TOKENGAZER: LEARNING WHERE TO LOOK IN TOKEN SPACE FOR VISUOMOTOR CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

In robot policies, using dense tokens from pretrained vision transformers, rather than a compressed global representation, preserves rich spatial structure and enables spatial reasoning and strong performance. However, dense representations come with two problems. First, policies that consume all tokens can be computationally expensive to train and can be slow at inference time. Second, dense token sets typically carry task-irrelevant information, such as background details, noise, clutter, and distractors. Conditioning on such redundant token sets can hurt policy generalization. In this work, we introduce TOKENGAZER, a policy that selects a small subset of tokens carrying task-relevant information, minimizing redundant information passed downstream while preserving the benefits of pretrained dense features and avoiding the need to fine-tune large model encoders. A key challenge is choosing the right training signal for this selection: a reconstruction objective rewards tokens that preserve visual fidelity rather than ones useful for control. Instead, we reward our token selection policy with a behavior cloning objective on a downstream action head conditioned on the selected tokens. We train the token policy and the action head alternately. Our policies conditioned on this sparse token subset outperform or at least match full-token-conditioned policy performance while processing far fewer tokens per step. This trend is more evident when the robot encounters out-of-distribution scenes. Code is available at `https://anonymous.4open.science/r/tokengazer-release-E075`.

1 INTRODUCTION

The choice of visual representation is crucial for visuomotor policy learning. Training from scratch can lead to overfitting to the training scenes, while fine-tuning a pretrained vision encoder such as DINOv2 (Oquab et al., 2023) on limited demonstration data can degrade the encoder's generalization. By contrast, using embeddings from a frozen encoder preserves the generalization learned from large, diverse image datasets. These features tend to transfer across tasks without task-specific training. Such encoders represent an image as hundreds of dense patch embeddings per frame, each a semantically rich representation of a local image region. Beyond generalization benefits, these tokens can represent the spatial structure manipulation tasks require (Zhou et al., 2024; Baldassarre et al., 2025).

However, it is unclear how to effectively leverage dense tokens from foundation model. Typically, practitioners pool the patch grid into one or a few vectors, or keep only a global token, which discards the spatial detail needed to localize objects, grippers, and their interactions. Recent work has shown that attending directly to the full grid of patch tokens substantially outperforms such compressed representations. However, a transformer conditioned on the full patch grid pays the computational cost of every token at every control step, even though many tokens cover background pixels that contribute little to the current action.

Reducing the number of visual tokens is a well-studied problem, but existing approaches do not fit this setting. Token pruning and merging methods for vision transformers (Rao et al., 2021; Bolya et al., 2023) operate inside the encoder, typically require training or modifying it, and decide which tokens to keep from image-level cues such as class-token attention or feature similarity. These cues reflect visual saliency rather than relevance to the next action, as required in robotics. Methods such as Perceiver (Jaegle et al., 2021) use cross-attention to summarize tokens in a new set of latent

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.

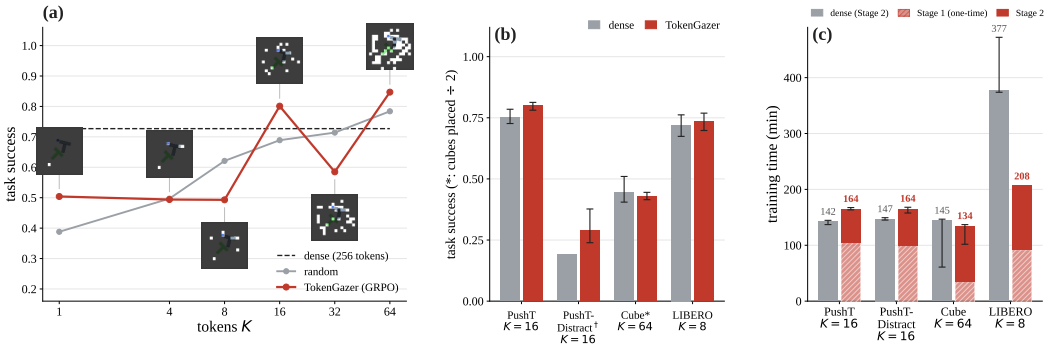

Figure 1: (a) Clean PushT success vs. token budget K; thumbnails show the selected patches. (b) Success of dense and TOKENGAZER on the same four configurations as (c), with 95% CIs (†single seed; *Cube as the fraction of cubes placed). (c) Training time: one-time Stage 1 (hatched) plus Stage 2 for TOKENGAZER, Stage 2 for dense; whiskers show the range over runs.

tokens. In robot learning, prior work on dense representations has likewise focused on compressing them (Pei et al., 2026). We argue that selecting manipulation-relevant tokens from a foundational vision model can combine the generalizability of such encoders with the structure and information the policy needs to solve the task, while increasing computational efficiency. In this paper, we propose a system to select task-relevant tokens called TOKENGAZER. In our approach, a gaze policy learns to select K patch tokens from each frame, and a downstream control policy conditions only on this subset, so computation scales with the number of selected tokens rather than the full patch grid. The selection policy is naturally a decision-making problem. The reward function for selecting useful tokens cannot be based on the image alone, but also on how well a policy conditioned on those tokens performs. Figure 1 illustrates this trade-off: conventional dense or pooled conditioning loses performance as the token budget shrinks, while TOKENGAZER's learned selection does not. Our contributions are:

- We introduce TOKENGAZER, which formulates token selection for visuomotor policies built on frozen dense features as a decision-making problem. A lightweight gaze policy selects K of the N DINOv2 patch tokens per frame, and a downstream diffusion policy conditions only on the selected tokens, so its sequence length and compute scale with K rather than with the full patch grid.
- We propose a task-driven reward for learning which tokens to selects, which scores each candidate subset by how accurately a frozen action-prediction model predicts the expert action from the selected tokens alone. We optimize the gaze with both GRPO and a Gumbel-softmax relaxation and compare the two throughout.
- We show that learned selection matches or exceeds full-grid conditioning with far fewer tokens. On PushT, 16 selected tokens outperform the full 256-token policy and every same-budget baseline, and on Cube, 64 tokens reach parity with the dense policy at a quarter of the tokens and 1.46× lower training cost.
- We show that selection is more robust to visual clutter than dense conditioning. On PushT-Distract, policies trained with distractors and deployed on clean scenes exceed the dense policy at five of six token budgets.

2 PRELIMINARIES

2.1 PROBLEM SETUP

We consider imitation learning from a dataset of expert demonstrations $\mathcal{D} = \{(o_t, \mathbf{a}_t)\}$, where o_t is an RGB observation, and $\mathbf{a}_t = (a_t, \ldots, a_{t+H-1})$ is a chunk of the next H expert actions. The goal is to learn a policy $\pi(\mathbf{a}_t \mid o_t)$ that predicts the action chunk from a conditioning context c_t derived from the observation.

Throughout, observations are represented by a frozen, pretrained vision transformer f. The encoder divides each image into a grid of patches and maps it to a set of N patch embeddings,

$$E_t = f(O_t) = \{e_1, \ldots, e_N\}, \qquad e_i \in \mathbb{R}^d.$$

Each e_i describes a local region of the image, and its index i identifies that region's position in the grid. The encoder is never updated, so E_t can be computed once and cached. A dense policy conditions on all of E_t; the question this paper studies is which subset of it the policy actually needs.

2.2 Diffusion Policies

We use diffusion policies (Chi et al., 2023), which model the action distribution through a learned denoising process. Starting from a clean action chunk $\mathbf{a}^0$, a forward process adds Gaussian noise over T diffusion steps,

$$\mathbf{a}^\tau = \sqrt{\bar{\alpha}_\tau}\,\mathbf{a}^0 + \sqrt{1-\bar{\alpha}_\tau}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I),$$

where $\tau \in \{1, \ldots, T\}$ and $\bar{\alpha}_\tau$ is a fixed noise schedule. A denoising network ϵ_ϕ is trained to recover the added noise given the noisy chunk, the step, and the conditioning context,

$$\mathcal{L}_{\mathrm{DP}}(\phi) = \mathbb{E}_{(c_t, \mathbf{a}^0)\sim\mathcal{D},\, \tau,\, \epsilon}\left[\left\|\epsilon - \epsilon_\phi(\mathbf{a}^\tau, \tau, c_t)\right\|^2\right].$$

At inference, an action chunk is generated by starting from Gaussian noise and applying the learned denoiser iteratively for T steps. When ϵ_ϕ is a transformer, the conditioning context enters as a sequence of tokens, and the cost of each denoising step grows with the length of that sequence. For a dense patch policy, c_t contains all N patch tokens, so every step of every control cycle pays for all of them. Policies are executed in a closed-loop, receding-horizon fashion: the policy predicts a chunk of H actions, the robot executes the first $h \le H$ of them, and the policy replans from the new observation. Smaller h makes the policy more reactive at the cost of more frequent inference.

