Sparse Planning in Visual World Models via Cost Gradients

Yingchen Xu
University College London
yingchen.xu.21@ucl.ac.uk

Edward Grefenstette
University College London
e.grefenstette@ucl.ac.uk

Abstract

Token-based world models enable fine-grained latent planning, but repeatedly processing large spatial token grids makes action search expensive. We introduce COSTGRAD, a training-free, goal-conditioned selector that ranks spatial tokens by the gradient norm of the planning cost with respect to each input token. By deriving importance from the downstream control objective, COSTGRAD targets tokens that matter for planning rather than merely for prediction. On AdaLN-conditioned predictors at 50% sparsity, COSTGRAD matches or exceeds full-token planning on three of four continuous-control benchmarks, while giving a measured $2.6\times$ wall-clock speedup per environment planning step. Combining token sparsity with reduced CEM search increases this to a $\sim 5\times$ total speedup while still exceeding the full-token baseline. We also identify an architecture-dependent failure mode: in a matched AdaLN-vs-concat comparison, concat maintains comparable full-token performance but pure COSTGRAD loses its advantage over random selection. This difference tracks action-pathway drift: gradient-selected removal produces less drift than random removal on AdaLN, but more on concat. These results highlight selector–architecture compatibility as a design axis for sparse world-model planning. Project page and demos: ycxuyingchen.github.io/costgrad/.

1 Introduction

Token-based world models [Zhou et al., 2025, Terver et al., 2025] predict future spatial feature grids produced by a frozen vision encoder. At inference time, planning runs the cross-entropy method (CEM) [Rubinstein, 1999] inside the learned model, rolling out many candidate action sequences against an encoded goal. This is expensive: at 16×16 tokens, horizon 6, and standard CEM widths, one planning step performs $\sim 10^7$ token-level forward computations. Much of this compute is often spent on weakly task-relevant tokens: in control tasks, the agent, object, obstacle, and goal occupy only a small part of the frame, while most patches describe background. Sparse Imagination [Chun et al., 2025] demonstrates efficient world-model planning through sparse training and random token selection. We ask how a fixed pretrained predictor can select tokens relevant to the current goal without sacrificing planning success.

A natural approach is to reuse the predictor's own signals, such as attention weights or prediction error. But prediction is not planning: improving dynamics prediction does not necessarily improve downstream control, an objective mismatch documented in model-based reinforcement learning [Lambert et al., 2020]. Attention can be underdetermined by prediction loss, and prediction error highlights visual unpredictability rather than control relevance: the object most important for control may be easy to predict because its motion follows directly from the action. This motivates using the planning objective itself as the source of token importance.

We instantiate this idea with COSTGRAD: given an encoded observation and goal, we compute the gradient of the planning cost with respect to each input token, and rank tokens by gradient norm.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Subsequent action search uses only the selected tokens. COSTGRAD requires one extra forward and backward pass per planning step, operates on a fixed pretrained predictor without retraining, and conditions selection on the current goal. On AdaLN-conditioned predictors at 50% sparsity, COSTGRAD matches or exceeds full-token planning success on three of four environments and gives a measured $2.6\times$ wall-clock speedup per environment planning step.

The more surprising finding is that COSTGRAD depends on how actions enter the predictor. We compare two otherwise matched predictors: AdaLN [Peebles and Xie, 2023] injects actions by modulating each visual token with action-dependent parameters, while concat appends the action to each token before self-attention. Pure COSTGRAD succeeds on AdaLN but loses its advantage on matched-concat, despite comparable full-token planning. This difference tracks action-pathway drift: gradient-selected token removal produces less drift than random removal on AdaLN, but more on concat. Sparse planning therefore depends on both choosing relevant tokens and on a predictor architecture that remains stable when restricted to those tokens. More generally, the planner is part of the model's operating distribution: token selection changes not just what the model sees, but how the planner uses the model.

Contributions. (i) We introduce COSTGRAD, a training-free, goal-conditioned token selector for sparse visual world-model planning. At 50% sparsity, COSTGRAD matches or exceeds full-token planning on three of four benchmarks, gives a measured $2.6\times$ wall-clock speedup per environment planning step, and can combine token sparsity with reduced CEM iterations for a $\sim 5\times$ total speedup while still exceeding the full-token baseline (§3.1). (ii) We identify a selector–architecture compatibility failure mode. In a matched AdaLN-vs-concat comparison, COSTGRAD beats random selection on AdaLN but loses its advantage on concat. Gradient-selected removal reduces action-pathway drift relative to random removal on AdaLN but increases it on concat. Anchor-mixing experiments further show that concat can require more coverage-preserving sparse supports, suggesting that sparse planning depends on matching the selector to the predictor architecture (§3.3).

2 Setup and Method

2.1 Token-based world model planning

We build on the DINO-WM framework [Zhou et al., 2025] using the open-source JEPA-WMs codebase [Terver et al., 2025]. A frozen DINOv2 ViT-S/14 encoder [Oquab et al., 2024] maps each 224×224 image to a 16×16 grid of 384-dimensional tokens, $\boldsymbol{z} \in \mathbb{R}^{P \times D}$ with $P = 256$. Our main results use an AdaLN-Zero-conditioned predictor [Peebles and Xie, 2023]: a 6-block transformer with 16 attention heads and GELU activation that maps the encoded context and an action to the next-step token grid. It is trained on trajectory data from Terver et al. [2025] and kept fixed during planning; the block equations are given in Appendix D.

Planning uses CEM [Rubinstein, 1999] in latent space: at each environment step, CEM samples 300 action sequences over horizon $H{=}6$, ranks them by cost to the encoded goal, and refits the sampling distribution to the top elites for 30 iterations (JEPA-WMs defaults). Each candidate is rolled out autoregressively, calling the predictor $H{=}6$ times on the full $P{=}256$-token grid, so one planning step performs $300 \times 30 \times 6 \times 256 \approx 1.4 \times 10^7$ token-level forward computations.

2.2 COSTGRAD: select tokens, then plan

COSTGRAD scores tokens using the *planning* objective rather than the predictor's training objective. Figure 1 illustrates how a single selection probe defines the token subset reused throughout action search.

Select tokens. Given a context encoding $\boldsymbol{z}$ and goal encoding $\boldsymbol{z}_g$, we run a one-step prediction on all P tokens under a probe action $\boldsymbol{a}_0$ (zero by default) and compute

$$\mathcal{L}_{\text{plan}}(\boldsymbol{a}_0; \boldsymbol{z}, \boldsymbol{z}_g) = \left\| \text{predictor}(\boldsymbol{z}, \boldsymbol{a}_0) - \boldsymbol{z}_g \right\|_2^2. \tag{1}$$

A single backward pass gives the per-token score

$$s_j = \left\| \nabla_{\boldsymbol{z}_j} \mathcal{L}_{\text{plan}} \right\|_2, \qquad j = 1, \ldots, P. \tag{2}$$

We retain the K tokens with the largest scores, defining the selected set $\mathcal{S}$.

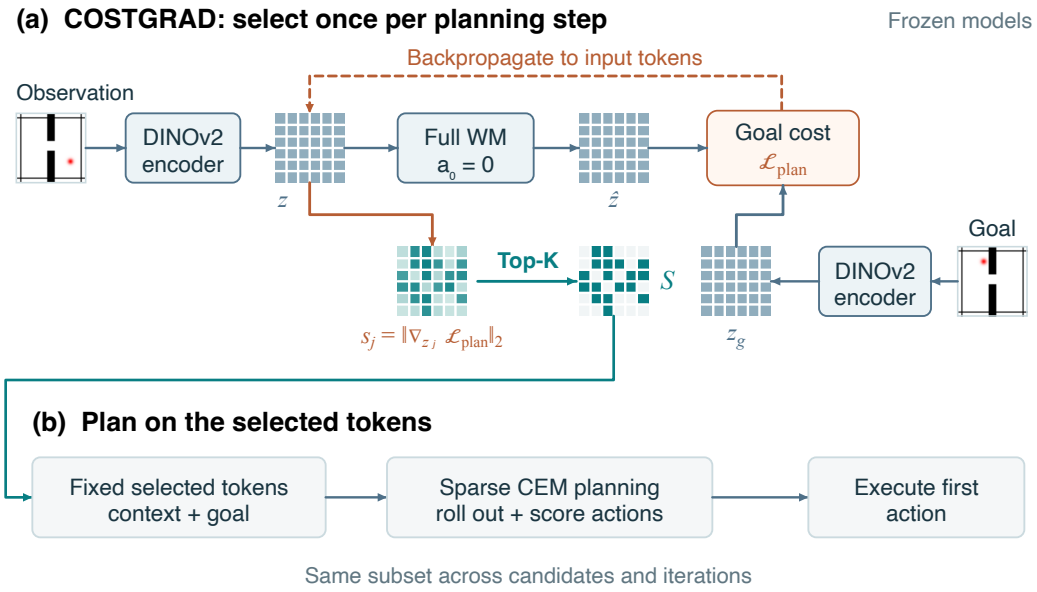


Figure 1: **COSTGRAD: select once, then plan on the selected tokens. (a)** A full-token, zero-action probe supplies planning-cost gradients to select top-K positions $\mathcal{S}$ for context and goal. **(b)** CEM searches for actions using only $\mathcal{S}$, then executes the first action. The subset stays fixed throughout action search and is reselected at the next observation; model weights remain frozen. Token grids are schematic; proprioception is omitted.

