END-TO-END DIFFERENTIABLE FEATURE SELECTION FOR DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

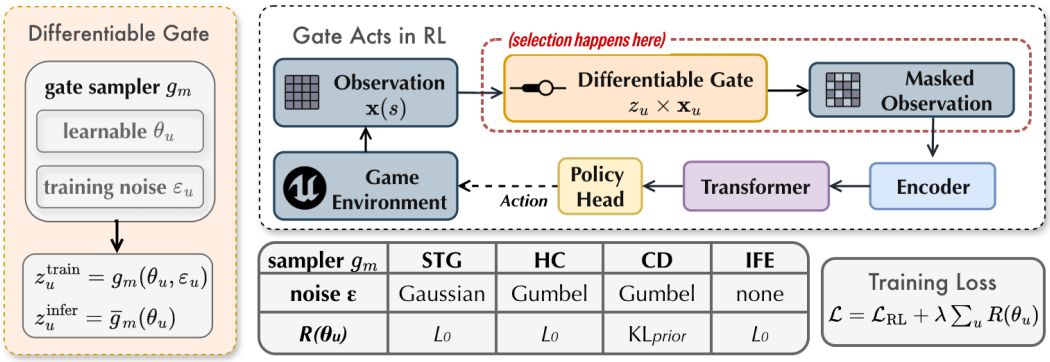
Lightweight reinforcement learning (RL) agents must remove unnecessary raw observation features without sacrificing control performance. Differentiable feature gates are a natural approach, but their standard theory assumes a fixed dataset, as in supervised learning. In RL that assumption breaks down: the gate reshapes the policy, the policy reshapes its trajectories, and those trajectories drive gate updates. We systematically study end-to-end differentiable feature selection under this policy-dependent feedback, comparing four gate mechanisms on a live commercial racing game with a $4{,}244$-dimensional observation across two key design choices: stochastic versus deterministic gating, and expected-L_0 versus KL-to-prior regularization. Sweeping the number of selected features, the best mechanism, Hard Concrete, removes 70.1% of the inputs while cutting completion time across twelve maps from 37.4 to 34.0 seconds, a 9.1% improvement. Gate masks, however, reveal which features are selected, not how strongly the trained policy relies on them. We introduce GateKL, which toggles one gate at inference—closing an open gate or opening a closed one while keeping the policy parameters and all other learned gates fixed—and measures the resulting change in the action distribution using KL divergence. Across all $4{,}244$ gates and twelve maps, gate values learned by Hard Concrete and Stochastic Gates closely reflect policy reliance, whereas those learned by Concrete Dropout and Interpretable Feature Extractor carry little such information. These findings distinguish sparse selection from policy reliance and show why gate masks alone are insufficient to assess learned feature selection.

1 INTRODUCTION

Modern reinforcement learning (RL) agents often consume high-dimensional, heterogeneous raw observations, even though only part of them may be useful for the task at hand. This creates a practical deployment question: can a policy discard unnecessary features without sacrificing control quality? Existing work addresses adjacent settings, but not this one. Feature selection for RL typically studies linear value-function approximation or a fixed feature basis (Parr et al., 2008; Kolter & Ng, 2009; Painter-Wakefield & Parr, 2015; Hao et al., 2021; Suzuki & Slavakis, 2025); deployment-oriented deep RL lightweighting usually compresses the network rather than the raw observation consumed by the policy (Rusu et al., 2016; Han et al., 2016). We study end-to-end differentiable feature selection inside a deep RL policy, a problem that has not been systematically studied before.

To do so, we place a learnable gate between the raw observation and the projection encoder, and optimize it jointly with the RL objective (Figure 1). Unlike wrapper methods that retrain a policy for every candidate subset, this design learns feature selection through one joint, differentiable training process and produces a deterministic feature mask at inference. We compare four differentiable gate samplers: Stochastic Gates (STG) (Yamada et al., 2020), Hard Concrete (HC) (Louizos et al., 2018), Concrete Dropout (CD) (Gal et al., 2017), and Interpretable Feature Extractor (IFE) (Pham & Cangelosi, 2025). STG, HC, and CD originate in fixed-data selection or representation-learning settings, whereas IFE uses a similar feature mask for deep-RL interpretability. Together, they span two central design choices: stochastic versus deterministic sampling, and an expected-L_0 versus a KL-to-prior regularizer. **Crucially, their gate-level selection designs do not model gate–policy–data coupling: in RL, the gate changes the policy, the policy changes the trajectories it visits,**

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.



sampler g_m	STG	HC	CD	IFE
noise ε	Gaussian	Gumbel	Gumbel	none
$R(\theta_u)$	L_0	L_0	KL_{prior}	L_0

Training Loss
$\mathcal{L} = \mathcal{L}_{\text{RL}} + \lambda \sum_u R(\theta_u)$

Figure 1: Overview of gate-based differentiable feature selection inside an RL policy. The lower table summarizes each mechanism's training noise and sparsity penalty (L_0: expected-L_0 surrogate; $\mathrm{KL}_{\text{prior}}$: KL-to-prior penalty). Section 3 formally defines the gate mappings and training objective.

and those trajectories determine subsequent gate updates. Fixed-data conclusions about which sampler selects a better or sparser subset therefore do not directly transfer to RL.

Gate masks alone do not reveal how good the learned feature subset is: they show which features are selected or removed, but not how much toggling a single gate changes the policy's actions. This motivates the question: at inference, do the gate values agree with the trained backbone's responses? We propose **GateKL**, which toggles one gate at inference—closing an open gate or opening a closed gate while holding every other gate fixed—and measures the resulting change in the action distribution. Our contributions are as follows:

- **A systematic study of end-to-end differentiable feature selection in deep RL** (Section 3). We gate the raw observation of a deep RL policy and compare four gate mechanisms under the RL training objective: STG, HC, and CD, previously used in supervised learning, and IFE, previously used for RL interpretability. We distinguish this setting from linear/fixed-basis RL selection, wrapper search, and network compression.
- **A GateKL analysis of single-gate toggles on the trained policy** (Sections 3.3 and 4.4). Across all $4{,}244$ gates and twelve maps, HC and STG learn gate values that faithfully reflect which gates the trained policy relies on, while those of CD and IFE carry little such information, making HC and STG the preferable gate mechanisms for feature selection in deep RL.
- **Evidence from a live commercial game** (Section 4). On a $4{,}244$-dimensional racing agent, sweeping the number of selected features k reveals a broad near-optimal region: at $k{=}1270$ (70.1% of the features removed), HC lowers the mean completion time from 37.4 to 34.0 seconds (-9.1%) across twelve maps, while more extreme compression ($k{=}210$) raises the mean back to 35.7 seconds.

