Extended Abstract

Motivation RL fine-tuning of robot policies, including modern Vision-Language-Action models, is expensive, and most of the cost goes to exploring action regions that never pay off. Prior work improves the reward signal or credit assignment but leaves the per-step action search unchanged. In this project, we instead try to shrink that search directly. We ask whether learning on the fly which discrete action tokens are persistently low-utility and pruning them from the sampling distribution can speed up training. Learned action elimination has accelerated RL when a reliable signal marks unhelpful actions. We therefore test whether the idea survives when the only signal is a noisy, on-policy utility estimate over a continuous-control token vocabulary.

Method We discretize each action dimension into a factored Categorical over $K{=}64$ token bins and train a PPO policy alongside an action-utility critic that scores every (dimension, bin) pair, regressing the taken token's utility onto the advantage with a VDN-style additive per-dimension credit. At each step the masker removes the lowest-utility tokens, up to a 25% budget, from the sampling distribution. To test whether learned selection is the source of any gain, we compare three arms at a matched budget: (1) learned utility masking, (2) a random-masking control that drops the same fraction uniformly, and (3) no masking.

Implementation We evaluate on four MetaWorld tasks of varying difficulty with a PPO implementation, three seeds per arm, and one million environment steps each. Because the argmax of a high-entropy factored categorical is off-distribution, we score every arm by its sampled-policy success rate rather than greedily.

Results Our key finding is that learned masking fails to beat either baseline. Only button-press separates the arms (reach and drawer saturate near 1.0 and pick-place near 0), and there, final success is 0.86 for no masking, 0.94 for random masking, and only 0.63 for learned masking, which also has the lowest area under the success curve (0.34 vs. 0.57 and 0.60) and is about twice as slow to reach 0.5 success. Ablations over bin count, masking strength, threshold, and warm-up never recover a selection benefit, and larger vocabularies ($K{=}128, 256$) instead degrade it sharply.

Discussion We find that random masking is a surprisingly strong baseline, and its benefit is stochastic regularization, which is a per-step perturbation of the available tokens, rather than the choice of which tokens to drop. This is because it improves over no masking while leaving policy entropy essentially unchanged. The learned selector forfeits that benefit by pruning deterministically, which collapses policy entropy onto a mis-ranked token set, and its per-token signal is too weak on-policy to compensate. At a matched budget the learned mask drives policy entropy lowest (about 8.0 versus 9.0–9.3 nats of a 16.6 maximum), and the critic's per-token ranking agrees only weakly with the policy's (rank correlation about 0.31). This means that even stochasticity-preserving learned masks fail to beat random at the same budget, confirming that perturbation as opposed to selection is what helps.

Conclusion We conclude that, at this scale, action-token pruning helps only as stochastic regularization, not as selection. A follow-up that applies the same utility signal as a soft loss-side penalty rather than a hard mask avoids the entropy collapse and recovers near-vanilla performance (0.90), indicating the failure mode is the deterministic masking mechanism, not the utility signal. This points to off-policy or contrastive utility estimation as the route to a learned selector that could outperform the random control.

Action-Token Pruning for Sample-Efficient RL Fine-Tuning of Robot Policies

Elvin Fu
Department of Electrical Engineering
Stanford University
elvinf@stanford.edu

Cole Van Hersett
Department of Data Science
Stanford University
cvh2112@stanford.edu

Abstract

RL fine-tuning of robot policies is expensive because much of the policy's exploration goes to action regions that never pay off. We ask whether learning, on the fly, which discrete action tokens are persistently low-utility and pruning them from the sampling distribution can speed up training. We discretize each action dimension into a factored Categorical over $K{=}64$ token bins and train a PPO policy alongside an action-utility critic, then mask the lowest-utility tokens up to a 25% budget; to isolate whether learned selection is the source of any gain, we compare against a budget-matched random mask and against no masking on four MetaWorld tasks. Learned masking fails to beat either baseline: on button-press, the only task that separates the arms, it reaches 0.63 final success against 0.86 for no masking and 0.94 for random masking, and is about twice as slow to learn. We find that random masking helps as stochastic regularization, a per-step perturbation of the available tokens, rather than through which tokens it drops, while the learned selector collapses policy entropy onto a mis-ranked token set. Applying the same utility signal as a soft loss-side penalty rather than a hard mask avoids the collapse and recovers near-vanilla performance, indicating the failure mode is the deterministic masking mechanism, not the utility signal. Action-token pruning at this scale therefore helps as regularization, not selection.

