_b^\star = 1\}$; freeze eight endpoint-disjoint, budget-neutral swaps.
Replay S, its eight single swaps, and 28 compatible pairs on the same saved inputs.
Form candidate actions for the chosen variant (Section 3.5):
  Conditional: measured anchor/singles; Additive: also predicted pair actions;
  Exact Pair: also measured pair actions (Equation 21).
Score candidates using fixed pose scales, episode aggregation, and CVaR_α.
$x^\star \leftarrow$ best candidate mask; replay the selected complete mask.
Freeze $x^\star$; record coefficients, constraints, solver gap, and source hashes.

Online: execute the frozen matrix

for each new action chunk **do**
  Initialize action noise and clear stored layer residuals $\bar{r}_l$.
  for $s = 1, \ldots, N$ **do**
    Form $h_{s,0}$ from the current noisy action and original denoising time t_s.
    for $l = 0, \ldots, L-1$ **do**
      if $x_{s,l}^\star = 0$ **then** ▷ S1 always computes
        $\bar{r}_l \leftarrow F_l(h_{s,l}, t_s, c) - h_{s,l}$ ▷ compute and refresh
      end if
      $h_{s,l+1} \leftarrow h_{s,l} + \bar{r}_l$ ▷ reuse never refreshes the cache
    end for
    Advance the original scheduler using the prediction from $h_{s,L}$.
  end for
  Execute the action prefix.
end for

Algorithm 1: CausalCache: offline compilation and online execution

A.2 Conditional interaction diagnostics

For the anchor S, single-swap sets S_i, S_j, and compatible joint set S_{ij} defined in Section 3.5, the conditional single-swap effect and loss interaction on episode e are

$$\begin{aligned} g_i(e) &= \ell_e(S_i) - \ell_e(S), \\ h_{ij}(e) &= \ell_e(S_{ij}) - \ell_e(S_i) - \ell_e(S_j) + \ell_e(S). \end{aligned} \tag{19}$$

Negative h_{ij} denotes compensation beyond loss additivity; positive values denote interference. For example, if the anchor loss is 1.0 and each swap alone gives 0.9, their loss-additive prediction is 0.8.

A measured joint loss of 0.7 gives $h_{ij} = -0.1$, whereas 0.95 gives $h_{ij} = +0.15$: the sign describes the difference from the prediction, not a negative pose loss or improvement over the anchor. Both signs are retained. To isolate non-additivity of action propagation, let $\delta_i(z) = A^{S_i}(z) - A^S(z)$. For a compatible swap subset T with at most two swaps, define

$$E_{ij}(z) = A^{S_{ij}}(z) - A^{S_i}(z) - A^{S_j}(z) + A^S(z), \tag{20}$$

$$\widehat{A}_T(z) = A^S(z) + \sum_{i \in T} \delta_i(z) + \sum_{\{i,j\} \subseteq T} E_{ij}(z). \tag{21}$$

Measured E_{ij} makes the reconstruction exact for up to two swaps. Conditional selection restricts $|T| \leq 1$; Additive and Exact Pair allow $|T| \leq 2$. Additive sets $E_{ij} = 0$, recovering Equation 11, and scores the predicted actions through inverse normalization and nonlinear FK. Exact Pair scores measured joint actions, equivalently retaining E_{ij}. Every variant computes CVaR90 only after evaluating and aggregating the resulting episode losses. The action residual E_{ij} measures propagation non-additivity; h_{ij} also includes FK and pose-loss geometry. Consequently, the action-additive prediction does not assume the loss-additive diagnostic $h_{ij} = 0$.

EE-Pose boundary-refinement cost. For the compiled anchor S, enumerate legal swaps using the saved episode surrogate $C_e(S) - c_a(e) + c_b(e)$. Rank proposals by its CVaR90, with mean loss as a tie-breaker. Freeze four low-cost endpoint-disjoint swaps, then four more that preferentially cover new restored-step/cached-step pairs. No new model evaluation is required for proposal generation. Exact replay measures the anchor, eight conditional single swaps, and 28 pairs: 37 matrices on each of 777 saved chunks, or 28,749 matrix–input queries per budget. All inputs come from the same 40 calibration episodes. Component scales, initial noise, and executed-prefix lengths remain fixed. Selection uses no rollout-success labels; identical selected matrices share one closed-loop evaluation. For each variant, candidates are ordered by episode CVaR90, then mean episode loss, then fewer swaps, then fixed candidate indices. Conditional selection considers the anchor and single swaps; Additive and Exact Pair also include compatible two-swap candidates. CVaR is computed only after composing actions, evaluating nonlinear FK, and aggregating each episode's executed-prefix errors.