2.3 Group Relative Policy Optimization

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a policy-gradient method originally introduced for fine-tuning language models. For each input x, it samples a group of G outputs $y^{(1)}, \ldots, y^{(G)}$ from the current policy π_θ, scores each with a reward $r^{(g)}$, and normalizes the rewards within the group,

$$A^{(g)} = \frac{r^{(g)} - \mathrm{mean}\big(r^{(1)}, \ldots, r^{(G)}\big)}{\mathrm{std}\big(r^{(1)}, \ldots, r^{(G)}\big) + \delta},$$

where δ is a small constant for numerical stability. Because the baseline comes from the group itself, GRPO needs no learned value function. The policy is updated to maximize

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{g=1}^{G} \min\Big(\rho^{(g)}A^{(g)},\ \mathrm{clip}\big(\rho^{(g)}, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}\big)A^{(g)}\Big)\right], \quad \rho^{(g)} = \frac{\pi_\theta(y^{(g)} \mid x)}{\pi_{\theta_{\mathrm{old}}}(y^{(g)} \mid x)}.$$

In our setting, the input x is a frame's patch embeddings E_t, an output $y^{(g)}$ is a subset of K token indices, and the reward is computed by a frozen action-prediction model.

2.4 Gumbel-Softmax Relaxation

As an alternative to policy-gradient optimization, discrete selection can be made differentiable. The Gumbel-max trick samples from a categorical distribution with probabilities p as $\arg\max_i (\log p_i + \gamma_i)$, where $\gamma_i = -\log(-\log u_i)$ and $u_i \sim \mathrm{Uniform}(0, 1)$. The Gumbel-softmax relaxation (Jang et al., 2016) replaces the $\arg\max$ with a temperature-controlled softmax,

$$y_i = \frac{\exp\big((\log p_i + \gamma_i)/\lambda\big)}{\sum_{j=1}^{N}\exp\big((\log p_j + \gamma_j)/\lambda\big)},$$

which approaches a one-hot sample as the temperature $\lambda \to 0$. Because y is differentiable with respect to p, gradients from a downstream loss can flow directly into the selector. In our setting, this lets the gaze policy be trained by backpropagating the action-prediction loss through the relaxed selection, in place of the sampled rewards used by GRPO.

3 Related Work

Visual representations for robot learning. Learning visual representations for control has progressed from end-to-end visuomotor policies and compact spatial keypoints to structured feature maps that preserve object geometry and interactions (Levine et al., 2016; Finn et al., 2016; Zeng et al., 2021). More recently, visual pretraining on large image and human-video datasets has improved downstream data efficiency through contrastive, masked-modeling, and value-based objectives (Nair et al., 2023; Xiao et al., 2022; Radosavovic et al., 2023; Ma et al., 2023; Majumdar et al., 2023). Spatially structured pretrained features also support predictive modeling, as demonstrated by DINO-WM and Back to the Features (Oquab et al., 2023; Zhou et al., 2024; Baldassarre et al., 2025). Most closely related, Patch Policy enables lightweight transformer policies to consume dense pretrained patch embeddings directly and demonstrates advantages over globally pooled representations (Zhou et al., 2026). We build on this observation but investigate whether the controller needs the complete patch grid. Rather than compressing every observation globally or retaining every patch, TokenGazer learns a small subset of frozen embeddings for downstream control.

Task-driven representations and attention for manipulation. Action supervision can shape visual representations toward information useful for manipulation, while spatial attention and learned aggregation allow policies to emphasize relevant scene content (Levine et al., 2016; Brohan et al., 2023; Shridhar et al., 2023). In particular, RT-1 incorporates TokenLearner to construct compact visual tokens under an action-prediction objective (Ryoo et al., 2021; Brohan et al., 2023). Task-driven discrete attention also has a longstanding connection to reinforcement learning, including policies that learn where to acquire image glimpses (Mnih et al., 2014). Our distinction is therefore not task-supervised attention alone. We select original frozen patch embeddings rather than learning aggregate representations or changing image acquisition. Selection is guided by how accurately an action head conditioned on the selected patches predicts demonstrated actions, allowing the visual selection strategy and action predictor to adapt to one another. We compare GRPO with a Gumbel–Softmax relaxation as alternative ways to optimize this discrete selection problem (Shao et al., 2024; Jang et al., 2016; Maddison et al., 2016).

Token reduction for efficient transformers. Token reduction exploits redundancy to shorten the sequences processed by transformers. Existing approaches include learned importance prediction, attention-guided pruning, adaptive token computation, and similarity-based merging (Rao et al., 2021; Liang et al., 2022; Yin et al., 2021; Fayyaz et al., 2021; Bolya et al., 2023). Other methods replace the input sequence with compact learned representations through latent bottlenecks or spatial aggregation (Jaegle et al., 2021; Ryoo et al., 2021; Alayrac et al., 2022). Reconstruction provides another selection signal. Building on masked image and video modeling, adaptive sampling and ranking methods identify informative patches through reconstruction feedback (He et al., 2021; Tong et al., 2022; Bandara et al., 2023; Luo et al., 2024; Rai et al., 2025). Related training-free approaches select tokens to preserve the original feature subspace (Lee et al., 2026). These paradigms differ in whether they retain or synthesize tokens, require additional training, and prioritize task accuracy or representation preservation. Our approach retains original embeddings and learns their selection from downstream action prediction rather than reconstruction, while leaving the pretrained encoder unchanged.

4 Method: TokenGazer

We consider a visuomotor policy that operates on dense patch representations produced by a frozen vision transformer. At time step t, using the notation of Section 2.1, a frozen encoder f maps the observation O_t into a set of N patch embeddings,

$$E_t = f(O_t) = \{e_1, \ldots, e_N\}, \qquad e_i \in \mathbb{R}^d, \tag{1}$$

where N is the number of patch tokens and d is the embedding dimension. Rather than conditioning a policy on the complete token grid, our goal is to determine whether a much smaller subset of K tokens, where $K \ll N$, contains the information required for control. Formally, we seek a subset

$$E_t^* = \{e_{i_1}, \ldots, e_{i_K}\}, \tag{2}$$

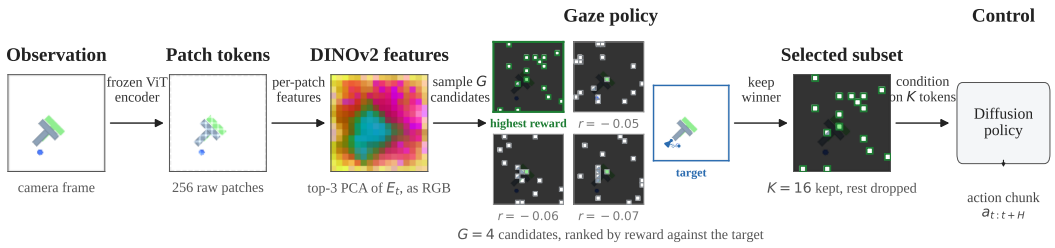

Figure 2: The TOKENGAZER pipeline on a real held-out frame: observation, patch tokens, DINOv2 feature space (top-3 PCA as RGB), the gaze policy ranking $G{=}4$ sampled candidate subsets by reward against the real expert action for this frame (target, blue), the winning subset, and the control policy's predicted action. Candidates, rewards, and the target are all a trained policy's actual output on this frame, not illustrative.

such that a control policy conditioned only on E_t^* performs as well as, or better than, one conditioned on all N tokens. The challenge is that the quality of a subset cannot be judged from the image alone: it depends on how well a downstream control policy performs when trained and evaluated using only those selected tokens. We therefore decompose the problem into two stages: first learning where to look through a gaze policy, and then learning how to act using only the selected tokens.

Frame encoding. Each camera view is encoded once by a frozen DINOv2 vision transformer (Oquab et al., 2023), and never fine-tuned. Throughout this work, the encoder produces $N{=}256$ patch embeddings of dimension $d{=}384$ per camera; LIBERO conditions on two cameras, agentview and wrist, so $N{=}512$ there. All features are computed once and cached before training, so every subsequent stage of TOKENGAZER, gaze policy, reward, and control policy alike, operates entirely in this cached latent space and never touches pixels again.

Gaze policy. The gaze is a lightweight transformer π_θ that attends to all N token embeddings and outputs one score for each,

$$\pi_\theta(E_t) \in \mathbb{R}^N. \tag{3}$$

A subset of K indices is then drawn sequentially without replacement,

$$i_1, \ldots, i_K \sim \text{Categorical}\big(\text{softmax}(\pi_\theta(E_t))\big), \tag{4}$$

masking each drawn index out of the distribution before the next draw, so that no patch is chosen twice and every candidate subset has a well-defined probability under the policy. We train π_θ with Group Relative Policy Optimization (Section 2.3): for every frame, a group of G candidate subsets $y^{(1)}, \ldots, y^{(G)}$ is sampled from the current policy, each is scored by the reward described below, and the rewards are normalized within the group,

$$A^{(g)} = \frac{r^{(g)} - \text{mean}\big(r^{(1)}, \ldots, r^{(G)}\big)}{\text{std}\big(r^{(1)}, \ldots, r^{(G)}\big) + \delta}. \tag{5}$$

The gaze policy is updated to minimize

$$\mathcal{L}_{\text{gaze}}(\theta) = -\,\mathbb{E}\Big[\tfrac{1}{G}\sum_{g=1}^{G} A^{(g)} \log \pi_\theta(y^{(g)} \mid E_t)\Big] \;+\; \beta\,\text{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big) \;-\; \lambda\,\mathcal{H}[\pi_\theta], \tag{6}$$

a plain group-relative policy gradient (one on-policy update per group, so no importance ratio or clipping is needed) plus a penalty that keeps π_θ close to its own frozen initial checkpoint π_{ref} and a bonus on the policy's entropy $\mathcal{H}[\pi_\theta]$. The KL term stops the gaze from latching onto the first subset that scores well; the entropy term keeps it exploring long enough to compare subsets fairly, rather than collapsing onto a single choice too early. On LIBERO, where the same fixed scene supports ten different language-specified tasks, both the gaze and the reward head additionally receive the task instruction as one extra token, since vision alone cannot tell the tasks apart. We also train the same architecture with a differentiable Gumbel-softmax relaxation (Section 2.4) in place of Equation equation 6, backpropagating the reward directly through the relaxed selection, and compare both optimizers throughout the paper.