1 Introduction

RL fine-tuning of robot policies, including modern Vision-Language-Action (VLA) models like OpenVLA [Kim et al., 2024], is slow because much of the policy's exploration goes to action regions that turn out to be useless for the task. We investigate whether we can speed up training by learning, on the fly, which discrete actions are persistently bad and pruning them from the policy's sampling distribution. The goal of the project is to build this pruning mechanism, test it on a small-scale RL benchmark, and characterize when it helps and when it hurts. Compute is the dominant cost of fine-tuning large robot policies, so even modest reductions in wasted exploration are useful.

Prior work on RL fine-tuning of VLAs, such as VLA-RL [Lu et al., 2025], improves the reward signal or credit assignment but does not reduce the per-step action search, so the exploration cost is unchanged. Instead, we attempt to shrink that space directly. Our pruning mechanism adapts action elimination [Zahavy et al., 2018] to the per-dimension token vocabularies of modern VLAs, masking persistently low-utility tokens rather than the strictly invalid actions that method originally targets, which a continuous VLA token vocabulary cannot supply. Unlike failure-replay approaches such as FEMA [Miao, 2026], which adjust what the agent learns from but do not shrink the action space, we retain informative near-miss failures and prune tokens that are low-utility across many states. We hypothesize that if a per-token utility signal is learnable on-policy, learned pruning should beat random masking at an equal budget.

Stanford CS224R 2026 Final Report

2 Related Work

The most direct precedent for our method is a line of work that removes actions from an agent's choices by learning which ones to suppress. Action elimination [Zahavy et al., 2018] trains a separate network to flag context-dependent invalid actions and mask them from a value-based agent, and in policy-gradient methods the same idea is realized by setting the logits of invalid actions to a large negative value before the softmax, which reliably accelerates learning when validity is known from the environment [Huang and Ontañón, 2020]. This is the same logit-masking mechanism we use. More recent work prunes not merely invalid but low-value actions: CEE [Liu et al., 2025] estimates each action's causal effect on the next-state distribution with a pretrained model and masks those below a threshold. We adapt this family to a factored, per-dimension discretized action vocabulary and mask the lowest-utility tokens at every step. Unlike a binary validity flag [Huang and Ontañón, 2020] or a frozen, dynamics-based estimate [Liu et al., 2025], our utility is a weak, on-policy estimate of each token's task advantage learned alongside the policy. Since the masking mechanism is identical, the open question is whether such a noisy learned signal can drive it.

Our policy rests on a method that discretizes each continuous action dimension into a categorical over bins and factorizes the joint action as a product of independent per-dimension distributions. Tang and Agrawal [2020] show that discretizing continuous control into per-dimension categoricals is competitive when optimized on-policy with PPO and TRPO, and the branching architecture of Tavakoli et al. [2018] places one head per action dimension on a shared trunk to avoid the combinatorial blow-up of the joint action space. This factorization mirrors the action-token heads of modern VLAs [Kim et al., 2024]. One known cost of this choice bears directly on our results. When the ordinal structure of the bins is ignored, finer discretization inflates the variance of the policy-gradient estimator [Zhu et al., 2024], which helps explain why a per-token utility signal over $K{=}64$ unordered bins is hard to estimate on-policy.

Beyond learned elimination, a broad body of work reshapes the action space by hand to ease learning. Kanervisto et al. [2020] systematically show that deliberately removing, discretizing, or restricting actions frequently helps, but every condition they study chooses what to remove from domain knowledge, and none compares against removing the same number of actions uniformly at random. This leaves open the question of whether the gains come from removing the useless actions specifically or merely from shrinking the space. That budget-matched random-removal control is exactly what our experiments supply.

Finally, our motivation comes from online RL fine-tuning of large robot and vision-language-action policies, where two strategies dominate. Reward-side methods such as VLA-RL [Lu et al., 2025] improve the reward or credit-assignment signal with a learned process-reward model but leave the action distribution untouched, so the per-step search is unchanged, while failure-centric methods such as FEMA [Miao, 2026] reuse bad episodes to steer the agent away from hazardous states without shrinking the action space. We instead reduce the per-step action search directly, while retaining the informative near-miss failures that carry credit-assignment signal.