B Conditional EE-Pose Refinement and Loss Interactions

Table 4 reports the updated five-run success comparison. Here we report the supporting local calibration interaction audit, separately from those repeated success means. The B242 interaction mean ($+1.534$) in the main table comes from an independent-platform calculation; Table 5 covers the nine locally audited budgets.

Matched interventions. We audit nine budgets using the 37-matrix design of Section 3.5. Each anchor is independently compiled at its budget. All candidates preserve that budget, protect only S1, and share the same 40 calibration episodes, 777 saved inputs, and fixed modality scales. The audit checks matrix hashes, disjoint swap endpoints, episode alignment, and exact cache counts before computing Equation 19. The 28 pairs yield 1,120 pair–episode observations per budget; these share episodes and are descriptive measurements, not 1,120 independent trials.

Compensation and interference. Table 5 reports signed and absolute loss interactions; $Q_{.9}(|h|)$ is the 90th percentile of absolute interaction. For each pair we also compute

$$\rho_{ij} = \frac{|\mathbb{E}_e[h_{ij}(e)]|}{|\mathbb{E}_e[g_i(e)]| + |\mathbb{E}_e[g_j(e)]| + 10^{-12}}. \tag{22}$$

The reported ρ is the median over 28 pairs. Signed averages retain the direction of the effect; absolute moments expose compensation and interference that cancel across episodes. The zero-interaction reference is loss additivity, not zero action error. Negative h_{ij} means better than the additive prediction, while improvement over the anchor depends on the complete loss.

At B210–B230, mean interactions are slightly negative while absolute interactions remain measurable. At B239–B241, 74.6–94.1% of observations are positive, and median ρ rises to 20.7–37.4%. This identifies distinct local regimes around the compiled anchors: mixed compensation at practical reuse levels and predominantly interfering candidate changes near the boundary. Budgets change both the anchor and its proposed swaps; the table characterizes those neighborhoods rather than a controlled budget-only intervention.

Table 5: **Loss interactions around EE-Pose anchors.** Moments and sign fractions pool pair–episode observations; ρ is the median across pairs.

Budget	$\mathbb{E}[h]$	$\mathbb{E}[\lvert h\rvert]$	$Q_{.9}(\lvert h\rvert)$	$h > 0$ (%)	ρ (%)	Reversals (%)
B210	-0.009	0.053	0.111	44.6	13.3	6.9
B220	-0.006	0.057	0.122	47.0	11.9	7.9
B230	-0.017	0.067	0.145	44.6	11.0	4.5
B239	+1.089	1.247	2.658	74.6	20.7	17.7
B240	+1.556	1.587	2.855	94.1	37.0	20.4
B241	+1.877	2.153	3.724	90.2	37.4	16.1
B243	+1.297	1.944	4.030	71.3	13.1	5.6
B245	+1.158	2.088	5.353	60.2	6.5	4.0
B250	+3.173	4.254	9.044	75.8	7.6	4.0

Interactions change the preferred combination. For each pair, compare $\mathrm{CVaR}_{0.9}[\ell_e(S_i) + \ell_e(S_j) - \ell_e(S)]$ with $\mathrm{CVaR}_{0.9}[\ell_e(S_{ij})]$. The reversal fraction in Table 5 counts opposite strict orderings among the 378 comparisons between candidate pairs. At B220 the best loss-additive pair and the best measured pair differ:

Swap pair (zero-indexed)	Loss-additive CVaR90	Measured CVaR90
$(1, 3)$	**1.865**	1.923
$(2, 3)$	1.875	**1.818**

This is a selection-relevant interaction: direct joint measurement changes which combination minimizes calibration risk. It is separate from the action-additive control, which retains nonlinear FK and pose-loss geometry.