2 Related Work

Feature selection for reinforcement learning. A first line treats feature selection as part of value-function approximation: classical analyses of linear value functions (Parr et al., 2008), sparsity-inducing penalties in temporal-difference learning (Kolter & Ng, 2009; Geramifard et al., 2012; Suzuki & Slavakis, 2025), and selection via low-rank or mutual-information structure (Behzadian & Petrik, 2019; Hachiya & Sugiyama, 2010). Most relevant, Hao et al. (2021) prove that sparse selection improves batch-RL sample efficiency under a group-sparse linear model, with minimax rates that we build on; Liu et al. (2015) survey this setting, and high-dimensional action selection has been studied with false-discovery-rate control (Zhang & Cai, 2025). These works assume linear approximation and a **fixed** feature basis; the closest feature-side method, dynamic sparse training for automatic noise filtering (Grooten et al., 2023), instead relies on discrete prune-and-regrow masks. Our setting is a deep, non-linear policy whose feature dependence co-evolves with training (full account in Appendix G).

Table 1: Task-level contrast with prior lines of work. **Inputs**: selects the policy's raw observation features; **Diff. gate**: selection is an end-to-end differentiable gate trained with the policy; **On-policy**: the training data are generated by the policy being gated.

Task	Inputs	Diff. gate	On-policy
Value-function selection (Parr et al., 2008; Kolter & Ng, 2009; Hao et al., 2021)	✓	×	×
Wrapper selection (Hazrati Fard et al., 2013; Rasoul et al., 2021)	×	×	×
Embedded differentiable selection (Louizos et al., 2018; Yamada et al., 2020; Gal et al., 2017)	×	✓	×
Network lightweighting (Rusu et al., 2016; Han et al., 2016)	×	×	×
Input noise filtering (Grooten et al., 2023)	✓	×	✓
This work	✓	✓	✓

Differentiable embedded feature selection. Our gates derive from differentiable L_0 relaxations: Louizos et al. (2018) introduce the Hard Concrete distribution for sparse networks, and Yamada et al. (2020) propose Stochastic Gates, a Gaussian relaxation selecting at $z>0$; both give closed-form expected-L_0 surrogates (Table 2). Concrete autoencoders (Balın et al., 2019) pair Gumbel–Softmax selection with a reconstruction objective. All build on concrete reparameterizations (Jang et al., 2017; Maddison et al., 2017). We run HC and STG in the same high-dimensional RL setting, each under its own inference rule, and show that—even at matched sparsity—they differ in how the policy relies on selected features, a difference GateKL exposes at the current inference gate (Proposition 1), unlike gate counts alone.

Lightweighting deep RL agents. Dominant lightweighting methods compress the network rather than select the agent's features: policy distillation (Rusu et al., 2016; Furlanello et al., 2018), pruning–quantization pipelines and architecture transforms (Han et al., 2016; Chen et al., 2016), and compact representations via successor features or bisimulation abstractions (Barreto et al., 2017; Kemertas & Aumentado-Armstrong, 2021). The closest feature-side precedent, automatic noise filtering with dynamic sparse training (Grooten et al., 2023), updates binary masks by discrete prune-and-regrow rules; we instead train the gate end-to-end by gradient through differentiable L_0 relaxations (full account in Appendix G).

Positioning. No prior line selects a deep RL policy's raw features end-to-end with the policy: classical RL-side selection assumes linear value functions over a fixed basis; differentiable selection targets supervised data; deployment-oriented methods compress the network or latent state, not the features (Table 1). We instead gate features end-to-end inside a high-dimensional deep RL policy.

3 Policy-Conditioned Feature Selection

This section separates three questions that are often mixed up in gated RL: which features a gate keeps, what happens to the policy's actions when we toggle one gate after training, and whether gate values predict how large those action changes are. Our observation is high-dimensional, and its feature groups are processed by shared encoders, so the gate acts on a structured observation rather than an arbitrary flattened vector. We introduce the four gate mechanisms, then define the current-gate intervention. Section 4 evaluates the resulting selection and gate-value–GateKL alignment separately.

3.1 Setup

The model gates the raw observation at state s with scalar gates indexed by $u \in \mathcal{U}$, applied **before** the projection encoder:

$$\bar{x}_u(s) = z_u \, x_u(s), \qquad z_u \in [0, 1],$$

where $x_u(s)$ is the scalar feature controlled by gate u (one of the 4,244 input dimensions). At training time, a mechanism $m \in \{\text{STG}, \text{HC}, \text{CD}, \text{IFE}\}$ turns a learnable scalar θ_u and, when applicable, a noise sample ϵ_u into the training-time gate value

$$z_u^{\text{train}} = g_m(\theta_u, \epsilon_u),$$

where g_m is the mechanism's training-time rule. At inference, it uses the deterministic counterpart $z_u^{\text{infer}} = \bar{g}_m(\theta_u)$. Thus the gate value z_u is not an independent model weight: it controls which raw signal reaches the encoder. We refer to the encoder and policy head together as the **backbone**. Table 2 gives the exact training and inference rules, the noise each mechanism adds, and the sparsity regularizer for the four mechanisms.

During training, the learner jointly minimizes its RL loss and the gate penalty:

$$\mathcal{L} := \mathcal{L}_{\text{RL}} + \lambda \sum_{u \in \mathcal{U}} R(\theta_u), \tag{1}$$

where $\mathcal{L}_{\text{RL}}$ is the RL loss (in our case, the PPO surrogate loss with an entropy bonus), R is the selected mechanism's sparsity regularizer, and $\lambda \geq 0$ controls the strength of that penalty.

Table 2: Four differentiable gate mechanisms compared in this work: training and inference gate forms, sparsity regularizer, and noise source. θ_u is a learnable scalar per gate; L is a closed-form bias aligning the surrogate with the expected L_0; $\sigma(\cdot)$ is the logistic sigmoid, σ_{noise} the STG Gaussian noise scale, $\Phi(\cdot)$ the Gaussian CDF, g Gumbel noise, τ the temperature, (γ, ζ) the Hard Concrete stretch bounds, p_0 the sparsity prior of CD. Default parameters follow the original papers.