The gradient norm measures local planning-cost sensitivity to each input token, including its influence on predictions elsewhere. Thus, a token can receive a high score by providing context for predicting goal-relevant regions, even when its own predicted feature is close to the goal.

Plan on the selected tokens. CEM rolls out each candidate action sequence autoregressively for the full horizon H using only $\mathcal{S}$ and evaluates the final goal-distance cost at the same kept positions. The subset remains fixed across all candidates and CEM iterations; we reselect tokens at the next observation. The one-step probe is used only for selection, while CEM samples and optimizes its own actions (Algorithm 1). We retain CEM to match the JEPA-WMs evaluation protocol and isolate the effect of token selection.

Implementation. Retained tokens keep their original positional indices, shared across context frames and the goal. Gradients score the latest observed frame; rollout uses up to two context frames. The probe uses only visual squared error and zero normalized action/proprioceptive inputs. CEM combines final-step mean squared visual and proprioceptive-feature errors with weights 1 and 0.1, respectively (objective and normalization in Appendix B).

Why the planning objective? Prediction error measures what the model gets wrong, whereas planning-cost gradients measure how its inputs affect the goal-reaching objective. Even a perfectly predicted next state can remain far from the goal, so prediction accuracy alone does not identify which tokens matter for planning. Appendix M develops this objective distinction and the local-sensitivity interpretation.

Computational cost. COSTGRAD is training-free and goal-conditioned: selection depends on the current goal while predictor weights remain fixed. Its overhead is one full-token forward and backward pass per planning step, amortized over all CEM rollouts. Attention compute scales quadratically in token count while MLP scales linearly, so halving tokens reduces attention flops by $4\times$ and MLP flops by $2\times$.

Algorithm 1 COSTGRAD sparse planning step

Input: encoded context $\boldsymbol{z}$, goal $\boldsymbol{z}_g$, keep ratio K/P
// Step 1: select tokens once using a probe action $\boldsymbol{a}_0$
Set probe action $\boldsymbol{a}_0 \leftarrow \mathbf{0}$ *// zero-action by default; Appendix E for alternatives*
$\hat{\boldsymbol{z}} \leftarrow \text{predictor}(\boldsymbol{z}, \boldsymbol{a}_0)$ *// full-token forward, retain graph*
$\mathcal{L} \leftarrow \|\hat{\boldsymbol{z}} - \boldsymbol{z}_g\|_2^2$
$s_j \leftarrow \|\nabla_{\boldsymbol{z}_j} \mathcal{L}\|_2$ for all tokens j *// single backward pass*
$\mathcal{S} \leftarrow \text{top-}K(s_j)$ *// sparse token set, fixed for the rest of this planning step*
// Step 2: run CEM as usual on the sparse token set; CEM samples its own actions
return $\text{CEM}(\boldsymbol{z}_{\mathcal{S}}, \boldsymbol{z}_{g,\mathcal{S}};$ horizon $H, 30$ iters, 300 candidates$)$

2.3 Baselines

We compare COSTGRAD with **Full** and four token-reduction baselines, sharing the trained predictor and CEM settings within each environment. Full retains all $P = 256$ tokens; all token-reduction methods use $K = P/2 = 128$.

PREDATTN ranks spatial keys by received attention in the predictor's last block under a full-grid, zero-action probe: sum over all queries and average uniformly over heads and context frames, with causal masking and conditioning-token keys excluded.

PREDGRAD ranks tokens by prediction-loss gradient norms, using earlier observed frames as input and the latest observed encoding as $\boldsymbol{z}_{\text{target}}$, without privileged future information. It uses a zero-action probe; the observed target need not match that action. Appendix B specifies the loss and temporal alignment.

Random samples K positions uniformly without replacement, once per planning step, and fixes them throughout CEM.

ToMe [Bolya et al., 2023] merges observation tokens by cosine-based bipartite matching, averaging each group at its surviving position. The mapping is shared across context frames; rollout predicts at the surviving positions and compares against unmerged goal features there (Appendix B).

3 Results

We evaluate planning performance and speed (§3.1), the task relevance and stability of selected tokens (§3.2), and the dependence of sparse planning on predictor architecture (§3.3).

We evaluate on four continuous-control benchmarks from the DINO-WM and JEPA-WMs suites — **PointMaze (MZ)**, **Wall**, **PushT (PT)**, and **MetaWorld (MW)** — detailed in Appendix A.

3.1 Sparse planning performance and speed on AdaLN-Zero

Table 1 reports planning success at $K{=}128$ tokens. COSTGRAD preserves full-token performance on PointMaze (-2.0 pp, within noise) and exceeds Full on Wall ($+7.3$ pp), PushT ($+2.8$ pp), and MetaWorld ($+6.9$ pp), giving an average gain of $+3.8$ pp while using half the tokens. Prediction-derived selectors are substantially weaker: the closest baseline, PREDATTN, trails COSTGRAD by 12.1 pp on average and by more than 20 pp on Wall and MetaWorld. Random selection is far worse (-19.2 pp on average), showing that the gain on AdaLN-Zero is not a generic sparsification effect; §3.3 examines what changes when the predictor's action-conditioning pathway is replaced.

On released JEPA-WMs checkpoints, COSTGRAD also preserves full-token performance across all four environments without retraining, using half the tokens and outperforming Random by 11.3–46.3 pp (Appendix L, Table 8).

Table 1: **Main results at 50% tokens.** Planning success rate (%) across four continuous-control environments. Mean ± std over 3 evaluation seeds (96 episodes each). Full uses all 256 tokens; all other methods keep 128 tokens. **Bold**: best non-Full method per environment. COSTGRAD preserves or exceeds Full on three of four environments while giving a measured 2.6× wall-clock speedup per environment planning step.

Method (50%)	PointMaze (MZ)	Wall	PushT (PT)	MetaWorld (MW)	Avg
Full (100%)	88.2 ± 8.7	84.7 ± 2.6	62.4 ± 4.4	58.7 ± 1.8	73.5
COSTGRAD	86.2 ± 6.1	$\mathbf{92.0 \pm 1.8}$	$\mathbf{65.2 \pm 6.5}$	$\mathbf{65.6 \pm 4.7}$	**77.3**
PREDATTN	$\mathbf{86.7 \pm 9.7}$	67.3 ± 1.7	62.0 ± 8.0	44.8 ± 6.1	65.2
PREDGRAD	76.8 ± 9.2	67.2 ± 3.3	41.3 ± 5.9	51.4 ± 3.5	59.2
ToMe	74.4 ± 9.1	84.4 ± 2.6	51.4 ± 5.9	39.1 ± 4.1	62.3
Random	77.0 ± 9.4	67.0 ± 3.6	41.3 ± 5.9	46.9 ± 3.1	58.1

Sparsity sweep. Figure 2 varies the keep ratio from 10% to 100%. COSTGRAD remains near Full at 25% tokens on Wall, PushT, and MetaWorld, and approaches Full at 50% on PointMaze. Below roughly 25%, all sparse methods degrade sharply, indicating that the selected subset must still contain enough state information for rollout. The gap to PREDATTN and Random widens as the token budget shrinks: planning-gradient selection becomes more important in the high-sparsity regime.

COSTGRAD dominates the measured speed–success frontier. Retaining 50% of the tokens with the standard CEM budget gives the 2.6× speedup reported in Table 1. The extra forward and backward pass adds 27.5 ms on a single H100, amortized over the subsequent CEM iterations.

Token selection and CEM budget are complementary compute axes: predictor cost per CEM iteration scales with the number of kept tokens K, and total planning-step time scales linearly with the number of CEM iterations. Combining 50% tokens with 15 CEM iterations reaches 92.0% success at $\sim$ 17 s per planning step — a 5.2× total speedup over Full (84.7% at 88.6 s) at higher mean success (Appendix G). At the same 15-iteration budget, Full achieves 84.4 ± 1.0% success, whereas COSTGRAD achieves 92.0 ± 5.1% while using half the tokens (Appendix G).

Across the measured token budgets and CEM settings (Figure 3), COSTGRAD configurations trace the entire non-dominated frontier; Full, PREDATTN, and Random are dominated. For example, COSTGRAD at K=192 reaches 92.7% success at 60.4 s per step (+8.0 pp and 1.47× faster than Full), while the PREDATTN and Random sweep curves trail COSTGRAD by large margins. Success enters a high-performance plateau around $K \geq 64$ across the measured COSTGRAD configurations, with the best points near $\sim$92%.

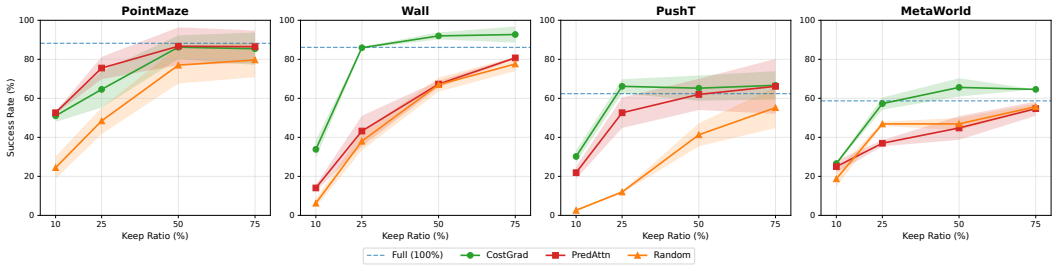


Figure 2: **Sparsity sweeps on four environments.** Planning success (%) vs. keep ratio for COSTGRAD, PREDATTN, and Random; horizontal dashed line is Full (100%). Shaded bands: ±1 std across 3 evaluation seeds at keep ratios 10/25/75 (96 episodes per cell per seed); the 50% point uses Table 1's 3-eval-seed mean. COSTGRAD approaches or exceeds Full at 25% tokens on three of four environments (Wall, PushT, MetaWorld); the gap to prediction-derived baselines (PREDATTN, Random) widens as sparsity increases. All sparse methods degrade sharply below $\sim$25%.