C Full-Benchmark Geometry Boundary

The B230/B240/B242/B245/B250 promotions use the same Full50 initial-state and instruction bank, released checkpoint, 40 calibration episodes, and exact-budget residual executor. All matrices protect S1 only. Each policy is evaluated on 50 tasks, two scene conditions, and ten episodes per task-condition. These are unrefined unary masks that isolate the scoring objective, not the refined five-run operating points in Table 1. Aggregation is checked against execution receipts, matrix hashes, and paired seed manifests.

Paired uncertainty. We resample tasks with replacement, keeping both methods paired, then resample episodes within each selected task-condition. Tables 6 and 7 use 5,000 bootstrap draws and seed 270917. Intervals are descriptive pointwise 95% intervals without a multiple-budget adjustment. Wins and losses denote episodes in which only EE-Pose or only MSE succeeds, respectively. B240 has one shared matrix, not two independent runs.

Table 6 focuses on B240–B250, the high-reuse boundary. As a lower-reuse reference, B230 achieves 91.3% with MSE and 91.1% with EE-Pose: a -0.2-point difference, paired 95% interval $[-2.5, 2.1]$, and 33 wins versus 35 losses.

Table 6: **Full50 high-reuse objective frontier.** Success (%) and EE-Pose-minus-MSE differences with paired 95% intervals (pp).

Budget	MSE	EE-Pose	Difference	95% interval	Wins / losses
B240	90.9	90.9	0.0	$[0.0, 0.0]$	0 / 0
B242	87.6	89.6	$+2.0$	$[-1.0, 5.0]$	60 / 40
B245	72.1	77.7	$+5.6$	$[2.2, 9.0]$	102 / 46
B250	31.2	19.9	-11.3	$[-16.2, -6.6]$	43 / 156

At B245, EE-Pose gains 5.6 points with a paired interval above zero; the smaller B242 gain has an interval spanning zero. At B250 (83.33% reuse), both policies collapse and MSE performs better.

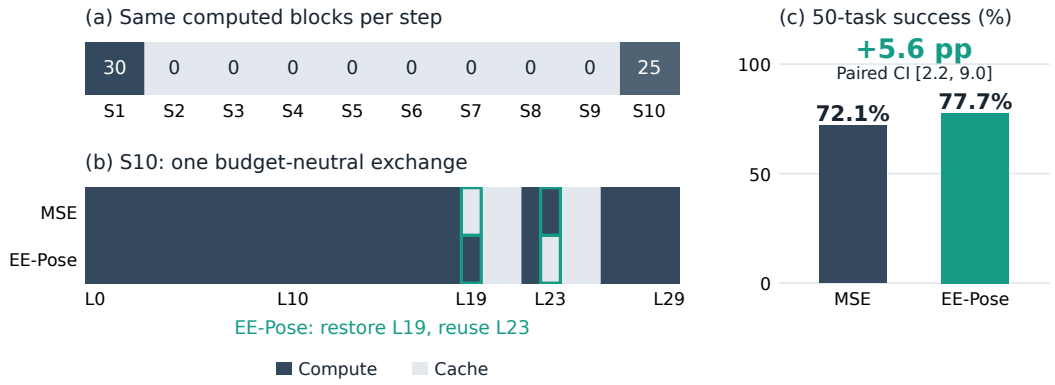

Figure 7: **Budget-neutral layer exchange at B245.** **(a)** Identical step quotas. **(b)** S10 L19/L23 exchange (teal); dark computes, light reuses. **(c)** Full50 success and paired 95% interval for EE-Pose minus MSE.

The measured frontier locates the compression regime in which geometry-aware allocation preserves control.

Transfer beyond calibration tasks. Removing the four calibration task identities leaves 46 tasks and 920 episodes per matrix (Table 7). Full achieves 93.70% on this subset. At B245, EE-Pose gains 4.89 points over MSE, with paired interval $[1.63, 8.26]$. This subset measures transfer beyond the calibration tasks within the same full50 evaluation.

Table 7: **Transfer to 46 calibration-disjoint tasks.** Differences and 95% intervals are EE-Pose minus MSE (pp).