3 Method

Our method augments a standard PPO agent with a learned mechanism for pruning unpromising actions on the fly. We discretize the continuous action space into a per-dimension token vocabulary and, alongside PPO, train an action-utility critic that scores every token in the current state. At each step the masker removes the lowest-utility tokens in each dimension up to a fixed budget from the sampling distribution so exploration is steered away from tokens the critic judges persistently bad. The rest of this section describes the discretized action policy, the action-utility critic, the token-masking rule, and the value normalization that keeps both critics stable (Figure 1).

Discretized action policy. To make token-level masking possible, we replace the usual Gaussian actor with a discretized one. Each of the four action dimensions is partitioned into $K = 64$ bins evenly spaced on $[-1, 1]$, and the policy emits an independent Categorical over the bins of each dimension, so the joint distribution and its log-probability factorize as

$$\pi(a \mid s) = \prod_{d=1}^{4} \pi_d(a_d \mid s), \qquad \log \pi(a \mid s) = \sum_{d=1}^{4} \log \pi_d(a_d \mid s).$$

This factored, per-dimension token head is the same structure modern VLAs use for their action heads [Kim et al., 2024], which makes the setup a faithful small-scale stand-in for VLA fine-tuning. It also exposes a discrete vocabulary that a masker can prune, which a continuous Gaussian head does not. A single two-layer, 64-unit MLP produces all $4 \times K$ logits, and the policy is trained with clipped PPO [Schulman et al., 2017] (clip 0.2, GAE $\lambda = 0.95$, $\gamma = 0.99$).

Action-utility critic. Alongside the policy we train a second MLP, the action-utility critic $U_\phi(s) \in \mathbb{R}^{4\times K}$, which predicts a scalar utility for every (dimension, bin) token from the state alone. It learns from the same on-policy rollouts as PPO and so adds no extra environment interaction. Rather than regress each taken token onto the raw return, we regress the sum of the per-dimension taken-token utilities onto the transition advantage $\hat{A}$, a value-decomposition (VDN-style) objective:

$$\mathcal{L}(\phi) = \mathbb{E}\left[\left(\sum_{d=1}^{4} U_\phi(s)[d, a_d] - \hat{A}\right)^2\right].$$

We regress onto the advantage rather than the return because the return is dominated by the state value $V(s)$ (which reaches $\sim 10^3$ on these tasks), so per-token differences would be a tiny residual on a large shared offset. The advantage subtracts that baseline and leaves a signal about a token's relative quality. We also use the additive decomposition because it forces each dimension's head to explain only its own share of the advantage given the others, disentangling dimensions that co-vary on-policy. Regressing every dimension's taken token onto the same scalar advantage (a "broadcast" target) is a weaker variant we keep only as an ablation. Supervision is sparse, since only the bin actually sampled in each dimension receives a target on a given step, so the masker relies on the ordering of utilities within a dimension rather than their absolute level.

Token masking. At each step the masker scores the current state with U_ϕ, ranks the K bins within every dimension, and removes the lowest-utility ones from the sampling distribution by pushing their logits to a large negative value before the Categorical is formed. The number removed is capped at 25% of K and ramped up from zero over an initial warm-up, giving the critic time to produce a meaningful ranking before any token is pruned. The mask that generated an action is stored in the rollout buffer and re-applied when PPO recomputes log-probabilities, so the importance ratios are evaluated under exactly the masked distribution that produced the data. To guarantee at least one admissible action per dimension we never prune that dimension's top-utility token, and we compare hard masking (an effectively $-\infty$ logit offset) against soft masking (a fixed finite penalty).

Value normalization. MetaWorld's dense shaping rewards push episode returns to $\sim 10^3$, a scale at which value regression becomes unstable because the value head and the utility critic would have to fit targets spanning several orders of magnitude. We therefore standardize the value targets online with a running estimate of their mean and variance, so both critics regress onto comparable $O(1)$ quantities, and de-normalize the predictions wherever real-scale values are needed (for example, when bootstrapping GAE). This normalization was enabled in all reported runs.