Why reconstruction is not enough. A natural way to score a candidate subset is to ask how well it lets the scene be reconstructed: if a subset preserves enough information to reconstruct the original frame, intuition suggests it should also preserve what a policy needs to act. We found that this assumption does not hold. Across several reconstruction objectives, reconstructing the current frame, predicting the next frame, and reconstructing only the held-out tokens, reconstruction quality tracked downstream policy performance weakly at best, and at some token budgets moved in the opposite direction: the subsets that reconstructed the scene most accurately were often among the least useful for action prediction. This observation motivates a reward that is tied to the control task directly rather than to image reconstruction.

Action-based reward. Instead of reconstruction, we score a candidate subset with a frozen action-prediction head g. Given a subset with indices $\mathbf{i} = \{i_1, \ldots, i_K\}$, every token outside it is masked before g sees the frame, and the reward is how accurately g recovers the expert action chunk $\mathbf{a}_t$ using only the tokens it was allowed to keep,

$$r(\mathbf{i}) = -\big\| g(E_t^{\mathbf{i}}, \mathbf{i}) - \mathbf{a}_t \big\|_2^2. \tag{7}$$

Depending on the benchmark, g is either a small action-prediction network trained from scratch on randomly chosen subsets and then frozen, or a pretrained dense transformer diffusion policy that already conditions on the full patch grid and is reused here, frozen, purely as a source of reward; Experimental Setup states which each benchmark uses. On LIBERO, g receives the task instruction as the same extra token given to the gaze, since it too must judge a subset against the correct task rather than a fixed one.

Stage 2: the control policy. Once the gaze policy is trained, we freeze it and train a diffusion policy (Section 2.2) to act on the tokens it selects. Rather than always conditioning on the gaze's own picks, the training subset is drawn per step as

$$S_t = \begin{cases} \mathrm{TopK}\big(\pi_\theta(E_t)\big) & \text{with probability } \frac{1}{2}, \\ \text{a uniformly random } K\text{-token subset} & \text{otherwise,} \end{cases} \tag{8}$$

and the diffusion loss (Section 2.2) is minimized over that mixture,

$$\mathcal{L} = \mathbb{E}\big\| \epsilon - \epsilon_\phi\big(\mathbf{a}_t^{(\tau)}, \tau, E_{t-1}^{S_t}, E_t^{S_t}\big) \big\|^2. \tag{9}$$

At deployment the policy always receives the gaze's own greedy choice, $S_t = \mathrm{TopK}(\pi_\theta(E_t))$. We do this because a policy trained on a single, fixed subset of tokens per frame tends to overfit to that particular subset; mixing in random subsets during training keeps the policy robust to whichever tokens actually arrive at deployment, while it still learns to make good use of the gaze when the gaze is what it gets. This is the same diffusion policy architecture for every method we compare, dense, pooled, random, and TOKENGAZER alike, so any difference in performance can only come from which tokens condition it and not from a difference in the policy itself. Because the policy attends only to the K tokens it is actually given, rather than a fixed-length sequence padded out to the full grid, both its sequence length and its computational cost scale directly with K; the resulting training-time savings, and their limits, are quantified in Figure 1(c) and Appendix Table 3.

Evaluation protocol. Every benchmark in this paper is evaluated vision-only: no method, dense included, ever receives proprioception, since a policy that can see the robot's own state can learn to bypass the image entirely, which is exactly the shortcut we ablate in Appendix A. We report closed-loop success under receding-horizon replanning, one action per replan on Cube and LIBERO, with seed and episode counts given in each results table.

5 EXPERIMENTAL SETUP

Figure 3 shows a representative frame from each benchmark.



Figure 3: A representative held-out frame from each benchmark, agentview only (LIBERO-Goal's wrist camera, where the gaze spends most of its budget, is in Figure 4). Both distractor variants show one live-composited severity-2 clutter frame from the same scene as their clean counterpart.

We evaluate on PushT (Chi et al., 2023), a two-dimensional manipulation task that asks a policy to push a T-shaped block into a target pose, and on a distractor variant of the same task that scatters visually similar but task-irrelevant shapes across the scene, distinguishable from the true block only by color, so that a representation which cannot ignore clutter pays for it directly. We use the same DINOv2 encoder as Patch Policy (Zhou et al., 2026), whose own published result reaches 0.81 ± 0.01 full-token coverage on this task, measured as raw per-episode coverage rather than the clipped success rate we report elsewhere (Table 4); they report no pooled variant under DINOv2, and their closest pooled number, 0.79 ± 0.02, uses a different encoder (WebSSL (Fan et al., 2025)), so we give it here only for context rather than as a matched comparison. Measured the same way, our own full-token policy reaches 0.691 coverage and our pooled baseline reaches 0.564 at $K{=}1$, the one budget at which pooling reduces to a single global-average token; both fall a little below the published numbers, and we do not paper over that gap (Appendix E.5 also compares Patch Policy's own learned patch-compression ablation against a plain random subset of ours at matching budgets).

We compare five ways of choosing what a control policy conditions on at a given token budget K: our learned gaze policy, trained with either group-relative policy optimization or the Gumbel-softmax relaxation; a fixed spatial pooling into K tokens; a fixed center crop; a training-free heuristic that selects by normalized attention; and a random subset, alongside a policy that conditions on the full, unselected token grid. Every method is evaluated through the same control policy architecture and the same closed-loop rollout procedure, so a comparison isolates the choice of which tokens to keep from everything downstream of it. Implementation details, including which architecture the reward head for each benchmark uses and its published action horizons, are in Appendix A.

5.1 Results

Protocol, all benchmarks: every benchmark is evaluated vision-only, dense included, and Stage 2 is trained on a 50/50 mix of the gaze's own token choices and a random subset, deployed with the gaze's top-K (Appendix A); replanning is one action per step on Cube and LIBERO. We use 50 episodes on PushT and PushT-Distract, 100 on Cube and Cube-Distract, and 200 on LIBERO. Multi-seed results below report the mean with an interval appropriate to the metric (percentile bootstrap over seed means for the continuous coverage/cubes-placed scores, Wilson score over pooled episodes for binary success; Appendix A); single-seed values are marked as such rather than given a fabricated one. Full per-seed values are in Appendix F.

PushT: TokenGazer reaches its best score, 0.847 (3 seeds, 95% bootstrap interval $[0.81, 0.91]$), at $K{=}64$, ahead of every baseline in Table 1. This is the one benchmark where we show the complete per-K sweep in the main body rather than only in the appendix: Figure 1(a) plots TokenGazer and random against dense across all six budgets we measured, real per-K points, not a smoothed curve, which is why TokenGazer dips below random at $K{=}32$ before recovering at $K{=}64$. Every other benchmark's full grid is appendix-only, referenced from its own paragraph below; for PushT the same numbers are also in Table 4.

PushT-Distract: policies here are trained with distractors composited into the training frames and deployed on the clean scene, so this is a test of whether selection transfers rather than of matched-clutter performance. TokenGazer's best score is 0.297 (3 seeds, $[0.24, 0.37]$) at $K{=}64$, and it

Table 1: Headline results: each cell is a method's best closed-loop score over its own measured token budgets on that benchmark (cubes placed, of 2, for Cube/Cube-Distract, success rate elsewhere), with that best K as a subscript, e.g. 0.85_{64}. *Pending* marks a configuration not yet measured; *n/a* marks one pooling cannot reach (non-square K) or that does not apply (pooling on LIBERO's two-camera candidate set). Bold marks the best-performing *selection* method per column (dense is the reference being selected against, not a competitor, so it is never bolded even where it is the numeric maximum, as on Cube: TokenGazer's 0.88_{32} is parity with dense's 0.89, not a loss). ‡Dense is 256 tokens except LIBERO, which is 512. TokenGazer's row is GRPO at every entry except PushT, where Gumbel is also measured but lower at every K; seed counts, the vision-only protocol, and full per-K grids for every method (including Patch Policy's own published numbers) are in Appendix B: Tables 4, 5, 7, 8, 9.

Method	PushT	PushT-Distract	Cube	Cube-Distract	LIBERO
Dense (256 tok.)‡	0.75	0.19	0.89	0.51	0.72
Pooled	0.80_{4}	0.11_{64}	0.26_{8}	0.31_{64}	n/a
Random	0.78_{64}	0.25_{64}	0.71_{64}	0.21_{64}	0.71_{64}
Attention-norm	0.69_{16}	0.14_{64}	0.06_{8}	0.17_{64}	0.24_{64}
Center	0.65_{16}	0.13_{64}	0.11_{8}	0.15_{64}	0.21_{64}
TOKENGAZER	$\mathbf{0.85_{64}}$	$\mathbf{0.30_{64}}$	$\mathbf{0.88_{32}}$	$\mathbf{0.53_{64}}$	$\mathbf{0.82_{64}}$

clears the dense reference at five of the six budgets we measured; the full grid is Table 5, and two further clutter settings, including deployment under matched clutter, are in Appendix C.