Budget	MSE (%)	EE-Pose (%)	Difference (pp)	95% interval (pp)
B230	92.72	93.04	$+0.33$	$[-1.63, 2.50]$
B240	92.61	92.61	0.00	$[0.00, 0.00]$
B242	89.46	91.74	$+2.28$	$[-0.54, 5.22]$
B245	74.89	79.78	$+4.89$	$[1.63, 8.26]$
B250	32.07	21.20	-10.87	$[-15.65, -6.20]$

Exact matrix differences. Both B242 masks compute $[30, 0, 0, 0, 0, 0, 0, 0, 0, 28]$ blocks across S1–S10; both B245 masks compute $[30, 0, 0, 0, 0, 0, 0, 0, 0, 25]$. At B242, MSE caches S10 layers $\{20, 21\}$ and EE-Pose caches $\{21, 25\}$; 241 cached blocks are shared. At B245, MSE caches S10 layers $\{19, 20, 21, 24, 25\}$ and EE-Pose caches $\{20, 21, 23, 24, 25\}$; 244 cached blocks are shared. All layer indices are zero-based. Each comparison therefore isolates one budget-neutral layer exchange under the same temporal profile. Its measured effect belongs to the exchange in that matrix context, rather than to either layer in isolation. Figure 7 visualizes the B245 exchange.

D Performance Protocols and Timing Replication

Table 1 reports five-run success means for the core operating points; the added LocalCos and terminal-unprotected TeaCache rows use their supplied full-benchmark aggregates. Module and complete-call latency are measured separately. The paired intervals in Appendix C belong to its unrefined objective ablation.

Independent paired timing replication. We additionally time 200 fixed inputs, balanced across clean and randomized scenes, on a single RTX 4090. Full and cached calls receive identical conditioning and initial noise, and execution order alternates. Three serial repeats run without concurrent GPU evaluation. This study measures the frozen EE-Pose unary B230 matrix from the paired objective cohort, which obtains 91.1% success in that cohort. Its Action-DiT speedup ranges

from 3.807× to 3.809×; complete-call speedup ranges from 2.809× to 2.811×. The independently compiled MSE B230 matrix has the same rounded speedups.

Table 8: **Independent B230 timing replication.** Mean CUDA-event latency (ms); complete calls exclude simulation and rendering.

Component	Full (ms)	Cached (ms)	Reduction	Speedup
Other policy computation	48.63	48.75	−0.2%	1.00×
Action-DiT	337.18	88.55	**73.7%**	**3.81**×
Complete policy call	385.82	137.30	**64.4%**	**2.81**×

D.1 Offline calibration cost and amortization

Table 9: **Parallel-optimized preparation time and amortization.**

Configuration	Total wall (min)	Refinement increment (min)	Wall-time break-even calls
Singleton	23.9	—	5,701
Singleton + Conditional	24.8	0.9	5,908
Singleton + Exact Pair	26.3	2.5	6,264

Table 9 covers eight episodes, 233 chunks, and 270 singleton candidates at B230, applying the reported fivefold parallel speedup to end-to-end preparation from Full-rollout collection through compilation and refinement, excluding inference benchmarking. For a fixed policy, robot, and sampling schedule, the singleton bank is reused across budgets; only allocation and anchor-dependent refinement are rerun. Exact Pair adds 2.5 minutes (10.4%) without introducing online optimization. Break-even is $\lceil T_{\text{prep}}/\Delta t \rceil$, where T_{prep} is optimized preparation wall time and $\Delta t > 0$ is the paired Full-minus-cached complete-policy wall-clock latency saving, both in seconds. Counts use unrounded timings from three paired repeats on 200 fixed inputs per mask on one isolated RTX 4090. Each count refers to action-chunk inference calls, amortizable across tasks and episodes, rather than individual robot actions or reductions in physical task duration.

E Additional Evaluation Settings

E.1 Simulation controls

Calibration and hard-panel selection. The four RoboTwin2 calibration tasks are moving a stapler to a pad, opening a microwave, stacking three bowls, and handing over a block. Each contributes five episodes under each of the clean and randomized conditions, using the first five valid fresh-seed initializations without success filtering. The hard-six panel is selected by the lowest historical Full-policy success averaged across both conditions, excluding these four tasks and breaking ties alphabetically. It comprises hanging mug, turning a switch, placing a can in a basket, picking diverse bottles, putting bottles in a dustbin, and placing a mouse pad. Selection precedes objective comparisons and does not use relative cache-selector results. Action-MSE averages squared differences over all 14 model-normalized action channels (12 arm joints and two grippers). TCP position compares the two end-effector positions after inverse normalization and FK, excluding orientation and gripper aperture.