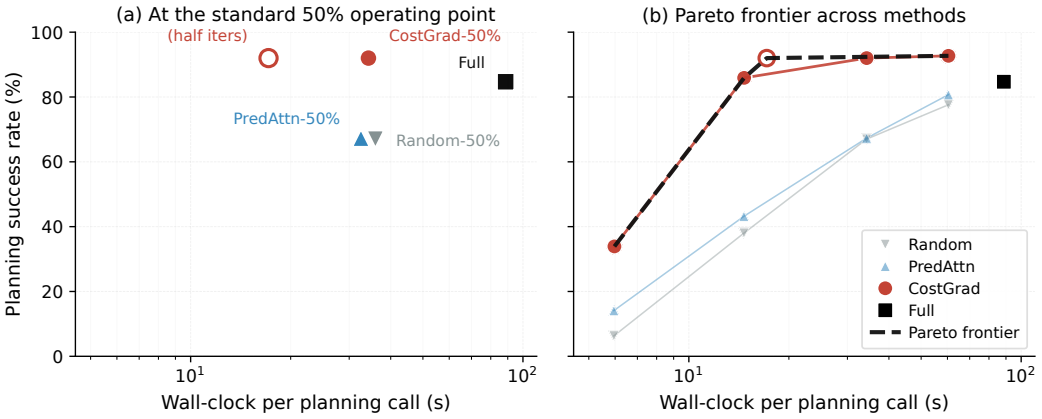


Figure 3: **Wall speed–success trade-off on AdaLN-Zero. (a)** Five canonical operating points at the standard $K{=}128$ sparse setting. CostGrad-50% with 15 CEM iterations achieves the same mean success as standard CostGrad-50% while reducing planning time to $\sim$17 s, a $5.2\times$ speedup over Full. **(b)** Pareto frontier across methods at 30 CEM iterations, plus the CostGrad $K{=}128$, 15-iteration point. Among measured points, the non-dominated frontier is traced by CostGrad configurations; Full, PredAttn, and Random are dominated. Timing details and the full (K, iters) marker convention are in Appendix K and Appendix H.

Why can sparsity exceed Full? Sparse planning also changes the effective planning cost: Full includes background and weakly task-relevant tokens that may add noise to the objective. Restricting rollout and goal comparison to high-sensitivity tokens can sharpen the cost landscape around controllable objects and goal-relevant regions, provided the subset retains enough state for prediction. This interpretation is consistent with the cost-function ablation on Wall: CostGrad's gain over Full *triples* when the cost is more degenerate — $+22.9$ pp under L_1 versus $+7.3$ pp under L_2 (Appendix F).

3.2 Task relevance and stability of planning-gradient selection

§3.1 showed that planning-gradient selection outperforms attention- and prediction-error-based selection by large margins on AdaLN-Zero (Table 1). We first examine what CostGrad selects and how those selections align with task relevance, then test their stability across trained predictors and probe actions.

Goal-conditioned selection. Figure 4 visualizes CostGrad selections across environments. The kept tokens concentrate on task-relevant regions — maze corridors, the agent and wall, the pusher and block, the robot arm and object — while dropping much of the background. Decoded one-step predictions from the same sparse token sets remain qualitatively faithful (Appendix C, Figure 5).

Prediction difficulty is not task relevance. Prediction error selects visually hard-to-predict tokens, whereas planning needs tokens whose state affects the goal-reaching cost. This distinction is clearest on PushT: the pusher and T-block are task-critical but move predictably under the action, so they receive low prediction error while still receiving high CostGrad score. A quadrant analysis confirms that PushT uniquely contains many "easy-but-critical" tokens — low prediction error but high planning gradient — at $\sim 2\times$ chance density, while the other three environments are depleted (Appendix J). Prediction-derived selection can therefore downweight precisely the objects that matter for control.

Selection stability. Prediction-derived attention is unstable across equivalent predictors. Three Wall AdaLN-Zero predictors trained with different seeds reach indistinguishable validation loss, but their top-128 attention selections overlap at only IoU 0.44 ± 0.02 and PredAttn planning success varies by 50 pp across seeds. CostGrad selections on the same predictors overlap at IoU > 0.82

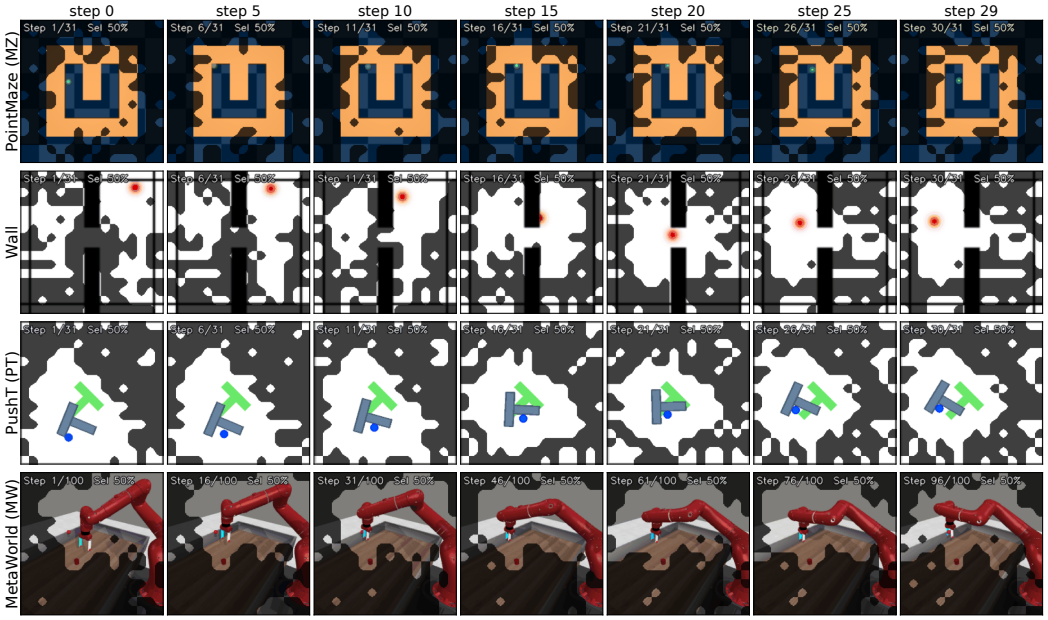


Figure 4: **CostGrad selection across environments.** Per-row: selection overlay at seven timesteps through one episode (bright = kept, dim = dropped, 50% kept throughout). CostGrad adapts to each environment's task structure: maze corridors on PointMaze, agent and wall on Wall, T-block and pusher on PushT, arm and object on MetaWorld. Decoded one-step predictions from the same sparse token sets are in Appendix C.

and yield planning success within evaluation noise. Planning-derived gradients therefore provide a more reliable selection signal than learned attention.

Probe-action robustness. CEM plans with action distributions far from $\boldsymbol{a}_0 = \mathbf{0}$, so a fixed probe action could in principle give unstable rankings. CostGrad rankings remain stable across 10 tested probe actions: $\rho = 0.918$ on AdaLN-Zero and 0.958 on matched-concat (Appendix I). On AdaLN-Zero, replacing the default zero-action probe with Gaussian-noise or CEM-mean probes changes Wall planning success by at most ± 5 pp (Appendix E). Stable token rankings do not guarantee reliable predictions after token removal; we next examine how sparse planning depends on the predictor's architecture (§3.3).

3.3 Sparse planning depends on selector–architecture compatibility

We evaluate selector–architecture compatibility on Wall and PointMaze, using three independently trained concat checkpoints per environment. The concat predictors share the encoder, training data, depth, heads, attention head dimension, MLP width, optimizer, and schedule with AdaLN-Zero. AdaLN supplies action-dependent scales, shifts, and gates at every block, with modulation parameters independent of the retained token set. Concat appends action features to each token at the input, before self-attention.

Across both environments, CostGrad has no consistent advantage over Random on concat: the paired success-rate gaps are -0.1 ± 4.9 pp on Wall and $+2.0 \pm 4.5$ pp on PointMaze (Appendix I, Table 5). This contrasts with the gains over Random on AdaLN-Zero (Table 1).

Table 2 contrasts full and sparse planning on Wall. Full-token success is comparable between concat and AdaLN-Zero (85.1% vs. 84.7%). Under sparsification, CostGrad beats Random by 25.0 pp and Full by 7.3 pp on AdaLN-Zero, whereas it falls 5.1 pp below Full on concat. We next examine how token removal changes the action-dependent predictions used by CEM.

Table 2: **Full and sparse planning on Wall.** AdaLN results are from Table 1; concat reports mean $\pm$ std across three training checkpoints, with 96 episodes per method per checkpoint. CG$-$Full > 0: sparsification is profitable; < 0: net harmful.