Gate	Training z_u	Regularizer $R(\theta_u)$	Inference z_u	Noise
STG	$\text{clip}(\theta_u{+}0.5{+}\sigma_{\text{noise}}\varepsilon,\ 0,\ 1)$	$\Phi\left(\frac{\theta_u+0.5}{\sigma_{\text{noise}}}\right)$	$\text{clip}(\theta_u{+}0.5,\ 0,\ 1)$	**additive Gaussian** ε on raw θ, $\sigma_{\text{noise}}{=}0.5$
HC	$\text{clip}\left(\sigma\left(\frac{\theta_u+L+g}{\tau}\right)(\zeta{-}\gamma){+}\gamma,\ 0,\ 1\right)$	$\sigma(\theta_u - \tau\log(-\gamma/\zeta))$	$\text{clip}(\sigma(\theta_u)(\zeta{-}\gamma){+}\gamma,\ 0,\ 1)$	**Gumbel** g, τ:1$\to$0.1
CD	$\text{clip}\left(\sigma\left(\frac{\theta_u+L+g}{\tau}\right)(\zeta{-}\gamma){+}\gamma,\ 0,\ 1\right)$	$\text{KL}(\sigma(\theta_u) \,\|\, p_0)$	$\text{clip}(\sigma(\theta_u)(\zeta{-}\gamma){+}\gamma,\ 0,\ 1)$	**Gumbel** g, $\tau{=}0.5$
IFE	$\text{clip}(\sigma(\theta_u)(\zeta{-}\gamma){+}\gamma,\ 0,\ 1)$	$\sigma(\theta_u)$	$\text{clip}(\sigma(\theta_u)(\zeta{-}\gamma){+}\gamma,\ 0,\ 1)$	**none**

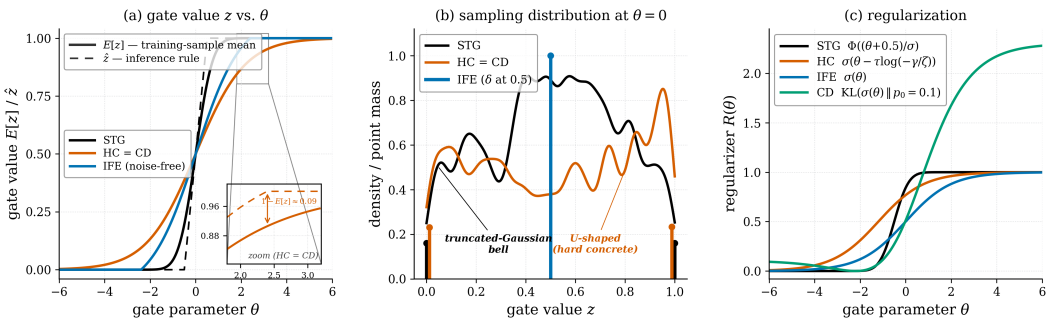


Figure 2: How the four gate mechanisms of Table 2 differ, panel by panel: (a) the gate value as a function of θ: the expected training-time value $E[z]$ (solid: mean of the stochastic training samples z^{train}) and the deterministic inference rule z^{infer} (dashed); the inset magnifies the region where the HC/CD stretch-and-clip rule saturates at exact 0 and 1. (b) Sampling distribution at the transition midpoint $\theta{=}0$ ($E[z]{=}0.5$ for every mechanism), from 20,000 seeded Monte-Carlo draws per mechanism: stems mark point masses at exact 0 and 1 (stem height = probability), curves the continuous part of each sampling density; IFE is a single point mass at $z{=}0.5$. (c) Sparsity regularizers $R(\theta)$. Illustrative parameters: $\sigma_{\text{noise}}{=}0.5$, $\tau{=}0.5$ (mid-annealing for HC, fixed for CD), $(\gamma, \zeta){=}(-0.1, 1.1)$.

3.2 Comparing gate mechanisms

All four mechanisms compute the gate value z_u from θ_u (plus noise where applicable) at the same position before the encoder. They differ in three ways: how the gate value depends on θ, the shape

of the training-time sampling distribution, and the sparsity regularizer. Figure 2 visualizes the three differences panel by panel.

The gate value depends differently on θ across mechanisms (Figure 2a): STG's expected training-time value $E[z]$ departs visibly from its clipped-linear inference rule (by up to ≈ 0.2) because its noise acts directly in gate-value space, whereas the logit-space noise of HC/CD keeps $E[z]$ within ≈ 0.1 of their stretched-sigmoid rule, and for the noise-free IFE the two coincide exactly.

The training-time sampling distributions differ as well (Figure 2b): the noisy mechanisms accumulate point mass at exact $\{0, 1\}$ (through clamping for STG and stretch-and-clip saturation for HC/CD), and their continuous parts have distinct shapes: a truncated-Gaussian bell for STG versus a U-shaped hard concrete density for HC/CD, whereas deterministic IFE has no noise and no point mass. This shape difference comes from the noise family itself and survives any temperature change. For HC/CD, the endpoint mass follows the annealing temperature ($8\% \to 88\%$ of total mass as τ:$1 \to 0.1$, against a fixed $\approx 16\%$ per endpoint for STG).

The sparsity regularizers differ too (Figure 2c): STG, HC, and IFE use expected-L_0 surrogates from the same monotone family, while CD applies a bidirectional KL to a sparsity prior (p_0=0.1). HC and CD share the same Hard Concrete sampler, so they map θ to z identically and draw from the same distribution, differing only in the regularizer. These three are mechanism properties, not claims about which selected features the final policy behaviorally relies on.