Selector definitions. Uniform refreshes all submodules at S1/S4/S7/S10. TeaCache uses accumulated predicted feature change; its unprotected variant removes forced S10 refresh while retaining S1 initialization. Native-parameter BAC assigns submodule schedules with ACS and applies BUA; S4/BUA0 is the operating point in the main frontier. Here S denotes the pre-BUA number of refreshes per submodule, not a denoising-step position. Low NFE=6 computes every retained step and has zero cache reuse. LocalCos scores adjacent-step residual similarity; its B120/B150/B180 operating points in Table 1 retain the original four-step reuse cap.

Ablation cohorts. The independent suffix comparison in Section 4.4.1 uses action-space Causal Unary without task-space scoring or refinement; both selectors protect S1 only, without a consecutive-reuse cap. The primary CAUSALCACHE frontier also protects only S1. Its refinement compares the anchor with Conditional and Exact Pair selection. Objective ablations fix the bank and solver while changing discrepancy coefficients; risk ablations additionally vary mean versus CVaR90. Table 3 reports a repeated six-task evaluation over $B \in \{220, 230, 240, 242, 245\}$ with two runs per budget. Its four objectives use CVaR90, CVaR90, Mean, and CVaR90, respectively; the final Mean row instead equally averages success across the six tasks. Identical masks share one audited evaluation; paired rollouts share initial states, instructions, and saved RNG state.

Replay and executor checks. Each record identifies its task, scene condition, episode, observation step, initial noise, matrix, and signed action delta. Full-reference telemetry defines the executed prefix: scores neither cross episode boundaries nor include unexecuted predictions. The official inverse normalizer recovers joint commands before URDF FK; TCP frames, root rotation, and end-effector offsets are checked against simulator observations. Residuals are reset per action chunk and updated only by real block evaluations, never by reuse. Every evaluated static matrix realizes exactly B cached blocks per chunk with zero fallback.

E.2 SO-101 CONFIGURATION

Task-specific data and policies. The datasets are stored in LeRobot format. Sequential pick-and-place uses `ICLR/Grab_3C` with prompt "Grab the Cube"; oriented T placement uses `ICLR/T_3C` with "Grab the T"; blue-on-red stacking uses `ICLR/Stack_3C` with "Stack the cube". The task definitions in Section 4.3 specify the required order, orientation, and stacking relation beyond these short prompts. For sequential pick-and-place (Pick-place in the tables), success requires grasping and placing the blue cube inside the box, followed by grasping and placing the red cube inside the box. The trained prompt remains unchanged. All three policies start from `lerobot/smolvla_base`, use 20,000 training steps and batch size 32, and save checkpoints every 2,000 steps. Evaluation loads the task-specific `checkpoints/last/pretrained_model` directory. W&B logging and Hub uploads are disabled in the supplied configuration.

Camera and robot configuration. LeRobot uses `so101_follower` on `/dev/ttyACM0`. The training rename map sends the image fields `front`, `wrist`, and `side` to `camera1`, `camera2`, and `camera3`, respectively. All cameras use the OpenCV backend at 640×480 and 30 FPS:

Policy input	Dataset image field	Device index	Capture format
`camera1`	`front`	0	YUYV
`camera2`	`wrist`	4	MJPG
`camera3`	`side`	2	YUYV

Rollout configuration. Each of the three tasks comprises 20 physical trials for each of Full, EE-Pose B40/B60/B80/B90, and Action-MSE B40/B60/B80/B90, totaling 540 trials. In that order, success counts are 16, 16, 15, 14, 13, 14, 14, 13, and 12 for pick-and-place; 16, 16, 16, 15, 14, 14, 13, 12, and 10 for T placement; and 16, 16, 16, 14, 12, 15, 13, 12, and 11 for stacking. Table 2 reports percentages and their equal-weight three-task mean. The recorder configuration uses live visualization, a 1,200-second episode recording limit, and a 30-second reset interval. The 1,200-second setting is a recording limit, not a measured task-completion time. Evaluation records remain local, with Hub uploads disabled. The three SmolVLA checkpoints use ten denoising steps, sixteen action-expert layers, and a 50-action chunk and execution horizon. Thus $B \in \{40, 60, 80, 90\}$ corresponds to 25%, 37.5%, 50%, and 56.25% reuse out of 160 blocks, respectively; the Fast-WAM 10×30 grid is specific to simulation. For single-arm EE-Pose scoring, Equations 4–5 average over one end effector, so the position and orientation denominators become $3H_z$ and H_z, respectively, using the SO-101 kinematic map.