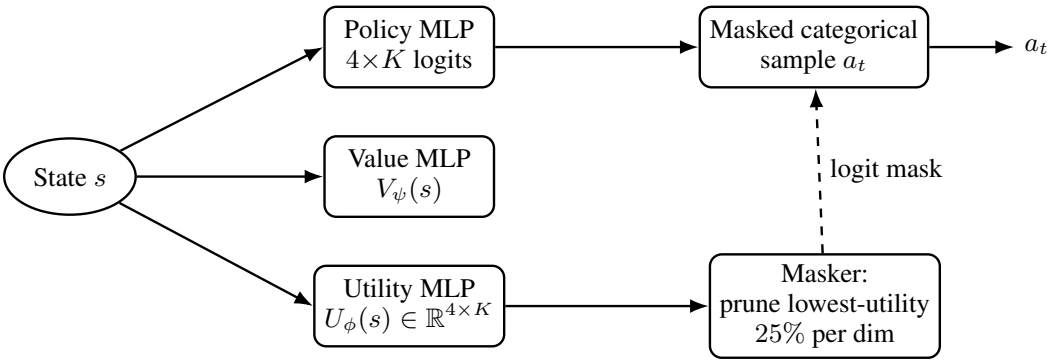

Figure 1: Method overview. Three MLPs map the state to a factored-categorical policy ($4\times K$ logits), a value head $V_\psi(s)$, and an action-utility table $U_\phi(s)$; each step, the masker prunes the lowest-utility 25% of tokens per dimension before sampling and stores the mask so PPO ratios are recomputed under the same distribution.

4 Experimental Setup

We evaluate the three masking arms on a suite of MetaWorld manipulation tasks, holding the policy, critic, and training budget fixed so that any difference in outcome is attributable to the masker alone.

Tasks. We use four tasks of increasing difficulty: reach, button-press, drawer-close, and pick-place, chosen to span the range over which masking could plausibly matter. Reach and drawer-close are easy enough that every arm saturates near a 1.0 success rate, and pick-place is hard enough that every arm floors near 0, so none of the three can separate the methods. Almost all of the signal therefore comes from button-press, which has enough headroom to reveal a difference yet is solvable within the budget. We therefore treat button-press as the primary task throughout.

Arms. The three arms share an identical PPO policy, value critic, and step budget and differ only in the masker. **Vanilla** applies no mask and is ordinary discretized PPO. **Random** removes the same 25% of tokens per dimension as the learned method but chooses them uniformly at random, holding the size of the perturbation fixed while discarding any information about which tokens are dropped. **Utility** is our learned mask, which removes the 25% of tokens the utility critic scores lowest. The comparison that matters is utility against random because the two differ only in whether the dropped tokens are chosen or arbitrary. This isolates the value of learned selection from the value of masking itself.

Configuration. Every run uses $K = 64$ bins per dimension and ramps the mask budget from 0% to 25% over steps 50k–150k, giving the critic time to warm up before any token is pruned. PPO uses a rollout of 2048 steps, 10 epochs per update, a minibatch of 64, a clip ratio of 0.2, a learning rate of 3×10^{-4}, and target-KL early stopping; the full hyperparameters are listed in Table 1. We train for 1M environment steps with three seeds per arm on Modal T4 GPUs and log to Weights & Biases.

Ablations. On button-press we additionally sweep the design choices most likely to affect the learned mask, which include the bin count $K \in \{32, 64, 128, 256\}$, hard versus soft masking, the utility threshold, the warm-up length, and the credit-assignment scheme (additive versus broadcast).

Evaluation. We report each arm's success rate under its sampled policy as a function of environment steps, averaged over seeds. We deliberately avoid greedy (argmax) evaluation. With a high-entropy factored categorical the argmax action lies off the distribution the policy actually visits and scores near 0 even when the sampled policy succeeds reliably so a greedy protocol would understate every arm. Scoring all arms under the same sampled protocol keeps the comparison fair.

Table 1: Training configuration (experiment values, not code defaults).