3.3 GateKL: what toggling one gate does

At inference, z_u only decides whether the corresponding raw signal reaches the backbone; it does not tell us whether toggling that decision changes the policy's action distribution. We therefore toggle each gate's current decision directly and measure how much the action distribution changes. Fix a trained backbone π_θ and its inference gate values $\mathbf{z}$. Write $\pi_{\theta,\mathbf{z}}$ for the resulting policy, and for $b \in \{0, 1\}$ let $\mathbf{z}^{u\to b}$ agree with $\mathbf{z}$ except that its u-th entry is set to b. GateKL is defined in two branches, one for each current decision of gate u:

$$D_{\mathbf{z}}^{\text{close}}(u) := \mathbb{E}_{s\sim\rho_{\pi_{\theta,\mathbf{z}}}}\left[D_{\mathrm{KL}}\big(\pi_{\theta,\mathbf{z}}(\cdot \mid s) \,\big\|\, \pi_{\theta,\mathbf{z}^{u\to 0}}(\cdot \mid s)\big)\right], \tag{2}$$

$$D_{\mathbf{z}}^{\text{reopen}}(u) := \mathbb{E}_{s\sim\rho_{\pi_{\theta,\mathbf{z}}}}\left[D_{\mathrm{KL}}\big(\pi_{\theta,\mathbf{z}}(\cdot \mid s) \,\big\|\, \pi_{\theta,\mathbf{z}^{u\to 1}}(\cdot \mid s)\big)\right]. \tag{3}$$

- **Closing an open gate** ($z_u > 0$): the feature currently reaches the backbone, and $\mathbf{z}^{u\to 0}$ blocks it. A large $D_{\mathbf{z}}^{\text{close}}(u)$ means the trained policy relies on this feature.
- **Reopening a closed gate** ($z_u = 0$): the feature is currently blocked, and $\mathbf{z}^{u\to 1}$ lets it through. A large $D_{\mathbf{z}}^{\text{reopen}}(u)$ means the backbone responds once the feature is let in; a zero gate value is therefore not, by itself, evidence that the feature is behaviorally unimportant.

In both branches the inner KL compares the action distributions the same network produces at one state, with and without the toggle, and the outer expectation averages this change over the states the policy visits.

The two branches are the same operation—flip the decision of exactly one gate and leave every other gate at its learned value—so we write them as a single quantity. The one-gate toggle mask $\mathcal{T}_u(\mathbf{z})$ picks the branch that follows gate u's current decision:

$$\mathcal{T}_u(\mathbf{z}) := \begin{cases} \mathbf{z}^{u\to 0}, & z_u > 0, \\ \mathbf{z}^{u\to 1}, & z_u = 0, \end{cases} \tag{4}$$

and the resulting GateKL is

$$\mathrm{GateKL}_{\mathbf{z}}(u) := \mathbb{E}_{s\sim\rho_{\pi_{\theta,\mathbf{z}}}}\left[D_{\mathrm{KL}}\big(\pi_{\theta,\mathbf{z}}(\cdot \mid s) \,\big\|\, \pi_{\theta,\mathcal{T}_u(\mathbf{z})}(\cdot \mid s)\big)\right]. \tag{5}$$

GateKL asks (Figure 3, left): at the current gate values, how much does toggling one gate change the action distribution? For an open gate it equals the closing branch above; for a closed gate, the reopening branch. GateKL therefore evaluates the trained policy itself, one gate at a time. It does not prove that an open gate is necessary for the task, that closing a gate was a mistake, or that the final gate values preserve task performance.

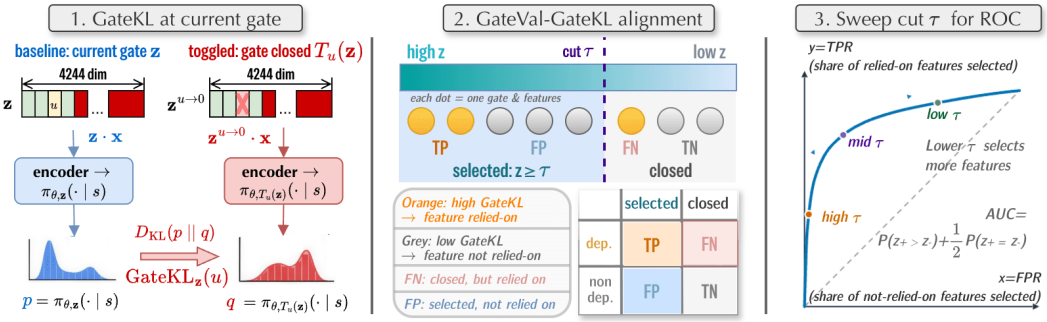


Figure 3: Conceptual illustration of the GateKL procedure, not an empirical ROC or a method ranking. *Left:* GateKL compares the policy under its learned gate $\mathbf{z}$ with the same backbone after one gate is toggled ($\mathcal{T}_u(\mathbf{z})$). *Middle:* a GateKL cutoff splits gates into relied-on and not relied-on (the ground truth); the gate-value rule $z_u \geq \tau$ predicts this split, and crossing the two gives TP/FP/FN/TN. *Right:* sweeping τ yields the ROC of the gate-value–GateKL alignment. The 4,244 dimensions are those of our RL setup; see Section 4.1 for the experimental gate grouping.

For a pre-specified GateKL cutoff κ, say that gate u is **relied on** when $\mathrm{GateKL}_{\mathbf{z}}(u) \geq \kappa$, and write $y_u = \mathbb{I}[\mathrm{GateKL}_{\mathbf{z}}(u) \geq \kappa]$ for this binary label and $\hat{y}_u(\tau) = \mathbb{I}[z_u \geq \tau]$ for the selection prediction. The cutoff κ is fixed before looking at any gate values; our experiments set it to the 80th percentile of the GateKL values within each method (Section 4.1.3). At one τ, crossing the label with the prediction gives the TP, FP, FN, and TN cells in the middle panel of Figure 3; sweeping τ gives the ROC in the right panel, with

$$\mathrm{AUC} = \Pr(z_+ > z_-) + \tfrac{1}{2}\Pr(z_+ = z_-),$$

where z_+ and z_- are gate values drawn uniformly at random from the relied-on gates and the rest, respectively. An AUC near 1 means that relied-on gates almost always have larger gate values than gates that are not relied on; an AUC near 0.5 means the two groups overlap, so the gate values carry no information about reliance. AUCs can be compared across mechanisms, but raw GateKL values cannot: a high AUC says the gate values order reliance well, not that the policy depends more strongly on its features. A false negative is a gate with a low value but high GateKL. If the gate is open, closing it changes the policy; if it is closed, reopening it does.

4 Experiments

We report how completion time varies with the number of selected features, then the multi-seed learning dynamics, and finally how well gate values align with GateKL across all maps.