Predictor	Full	CostGrad	Random	CG$-$Rand	CG$-$Full
AdaLN-Zero	84.7 ± 2.6	92.0 ± 1.8	67.0 ± 3.6	$+25.0$	$+7.3$
Concat (matched)	85.1 ± 5.2	80.0 ± 2.9	80.2 ± 3.8	-0.1	-5.1

Table 3: **Multi-seed action-pathway diagnostic on Wall.** Mean $\pm$ sample standard deviation across three training seeds per architecture; K=128, 20 observations $\times$ 5 action probes per checkpoint. Selected/Random ratios below one indicate less drift than Random. α=0.8 uses 80% random anchors.

Predictor	Selection	KL random	KL selected	Sel. / random
AdaLN-Zero	CostGrad	0.265 ± 0.005	0.077 ± 0.009	$(0.29 \pm 0.03)\times$
AdaLN-Zero	α=0.8	0.265 ± 0.005	0.116 ± 0.009	$(0.44 \pm 0.02)\times$
Concat (matched)	CostGrad	0.287 ± 0.017	0.380 ± 0.030	$(1.33 \pm 0.17)\times$
Concat (matched)	α=0.8	0.287 ± 0.017	0.298 ± 0.016	$(1.04 \pm 0.11)\times$

Action-pathway drift. CEM ranks candidate actions by their predicted outcomes. Sparse Imagination [Chun et al., 2025] identifies a related failure: fixed token subsets can create blind spots that make candidate actions indistinguishable. We therefore examine whether token removal changes the spatial distribution of action effects across retained positions. Let $f_i(\mathcal{U}, \boldsymbol{a})$ denote the predictor's next-step visual output at position i using token set $\mathcal{U}$ and action $\boldsymbol{a}$. Holding context and proprioception fixed, define the action effect by

$$\boldsymbol{\delta}_i^{\mathcal{U}} = f_i(\mathcal{U}, \boldsymbol{a}) - f_i(\mathcal{U}, \mathbf{0}). \tag{3}$$

For the full set $\mathcal{P} = \{1, \ldots, P\}$ and kept set $\mathcal{S}$, normalize squared action effects over the same kept positions $i \in \mathcal{S}$:

$$p_i = \frac{\|\boldsymbol{\delta}_i^{\mathcal{S}}\|_2^2}{\sum_{j \in \mathcal{S}} \|\boldsymbol{\delta}_j^{\mathcal{S}}\|_2^2}, \qquad q_i = \frac{\|\boldsymbol{\delta}_i^{\mathcal{P}}\|_2^2}{\sum_{j \in \mathcal{S}} \|\boldsymbol{\delta}_j^{\mathcal{P}}\|_2^2}. \tag{4}$$

We measure drift as $D_{\mathrm{KL}}(p\|q) = \sum_{i \in \mathcal{S}} p_i \log(p_i/q_i)$. Using the same retained positions isolates redistribution from the absence of dropped outputs; normalization focuses on relative spatial magnitudes, discarding overall scale and vector direction. Random selection at the same token budget provides a reference for generic sparsification. The diagnostic does not itself bound planning error or establish that candidate-action rankings are preserved. See Appendix D for details.

Table 3 reports drift across three independently trained Wall checkpoints per architecture: CostGrad yields $(0.29 \pm 0.03)\times$ Random on AdaLN-Zero and $(1.33 \pm 0.17)\times$ on matched-concat, with the same direction in every checkpoint.

Coverage through random anchors. To examine whether the concentration of selected tokens matters, we replace part of the gradient-selected subset with random anchors while keeping K fixed. Mixing 20% top-CostGrad tokens with 80% random anchors (α=0.8) brings concat drift to $(1.04 \pm 0.11)\times$ Random, while AdaLN remains below Random at $(0.44 \pm 0.02)\times$ (Table 3). The smaller concat drift motivates broader coverage, but planning recovery is checkpoint-dependent (Appendix I); anchor mixing is therefore a diagnostic intervention, not a generally better selector.

Mechanistic interpretation. Our hypothesis is that AdaLN's repeated local conditioning reduces dependence of the action response on removed context, whereas concat relies on learned transformations of the action features supplied at the input; neither architecture guarantees this stability. Reversing the ranking does not resolve the failure: keeping the bottom-gradient tokens collapses planning on both predictors (Appendix I). Appendix M separates visual cost restriction from changes caused by sparse dynamics. For sparse planning, token relevance and the predictor's response to token removal must be evaluated together.

4 Related Work

World models and sparse planning. Token-based visual world models such as DINO-WM [Zhou et al., 2025] and JEPA-WMs [Terver et al., 2025] preserve a spatial grid of visual features for planning. This spatial structure is useful for goal-conditioned control, but it makes CEM rollouts expensive because every candidate action sequence is evaluated over many tokens. V-JEPA 2-AC [Assran et al., 2025] also uses an action-conditioned latent world model for image-goal robot planning. Among compact-latent approaches, DreamerV3 [Hafner et al., 2023] learns a policy from imagined trajectories, while TD-MPC2 [Hansen et al., 2024] and LeWorldModel [Maes et al., 2026] optimize actions in lower-dimensional latent spaces. COSTGRAD addresses the complementary setting: given a token-based world model, it reduces the number of spatial tokens used during planning without retraining the model.

Sparse Imagination [Chun et al., 2025] combines randomized grouped attention during training with random token selection during planning. It also analyzes spatial blind spots from importance-based selection that make candidate actions difficult to distinguish. COSTGRAD instead uses planning-cost gradients to select a goal-conditioned subset in a fixed pretrained predictor and studies its interaction with action conditioning. The approaches are complementary: sparse training improves tolerance to missing tokens, while planning-gradient selection identifies which tokens matter for the current goal.

Contemporaneous Scope-WM [Li et al., 2026] distills prediction-loss sensitivity into an action-conditioned selector and trains sparse dynamics with lightweight background updates. Its selector–background-update interaction provides complementary evidence for matching selection to sparse dynamics. COSTGRAD instead selects directly from the current goal's planning-cost gradients without retraining the predictor.

Token selection and importance estimation. Token-reduction methods for ViTs include token merging [Bolya et al., 2023], learned token pruning [Rao et al., 2021], class-token-driven reorganization [Liang et al., 2022], and patch dropout during training [Liu et al., 2023]. These methods were primarily developed for single-pass perception, where the reduced token set supports one forward computation. EfficientVLA [Yang et al., 2025] extends training-free token reduction to robot policies, combining task-aware attention scores with feature diversity. Sparse visual planning is different: the selected tokens must support many action-conditioned rollouts inside CEM.

Gradient saliency has a long history, from input-gradient visualizations [Simonyan et al., 2014] to axiomatic attribution methods such as Integrated Gradients [Sundararajan et al., 2017] and class-discriminative localization methods such as GradCAM [Selvaraju et al., 2017]. Sensitivity also guides computation reduction: Taylor-based pruning [Molchanov et al., 2017] approximates the loss change caused by removing convolutional filters, and ZOO-Prune [Kim et al., 2026] estimates projected-feature sensitivity for training-free visual-token pruning. COSTGRAD uses downstream planning-cost gradients to choose the token subset for repeated action-conditioned rollouts in a fixed predictor. Its contribution couples this goal-conditioned selection rule with an analysis of whether the selected subset preserves the predictor's action response (§3.3).

Decision relevance and predictor compatibility. Objective mismatch [Lambert et al., 2020], value-aware model learning [Farahmand et al., 2017], and value equivalence [Grimm et al., 2020] distinguish predictive fidelity from usefulness for downstream decisions. COSTGRAD applies this principle to inference-time token selection in a fixed predictor: the planning objective determines which representation supports action search, without changing model training.

Underspecification [D'Amour et al., 2022] describes how models with similar in-distribution metrics can differ under downstream use. The attention-as-explanation debate [Jain and Wallace, 2019, Wiegreffe and Pinter, 2019] likewise highlights the need to evaluate attention's usefulness in context, rather than assume universal reliability or unreliability. Our Wall seed study (§3) demonstrates an operational consequence: predictors with matched validation loss yield unstable attention-based selections and planning outcomes, while planning-gradient selection is more stable in the same setting.

Feature-wise linear modulation (FiLM) [Perez et al., 2018] is a standard way to condition neural representations; adaptive normalization variants such as AdaLN-Zero [Peebles and Xie, 2023] apply

this idea inside transformer blocks. DiT [Peebles and Xie, 2023] compares adaptive normalization with cross-attention and in-context conditioning through appended tokens. Our matched AdaLN-vs-channel-concat comparison (§3.3) adds stability under token removal as a criterion for action-conditioning design. In our experiments, AdaLN supports pure planning-gradient sparsification, whereas concat makes concentrated subsets unstable and can require coverage-preserving selectors.

5 Discussion and Limitations

When does sparse planning work? Our results identify two coupled design considerations for sparse planning: relevance to the control objective and compatibility with the predictor. COSTGRAD derives token importance from the current goal, since importance for prediction need not coincide with importance for control. In our matched comparison, COSTGRAD reduces measured action-effect drift relative to Random on AdaLN but increases it on concat, tracking its loss of planning advantage (§3.3). Token-selection performance should therefore be assessed jointly with the predictor at the intended token budget.

Limitations and open questions. Our evaluation covers four simulated environments, one encoder family (DINOv2 ViT-S/14) and one planner (CEM). Physical-system evaluation with perception noise, calibration errors, and model mismatch remains future work. Our selector–architecture finding is supported for the tested AdaLN and channel-wise concat predictors; its generality beyond this comparison remains untested. Future work should evaluate FiLM, cross-attention, and prefix-style conditioning to test whether action-pathway drift predicts sparse-planning performance across conditioning mechanisms, and explore sparse training to improve robustness to gradient-selected subsets. The strength of COSTGRAD is also environment-dependent: attention-based selection is competitive on PointMaze and PushT, while the gap is much larger on Wall and MetaWorld. All experiments use fixed pretrained predictors and inference-time selection; online adaptation and learned selectors remain future work. Selection uses a one-step probe, while CEM evaluates longer action sequences. Whether longer-horizon probes improve selection enough to justify their cost remains open.