Component	Value
Bins per dimension K	64
Action dimensions	4
Mask budget	25% (ramp $0 \rightarrow 25\%$ over 50k–150k steps)
Rollout length	2048
Epochs / minibatch	10 / 64
PPO clip ratio	0.2
Learning rate	3×10^{-4}
Discount γ / GAE λ	0.99 / 0.95
Value-target normalization	on (running mean/std)
Early stopping	target-KL
Training steps	1×10^{6}
Seeds per arm	3
Hardware	Modal T4 GPU
Tasks	reach, button-press, drawer-close, pick-place

5 Results

5.1 Quantitative Evaluation

Our main result is negative. Learned utility masking never beats both baselines, and on the one task that can separate the methods it is the weakest and the slowest to learn. Table 2 gives final success and area-under-curve for every task and arm, and Figures 2 and 3 summarize the button-press curves and sample efficiency.

Only button-press tells the methods apart. As anticipated in the setup, reach and drawer-close saturate near 1.0 for all three arms while pick-place floors near 0 for all of them, so on three of the four tasks the arms are indistinguishable (Table 2). Button-press is the only task with both headroom and a solvable budget, so the remaining analysis focuses there.

Utility loses where it should win. On button-press, final rollout success is 0.86 for vanilla, 0.94 for random, and only 0.63 for learned utility masking (Figure 2). The learned method is thus worse than masking nothing at all, and worse still than masking the same number of tokens at random. This is the exact opposite of what it is designed to do. We report the utility arm with the additive (VDN) credit critic, the stronger of the two credit schemes; the earlier broadcast variant is weaker yet, at 0.51.

It is also the slowest to get there. Beyond the final number, utility masking is the least sample-efficient arm. It has the lowest normalized area under the success curve (0.34, against 0.57 for vanilla and 0.60 for random) and needs nearly twice as many steps to first reach 0.5 success (628k, against 321k for vanilla and 338k for random; Figure 3). The learned mask therefore does not merely converge to a worse policy, but also slows learning throughout.

Ablations do not rescue it. No configuration we tried recovers a selection benefit (Table 3). Hardening the mask or lengthening the warm-up helps relative to the default (0.73 and 0.80 respectively) but still trails vanilla and random, and enlarging the vocabulary hurts sharply (0.27 at both $K{=}128$ and $K{=}256$), consistent with the higher policy-gradient variance of fine, unordered discretization. With only three seeds per cell, however, the standard deviations are large (up to about ± 0.17), and the random-over-vanilla gap is small relative to that spread.

Table 2: Final rollout success (mean $\pm$ std over 3 seeds) with normalized AUC in parentheses. Only button-press separates the arms. †Button-press utility uses additive/VDN credit (the final method); the earlier broadcast-credit run scored 0.513 ± 0.145 (AUC 0.366).

Task	vanilla	random	utility
reach	1.000 ± 0.000 (0.795)	0.997 ± 0.005 (0.803)	0.990 ± 0.008 (0.810)
button-press	0.857 ± 0.045 (0.567)	0.940 ± 0.037 (0.596)	0.633 ± 0.088 (0.343)†
drawer-close	1.000 ± 0.000 (0.951)	1.000 ± 0.000 (0.948)	0.990 ± 0.014 (0.910)
pick-place	0.000 ± 0.000 (0.002)	0.007 ± 0.005 (0.002)	0.003 ± 0.005 (0.002)

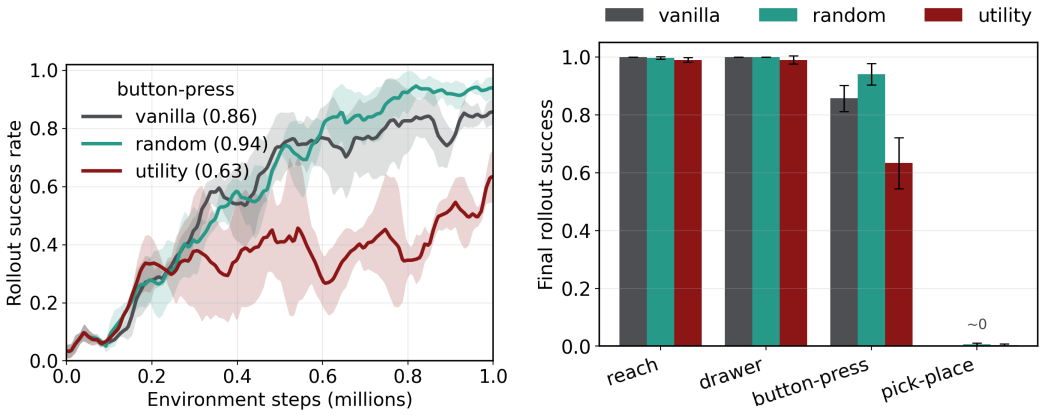

Figure 2: Left: button-press rollout success vs. environment steps (3 seeds, shaded $\pm$ std). Right: final rollout success across the four tasks.