4.1 Setup

4.1.1 RL environment

We test the theory of Section 3 on a high-dimensional RL policy trained with an on-policy, PPO-style learner. The task is a racing game: the agent must reach the finish line as fast as possible. The gated observation has D=4,244 dimensions, each carrying its own feature-level gate. Throughout, k denotes the number of raw features a trained model retains, counted by the rules of Section 4.1.3 and swept in Section 4.2. Evaluation uses twelve maps (map-1 to map-12), each method trained with three seeds. Samples are collected on 500 machines with 16-core CPUs; training runs on 8 V100 32GB GPUs with batch size 4096 for 20 hours ($\approx$ 200K learner steps), by which point the training loss has converged. The PPO learner uses a clip range of 0.2, generalized advantage estimation with λ_{GAE}=0.94 and discount γ=0.92, the Adam optimizer with a fixed learning rate of 2×10^{-4}, an entropy coefficient of 0.005, and gradient-norm clipping at 0.5. Gate training keeps the coefficient λ in Equation 1 at zero for a 150K-step warm-up and then ramps it to its peak $\lambda_{\max}$=0.01 over a further 80K steps; all comparisons use converged models at this matched $\lambda_{\max}$.

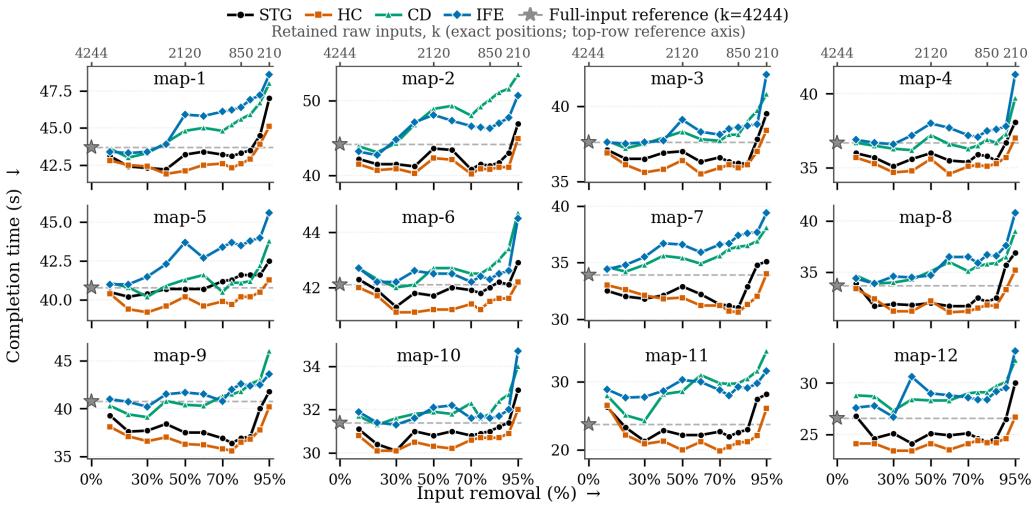


Figure 4: Completion-time sweep over the number of selected features across twelve maps. Each colored marker is a measured method–map completion time at k selected raw features (lower is better). The bottom axis gives feature removal relative to D=4,244, while the top reference labels give the selected feature count k. The gray star and dashed line mark the full-feature reference at k=4,244. Each panel uses an independent y-axis.

4.1.2 Comparison methods

For each mechanism we adopt the best-performing hyperparameter configuration reported in its original paper. HC follows Louizos et al. (2018) with stretch bounds $(\gamma, \zeta)=(-0.1, 1.1)$ and temperature τ annealed $1\to0.1$, sparsity pressure being delayed until after an initial fully open period; at inference, noise is removed and the stretch-and-clip gives $z=\mathrm{clip}(\sigma(\log\alpha)\cdot(\zeta-\gamma)+\gamma,\ 0,\ 1)$. STG follows Yamada et al. (2020) with noise standard deviation $\sigma_{\text{noise}}=0.5$ and no annealing, giving $z=\mathrm{clip}(\mu+0.5, 0, 1)$ at inference. CD (Gal et al., 2017) reuses this sampler unchanged—the same stretch-and-clip with the same (γ, ζ)—and differs only in its regularizer (KL-to-prior $\mathrm{KL}(\sigma(\theta)\,\|\,p_0)$ with $p_0=0.1$). IFE (Pham & Cangelosi, 2025) keeps the same stretch-and-clip and the same L_0 surrogate, but trains deterministically (no Gumbel noise). All four use the objective in Equation 1 and are trained under the same $\lambda_{\max}$. Table 2 summarizes the sampling, regularizer, inference, and default-parameter settings of every relaxation.

4.1.3 Evaluation protocol

Metrics. The primary metric, completion time in seconds (lower is better), is measured over 2000 episodes per model. GateKL is measured on a separate sample of 100 episodes per model, with 60 frames drawn at random within each episode, giving 6000 evaluation frames; both use the deterministic inference gate and no exploration noise. The gate-distribution statistics (selected-feature counts, endpoint masses, soft-value histograms) need no episodes, since $z_u = \bar{g}_m(\theta_u)$ is a deterministic function of the trained parameter θ_u alone. The relied-on-gate cutoff κ of Section 3.3 is fixed in advance: within each method, the top 20% of gates by aggregated GateKL form the relied-on class.

Counting selected features. All four mechanisms produce deterministic gate values in $[0, 1]$ at inference (Table 2), but their endpoint masses, soft-value distributions, and hence counting rules are mechanism-specific. We therefore follow each mechanism's own L_0 inference rule: under STG a feature is selected when $z>0$; under HC when its gate's closed-form L_0 expectation exceeds 0.5, a rule IFE and CD inherit with the same inference form. These counts say how many features a mechanism selects, not that the underlying gate-value distributions match.

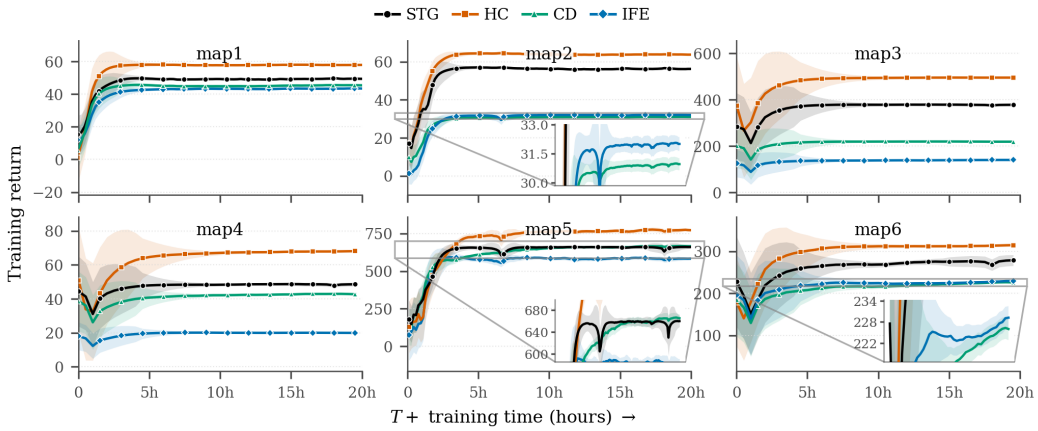


Figure 5: Multi-seed training-return trajectories on six displayed maps. Solid curves show the seed-mean return (higher is better); shading, the across-seed variation. All runs plateau within the shown 0–20 hours, so the window highlights early learning differences. Insets magnify late-stage differences where shown. Panels have independent y-axes.