Calibration. Each task uses eight demonstration episodes, with three evenly spaced queries per episode, and four disjoint episodes for held-out action-error evaluation. All 144 blocks outside the protected first step receive singleton interventions under identical inputs and noise. Scores use the configured 50-action execution horizon, capped at the demonstration episode boundary. EE-Pose

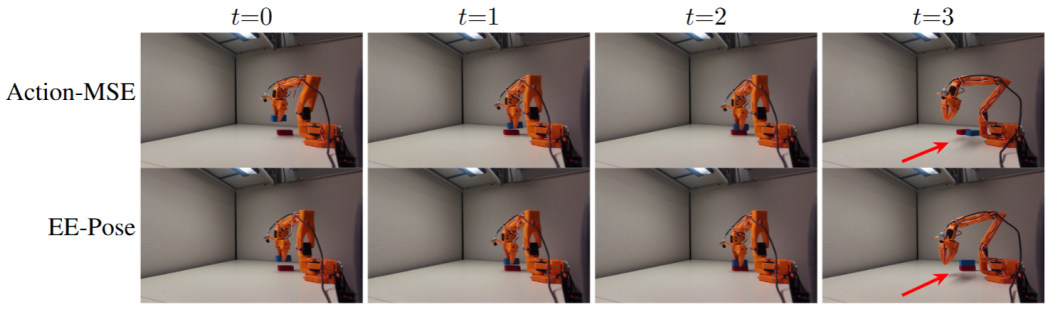

Figure 8: **Qualitative stacking comparison on SO-101 with SmolVLA.** Red arrows highlight the post-release outcomes: Action-MSE (top) leaves the blue cube beside the red cube, whereas EE-Pose (bottom) stacks it on top.

Table 10: **Per-task SmolVLA latency (ms).** Expert: action-expert denoising; Model: complete model inference.

	Pick-place		T placement		Stacking	
Configuration	Expert	Model	Expert	Model	Expert	Model
Full	222.2	272.9	221.3	279.5	223.5	275.8
Action-MSE B40	170.4	222.5	174.9	226.4	173.3	230.3
Action-MSE B80	119.2	172.8	121.7	176.5	120.7	176.4
EE-Pose B40	173.9	224.8	176.7	231.2	171.2	226.8
EE-Pose B80	121.0	175.0	123.4	176.8	120.5	175.2
Full (B60 batch)	221.5	276.3	226.8	279.9	225.2	277.5
Action-MSE B60	149.7	202.6	148.8	205.4	148.3	201.9
EE-Pose B60	147.9	201.2	153.7	208.4	151.9	203.8
Full (B90 batch)	227.1	279.0	222.7	278.4	226.9	279.7
Action-MSE B90	106.6	162.9	109.9	162.9	110.5	165.8
EE-Pose B90	108.7	164.5	109.5	170.1	109.3	164.1

applies SO-101 forward kinematics to de-normalized joint commands; its position and rotation scales are fixed using calibration data. Action-MSE includes all six normalized action channels, including the gripper. The compiler minimizes empirical CVaR_{90} of episode-averaged additive costs at an exact budget, without a maximum reuse streak or a terminal-step protection constraint. With eight equally weighted episodes, empirical CVaR_{90} equals the maximum episode cost. These masks use singleton compilation without pair refinement, and remain task-specific; the held-out split evaluates cache calibration rather than policy-training generalization.

Timing. For each checkpoint, a fixed observation and initial noise are reused across Full and both objectives at B40/B80 on one RTX 3090 at batch size one. B60 and B90 each use a separate timing batch with its own paired Full reference. We discard two warm-up rounds and average ten interleaved rounds, reversing method order on alternating rounds. Wall-clock timing is bracketed by CUDA synchronization. Action-expert timing excludes visual–language–state prefix construction; complete-model timing includes it. Both exclude action/observation preprocessing and postprocessing, camera capture, motor I/O, and physical task execution. Table 10 reports per-task measurements; aggregate speedups are ratios of the equally weighted task means. The paired Full averages (expert/model) are 224.5/277.9 ms for B60 and 225.6/279.0 ms for B90; Table 2 uses these paired references.