Cube: TOKENGAZER's best score is 0.88 cubes placed (3 seeds, $[0.72, 0.96]$) at K=32, an eighth of the dense token count, and this is parity with dense's 0.89 ($[0.81, 1.02]$) rather than a clear win, since the two intervals overlap almost entirely; every budget is in Table 7. Figure 4 shows TOKENGAZER's actual selected patches on a held-out frame from each benchmark.

Cube-Distract: with severity-2 clutter composited into both training and deployment, TOKENGAZER's best score is 0.53 cubes placed (3 seeds, $[0.46, 0.65]$) at K=64, again parity with dense's 0.51 (3 seeds, $[0.41, 0.60]$) rather than a win – both intervals are wide at this seed count and should not be read as more settled than they are. This does not transfer across a training/deployment clutter shift (Appendix C), so the matched-clutter comparison here, not the transfer result that holds for PushT-Distract, is the Cube-Distract claim; the full grid is Table 8.

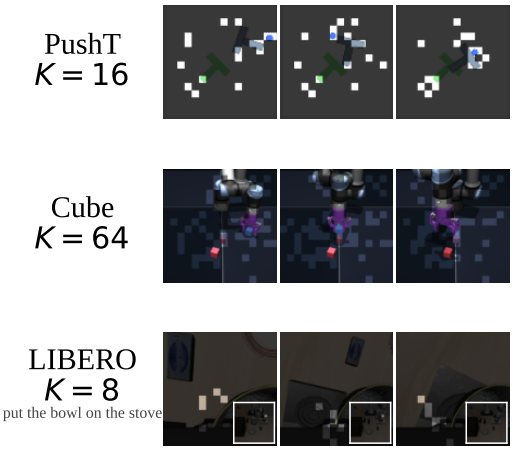

Figure 4: TOKENGAZER's selected patches (unshaded) on held-out frames; LIBERO shows the wrist camera with the agentview as an inset. Rollout videos are in the supplementary material.

LIBERO Goal, language on, vision only: TOKENGAZER beats the dense policy at every budget we measured, from 0.735 with 8 of 512 tokens to 0.815 with 64, and beats random selection at every K as well; proprioception reverses this, hurting every method and TOKENGAZER most of all (0.31 versus 0.735 at K=8), the seed-dependent collapse detailed in Appendix E.4; the full grid is Table 9 (a compact copy is Table 2).

Compute (Figure 1(c); full numbers in Appendix Table 3): Stage 2 alone is 2.3, 2.3, 1.46, and 3.2 times cheaper than dense on PushT, PushT-Distract, Cube, and LIBERO, and deployment conditions on K tokens instead of 256 (512 for LIBERO). Counting Stage 1 (reward-head and selector training) changes the picture: end-to-end TOKENGAZER training is cheaper than dense on LIBERO ($0.55\times$) and Cube ($0.92\times$), but slightly more expensive on PushT ($1.15\times$) and PushT-Distract ($1.12\times$). Stage 1 is a real cost that does not

always pay for itself within a single training run, but it is paid once per selector, amortised across every K and every Stage-2 policy trained on it, unlike Stage 2 and the deployment token count, which recur every time and are where the saving holds (Appendix A).

Table 2: LIBERO-Goal, vision-only success by budget; last column: with proprioception (Appendix E.4).

Method	K=8	K=16	K=32	K=64	K=8, w/ proprio.
TOKENGAZER (GRPO)	**0.74**	**0.77**	**0.80**	**0.82**	0.31
Random	0.65	0.67	0.70	0.71	0.48
Dense			0.72		0.70

6 CONCLUSION

TOKENGAZER separates the choice of which patch tokens a policy conditions on from how much of the image those tokens compress. The natural, self-supervised reward for training a token selector, reconstruction quality, is not just a weak proxy for task performance but an actively misleading one, and replacing it with a reward drawn from a strong, independently trained action-prediction model corrects this failure. Because our control policy conditions on exactly the tokens it is given rather than a fixed, padded sequence, selection is not only a performance question but a compute one: a smaller, well-chosen subset is a cheaper policy to run, not only a sparser one. The results span PushT and its distractor variant, Cube, and language-conditioned LIBERO; on clean PushT, TOKENGAZER now beats the dense reference and every same-budget baseline at K=16 and K=64. Cube and LIBERO report three seeds at the budgets in Figure 1 and single seeds elsewhere, as stated in each table's own caption.

Two observations explain why selection works where it does. A policy trained on one fixed subset per frame overfits to that subset, so alternating the gaze's selection with random subsets during Stage 2 keeps the policy robust to whichever tokens arrive while it still learns to exploit the gaze. And the gaze learns where the task happens: on LIBERO it spends almost its entire budget on the wrist camera (Figure 4), the view in which grasping and placing are decided.

Limitations and future work. Stage 1 is an extra, one-time cost, which makes end-to-end training slightly more expensive than dense conditioning on PushT even though every Stage-2 policy is cheaper. Every method, TOKENGAZER included, fails when the clutter distribution shifts between training and deployment (Appendix C), and policies given proprioception learned to ignore vision on some seeds, which the vision-only protocol removes. A natural next step is a generalist TOKENGAZER trained once across many tasks and embodiments, so that Stage 1 becomes a single pretraining cost shared by every downstream policy.

REPRODUCIBILITY STATEMENT

Full training and evaluation details for the gaze policy, the reward model, and the control policy, including hyperparameters and the exact closed-loop evaluation protocol (replanning frequency, proprioception ablation, live-clutter compositing, and every correction made to the protocol along the way), are in Appendix A. Every per-seed value behind the means and confidence intervals reported in the main tables is listed individually in Appendix F, so every number in the paper can be checked against the runs it comes from. Training and inference wall-clock, by token budget and benchmark, is in Table 3. Code is available at `https://anonymous.4open.science/r/tokengazer-release-E075`: the full pipeline, including feature caching, the Stage 1 selector and reward heads, the Stage 2 policy, rollouts, and per-benchmark scripts with the protocol flags used for every result in this paper.

Ethics statement

This work involves no human subjects and no personal data: every result is on simulated manipulation benchmarks (PushT, OGBench Cube, LIBERO), built on public, freely available datasets and simulators.

The Use of Large Language Models (LLMs)

We used generative AI tools to polish writing, improve readability, and suggest structure for this paper, to create and edit software code, and to create or modify figures; every such use was reviewed by the authors. We did not use generative AI to generate any dataset or synthetic data, or to formulate or prove any mathematical claim in this paper; the remaining disclosure categories in the call for papers do not apply to this work. All AI-assisted code was verified by running the experiments reported in this paper, and all AI-assisted text was reviewed and edited by the authors. The authors take full responsibility for the content of this paper, including all text, claims, and artifacts produced with AI assistance.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Milli-can, Malcolm Reynolds, et al. Flamingo: A visual language model for few-shot learning. In *Neural Information Processing Systems*, volume 35, pp. 23716–23736. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-1723. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/960a172bc7fbf0177ccccbb411a7d800-Paper-Conference.pdf.

Federico Baldassarre, Marc Szafraniec, Basile Terver, Vasil Khalidov, Francisco Massa, Yann LeCun, Patrick Labatut, Maximilian Seitzer, and Piotr Bojanowski. Back to the features: Dino as a foundation for video world models. *arXiv preprint arXiv:2507.19468*, 2025.

Wele Gedara Chaminda Bandara, Naman Patel, Ali Gholami, Mehdi Nikkhah, Motilal Agrawal, and Vishal M. Patel. AdaMAE: Adaptive masking for efficient spatiotemporal learning with masked autoencoders. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14507–14517. IEEE, 2023. doi: 10.1109/cvpr52729.2023.01394. URL https://openaccess.thecvf.com/content/CVPR2023/html/Bandara_AdaMAE_Adaptive_Masking_for_Efficient_Spatiotemporal_Learning_With_Masked_Autoencoders_CVPR_2023_paper.html.

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*, 2023. doi: 10.48550/arXiv.2210.09461. URL https://arxiv.org/abs/2210.09461.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://roboticsproceedings.org/rss19/p025.html.

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. *Robotics: Science and Systems Conference*, 44(10-11):1684–1704, 2023. doi: 10.1177/02783649241273668. URL https://journals.sagepub.com/doi/10.1177/02783649241273668.

David Fan, Shengbang Tong, Jiachen Zhu, Koustuv Sinha, Zhuang Liu, Xinlei Chen, Michael Rabbat, Nicolas Ballas, Yann LeCun, Amir Bar, et al. Scaling language-free visual representation learning, 2025.

Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Juergen Gall. Adaptive token

sampling for efficient vision transformers. In *European Conference on Computer Vision*, pp. 396–414. Springer Nature Switzerland, 2021. doi: 10.1007/978-3-031-20083-0_24. URL https://arxiv.org/abs/2111.15667.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *IEEE International Conference on Robotics and Automation*, 2016. doi: 10.1109/ICRA.2016.7487173. URL https://rll.berkeley.edu/dsae/dsae.pdf.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Doll'ar, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *Computer Vision and Pattern Recognition*, pp. 15979–15988. IEEE, 2021. doi: 10.1109/CVPR52688.2022.01553. URL https://arxiv.org/abs/2111.06377.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2016. URL https://openreview.net/forum?id=rkE3y85ee.

Jaeyeon Lee, Shunjie Wen, and Dong-Wan Choi. Moving beyond diversity: Visual token pruning as subspace reconstruction for efficient vlms, 2026.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. URL https://jmlr.org/papers/v17/15-522.html.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. EViT: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*, 2022. URL https://iclr.cc/virtual/2022/poster/6168.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems 36*, volume 36, pp. 44776–44791. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-1939. URL https://arxiv.org/abs/2306.03310.

Yang Luo, Zhineng Chen, Peng Zhou, Zuxuan Wu, Xieping Gao, and Yu-Gang Jiang. Learning to rank patches for unbiased image redundancy reduction. In *Computer Vision and Pattern Recognition*, pp. 22831–22840. IEEE, 2024. doi: 10.1109/CVPR52733.2024.02154. URL https://openaccess.thecvf.com/content/CVPR2024/html/Luo_Learning_to_Rank_Patches_for_Unbiased_Image_Redundancy_Reduction_CVPR_2024_paper.html.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *arXiv (Cornell University)*. Cornell University, 2023. doi: 10.48550/arxiv.2210.00030. URL https://iclr.cc/virtual/2023/poster/11127.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1611.00712.

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? In *Advances in Neural Information Processing Systems 36*, volume 36, pp. 655–677. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-0031. URL https://arxiv.org/abs/2303.18240.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Neural Information Processing Systems*, volume 27, 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/3e456b31302cf8210edd4029292a40ad-Paper.pdf.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 892–909, 2023. doi: 10.48550/arXiv.2203.12601. URL https://proceedings.mlr.press/v205/nair23a.html.

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. *arXiv preprint arXiv:2304.07193*, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.20092. URL https://arxiv.org/abs/2410.20092.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 10832–10851, 2026.

Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 416–426, 2023. doi: 10.48550/arXiv.2210.03109. URL https://proceedings.mlr.press/v205/radosavovic23a.html.

Ayush K. Rai, Kyle Min, Tarun Krishna, Feiyan Hu, Alan F. Smeaton, and Noel E. O'Connor. Reinforcement learning meets masked video modeling: Trajectory-guided adaptive token selection. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 7071–7081. IEEE, 2025. doi: 10.1109/ICCVW69036.2025.00730. URL https://openaccess.thecvf.com/content/ICCV2025W/LongVid-Foundation/papers/K._Reinforcement_Learning_meets_Masked_Video_Modeling_-_Trajectory-Guided_Adaptive_Token_ICCVW_2025_paper.pdf.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2106.02034.

Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Neural Information Processing Systems*, volume 34, pp. 12786–12797, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/6a30e32e56fce5cf381895dfe6ca7b6f-Abstract.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-Actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 785–799, 2023. doi: 10.48550/arXiv.2209.05451. URL https://proceedings.mlr.press/v205/shridhar23a.html.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems 35*, volume 35, pp. 10078–10093. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-0732. URL https://arxiv.org/abs/2203.12602.

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control, 2022. URL https://arxiv.org/abs/2203.06173.

Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *Computer Vision and Pattern Recognition*, pp. 10799–10808. IEEE, 2021. doi: 10.1109/CVPR52688.2022.01054. URL https://research.nvidia.com/publication/2022-06_vit-adaptive-tokens-efficient-vision-transformer.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter Networks: Rearranging the visual world for robotic manipulation. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 726–747, 2021. URL https://proceedings.mlr.press/v155/zeng21a.html.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Gaoyue Zhou, Zichen Jeff Cui, Ada Langford, Bowen Tan, Yann LeCun, and Lerrel Pinto. Patch policy: Efficient embodied control via dense visual representations. *arXiv preprint arXiv:2607.18236*, 2026.

A PROTOCOL AND CORRECTIONS

Implementation details. For Cube and LIBERO (Liu et al., 2023), the reward head is a small action-prediction network trained from scratch. For PushT and PushT-Distract, it is a pretrained dense transformer diffusion policy that conditions on the full patch grid, reused here, frozen, as a source of reward; its architecture follows Patch Policy (Zhou et al., 2026). All methods are evaluated under closed-loop, receding-horizon replanning; for Cube and LIBERO Goal this matches Patch Policy's own published action horizon (1 and 3 actions respectively).

Confidence intervals. Every interval in this paper, in the tables, the Results text, and every bar plot, uses one of two methods chosen to respect the metric it describes, never a t-interval applied indiscriminately. PushT and PushT-Distract's clipped-coverage score and Cube/Cube-Distract's cubes-placed count are continuous per-episode metrics (cubes-placed ranges over $[0, 2]$, not $[0, 1]$, since two cubes can be placed); for these, a multi-seed cell reports a 95% percentile bootstrap interval over the seed-level means, 10,000 resamples with replacement. At $n{=}3$ seeds this interval is exactly the seed range, $[\min, \max]$: with only three possible values to resample from, the 2.5th and 97.5th percentiles of the resampled means land on the extremes far more often than not. This is a real property of the bootstrap at this sample size, not a bug, and it is why some of these intervals look as wide as the raw seed spread. Cube and LIBERO's task-success flag, and PushT's own reported success rate where treated as a completion flag, are binary per episode; for these, a cell's interval is a 95% Wilson score interval computed over all episodes pooled across every seed (e.g. LIBERO's $K{=}8$ TOKENGAZER row pools 3 seeds $\times$ 200 episodes into one 441/600 interval rather than three separate ones). A single-seed cell for a continuous metric gets no interval, since there is nothing to resample; a single-seed cell for a binary metric still gets a Wilson interval, computed over that one seed's own episodes.

Every bound this produces was checked directly: every success-rate and clipped-coverage interval in this paper, a metric with a $[0, 1]$ range by construction, has both bounds inside $[0, 1]$, as the bootstrap and Wilson constructions guarantee once the underlying metric itself is bounded that way. Cube and Cube-Distract's cubes-placed intervals are the one exception, by construction rather than error: that metric ranges over $[0, 2]$ (two cubes can be placed), so its tables report the raw count, headed *(of 2)*, and Figure 1(b) instead plots it as a fraction (cubes placed $\div$ 2) specifically so that panel's bars and intervals sit in $[0, 1]$ alongside the success-rate benchmarks.

Replanning. Replanning frequency changes the numbers a great deal. LIBERO dense: 0.41 replanning every 8 actions, 0.67 every 3, 0.795 every 1. Cube dense: 0.16 every 8, 0.30 every 1. All Cube and LIBERO numbers in this paper replan after every action, not Patch Policy's own published LIBERO horizon of 3 (their Table 11): at 3 actions per replan, the LIBERO seeds that already collapse at 1 partly recover (random $K{=}16$: 0.11 at 1, 0.225 at 3), but seeds that work at 1 get worse (TOKENGAZER $K{=}8$ language on: seed 0 from 0.70 to 0.415, seed 2 from 0.035 to 0.095), so 1 action per replan is kept throughout.

Proprioception. On Cube, a policy given proprioception can bypass the image entirely. Dense with proprioception places 0.30 cubes; without it, 0.83. All Cube numbers are vision-only. PushT is vision-only throughout.

Stage-1 cost. Figure 1(c) and Table 3 give the full, end-to-end picture: once reward-head and selector training are counted, TOKENGAZER is cheaper than dense on LIBERO ($0.55\times$) and, more narrowly, on Cube ($0.92\times$), but slightly more expensive on PushT ($1.15\times$) and PushT-Distract ($1.12\times$). Stage 1 is a real cost, not something the paper's efficiency claim hides, and it does not always pay for itself within a single training run. It is, however, paid once per selector: the same frozen selector trains every Stage-2 policy in this paper at its K, across seeds, and would train any further policy at that K for free. Stage 2 – the cost paid again for every such policy, and the one that recurs at deployment – is where the saving is, 1.5 to 3.2 times cheaper with TOKENGAZER, alongside conditioning on 8 to 64 tokens at deployment instead of 256 to 512. Figure 1(c) plots this breakdown; the same numbers, per benchmark and stage, are below.

Table 3: End-to-end training wall-clock, minutes, at each benchmark's reported K, GPU-matched (V100-SXM2) medians across the available seeds, feature caching excluded (about 4 minutes, shared by every method so it cancels in the comparison). Stage 1 is head-training plus selector-training (GRPO); dense has no Stage 1. †Clean PushT's head-training run did not survive the V100-only filter used here; its Stage 1 reuses the PushT-Distract head-training time (same reward-head configuration) as the best available estimate, so its Total is an estimate, not a direct measurement, unlike every other cell.