Design implication. A planner is part of a world model's operating distribution. Token restriction can change the model's response to candidate actions, even when the retained tokens appear relevant to the task. World models intended for sparse planning should be evaluated under the token subsets and action searches used at deployment, because full-token prediction quality alone does not establish their suitability for sparse control.

Acknowledgements and Disclosures

The authors received no third-party financial or in-kind support for this work during the 36 months prior to submission. The authors declare no competing interests.

References

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations (ICLR)*, 2023.

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

Junha Chun, Youngjoon Jeong, and Taesup Kim. Sparse imagination for efficient visual world model planning. *arXiv preprint arXiv:2506.01392*, 2025.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61, 2022.

Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. *International Conference on Learning Representations (ICLR)*, 2024.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *NAACL-HLT*, 2019.

Youngeun Kim, Youjia Zhang, Huiling Liu, Aecheon Jung, Sunwoo Lee, and Sungeun Hong. ZOO-Prune: Training-free token pruning via zeroth-order gradient estimation in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39572–39582, 2026.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Learning for Dynamics and Control (L4DC)*, 2020.

Chunzheng Li, Zesheng Jia, Hongda Zhang, Jiaying Tang, Yuntian Wang, Siao Liu, and Jin Wang. Scope-WM: Scoped computation for efficient visual world models. *arXiv preprint arXiv:2609.33218*, 2026.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations (ICLR)*, 2022.

Yue Liu, Christos Matsoukas, Fredrik Strand, Hossein Azizpour, and Kevin Smith. PatchDropout: Economizing vision transformers using patch dropout. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations (ICLR)*, 2017.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. In *Transactions on Machine Learning Research (TMLR)*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *International Conference on Computer Vision (ICCV)*, 2023.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2014.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *arXiv preprint arXiv:2512.24497*, 2025.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *EMNLP-IJCNLP*, 2019.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. EfficientVLA: Training-free acceleration and compression for vision-language-action models. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations (ICLR)*, 2017. URL `https://arxiv.org/abs/1612.03928`.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 79115–79135. PMLR, 2025.

A Environments

We evaluate on four continuous-control benchmarks following Terver et al. [2025]. All four are visual planning tasks where the agent is given an initial frame and a goal frame and must produce an action sequence that drives the environment to the goal.

PointMaze (MZ). A 2D point navigates a U-Maze configuration drawn from Terver et al. [2025]'s suite (the underlying maze layout follows the D4RL maze suite, Fu et al., 2020). Action space: 2-dimensional continuous (x, y motor controls). Visual: a 224×224 rendering of the agent and maze walls. Episode horizon: 30 environment steps; success = reaching within 0.5 of the goal location.

Wall. A 2D agent navigates around a vertical wall with a single horizontal gap to reach a goal on the other side. Action space: 2-dimensional continuous; frame-skip 5, so each predictor step receives a 10-dim concatenated action. Visual: 224×224 rendering with the agent as a small red dot, the wall as a black bar, and the gap as the wall's discontinuity. Episode horizon: 30; success = L_2 distance to goal < 4.5.

PushT (PT). A 2D pusher pushes a T-shaped block to a target pose, following the PushT environment used by Chi et al. [2023]. Action space: 2-dimensional continuous (pusher target position). Visual: 224×224 rendering with a small blue pusher dot, a blue T-block, and a green target T over a plain light-gray background. Episode horizon: 30; success = L_2 error in the concatenated pusher/block positions < 20 and block-angle error $< \pi/9$.

MetaWorld (MW). A 3D 7-DoF arm performs the Reach task from MetaWorld's training suite [Yu et al., 2020]. Action space: 4-dimensional continuous (three Cartesian end-effector increments and one gripper command). Visual: 224×224 third-person view; proprio: 4-dimensional vector (end-effector position and gripper opening). Episode horizon: 100 environment steps; success per the MetaWorld task definition (end-effector–target distance ≤ 0.05).

B Implementation and hyperparameters

Encoder. Frozen DINOv2 ViT-S/14 [Oquab et al., 2024]; we do not fine-tune any encoder weights. Input image 224×224; patch size 14; produces a $16 \times 16 = 256$ grid of 384-dimensional tokens per frame.

Predictor. Six-block transformer with 400-dimensional internal embedding and 16 heads. RoPE positional encoding; GELU activation in the FFN, SiLU in the AdaLN modulation MLP; per-block AdaLN modulation parameters $(\boldsymbol{\gamma}_1, \boldsymbol{\beta}_1, \boldsymbol{g}_1, \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2, \boldsymbol{g}_2)$ derived from the action via a single linear layer. Action embedding dimension 400 (matching internal embedding); action tokens: 1. Total predictor parameters: ≈ 17.6M.

Concatenation predictor (matched-concat). Same depth (6 blocks), heads (16), attention head dimension (25), MLP hidden dimension (1600), GELU activation, RoPE positional encoding, training data, and optimizer as the AdaLN predictor. Action conditioning is replaced: a 10-dimensional frame-skip action vector is channel-wise concatenated to each visual token before the q/k/v projections (DINO-WM-style). The only architectural difference is the action pathway.

Planner. Cross-Entropy Method (CEM, Rubinstein, 1999): horizon H=6, 300 candidate sequences per iteration, 30 iterations, 10 elites for refit. Cost: L_2 between predicted final-step encoding and goal encoding (cosine and L_1 alternatives in Appendix F). $\alpha = 0.1$ proprio weight when applicable.

Pretraining. Our retrained predictors are trained for 20 epochs with AdamW, learning rate 5×10^{-4}, and batch size 8 per GPU, following the reference JEPA-WMs hyperparameters. Unless otherwise stated, main results use our retrained checkpoints; CostGrad itself is training-free and can also be applied directly to released checkpoints.

Objective and normalization. For retained positions $\mathcal{S}$, $K = |\mathcal{S}|$, the default final-step CEM cost is

$$J_{\mathcal{S}}(A) = \frac{1}{KD} \sum_{j \in \mathcal{S}} \|\hat{\boldsymbol{z}}_{H,j}(A) - \boldsymbol{z}_{g,j}\|_2^2 + \frac{\alpha}{d_p} \|\hat{\boldsymbol{p}}_H(A) - \boldsymbol{p}_g\|_2^2, \qquad \alpha = 0.1,$$

where $\boldsymbol{p}$ is the flattened proprioceptive representation and d_p its number of entries; the second term is omitted when proprioception is absent. Full uses all P positions. Both terms average over their own feature entries, so reducing K does not implicitly reduce the visual term's weight. The selection probe uses only the full-grid visual sum of squared errors; replacing this sum by a mean would uniformly rescale its gradients without changing their ranking. RGB inputs are scaled to $[0, 1]$ and standardized with ImageNet channel statistics before DINOv2 encoding; its output features receive no additional feature normalization. Proprioceptive inputs are standardized using dataset means and standard deviations before encoding, and actions use the training-normalized coordinates. Probe-based selectors use zero action and proprioceptive inputs in these coordinates; CEM rollout uses the observed proprioceptive input.

Attention aggregation. Let $A^{(h)}$ be the last block's attention matrix for head h, after the scaled query–key softmax with block-causal masking, and let $k(t, j)$ index spatial position j in context frame t. With M heads and T context frames, PREDATTN uses

$$s_j^{\text{attn}} = \frac{1}{MT} \sum_{h=1}^{M} \sum_{t=1}^{T} \sum_{q} A^{(h)}_{q,k(t,j)}.$$

The query sum includes all queries; only spatial keys are ranked. Heads and frames receive equal weight, with no entropy weighting.

Prediction-gradient target. For an observed context with $T \geq 2$, PREDGRAD computes

$$\mathcal{L}_{\text{pred}} = \|f(\boldsymbol{z}_{1:T-1}, \mathbf{0})_{\text{last}} - \text{stopgrad}(\boldsymbol{z}_T)\|_2^2, \qquad s_j^{\text{pred}} = \|\nabla_{\boldsymbol{z}_{T-1,j}} \mathcal{L}_{\text{pred}}\|_2.$$

Here f returns visual predictions and the subscript selects its last temporal output. The target is a detached encoding from observed history, not an outcome obtained by executing the probe action; the historical action that produced it is not substituted for the zero probe. Top-K spatial indices from this retrospective residual are applied to the current observation and goal.

Token-merging adaptation. ToMe partitions the current grid into checkerboard source and destination sets, matches each source to its most cosine-similar destination, and merges the $P - K$ highest-similarity source–destination pairs. Each group is replaced by the equal-weight average of its original features at the destination's original positional index. The mapping is computed once per planning step and applied to every observed context frame; subsequent rollout predicts only the surviving positions without rematching. The visual cost compares these predictions with the unmerged goal features at the same positions, averaging uniformly over surviving tokens without merge-size weights.

C Decoded one-step predictions from COSTGRAD-50% sparse tokens

Figure 5 shows the predictor's decoded next-step predictions from the COSTGRAD-selected 50% token subsets visualized in Figure 4. On all four environments, the prediction quality at 50% tokens is qualitatively faithful: agent and goal positions are recovered on PointMaze and Wall, the T-block and pusher poses are recovered on PushT, and the arm/object configuration is recovered on Meta-World. Visual artifacts from missing tokens are concentrated in background regions COSTGRAD chose to drop.