4.2 Completion time versus the number of selected features

To probe how sparsity affects completion time, we sweep the number of selected features k for STG, HC, IFE, and CD on the same twelve maps (Section 4.1). The feature sets come from the gate evolution itself: for each sweep point we locate the training time at which a gated run retains approximately k raw features, then train a fresh policy for 24 hours on the features open at that time. All twelve maps share the same k at every sweep point, rather than a per-map optimum; Figure 4 gives the per-map curves, whose k=4,244 value is the full-feature reference from the same sweep. Across the twelve-map means, HC has the lowest completion time at every measured k. At the common analysis point k=1270, HC retains 29.9% of the raw features (removing 70.1%) and reduces the mean completion time from 37.4 to 34.0 seconds (-3.4 seconds, -9.1%) relative to the full-feature references on all twelve maps. The nearby settings k=850, 1060, and 1270 yield 34.3, 34.1, and 34.0 seconds, a broad near-optimal region rather than a sharp optimum. The gain is not monotonic: extreme compression to k=210 raises HC's mean to 35.7 seconds and improves only 10 of 12 maps (one ties, one degrades).

The sweep therefore shows a trade-off between sparsity and control quality: more compression is not always better. At the same k, the four mechanisms reach different completion times, so the selected-feature count alone does not determine control performance, complementing the alignment analysis of Section 4.4. The sweep itself, however, measures neither policy reliance nor the reason a given k helps: each k retrains a fresh policy from features selected at one moment of the gate evolution, not one run in which gates are removed one by one.

4.3 Learning dynamics: multi-seed training-return trajectories

To describe how each policy reaches its final performance, Figure 5 reports multi-seed training-return trajectories (higher is better) for STG, HC, CD, and IFE on six maps. We show only the first 20 hours, by which every run has plateaued. Panels use independent y-axes, so cross-map comparisons should rely on reported values rather than panel height.

Across the six maps, HC earns high returns earlier and has the highest final return in every panel. STG is generally next; on one map, CD approaches STG late in training. Training return is therefore an early indicator of learning quality, complementary to the converged completion times in Table 3 (Appendix F). It neither replaces task-level evaluation nor explains why the mechanisms diverge; Section 4.4 answers a different question about the features.

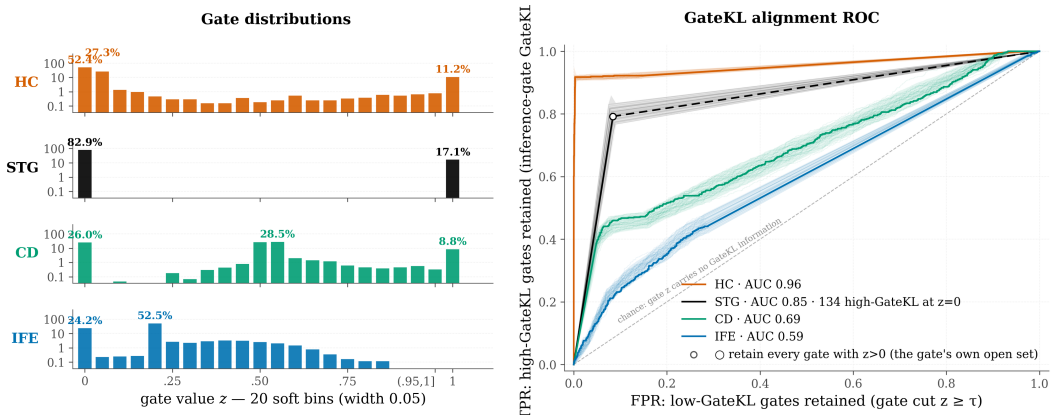


Figure 6: All-map gate distributions and gate-value–GateKL alignment. *Left:* deterministic inference-time gate values aggregated from the checkpoints of every map, method, and training seed across map-1 to map-12; exact 0 and 1 masses are separated from the twenty soft-value bins. *Right:* the ROC over the 4,244 feature gates across all maps, produced by sweeping $z_u \geq \tau$ against the relied-on gates (the top 20% of the aggregated GateKL values within each method).

4.4 Gate distributions and gate-value–GateKL alignment

Figure 6 is the empirical realization of the procedure sketched in Figure 3, aggregated across all maps. We measure GateKL on every trained checkpoint—one per map, method, and seed—against that checkpoint's own action distribution under its current inference gate: an open gate is set to 0, a closed gate to 1, all others keep their inference-time value, and we compute the action-distribution KL. For each of the 4,244 gates we average this over the checkpoint's 6000 evaluation frames, then average GateKL and gate value over all checkpoints of the same method.

The left panel of Figure 6 reports each mechanism's deterministic inference-time gate distribution. Endpoint masses and soft gate values reveal sharply different closure patterns: STG is almost binary (82.9% exact zeros, 17.1% exact ones), CD and IFE concentrate most gate values at intermediate levels, and HC combines sizable endpoint masses with a broad soft middle. The right panel applies the threshold rule $z_u \geq \tau$ to these aggregated gate values and compares it with the relied-on gates of Section 3.3, using the cutoff κ specified in Section 4.1.3.

The resulting AUCs are 0.96 for HC, 0.85 for STG, 0.69 for CD, and 0.59 for IFE. Among the evaluated mechanisms, then, HC's gate values align best with the relied-on gates, STG remains substantially but less strongly aligned, and CD and IFE sit closer to the chance diagonal. The analysis answers exactly one question: does a gate's value predict whether toggling it substantially changes the trained policy's actions? A high AUC means only that a mechanism's gate values rank the relied-on gates above the rest. It says nothing about how heavily that policy relies on features overall, whether an open feature is necessary for the task, whether a closed one was closed in error, or whether that mechanism completes the task faster.