Benchmark	Method	Stage 1 (head + selector)	Stage 2	Total
PushT K=16	TOKENGAZER	$63^{\dagger} + 42 = 105$	62	167
	Dense	–	145	145
PushT-Distract K=16	TOKENGAZER	$63 + 37 = 100$	64	164
	Dense	–	147	147
Cube K=64	TOKENGAZER	$17 + 18 = 35$	99	134
	Dense	–	145	145
LIBERO K=8	TOKENGAZER	$26 + 66 = 92$	117	209
	Dense	–	377	377

LIBERO random baseline. The random baseline originally drew only from the agentview camera's 256 tokens. Drawing from both cameras' 512 tokens gives 0.645 at K=8, which is the number reported.

Stage-2 subset randomisation. A policy trained on random K=64 subsets scores 0.813 when deployed on random subsets, 0.780 on the gaze's subsets, 0.727 on spare, 0.427 on center, and 0.177 on attention-norm; a policy trained on the gaze's greedy top-K scores 0.552 on random subsets. At K=16 the random-trained policy scores 0.695 on random subsets, 0.629 on the gaze's, and 0.342 on center.

Live clutter. Earlier PushT-Distract rollouts ran in the clean environment: Stage 2 was trained on cluttered features but never evaluated with clutter present in the live frames. Numbers with distractors composited onto the live frames, and with clean-trained policies evaluated under clutter they never saw in training, are in Appendix C.

B FULL BENCHMARK TABLES

The main text reports each benchmark at its final-protocol K (Table 1). The full token-budget grid behind that headline number, every method at every measured K, is here for each benchmark in turn.

Table 4: PushT, closed-loop success rate by method and token budget K. *Pending* marks a configuration not yet measured, not a zero; *n/a*† marks K pooling cannot reach. Bold marks TOKENGAZER's best cells. Cells are 2-decimal point estimates; 95% CIs for multi-seed cells are in the Results text and Appendix F, not repeated here to keep columns narrow. Seeds: Dense, TOKENGAZER (GRPO) at K=16, 64, and Random at K=32, 64 are means of 3; every other cell is a single seed (exact per-seed values in Appendix F, along with attention-norm and center, moved out of this table).

Method	K=1	K=4	K=8	K=16	K=32	K=64
Dense (256 tok.)				0.75		
Patch Policy				0.81		
Pooled	0.59	**0.80**	n/a†	0.77	n/a†	0.74
Random	0.39	0.50	**0.62**	0.69	**0.71**	0.78
TOKENGAZER (GRPO)	0.50	0.49	0.49	**0.80**	0.58	**0.85**
TOKENGAZER (Gumbel)	0.59	0.52	0.52	0.57	0.63	0.58

Table 6: PushT-Distract, additional training-free baselines, same protocol as Table 5, trained with distractors and deployed on the clean scene; column definitions as in Table 5.

Method	K=1	K=4	K=8	K=16	K=32	K=64
Pooled				0.108		0.114
Attention-norm				0.124		0.141
Center crop				0.118		0.132

Table 5: PushT-Distract, closed-loop success rate by method and token budget K. Policies are trained with distractors composited into the training frames and deployed on the clean scene. *Pending* marks a configuration not yet measured, not a zero. Bold marks TOKENGAZER cells that clear the dense reference (0.192, shown rounded below). Cells are 2-decimal point estimates; 95% CIs for multi-seed cells are in the Results text and Appendix F (which also has every per-seed value behind the means below). Seeds: TOKENGAZER rows are means of 3 (GRPO at K=16 uses a tuned exploration setting, Section E); Random is a single seed. Pooled, attention-norm and center follow the same protocol and are reported at K=16 and 64 in Table 6. The bars give the same GRPO/Gumbel numbers with 95% CI error bars.

Method	K=1	K=4	K=8	K=16	K=32	K=64
Dense (256 tok.)				0.19		
Patch Policy				0.19		
Random	0.13	0.11	0.14	0.14	0.18	0.25
TOKENGAZER (GRPO)	**0.21**	0.21	0.18	**0.24**	**0.29**	**0.30**
TOKENGAZER (Gumbel)	0.11	0.18	**0.28**	**0.29**	**0.25**	**0.27**

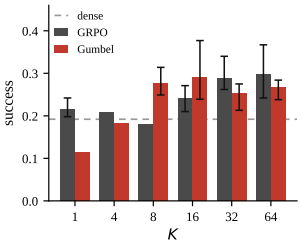

Table 7: Cube (via OGBench (Park et al., 2025)), cubes placed (of 2) / task success by method and token budget K, vision only, 100 episodes. *Pending* marks a configuration not yet measured, not a zero; *n/a*† marks K pooling cannot reach. Bold marks TOKENGAZER's cell wherever it beats every measured baseline in that column. Seeds: K=16, K=32, K=64, and the full token set are means of 3 seeds (TOKENGAZER K=16: 0.69/0.20, 0.75/0.28, 0.61/0.18; TOKENGAZER K=32 (seeds 0, 800, 801): 0.72/0.22, 0.95/0.32, 0.96/0.32; TOKENGAZER K=64: 0.89/0.32, 0.83/0.29, 0.86/0.30; full token set: 0.83/0.27, 1.02/0.40, 0.81/0.24); TOKENGAZER K=8 and pooled are single seed, and random follows the same protocol. Attention-norm and center are moved to Appendix F, along with every per-seed value behind the means here. K=32 and K=64 both reach parity with the dense policy, at an eighth and a quarter of the tokens respectively. †Pooled is only defined at perfect-square K; its measured point at K=8 is really K=9 (footnoted in-cell), placed in that column for comparison. Pooled is now complete: K=9 →col-8 0.26/0.02, K=16 0.23/0.01, K=32 n/a, K=64 0.65/0.17, all seed 0. TOKENGAZER (Gumbel) is not yet run on this benchmark. Cells are 2-decimal point estimates; 95% CIs for multi-seed cells are in the Results text and Appendix F.

Method	K=8	K=16	K=32	K=64
Dense (256 tok.)		0.89 / 0.30		
Patch Policy		1.24		
Pooled†	0.26 / 0.02	0.23 / 0.01	n/a†	0.65 / 0.17
Random	0.06 / 0.00	0.28 / 0.04	0.58 / 0.14	0.71 / 0.21
TOKENGAZER (GRPO)	**0.40 / 0.08**	**0.68 / 0.22**	**0.88 / 0.29**	**0.86 / 0.30**

Table 8: Cube-Distract, cubes placed (of 2) / task success by method and token budget K, vision only, 100 episodes. Trained with severity-2 clutter composited into the training frames, deployed with clutter also composited onto the live frames. *Pending* marks a configuration not yet measured, not a zero; *n/a*† marks K pooling cannot reach. TOKENGAZER $K{=}8$, $K{=}16$, and $K{=}32$ are seed 0; $K{=}64$ is 3 seeds (0, 1, 2: cubes 0.65, 0.47, 0.46, mean 0.53, 95% bootstrap interval $[0.46, 0.65]$; success 0.14, 0.07, 0.06, mean 0.09, 95% Wilson interval over the pooled 300 episodes, $[0.06, 0.13]$; Appendix A) – parity with dense under matched clutter at a quarter of the tokens, not a win, so it is not bolded. Dense is 3 seeds (0, 1, 2: cubes 0.41, 0.52, 0.60, mean 0.51, $[0.41, 0.60]$; success 0.02, 0.10, 0.09, mean 0.07, Wilson $[0.05, 0.10]$), same reference at every K. Pooled and Random are seed 0 at every K shown. Attention-norm and center, seed 0, are in Appendix F alongside every per-seed value here. Clean-Cube values at the same K's are in Table 7.

Method	$K{=}8$	$K{=}16$	$K{=}32$	$K{=}64$
Dense (256 tok.)		0.51 / 0.07		
Pooled	n/a†	0.12 / 0.00	n/a†	0.31 / 0.03
Random	0.04 / 0.00	0.09 / 0.00	0.15 / 0.00	0.21 / 0.00
TOKENGAZER (GRPO)	0.25 / 0.04	0.45 / 0.06	0.44 / 0.04	0.53 / 0.09

Table 9: LIBERO Goal (10 tasks, language on), closed-loop success rate by method and token budget K, **vision only**, 1 action per replan, 200 episodes. The earlier protocol conditioned on proprioception in addition to vision; that version is retired to Appendix E.4 as the ablation that exposed a proprioception shortcut some training seeds exploit instead of vision, the same failure mode found on Cube and fixed there the same way. Dense is 2 seeds (0.715, 0.725, mean 0.72); TOKENGAZER $K{=}8$ is 3 seeds (0.765, 0.72, 0.72, mean 0.735) and random $K{=}8$ is 3 seeds (0.585, 0.835, 0.530, mean 0.65); $K{=}16$, $K{=}32$, and $K{=}64$ are single seed for TOKENGAZER/random. Attention-norm and center, under the same protocol, are in Appendix F. Success is binary per episode; intervals are Wilson score intervals over all episodes pooled across seeds (Appendix A), stated in the Results text rather than repeated here. †Pooled is n/a throughout: LIBERO's candidate set is two separate camera grids (Table 7's single-grid pooling has no well-defined analogue here).