Figure 5: **Decoded one-step predictions from CostGrad-50% sparse tokens.** Per-row: the predictor's decoded $t{+}1$ output at seven timesteps through one episode, when only the top-128 CostGrad-scored tokens are kept (corresponding to the selection overlays in Figure 4).

D Action conditioning and token-removal sensitivity

We summarize the conditioning mechanisms and define the drift diagnostic used in §3.3. Let $\boldsymbol{z}_i$ denote a block's input token, $\boldsymbol{a}$ the action input, and $\mathcal{S}$ the retained spatial indices. The schematic block updates below omit temporal indexing and proprioceptive channels.

AdaLN-Zero block (predictor used in §2.1):

$$\boldsymbol{h}_i = \boldsymbol{z}_i + \boldsymbol{g}_1(\boldsymbol{a}) \odot \mathrm{Attn}_i\big(\{(1+\boldsymbol{\gamma}_1(\boldsymbol{a})) \odot \mathrm{LN}(\boldsymbol{z}_j) + \boldsymbol{\beta}_1(\boldsymbol{a}) : j \in \mathcal{S}\}\big), \tag{5}$$

$$T_i^{(\ell+1),\text{AdaLN-Zero}} = \boldsymbol{h}_i + \boldsymbol{g}_2(\boldsymbol{a}) \odot \mathrm{MLP}\big((1+\boldsymbol{\gamma}_2(\boldsymbol{a})) \odot \mathrm{LN}(\boldsymbol{h}_i) + \boldsymbol{\beta}_2(\boldsymbol{a})\big). \tag{6}$$

The action enters at six per-token modulation parameters $(\boldsymbol{\gamma}_1, \boldsymbol{\beta}_1, \boldsymbol{g}_1, \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2, \boldsymbol{g}_2)$, all functions of $\boldsymbol{a}$ alone, broadcast per-token. The gates $\boldsymbol{g}_1, \boldsymbol{g}_2$ are zero-initialized, so each block starts as the identity (the "Zero" in AdaLN-Zero). The action-dependent modulation is broadcast across tokens; this structure, rather than the zero initialization, is what §3.3 returns to when comparing AdaLN against concat conditioning.

Concat block (DINO-WM style; channel-wise feature concatenation): with $\boldsymbol{z}_i^{\text{aug}} := [\boldsymbol{z}_i; \pi(\boldsymbol{a})] \in \mathbb{R}^{D+D_a}$,

$$\boldsymbol{h}_i = \boldsymbol{z}_i^{\text{aug}} + \mathrm{Attn}_i\big(\{\mathrm{LN}(\boldsymbol{z}_j^{\text{aug}}) : j \in \mathcal{S}\}\big), \tag{7}$$

$$T_i^{(\ell+1),\text{concat}} = \boldsymbol{h}_i + \mathrm{MLP}(\mathrm{LN}(\boldsymbol{h}_i)). \tag{8}$$

The action enters once, channel-wise per token, before the q/k/v projections.

Dependence on the retained tokens. AdaLN's modulation parameters depend on the action alone, but its modulated tokens still enter self-attention. In both architectures, removing tokens changes attention normalization and aggregated values, so action effects at the block output can change. The conditioning mechanisms therefore motivate an empirical comparison of sensitivity to token removal; subset-independent modulation parameters alone do not establish output invariance.

Output-level action-pathway drift. Let $f_i(\mathcal{U}, \boldsymbol{a})$ be the predictor's next-step visual output at position i using token set $\mathcal{U}$. With context and proprioception fixed, define

$$\boldsymbol{\delta}_i^{\mathcal{U}} = f_i(\mathcal{U}, \boldsymbol{a}) - f_i(\mathcal{U}, \boldsymbol{0}). \tag{9}$$

Table 4: Wall planning success vs. choice of CEM cost function. The selector uses an L_2 probe throughout; its largest gain is under the L_1 CEM cost, where full-token planning performs worst.

Cost	Full	CostGrad (50%)	Δ
L_2	84.7	92.0	+7.3
Cosine	82.3	87.5	+5.2
L_1	47.9	70.8	+22.9

For $\mathcal{P} = \{1, \ldots, P\}$ and $i \in \mathcal{S}$, normalize the squared action effects of both predictors over the same retained positions:

$$p_i = \frac{\|\boldsymbol{\delta}_i^{\mathcal{S}}\|_2^2}{\sum_{j \in \mathcal{S}} \|\boldsymbol{\delta}_j^{\mathcal{S}}\|_2^2}, \qquad q_i = \frac{\|\boldsymbol{\delta}_i^{\mathcal{P}}\|_2^2}{\sum_{j \in \mathcal{S}} \|\boldsymbol{\delta}_j^{\mathcal{P}}\|_2^2}. \tag{10}$$

We report $D_{\mathrm{KL}}(p\|q) = \sum_{i \in \mathcal{S}} p_i \log(p_i/q_i)$, with 10^{-10} numerical stabilization in the implementation. This measures changes in the spatial distribution of squared action-effect magnitudes; it does not measure their vector directions or a common change in magnitude.

Diagnostic design. Subtracting the zero-action output removes action-independent offsets, so the comparison concerns the response to an action rather than raw prediction agreement. We compare finite action responses of the same predictor under full and sparse inputs. Summing squared responses over feature channels produces a nonnegative spatial map, analogous to the squared-activation maps used in attention transfer [Zagoruyko and Komodakis, 2017]. We normalize over the same retained positions to compare relative spatial allocation without counting missing positions or a uniform rescaling as drift. Our sum normalization and KL divergence differ from attention transfer's normalized-map distance: $D_{\mathrm{KL}}(p\|q)$ penalizes sparse-response mass where the full predictor assigns little relative mass. This is an output-response diagnostic, not a direct measurement of internal computational pathways or a guarantee of action-ranking preservation.

Empirical architecture interaction. Across three Wall training seeds per architecture, Table 3 reports a CostGrad-selected/Random drift ratio of 0.29 ± 0.03 for AdaLN-Zero and 1.33 ± 0.17 for matched-concat, with the direction holding for every checkpoint. The proposed conditioning-based explanation is discussed in §3.3.

E Probe action sensitivity (CostGrad and PredAttn)

Both CostGrad and PredAttn require a probe action sequence to compute their importance scores. Both use a zero-action probe by default, corresponding to the mean action in normalized training space and providing a neutral starting point before action search. For CostGrad, replacing zero with small Gaussian noise changes planning success by at most 5 pp across the four environments. A CEM-mean probe was additionally tested on Wall, also within 5 pp; we have not tested CEM-mean on every environment. The probe-stability of CostGrad's selection extends to the per-token level: rank Spearman across 10 probe actions is $\rho > 0.9$ on both AdaLN-Zero and the matched-concat predictor (Wall, seed 1; Appendix I).

F Cost function robustness

CostGrad's benefit is robust across the choice of planning cost function; in fact the benefit is largest when the cost is poorest. Table 4 reports Wall planning success with three cost functions: L_2 (default), cosine similarity, and L_1. This ablation varies the CEM planning cost while retaining the L_2 selection probe for CostGrad in every row.

Cross-cost selection IoU. In a separate diagnostic, we vary the selection-probe objective itself: top-128 Jaccard IoU between L_2 and cosine selections is 0.917; between L_2 and L_1 it is 0.841. The three probe objectives therefore select substantially overlapping token sets in this diagnostic.

G Matched compute baseline

A natural concern: CostGrad saves predictor compute by using fewer tokens, but the saved compute could alternatively be spent on more CEM iterations or candidates. We test this directly on Wall: CostGrad at 50% tokens with half the CEM iterations (15 instead of 30) achieves $92.0 \pm 5.1\%$ planning success across 3 eval seeds (per-seed: $89.58, 88.54, 97.92$) at $\sim 5\times$ overall wall-clock speedup ($2.6\times$ from token sparsity $\times$ $2\times$ from halving CEM iterations; Appendix K). The mean matches the standard CostGrad-50% rate ($92.0 \pm 1.8\%$, Table 1) and exceeds the 84.7% Full baseline by $+7.3$ pp. Eval-seed variance is wider here than for standard CostGrad-50% ($\sigma = 5.1$ vs. $\sigma = 1.8$), but the worst-case seed (88.54%) still beats Full.

Full at half CEM iters as a matched-compute reference. The natural reviewer comparison is whether halving *Full*'s CEM iterations would achieve a similar speedup–success trade-off without any token selection. Three eval seeds at Full (K=256) with 15 iterations on Wall give $84.38, 83.33, 85.42 = 84.4 \pm 1.0\%$, within eval-seed noise of the standard $84.7 \pm 2.6\%$ for $2\times$ wall-clock speedup ($88.6 \rightarrow 44.3$ s; predictor compute scales linearly with iter count and CostGrad adds no overhead in this case since it is not used). CostGrad-50%-with-half-iters beats Full-with-half-iters by $+7.6$ pp (92.0 vs. 84.4) at the same iteration budget and a further $2.6\times$ per-iter cost reduction. Token selection and increased search are therefore complementary: they address different sources of inefficiency.