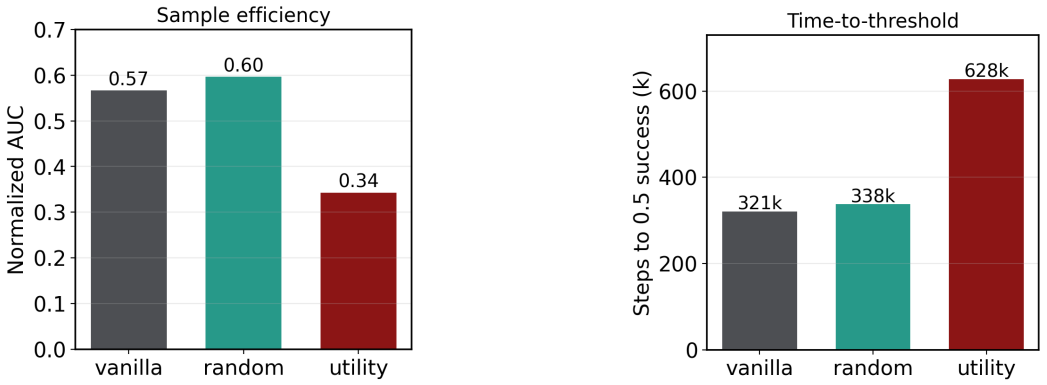

Figure 3: Sample efficiency on button-press: normalized area under the success curve (left) and steps to 0.5 success (right).

Table 3: Button-press ablations (final success $\pm$ std, normalized AUC), broadcast/default config unless noted. Reference: vanilla 0.857, random 0.940.

Variant	Final success	AUC
K=32	0.717 ± 0.173	0.517
K=64 (default)	0.513 ± 0.145	0.366
K=128	0.270 ± 0.110	0.199
K=256	0.277 ± 0.095	0.137
hard mask	0.727 ± 0.037	0.451
soft mask (default)	0.513 ± 0.145	0.366
threshold 0.25 (default)	0.513 ± 0.145	0.366
threshold 0.5	0.637 ± 0.165	0.435
warm-up 150k	0.797 ± 0.026	0.448
additive credit	0.633 ± 0.088	0.343

5.2 Qualitative Analysis

Three measurements on button-press characterize the failure. Policy entropy (Figure 4) begins near the $4 \ln 64 \approx 16.6$-nat maximum for every arm, but the learned mask drives it lowest late in training (about 8 nats, versus 9.0–9.3 for no masking and random), while random masking tracks no masking

closely. The policy is nonetheless far from collapsed when masking begins: at 150k steps it still spreads its mass over roughly 30 effective bins per dimension (the exponential of its per-dimension entropy), so each dimension offers many live candidates to rank. The critic ranks them poorly: on a held-out checkpoint its per-bin ordering matches the policy's at a Spearman correlation of only about 0.31. Many candidate tokens paired with a weak, noisy per-token signal is precisely the regime in which confident deterministic pruning discards useful actions.

6 Discussion

Why does utility masking fail? We propose that at an equal token budget, the learned mask collapses the policy's entropy onto a token set that the imperfect critic has mis-ranked, committing the agent early to a narrow and suboptimal slice of the action space. Random masking removes the same fraction of tokens yet leaves entropy essentially unchanged, so its benefit is a per-step stochastic regularization rather than any increase in exploration. The analysis below develops this account and traces it through the entropy curves (Figure 4) and a set of further experiments.

Premature entropy collapse. The clearest symptom is in the entropy curves. At the same 25% budget the learned mask drives policy entropy lower than any other arm, to roughly 8.0 nats late in training against 9.0–9.3 for no masking and random (out of a 16.6-nat maximum). By repeatedly deleting whichever tokens the critic scores lowest, the mask concentrates probability mass on a small set of bins early, before the policy has discovered which actions actually succeed. Because that ranking is imperfect, the set the policy commits to is itself suboptimal, so utility masking both learns more slowly and plateaus at a lower success rate than simply leaving the action space intact.