Method	$K{=}8$	$K{=}16$	$K{=}32$	$K{=}64$
Dense (512 tok.)		0.72		
Patch Policy		0.98		
Pooled	n/a†	n/a†	n/a†	n/a†
Random	0.65	0.67	0.70	0.71
TOKENGAZER (GRPO)	**0.74**	**0.77**	**0.80**	**0.82**

C ADDITIONAL CLUTTER SETTINGS

Table 10 reports two settings beyond the main PushT-Distract table. When policies are also deployed under the clutter they were trained with, TOKENGAZER at $K{=}64$ matches dense (0.350 against 0.345) and trails it at smaller budgets. When policies trained clean are deployed under clutter they never saw, every method drops together (dense 0.126, TOKENGAZER 0.139 at $K{=}16$ and 0.126 at $K{=}64$, random 0.119, pooled 0.102). What the main table shows is that selection is far less sensitive than dense conditioning to the clutter distribution shifting between training and deployment.

Table 10: PushT-Distract, two further settings, single seed, 50 episodes. *Deployed under clutter*: Stage 2 trains on distractor-composited features and distractors (severity 2) are also composited onto the live rollout frames. *Trained clean, deployed under unseen clutter*: the clean-PushT policies of Table 4, rolled out with live clutter they never saw in training. Bold marks cells that clear the full-token-set row of the same block.

Method	K=1	K=4	K=8	K=16	K=32	K=64
Deployed under clutter						
TOKENGAZER (GRPO)					0.319	**0.350**
TOKENGAZER (Gumbel)			0.206	0.276		0.312
Full token set (K=256)				0.345		
Trained clean, deployed under unseen clutter						
TOKENGAZER (GRPO)				0.139		0.126
Random						0.119
Pooled		0.102				
Full token set (K=256)				0.126		

Table 11 reports the same two settings for Cube. Both collapse to near zero: trained under clutter and deployed clean, dense places 0.04 cubes and TOKENGAZER 0.05; trained clean and deployed under unseen clutter, dense places 0.04 cubes (0.00 success) and TOKENGAZER 0.08 (0.00 success). Unlike PushT, where selection transfers across the training/deployment clutter shift, on Cube every policy collapses when the clutter distribution changes between training and deployment, so the Cube-Distract claim is the matched-clutter one in Table 8, not a transfer claim.

Table 11: Cube, two further train/deploy combinations, cubes placed (of 2), seed 0. *Trained under clutter, deployed clean*: Stage 2 trains on distractor-composited features, rollouts run in the clean environment. *Trained clean, deployed under unseen clutter*: the clean-Cube policies of Table 7, rolled out with live clutter they never saw in training. Both at K=64/full token set.

Method	Trained under clutter, deployed clean	Trained clean, deployed under clutter
TOKENGAZER (GRPO) K=64	0.05	0.08
Full token set (K=256)	0.04	0.04

LIBERO-Distract (severity-2 clutter composited onto both cameras, matched, 1 action per replan, seed 0): dense drops from 0.795 clean to 0.085, TOKENGAZER at K=8 from 0.70 to 0.07. At this clutter severity every LIBERO policy fails; the benchmark is left for future work rather than reported as a claim.

D ADDITIONAL QUALITATIVE RESULTS

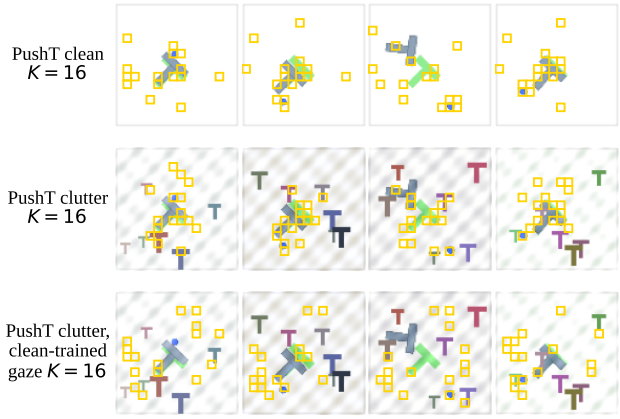

Figure 5: The same four PushT frames, clean and with severity-2 clutter composited onto them (deterministic per episode and step), with the K=16 selection outlined. Rows 1-2: the PushT-Distract2 selector on the clean and the cluttered version of each frame. Row 3: the clean-PushT selector, never trained on clutter, evaluated on the same cluttered frames as row 2.

E ABLATIONS

E.1 LANGUAGE OFF

Table 12 removes the task instruction on LIBERO Goal.

Table 12: LIBERO Goal language ablation at K=8, single seed, with-proprioception protocol (Table 16). Without the instruction, every policy is at chance on this fixed-scene, ten-task suite; the instruction is the task signal, and the gaze needs it too, which is why Stage 1 is language-conditioned.

Method	Language on	Language off
TOKENGAZER (GRPO)	0.70	0.08
Random	0.645	0.045
Full token set	0.795	0.045

E.2 STAGE-2 RECIPE

Table 13 compares the three Stage-2 training recipes. Mixing the gaze's subsets with random ones matches sampling alone on clean PushT and both are far above training on the gaze's greedy top-K. On PushT-Distract the mix does not repeat that win: at K=4 it reaches 0.199 and at K=16 0.210 (3 seeds each), below the original recipe's 0.207 and 0.241, so the main PushT-Distract table keeps the original recipe.

Table 13: Stage-2 training recipe, GRPO gaze, deployed with the gaze's top-K. *Original*: Stage 2 trains on the gaze's greedy top-K only. *Sample-only*: trains on subsets sampled from the gaze, single seed. *Mix*: 50/50 sampled gaze subsets and random subsets. PushT and PushT-Distract original and mix are 3-seed means.

	PushT		PushT-Distract	
Stage-2 recipe	K=16	K=64	K=4	K=16
Original	0.528	0.586	**0.207**	**0.241**
Sample-only	0.797	0.845		
Mix	**0.801**	**0.847**	0.199	0.210

E.3 Reward and exploration

A natural way to score a candidate subset is by how well it reconstructs the original image: if a subset preserves enough information to faithfully reconstruct the scene, it should also preserve what control needs. This does not hold. Across several reconstruction objectives, including reconstructing the current frame and predicting the next one, reconstruction quality showed little relationship, and sometimes an inverse one, with downstream policy performance – subsets that reconstructed the scene most accurately were often among the least useful for action prediction. This is why the reward in Section 4 is tied to the control task directly rather than to reconstruction.

PushT and PushT-Distract score candidate subsets with the frozen Patch Policy backbone, blending its action-prediction accuracy $r_{\mathrm{act}}(i)$ with its attention mass $r_{\mathrm{attn}}(i)$ (the fraction of the backbone's own attention, over the complete token grid, that falls on the selected subset i): $r(i) = \alpha\, \hat{r}_{\mathrm{act}}(i) + (1-\alpha)\, \hat{r}_{\mathrm{attn}}(i)$, with both terms normalized by a running mean and variance. Cube and LIBERO use a small action-prediction head trained from scratch, language-conditioned on LIBERO, which has no comparable attention signal to blend in, so only r_{act} is used there. The Patch Policy backbone reward was also tried on Cube and did not improve on the action-head reward.

Table 14 sweeps α under the original Stage-2 recipe at K=64. Raising it toward 1 helps GRPO monotonically at this K, and the same change hurts Gumbel.

Table 14: Ablation: reward blend weight α at K=64, clean PushT, single seed per cell except the α=0.5 row (3-seed, matching Table 4). References, also 3-seed from Table 4: random 0.784, full token set 0.750.

α	GRPO	Gumbel
0.5 (default)	0.586	0.582
0.75	0.604	0.589
0.9	**0.639**	0.544

Table 15 varies the group size G and the entropy regularization weight on PushT-Distract at K=16. Every alternative to the untuned default outperforms it, most sharply reducing the group size from 16 to 8: two confirmed seeds both clear the reference, against zero of three under the default. This is the configuration reported in the main PushT-Distract table.

Table 15: Ablation over the gaze policy's exploration settings at $K{=}16$ on PushT-Distract: the number of candidate subsets scored per update, G (Section 4), and the entropy regularization weight λ_{ent}; "above ref." means above the 0.192 full-token reference. The first row, $G{=}16$, is the default used everywhere else in this paper; every alternative tried improves on it. For comparison, Gumbel at the same K with its default settings (Table 5) scores 0.291, 3 of 3 seeds above reference.

G	λ_{ent}	seeds	mean
16 (default)	0.01 (default)	3	0.170 (0/3 above ref.)
32	0.01	1	0.267
16	0.03	1	0.241
32	0.03	1	0.244
8	0.01	3	**0.241** (3/3 above ref.)

E.4 SEED-DEPENDENT LIBERO COLLAPSE

On LIBERO Goal, with the earlier, proprioception-conditioned protocol (Table 16), rollouts collapsed for some runs despite validation error that looked the same as for runs that succeeded. At $K{=}8$ with language, the same selector gave 0.70, 0.475, 0.035, 0.13, and 0.205 across policy seeds 0 through 4. At $K{=}16$, random selection scored 0.11 and the language-conditioned gaze 0.05. More environment steps per episode did not fix it (Appendix A), and neither did replanning less often: at 3 actions per replan, collapsed seeds partly recovered but working seeds got worse. A first check of two random seeds (0.645, 0.78) looked stable, but a third collapsed to 0.005: random showed the same seed-dependent instability as the gaze, not a stable floor, so the collapse affected every $K{=}8$ selection method rather than being specific to the learned gaze. The dense policy was not exempt either: its own two seeds reached 0.795 and 0.60 (mean 0.70). Table 16 reports the mean alongside the best seed rather than hiding the spread; $K{=}16$ was never reported past a single seed and, since it was not re-rolled before the protocol moved to vision-only, stays pending rather than shown as a result. At $K{=}64$ (single seed), the gaze reached 0.755, exceeding the dense mean of 0.70 even under this retired, proprioception-conditioned protocol.