5 Conclusion

We studied end-to-end differentiable feature selection in a deep RL policy with a 4,244-dimensional observation. Because the gate reshapes the policy and the policy reshapes the data driving gate updates, feature usefulness is conditioned on the current policy: a gate decision can only be judged relative to the policy that produced it. Mechanism choice then matters beyond the selected support: at k=1270 (70.1% removed) HC cuts mean completion time from 37.4 to 34.0 seconds across twelve maps, CD and IFE degrade at comparable sparsity, and GateKL separates them where counts cannot (0.96 AUC for HC, 0.85 STG, 0.69 CD, 0.59 IFE). Deploying gates in RL thus requires evaluating the selection, its control performance, and its alignment with the trained backbone (Appendix E).

AI USE STATEMENT

In this work, we used generative AI tools (LLM-based coding assistants) for writing and editing assistance: drafting and polishing prose, editing and cross-checking the LaTeX sources (tables, figures, and the appendix structure), unifying terminology across sections, and assisting with small data-processing and plotting scripts that render tables and figures from recorded experiment snapshots. We have not used generative AI tools for research ideation, experimental design, running the game experiments, or collecting data, and the reported numbers were not produced by AI: all completion times, GateKL divergences, alignment AUCs, and gate counts come from the evaluation platform's recorded logs and archived snapshots. We have reviewed all AI-assisted work: AI-suggested text was checked against the recorded experimental data, AI-generated scripts were executed and their outputs diff-checked against the source snapshots, and every quantitative claim in the paper was traced back to its source file or log. Citation suggestions made by AI tools were verified against the original papers by the authors. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Muhammed Fatih Balın, Abubakar Abid, and James Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *PMLR*, 2019.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Bahram Behzadian and Marek Petrik. Fast feature selection for linear value function approximation. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2019.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer for neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Wei Fan, Kunpeng Liu, Hao Liu, Pengyang Wang, Yong Ge, and Yanjie Fu. Interactive reinforcement learning for feature selection with decision tree in the loop. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2021.

Wei Fan, Kunpeng Liu, Hao Liu, Ahmad Hariri, Pengyang Wang, and Yanjie Fu. Feature and instance joint selection: A reinforcement learning perspective. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In *International Conference on Machine Learning (ICML)*, 2018.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Alborz Geramifard, Finale Doshi, Joshua Redding, Nicholas Roy, and Jonathan How. Online discovery of feature dependencies / bayesian model selection for value function approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2012.

Bram Grooten, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Automatic noise filtering with dynamic sparse training in deep reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023. arXiv:2302.06548.

Hirotaka Hachiya and Masashi Sugiyama. Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In *ECML-PKDD*, 2010.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

Botao Hao, Tor Lattimore, Csaba Szepesvári, and Mengdi Wang. Sparse feature selection makes batch reinforcement learning more sample efficient. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *PMLR*, 2021.

Seyed Mehdin Hazrati Fard, Ali Hamzeh, and Sattar Hashemi. Using reinforcement learning to find an optimal set of features. *Computers and Mathematics with Applications*, 66(10):1892–1904, 2013.

Rahma Hellali et al. A multi-objective multi-agent interactive deep reinforcement learning approach for feature selection. *HAL preprint hal-04723314*, 2024.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

J. Zico Kolter and Andrew Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.

De-Rong Liu, Hong-Liang Li, and Ding Wang. Feature selection and feature learning for high-dimensional batch reinforcement learning: A survey. *International Journal of Automation and Computing*, 12(3):229–242, 2015.

Kunpeng Liu, Yanjie Fu, Pengfei Wang, Le Wu, Rui Bo, and Xiaolin Li. Automated feature selection: A reinforcement learning perspective. In *IEEE International Conference on Data Mining (ICDM)*, 2019.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018. arXiv:1712.01312.

Karim Lounici, Massimiliano Pontil, Sara van de Geer, and Alexandre B. Tsybakov. Oracle inequalities and optimal inference under group sparsity. *The Annals of Statistics*, 39(4):2164–2204, 2011.

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 (ICLR)*, 2017.

Christopher Painter-Wakefield and Ronald Parr. Regularized feature selection in reinforcement learning. *Machine Learning*, 100(2-3):655–676, 2015.

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Tien Pham and Angelo Cangelosi. Pay attention to what and where? interpretable feature extractor in vision-based deep reinforcement learning. *arXiv preprint arXiv:2504.10071*, 2025.

Sali Rasoul, Sodiq Adewole, and Alphonse Akakpo. Feature selection using reinforcement learning. *arXiv preprint arXiv:2101.09460*, 2021.

Kezhou Ren, Yifan Zeng, Zhiqin Cao, and Yingchao Zhang. ID-RDRL: A deep reinforcement learning-based feature selection intrusion detection model. *Scientific Reports*, 12:15370, 2022.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations (ICLR)*, 2016.

Sumana Sanyasipura Nagaraju. Automation and feature selection enhancement with reinforcement learning. *arXiv preprint arXiv:2503.11991*, 2025.

Kyohei Suzuki and Konstantinos Slavakis. Nonconvex regularization for feature selection in reinforcement learning. *arXiv preprint arXiv:2509.15652*, 2025.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *PMLR*, 2020.

Wenbo Zhang and Hengrui Cai. Where to intervene: Action selection in deep reinforcement learning. *Transactions on Machine Learning Research*, 2025. arXiv:2507.04187.

A CONDITIONAL LOCAL-LINEAR AND RESTRICTED-EIGENVALUE BACKGROUND

Scope. This appendix collects optional local-linear and sparse-recovery background. It supports interpretation of the intervention measures in Section 3.3, but is not needed to define them. The assumptions below are conditional approximations: they neither assert that all four gate mechanisms optimize one common sparse-regression objective nor prove that a closed gate is task-irrelevant.

Fixed-data selection baseline. On fixed data, let $S^\star \subseteq \{1, \dots, U\}$ denote a sparse set of used features in a local linear parameterization.