Random masking regularizes rather than explores. The random control removes just as many tokens, yet its policy entropy stays essentially indistinguishable from no masking, while it still trains faster and reaches a higher final success rate. The benefit therefore cannot be coming from broader exploration. Instead it is a per-step perturbation of which tokens are available, a form of stochastic regularization that discourages the policy from over-committing to any single bin while leaving its overall spread intact. This is the precise contrast with the learned mask, which lowers entropy onto a mis-ranked set instead of supplying that regularizing churn.

The per-token signal is too weak to rank. For a learned mask to beat a random one, the critic must reliably tell good tokens from bad, and the diagnostics above show it cannot: the policy keeps many near-equivalent candidate tokens per dimension while the critic's per-bin ordering correlates only weakly with the policy's own. A single advantage shared across an entire transition simply does not carry enough information to order K tokens in each dimension. Deterministically pruning the bottom quarter of such a ranking is then about as likely to remove a useful action as a useless one, which is why the same budget that helps when spent at random hurts when spent on the critic's choices.

A narrow operating regime. Finally, masking can change outcomes only where two conditions hold at once: there is headroom to improve, and there is a learnable per-token signal to exploit. Easy tasks such as reach and drawer saturate regardless of masking, and very hard tasks such as pick-place bottom out for every arm, so neither can reveal a difference. Only button-press offers headroom, and there the second condition fails. This narrowness, rather than a defect in any single component, is what limits the reach of action-token pruning at the scale we study.

Mechanism versus signal. Two further comparisons separate the masking mechanism from the utility signal. First, at a matched masking budget, stochasticity-preserving learned variants still fail to beat random removal. A Gumbel-top-k directed mask reaches 0.50 at a 40% budget against 0.72 for random, and 0.66 at a 10% budget against 0.63 for random (a tie within noise), while an additive boost toward high-utility tokens reaches 0.62, below random. This confirms that the regularization value of stochasticity, not token selection, is what helps. Second, applying the utility signal as a soft loss-side penalty toward $\mathrm{softmax}(\tau U_\phi)$ rather than as a hard action mask avoids the entropy collapse and recovers near-vanilla performance: 0.90 for the utility-tilted penalty (τ=4) and 0.85 for the entropy-only version (τ=0), both well above the 0.63 of hard masking though still below random's 0.94. The failure mode is therefore the hard, deterministic masking mechanism, not the utility signal itself (Table 4).

Table 4: Button-press masking variants and loss-side shaping (final rollout success mean $\pm$ std, normalized AUC). The budget-sweep, directed, explore, and loss-shaping arms use a separate seed batch, so their random-25% value (0.810) differs from the headline run's (0.940); n is the number of seeds reaching $\geq$300k steps.

Family	Arm	n	Final (AUC)
Baseline	vanilla	3	0.857 ± 0.045 (0.566)
Baseline	random (25%)	3	0.940 ± 0.037 (0.596)
Baseline	utility mask (additive)	3	0.633 ± 0.088 (0.343)
Budget sweep	random (10%)	3	0.633 ± 0.184 (0.380)
Budget sweep	random (25%)	3	0.810 ± 0.078 (0.335)
Budget sweep	random (40%)	3	0.717 ± 0.074 (0.289)
Directed mask	directed (10%, α=4)	3	0.657 ± 0.274 (0.347)
Directed mask	directed (40%)	6	0.503 ± 0.062 (0.217)
Explore boost	explore (β=2)	3	0.617 ± 0.147 (0.299)
Loss shaping	KL penalty (τ=0)	3	0.853 ± 0.094 (0.384)
Loss shaping	KL penalty (τ=4)	4	0.895 ± 0.082 (0.395)