H CostGrad operating curve

Figure 6 shows every measured CostGrad operating point on AdaLN-Zero, sweeping both token budget $K \in \{26, 64, 128, 192\}$ and CEM iteration count iters $\in \{10, 15, 20, 30\}$. Color encodes the token budget K cluster; marker shape encodes iter count ($\Diamond$=10, open $\bigcirc$=15, $\triangle$=20, filled $\bullet$=30). Thin lines connect points within each K cluster, in increasing iter order.

Two qualitative features support the practical claim that CostGrad "saturates" at the standard 50% operating point. First, planning success is roughly flat across $K \in \{64, 128, 192\}$ at iters$=$ 30 (85.9%, 92.0%, 92.7%), so the marginal benefit of doubling tokens beyond K=64 is small. Second, within each K cluster, halving iterations ($30 \rightarrow 15$) does not consistently reduce success — measurements at iters$\in \{10, 15, 20\}$ scatter around the iters$=$ 30 mean rather than tracking it monotonically. Reducing CEM iterations therefore acts mostly as a horizontal slide along the time axis at roughly fixed success, until iterations get very low.

I Additional diagnostic checks

Sparse planning retains K=128 of 256 tokens. Table 5 gives the per-training-seed concat results on Wall and PointMaze summarized in §3.3; the remaining checks use Wall. Appendix D defines the action-pathway diagnostic, and Table 3 reports its multi-seed results.

Probe-action stability. Across 30 observations and 45 pairs of 10 probe actions per observation, per-token rank Spearman is 0.918 ± 0.058 on AdaLN-Zero and 0.958 ± 0.039 on matched-concat (one checkpoint per architecture; mean $\pm$ standard deviation over all 1,350 pairwise correlations). The probes comprise zero, four positive unit-axis directions, and five Gaussian samples with σ=0.5. Both predictors maintain stable rankings under these probes despite their different responses to token removal.

Inverted-CostGrad sanity check. Keeping the bottom-K tokens by gradient norm yields $0/60$ successful episodes on AdaLN-Zero (the run was stopped after 60 of 96 planned episodes) and $1/96$ (1.04%) on matched-concat. Reversing the ranking therefore collapses planning on both predictors; it does not recover CostGrad's advantage over Random on concat.

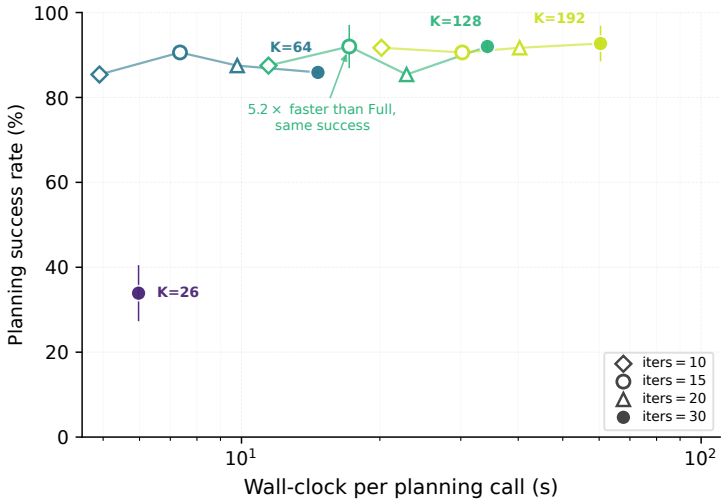


Figure 6: **CostGrad operating curve on Wall AdaLN-Zero.** Every measured (K, iters) CostGrad cell. Color encodes K cluster (sequential viridis: purple $K{=}26$, teal $K{=}64$, green $K{=}128$, yellow $K{=}192$); marker shape encodes iter count ($\Diamond{=}10$, open $\bigcirc{=}15$, $\triangle{=}20$, filled $\bullet{=}30$). Thin lines connect points within each K cluster (sorted by iter count). K labels at the iters$=30$ marker of each cluster. The headline operating point ($K{=}128$ with iters$=15$, $\sim$17 s, $92.0 \pm 5.1\%$) gives a $5.2\times$ wall-clock speedup over Full at higher mean success.

Table 5: **Concat planning across environments and training seeds.** Paired CostGrad–Random success-rate gaps (pp), with $K{=}128$ of 256 tokens and 96 episodes per method per checkpoint. Mean $\pm$ std is across the three training seeds.

Environment	Seed 1	Seed 2	Seed 3	Mean $\pm$ std
Wall	$+2.1$	$+3.2$	-5.7	-0.1 ± 4.9
PointMaze	$+4.1$	-3.2	$+5.2$	$+2.0 \pm 4.5$

Anchor mixing. The selector keeps $(1{-}\alpha)K$ top-CostGrad tokens and fills the remaining αK slots uniformly at random from the unselected tokens. At $\alpha{=}0.8$, planning improves over Random on two of three matched-concat training checkpoints. On the first checkpoint, success is $94.1 \pm 2.4\%$ versus Random's $79.5 \pm 2.1\%$ (three evaluation seeds). On the other two checkpoints, the respective anchor/Random results are $89.6/76.0\%$ and $76.0/83.3\%$ (one evaluation seed each); all evaluations use 96 episodes per seed. Together with Table 3, these results show that restoring drift toward the Random baseline does not guarantee a planning gain. We therefore use anchor mixing as a diagnostic of selector–architecture compatibility; its planning benefit is checkpoint-dependent.

J Quadrant analysis methodology

For each environment we compute, on 30 observations, the per-token prediction error $e_j = \|\text{pred}_j - \text{target}_j\|^2$ and the CostGrad score s_j. For each observation, the easy-but-critical intersection contains tokens in the bottom 25% of prediction error and the top 25% of CostGrad score. The reverse intersection contains tokens in the top 25% of prediction error and the bottom 25% of CostGrad score; these two intersections do not partition all 256 tokens. Under independence, the expected count in each intersection is $0.25 \times 0.25 \times 256 = 16$. The easy-but-critical intersection measures disagreement between prediction error and planning sensitivity: tokens with low prediction error but high planning-cost gradient.

PushT has almost twice the independence-reference count of easy-but-critical tokens; Wall, MZ, and MW have fewer than expected. This supports the interpretation that predictable motion can still matter for control: on PushT, the pusher and T-block can have low prediction error while receiving high planning-gradient scores. Depletion in the other environments describes this particular intersection,

Table 6: Easy-but-critical token counts per environment (30 obs each, P=256): bottom-quartile prediction error intersected with top-quartile CostGrad score. Counts are mean $\pm$ std; the independence reference is 16. PushT is the only environment with enrichment above this reference; the other three are depleted.

Environment	Easy + critical (count)	Multiple of expected
PushT (PT)	$\mathbf{31.9 \pm 4.3}$	$\mathbf{1.99}\times$
PointMaze (MZ)	10.7 ± 2.8	$0.67\times$
Wall	8.5 ± 4.3	$0.53\times$
MetaWorld (MW)	3.2 ± 2.2	$0.20\times$

Table 7: Timing breakdown for one CEM planning step (300 candidates $\times$ 30 iterations $\times$ $H = 6$). CostGrad's extra forward+backward is a one-time cost amortized over 30 CEM iterations, while the per-iteration cost drops by $2.61\times$ from the predictor-compute reduction. The reported total includes predictor rollout time and CostGrad overhead; the shared 7.64 ms encoder cost is shown separately and is negligible at this scale. Net wall-clock speedup at $K = 128$ vs. $K = 256$ is $2.60\times$.

Component	K=256 (Full)	K=128 (CostGrad)
Encode context + goal (ms)	7.64	7.64
CostGrad forward + backward (ms)	—	27.53
Predictor forward at batch 300 (ms)	492.25	188.90
Per CEM iter ($\times H$=6, ms)	2953.51	1133.42
Total predictor over CEM step ($\times$30 iters, s)	88.61	34.00
Planning call total, excl. shared encoder (s)	**88.61**	**34.04**

rather than establishing agreement between the complete token rankings. The reverse intersection is below the independence reference in every environment ($0.91\times$ on PushT; 0.20–$0.87\times$ elsewhere), so it does not show the same enrichment.

K Timing breakdown

We measure per-component latency on the AdaLN-Zero predictor using the Wall checkpoint and a single H100 GPU, mean $\pm$ std over 5 warm-cache trials with 3 warmups discarded.

The $2.60\times$ wall-clock speedup exceeds the naive $2\times$ that linear scaling in token count would predict. The reason is that predictor compute mixes attention ($O(N^2)$ in token count) and MLPs ($O(N)$): at $N = 128$ versus $N = 256$, attention flops drop by $4\times$ while MLP flops drop by $2\times$, and the weighted combination on this architecture lands at $\sim 2.6\times$. The CostGrad forward+backward overhead (27.53 ms) amortizes to $\sim$1 ms per CEM iteration over 30 iterations, an order of magnitude smaller than the per-iteration savings ($\sim$ 1820 ms), so the overhead is operationally invisible.

L Public-checkpoint control

We evaluate CostGrad on released JEPA-WMs checkpoints [Terver et al., 2025] without retraining (Table 8).

Table 8: Planning success (%) on released checkpoints without retraining. Mean $\pm$ standard deviation over three evaluation seeds, with 96 episodes per seed. Full uses all 256 tokens; CostGrad and Random keep 128 (50%).

Environment	Full	CostGrad (50%)	Random (50%)
PointMaze	87.7 ± 7.7	86.3 ± 11.9	70.7 ± 15.1
Wall	87.0 ± 2.8	87.0 ± 2.2	75.7 ± 4.6
PushT	66.3 ± 2.6	67.3 ± 3.4	21.0 ± 9.0
MetaWorld (Reach)	58.3 ± 10.0	63.2 ± 5.2	20.8 ± 9.0

M An analytical perspective on token selection

The arguments below concern the visual squared-distance cost; the evaluated planner also includes weighted proprioceptive error.