Table 16: LIBERO Goal, $K{=}8$, **with proprioception** (retired protocol): the ablation that exposed the proprioception shortcut, not a current result – Table 9 is the current, vision-only protocol. TOKENGAZER at $K{=}8$ is the mean of 5 seeds (0.70, 0.475, 0.035, 0.13, 0.205; best 0.70); Random is the mean of 3 seeds (0.645, 0.005, 0.78; best 0.78); dense is the mean of 2 seeds (0.795, 0.60). TOKENGAZER $K{=}32$ (0.305) and $K{=}64$ (0.755) are single seed, 1 action per replan, 200 episodes. $K{=}16$ under this protocol was never re-rolled past the single-seed collapse-diagnostic measurements (gaze 0.05, random 0.11) and is superseded by the vision-only grid, so it is left pending rather than shown as a result. Attention-norm and center are single seed. †Pooled is n/a for the same reason as Table 9.

Method	$K{=}8$	$K{=}16$	$K{=}32$	$K{=}64$
Dense (512 tok.)	0.70	*pending*	*pending*	*pending*
Patch Policy		0.98		
Pooled	n/a†	n/a†	n/a†	n/a†
Random	0.48	*pending*	*pending*	*pending*
Attention-norm	0.02	*pending*	*pending*	*pending*
Center	0.08	*pending*	*pending*	*pending*
TOKENGAZER (GRPO)	0.31	*pending*	0.31	0.76

Cross-gazer deployment at $K{=}8$ isolated the failure to training, not to which tokens were deployed: the working policy (seed 0) scored 0.70 with its own gaze and 0.465 under random selection at deployment, so it was exploiting what its gaze learned, while the collapsed policy (seed 2) scored 0.035 with its own gaze and 0.005 under random selection, so the failure was in the trained policy itself rather than the deployed subset; together with random-8 seed 1 also collapsing to 0.005, the

instability looked like a property of LIBERO Stage-2 training at this token budget rather than of any one selection method. A no-proprioception variant, the fix that worked on Cube's own proprioception shortcut, confirmed the same mechanism here: the collapsed seed 2 policy, which scored 0.035 with proprioception, reaches 0.72 when the identical $K{=}8$ gaze policy is retrained without it. Proprioception was a shortcut some training seeds latched onto instead of vision, and that shortcut failed closed-loop; the fix, as on Cube, is to drop it. LIBERO therefore now uses the same vision-only protocol as Cube (Table 9); the first vision-only pair, single seed each, is dense 0.715 (seed 0) and TOKENGAZER $K{=}8$ 0.72 (seed 2, the policy that had collapsed under proprioception), and a full vision-only grid ($K \in \{8, 16, 32, 64\}$, 3 seeds, with a random baseline) is queued to fill in Table 9 itself, whose cells stay *pending* until then.

E.5 COMPARISON WITH PATCH POLICY'S COMPRESSION

Table 17 places Patch Policy's own patch-compression ablation, a learned convolutional encoder trained end-to-end with their policy, against our random-dropping baseline at matching token budgets, on the same raw-coverage metric. This is not a claim that their method is random dropping; it is a comparison of their learned compression against ours, which uses no learned compression at all. At $K{=}1$ and $K{=}4$ their trained encoder scores higher (0.48 vs. 0.369, 0.51 vs. 0.473); at $K{=}16$, $K{=}64$, and $K{=}256$, plain random dropping matches or exceeds it (0.658 vs. 0.53, 0.751 vs. 0.52, 0.691 vs. 0.69).

Table 17: Clean PushT, raw per-episode coverage (mean_max_coverage, not the clipped success rate used elsewhere in this paper) at matching token budgets K. Patch Policy's row is their own published patch-compression ablation: a lightweight convolutional encoder, trained end-to-end with their policy, that learns to compress the patch grid down to K tokens. Our row is a plain random subset of K patches, with no learned compression of any kind; single seed, corrected for an earlier fixed-seed bug that had prevented it from actually randomizing. Bold marks whichever of the two is higher at a given K. $K{=}256$ is the full, uncompressed token set for both.

Method	$K{=}1$	$K{=}4$	$K{=}8$	$K{=}16$	$K{=}32$	$K{=}64$	$K{=}256$
Patch Policy (learned compression)	**0.48**	**0.51**	–	0.53	–	0.52	0.69
Ours (random dropping)	0.369	0.473	0.590	**0.658**	0.631	**0.751**	**0.691**

F FULL PER-SEED TABLES

Tables 18–21 list every per-seed value behind the means and confidence intervals reported in the main text, one table per benchmark, seeds as columns.

Table 18: PushT, per-seed values behind the means and 95% CIs in Table 4 and the Results text, plus the single-seed attention-norm and center baselines moved out of the main table.

Cell	Seed 0	Seed 1	Seed 2	Mean
Dense (256 tok.)	0.7265	0.7386	0.7849	0.750
TOKENGAZER (GRPO) $K{=}16$	0.810	0.781	0.813	0.801
TOKENGAZER (GRPO) $K{=}64$	0.905	0.829	0.808	0.847
Random $K{=}32$	0.663	0.725	0.755	0.714
Random $K{=}64$	0.788	0.804	0.761	0.784
Attention-norm $K{=}1$	0.686 (1 seed)			0.686
Attention-norm $K{=}4$	0.584 (1 seed)			0.584
Attention-norm $K{=}16$	0.687 (1 seed)			0.687
Center $K{=}1$	0.543 (1 seed)			0.543
Center $K{=}4$	0.641 (1 seed)			0.641
Center $K{=}16$	0.645 (1 seed)			0.645

Table 19: PushT-Distract, per-seed values behind the means and 95% CIs in Table 5 and the Results text. TOKENGAZER (GRPO) $K{=}16$ uses the tuned exploration setting ($G{=}8$, Section E); the same K under the untuned default scores 0.170 (seeds 0.174, 0.149, 0.185), reported only in Table 15, not here.

Cell	Seed 0	Seed 1	Seed 2	Mean
TOKENGAZER (GRPO) $K{=}1$	0.198	0.242	0.207	0.215
TOKENGAZER (GRPO) $K{=}16$, tuned	0.271	0.210	0.243	0.241
TOKENGAZER (GRPO) $K{=}32$	0.262	0.265	0.340	0.289
TOKENGAZER (GRPO) $K{=}64$	0.281	0.242	0.367	0.297
TOKENGAZER (Gumbel) $K{=}8$	0.266	0.249	0.314	0.276
TOKENGAZER (Gumbel) $K{=}16$	0.256	0.377	0.239	0.291
TOKENGAZER (Gumbel) $K{=}32$	0.271	0.213	0.275	0.253
TOKENGAZER (Gumbel) $K{=}64$	0.284	0.276	0.238	0.266

Table 20: Cube and Cube-Distract, per-seed values (cubes placed, of 2, / task success, except the two train-time rows in seconds) behind the means and 95% CIs in Tables 7 and 8 and the Results text, plus the single-seed attention-norm and center baselines moved out of the main tables (Cube at $K{=}8$; Cube-Distract at $K{=}16, 64$). Cube-Distract $K{=}64$ rows use seeds 0/1/2 for both TOKENGAZER and dense.

Cell	Seed A	Seed B	Seed C
TOKENGAZER $K{=}16$	0.69 / 0.20	0.75 / 0.28	0.61 / 0.18
TOKENGAZER $K{=}32$ (seeds 0, 800, 801)	0.72 / 0.22	0.95 / 0.32	0.96 / 0.32
TOKENGAZER $K{=}64$	0.89 / 0.32	0.83 / 0.29	0.86 / 0.30
Full token set	0.83 / 0.27	1.02 / 0.40	0.81 / 0.24
TOKENGAZER $K{=}64$, train time (s)	5977	5857	6054
Full token set, train time (s)	8789	8692	8671
Cube-Distract TOKENGAZER $K{=}64$, cubes placed	0.65	0.47	0.46
Cube-Distract TOKENGAZER $K{=}64$, success	0.14	0.07	0.06
Cube-Distract Full token set, cubes placed	0.41	0.52	0.60
Cube-Distract Full token set, success	0.02	0.10	0.09
Attention-norm $K{=}8$	0.06 / 0.00 (1 seed)		–
Center $K{=}8$	0.11 / 0.00 (1 seed)		–

Table 21: LIBERO Goal, vision-only protocol, per-seed values behind the means and 95% CIs in Table 9 and the Results text ($K{=}8$ for TOKENGAZER/random/dense), plus the single-seed attention-norm and center baselines moved out of the main table. This corrects an earlier version of this table that reproduced the retired, with-proprioception protocol's $K{=}8$ values instead (now in Table 16) under this table's vision-only label.

Cell	Seed 0	Seed 1	Seed 2
TOKENGAZER (GRPO) $K{=}8$	0.765	0.72	0.72
Random $K{=}8$	0.585	0.835	0.530
Dense (512 tok.)	0.715	0.725	–