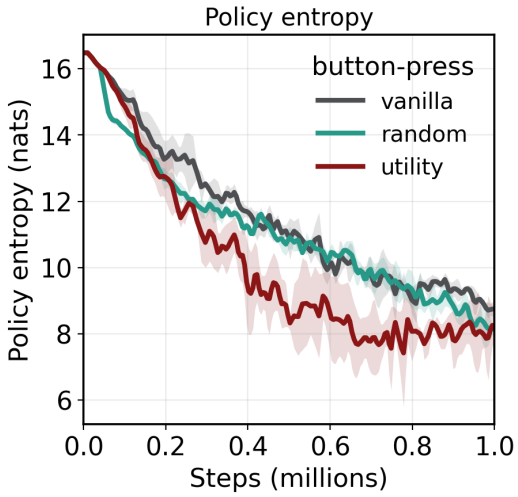

Figure 4: Policy entropy (nats) vs. environment steps on button-press; the maximum entropy is $4 \ln 64 \approx 16.6$ nats. The learned utility mask drives entropy lowest at an equal masking budget.

7 Conclusion

We set out to test whether learning which action tokens to prune could speed up RL fine-tuning, and at the scale we study the answer is no. Action-token pruning did not improve sample efficiency, and the surprise was the control. The random mask we included only to isolate the mechanism turned out to be a strong baseline in its own right, with a benefit that the analysis traces to stochastic regularization rather than to dropping the right tokens.

On the one task that separates the methods, learned utility masking underperforms both vanilla PPO and random masking, and it is also the slowest to learn. Since random masking helps just as much without choosing which tokens to drop, the benefit is coming from stochastic action selection, not token selection. Thus a learned selector has a high bar to match random pruning, much less beat it. Stochasticity-preserving learned masks still do not beat a budget-matched random mask, but moving the utility signal off the action space into a soft loss-side penalty avoids the entropy collapse and recovers near-vanilla performance (0.90), so the failure mode is the deterministic masking mechanism rather than the utility signal.

Future work. Promising directions include a stronger per-token signal through off-policy or contrastive utility estimation, combining pruning with exploration rather than treating it as a replacement, and larger budgets with real VLA fine-tuning, where random tuning may start to hurt and a learned selector could outperform.

8 Team Contributions

- **Elvin Fu:** Implemented PPO through the baseline, and resolved instabilities in our baseline models by adding the return and observation normalization needed for stable training. Synthesized all experimental data into the final results and graphics for the poster, and helped design and write the poster. After the presentation, continued iterating on the masking strategies to verify the negative results and to rule out code-structure issues.
- **Cole Van Hersett:** Set up the initial codebase (with Claude.ai assistance) and validated the baselines. Implemented the masking strategies that served as the main control and experimental methods, and helped run rollouts. Built the token-analysis tooling and the pruned-token heat maps. Coordinated the milestone, drove the analysis and write-up of the final results, and helped create the poster.

Changes from Proposal: The method changed from the proposed state-conditioned threshold on Monte-Carlo/GAE returns to regressing a per-token utility onto the PPO advantage with additive, VDN-style per-dimension credit, which gave a more stable signal. We also ran four MetaWorld tasks rather than five and dropped the proposed OpenVLA/LIBERO fine-tuning stretch under compute limits. The proposal assigned the masking mechanism and the stretch experiment to Cole and the baseline and infrastructure to Elvin, but reaching a stable baseline required substantial unplanned normalization work that Elvin took on. Once learned masking failed to beat the random control, jointly verifying and analyzing that negative result became the priority. Both of us converged on PPO debugging and the failure-mode follow-ups rather than the separate tracks we had planned.

9 AI-Tools Disclosure

We used AI tools to scaffold the initial codebase, wire up Modal and Weights & Biases, and lay out the repository. We also used them, under close supervision, as a sounding board for debugging and strategy iteration. All AI-assisted code and suggestions were reviewed and validated by the authors.

References

Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms, 2020.

Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. Action space shaping in deep reinforcement learning, 2020.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024.

Wenzhang Liu, Lianjun Jin, Lu Ren, Chaoxu Mu, and Changyin Sun. Reducing action space for deep reinforcement learning via causal effect estimation, 2025.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025.

Chenyang Miao. Learning from failures: Efficient reinforcement learning control with episodic memory, 2026.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization, 2020.

Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning, 2018.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2019.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J. Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning, 2018.

Yuanyang Zhu, Zhi Wang, Yuanheng Zhu, Chunlin Chen, and Dongbin Zhao. Discretizing continuous action space with unimodal probability distributions for on-policy reinforcement learning, 2024.