Local relevance. At a fixed probe, write $f = f(\boldsymbol{z}, \boldsymbol{a}_0)$ and $G_j = \partial f / \partial \boldsymbol{z}_j$. With fixed prediction target $\boldsymbol{z}_{\text{next}}$ and goal $\boldsymbol{z}_g$, squared losses give

$$\nabla_{\boldsymbol{z}_j} L_{\text{plan}} = 2G_j^\top (f - \boldsymbol{z}_g), \qquad \nabla_{\boldsymbol{z}_j} L_{\text{pred}} = 2G_j^\top (f - \boldsymbol{z}_{\text{next}}). \tag{11}$$

Exact prediction makes the second gradient zero without requiring the first to vanish. For the fixed full-token computation, let $g_j = \nabla_{\boldsymbol{z}_j} L_{\text{plan}}$ and let $\mathcal{S}$ contain the K retained tokens. Perturb only discarded tokens, independently within $\|\Delta \boldsymbol{z}_j\|_2 \le \eta$. Then

$$\max_{\{\|\Delta \boldsymbol{z}_j\|_2 \le \eta:\, j \notin \mathcal{S}\}} \left| \sum_{j \notin \mathcal{S}} g_j^\top \Delta \boldsymbol{z}_j \right| = \eta \sum_{j \notin \mathcal{S}} \|g_j\|_2. \tag{12}$$

Cauchy–Schwarz gives the upper bound; aligned perturbations attain it. Keeping the K largest norms minimizes this linearized criterion. Physical token removal also changes attention and subsequent rollouts, which this local perturbation model does not represent.

Exact cost decomposition. Fix an observation, goal, horizon H, and token set $\mathcal{S}$ shared across action candidates. Let $\Phi_{\mathcal{P}}(A)$ and $\Phi_{\mathcal{S}}(A)$ denote the full and sparse terminal predictions for action sequence A, with $\mathcal{P} = \{1, \dots, P\}$. For this visual component, sums and means induce the same ranking because the token count is fixed across candidates. Define

$$\begin{aligned} J(A) &= \sum_{i \in \mathcal{P}} \|\Phi_{\mathcal{P},i}(A) - \boldsymbol{z}_{g,i}\|_2^2, \\ J_{\mathcal{S}}^{\text{oracle}}(A) &= \sum_{i \in \mathcal{S}} \|\Phi_{\mathcal{P},i}(A) - \boldsymbol{z}_{g,i}\|_2^2, \qquad J_{\mathcal{S}}(A) = \sum_{i \in \mathcal{S}} \|\Phi_{\mathcal{S},i}(A) - \boldsymbol{z}_{g,i}\|_2^2. \end{aligned} \tag{13}$$

The oracle cost retains the selected positions but uses full-token dynamics. Set $D_{\mathcal{S}} = J - J_{\mathcal{S}}^{\text{oracle}}$ (discarded cost) and $E_{\mathcal{S}} = J_{\mathcal{S}} - J_{\mathcal{S}}^{\text{oracle}}$ (change from sparse dynamics). Exactly,

$$J_{\mathcal{S}}(A) - J(A) = E_{\mathcal{S}}(A) - D_{\mathcal{S}}(A). \tag{14}$$

Sparse planning therefore changes both which positions enter the cost and the predictions at retained positions. Only action-dependent variation in $E_{\mathcal{S}} - D_{\mathcal{S}}$ can change candidate rankings; an action-independent offset preserves them.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and §1 state (i) CostGrad matches or exceeds full-token planning at 50% sparsity on 3 of 4 environments at $2.6\times$ speedup; (ii) saved compute is reinvestable for $\sim 5\times$ total speedup; (iii) the result is architecture-dependent — replacing AdaLN with matched concatenation conditioning eliminates the advantage across three matched-concat training checkpoints (CG–Random -0.1 ± 4.9 pp); (iv) we identify action-pathway stability under token removal as the measured architectural discriminator. Claim (i) is supported by Table 1; claim (ii) by Figure 3; claims (iii)–(iv) by Tables 2 and 3, with the conditioning mechanisms and drift diagnostic described in §3.3 and Appendix D.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: §5 has a dedicated Limitations paragraph (four environments, one encoder family, one planner, one main sparsity level; only two conditioning mechanisms tested; no online predictor adaptation). Open questions are listed separately.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Appendix M states the assumptions and derivation for the local-sensitivity argument and derives the planning/prediction gradient identities and visual-cost decomposition. It distinguishes fixed-computation sensitivity from physical token removal.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results?

   Answer: [Yes]

   Justification: §2 specifies the predictor (depth, heads, activation, conditioning, RoPE), planner (CEM with 300 candidates, 30 iterations, horizon $H = 6$), and the CostGrad algorithm (Algorithm 1). Appendix B provides full hyperparameters. Appendix A describes the four environments. Code release is described under question 5 (Open access to data and code).

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results?

   Answer: [No]

   Justification: We will release the codebase (training, evaluation, and analysis scripts) at camera-ready time. The four benchmarks (PointMaze, Wall, PushT, MetaWorld) are all available through the public DINO-WM [Zhou et al., 2025] and JEPA-WMs [Terver et al., 2025] repositories with open-source datasets and trajectory generators. CostGrad itself is training-free and can be applied to publicly released JEPA-WMs checkpoints; our released code will include the scripts needed to run this drop-in evaluation.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details necessary to understand the results?

   Answer: [Yes]

   Justification: §2 specifies the planner setup (CEM hyperparameters), Appendix B provides the full predictor and pretraining schedule (AdamW, $\mathrm{lr}{=}5 \times 10^{-4}$, batch size 8 per GPU, 20 pretraining epochs at the same hyperparameters as Terver et al. [2025]), and Appendix A describes the environments, action spaces, episode horizons, and success criteria.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined?

   Answer: [Yes]

   Justification: Table 1 and Figure 3 report mean $\pm$ std across 3 evaluation seeds (96 episodes per seed) at $K{=}128$ and $K{=}256$. Table 2 reports 3 eval seeds for AdaLN-Zero. Table 3 reports mean $\pm$ sample standard deviation across three training seeds per architecture. Figure 2 shows shaded ± 1 std bands across 3 evaluation seeds at non-default keep ratios. Variance source is identified in each caption as eval-seed (or training-seed, where noted) RNG.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on compute resources?

   Answer: [Yes]

   Justification: Appendix K reports per-component timing on a single NVIDIA H100 GPU: encode 7.6 ms, predictor forward 493 ms (Full) / 189 ms (CostGrad-50%), end-to-end planning step 88.6 s (Full) vs. 34.0 s (CostGrad-50%). Predictor pretraining took $\sim$ 24 hours per seed per environment on a single A100; full evaluation of one (predictor, method) cell takes $\sim$ 1.5–3 hours. Total project compute (including failed/preliminary experiments and ablations not in the paper) is approximately 400 GPU-days.

9. **Code of ethics**

   Question: Does the research conform with the NeurIPS Code of Ethics?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and the work is in compliance. The research uses only synthetic simulation environments (no human subjects, no scraped data, no biometric or sensitive personal data), and the released artifacts (code and analysis scripts) pose no clear misuse pathway.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive and negative societal impacts?

    Answer: [Yes]

    Justification: The work is a methodological contribution to inference-time compute reduction in token-based world models for continuous-control planning. The main positive impact is improved efficiency for simulation-based planning and model-based control research. The main potential risk is that more efficient planning could lower the cost of deploying autonomous systems; however, our experiments are simulation-only, use standard benchmark environments, and do not release new high-risk models, datasets, or application-specific autonomous capabilities.

11. **Safeguards**

    Question: Does the paper describe safeguards for responsible release of high-risk data or models?

    Answer: [N/A]

    Justification: The paper does not release pre-trained generative models with high misuse risk. The proposed token selector CostGrad is a $\sim$ 100-line training-free inference-time procedure applied to existing world-model checkpoints; it does not introduce new generative capabilities.

12. **Licenses for existing assets**

    Question: Are creators of existing assets properly credited and licenses respected?

    Answer: [Yes]

    Justification: We cite DINOv2 [Oquab et al., 2024] (Apache-2.0), DINO-WM [Zhou et al., 2025] (MIT-style), JEPA-WMs [Terver et al., 2025] (open release), MetaWorld [Yu et al., 2020] (MIT), and the ToMe baseline [Bolya et al., 2023] (CC-BY-NC). All assets are used within their respective license terms; no proprietary or restricted assets are used.

13. **New assets**

    Question: Are new assets introduced in the paper well documented?

    Answer: [Yes]

    Justification: The new asset is the CostGrad selector and analysis scripts. Algorithm 1 provides the full procedure; Appendix B specifies all hyperparameters; the released codebase will include configs, training scripts, and evaluation scripts for every reported experiment.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, are details provided?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects.

15. **Institutional review board (IRB) approvals**

    Question: Does the paper describe risks and IRB review for human-subjects research?

    Answer: [N/A]

    Justification: The paper does not involve human subjects or any data derived from human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs as a core methodological component?

    Answer: [N/A]

    Justification: LLMs are not used as a component of the proposed method or the experimental pipeline. The vision encoder is DINOv2 (a vision transformer trained with self-supervised distillation, not a language model); the predictor and planner do not involve LLMs.