Assumption 1 (Feature sparsity)**.** *There is a set $S^\star \subseteq \{1, \dots, U\}$ of **used** features, $|S^\star| = s_0$, with $w_u^\star = 0$ for $u \notin S^\star$.*

Lemma 1 (Gating as reparameterized sparse regression)**.** *Under this fixed-data parameterization, scaling the feature controlled by gate u by $z_u \in [0, 1]$ reparameterizes its downstream coefficient as $z_u w_u$. A regularizer $R(z_u)$ that pushes $z_u \to 0$ therefore induces a sparse selection view in which gates with $z_u > 0$ retain their features and gates with $z_u \to 0$ remove them.*

This is a fixed-data baseline only; it is not a theorem about the behavior or relative quality of the learned RL policies.

Local intervention approximation. For a fixed trained policy and its inference gate values $\mathbf{z}$, write the action logits $f(s) \in \mathbb{R}^{|\mathcal{A}|}$ locally as a function of the encoded gate features, assuming the encoded input decomposes additively per gate: $\phi_u(s) \in \mathbb{R}^{d_u}$ is the encoded contribution of gate u at state s, and $J_{u,\mathbf{z}}(s) = \nabla_{\phi_u} f(s) \in \mathbb{R}^{|\mathcal{A}| \times d_u}$ is the logit Jacobian with respect to that contribution. Closing gate u perturbs its contribution by $-z_u \phi_u(s)$, so the first-order logit perturbation is $-J_{u,\mathbf{z}}(s)\, z_u \phi_u(s)$. For a softmax policy, the standard second-order KL expansion between the reference logits and the perturbed logits then gives

$$D_{\mathrm{KL}} \approx \tfrac{1}{2} z_u^2 \phi_u(s)^\top J_{u,\mathbf{z}}(s)^\top F(s) J_{u,\mathbf{z}}(s) \phi_u(s),$$

where $F(s) = \mathrm{diag}\big(\pi(\cdot \mid s)\big) - \pi(\cdot \mid s)\, \pi(\cdot \mid s)^\top$ is the action Fisher information at state s. Defining the Fisher-weighted perturbation $\tilde{\phi}_{u,\mathbf{z}}(s) = F(s)^{1/2} J_{u,\mathbf{z}}(s)\, z_u \phi_u(s)$ and stacking its rows over n sampled states (one per visited state) into $\tilde{\Phi}_{u,\mathbf{z}}$ yields the following conditional interpretation of the closing GateKL branch:

Proposition 1 (The closing GateKL branch measures current reliance)**.** *For a fixed policy and mask under the local expansion above, $D_{\mathbf{z}}^{\mathrm{close}}(u) \asymp \frac{1}{n} \|\tilde{\Phi}_{u,\mathbf{z}}\|_F^2$. Thus it measures action-distribution sensitivity to a feature that the inference gate still lets through.*

For an open gate ($z_u > 0$), GateKL uses this same closing intervention, so $\mathrm{GateKL}_{\mathbf{z}}(u) = D_{\mathbf{z}}^{\mathrm{close}}(u)$. For a closed gate ($z_u = 0$), GateKL instead uses the reopening branch $D_{\mathbf{z}}^{\mathrm{reopen}}(u)$; the close-only local expansion above does not apply to that branch.

Observation model for the recovery background. For the recovery rate below, consider the local model $V^\pi(s) \approx \sum_u \langle w_u^\star, \phi_u(s) \rangle$ with $w^\star$ as in Assumption 1, observed through $Y_i = V^\pi(s_i) + \varepsilon_i$ with sub-Gaussian noise of level σ_{noise} and states $s_i \sim \rho_\pi$ visited by the current policy. Let $\hat{w}$ minimize the group-lasso objective

$$\sum_{i=1}^{n} \Big(Y_i - \sum_u \langle w_u, \phi_u(s_i) \rangle \Big)^2 + \lambda \sum_u \sqrt{d_u}\, \|w_u\|_2, \qquad \hat{V}(s) = \sum_u \langle \hat{w}_u, \phi_u(s) \rangle.$$

Restricted-eigenvalue condition. The following condition states the policy-dependent restricted-eigenvalue (RE) background for this design. Here $\tilde{\Phi}$ denotes the $n \times \big(\sum_u d_u\big)$ block design whose u-th block stacks the feature rows $\phi_u(s_i)^\top$, $s_i \sim \rho_\pi$—distinct from the per-gate perturbation matrix $\tilde{\Phi}_{u,\mathbf{z}}$ above; it is policy-induced through the state distribution. When the encoding is the identity and the state distribution is a fixed data distribution, this design reduces to the classical data-only design Φ, and $\tilde{\kappa}(\pi)$ recovers the classical constant κ. It does not supply a universal guarantee for the mechanisms studied in this paper.

Assumption 2 (Policy-induced restricted eigenvalue). *There is* $\tilde{\kappa}(\pi) > 0$ *such that* $\frac{1}{n}\|\tilde{\Phi}\Delta\|_2^2 \geq \tilde{\kappa}\,\|\Delta\|_2^2$ *for all* Δ *in the sparse recovery cone* $\mathcal{C}(S^\star) = \{\Delta : \sum_{u\notin S^\star}\sqrt{d_u}\|\Delta_u\|_2 \leq 3\sum_{u\in S^\star}\sqrt{d_u}\|\Delta_u\|_2\}$.

Under Assumptions 1–2, standard sparse-regression analysis gives matching rates.

Theorem 1 (Matching upper and lower bounds). *With* $\lambda \asymp \sigma_{\text{noise}}\sqrt{(d_{\max}+\log U)/n}$*, with probability at least* $1-U^{-c}$*,*

$$\|\hat{V}-V^\pi\|_{2,\rho}^2 \;\leq\; C\,\frac{\sigma_{\text{noise}}^2\, s_0\,(d_{\max}+\log U)}{\tilde{\kappa}(\pi)^2\, n}, \tag{6}$$

where V^π *is the current policy's value function,* $\hat{V}$ *the group-lasso estimate above, and* $\|\cdot\|_{2,\rho}$ *the* L_2 *norm over the state distribution* ρ*;* $d_{\max}$ *the largest group dimension, and* C *a universal constant. This rate is minimax-optimal up to* $\tilde{\kappa}(\pi)^{-2}$*: for the feature-sparse class of Assumption 1,* $\inf_{\hat{V}}\sup_{w^\star}\mathbb{E}\|\hat{V}-V^\pi\|_{2,\rho}^2 \geq c'\,\sigma_{\text{noise}}^2\, s_0(d_{\max}+\log U)/n$*. Proof sketches follow Lounici et al. (2011); Hao et al. (2021).*

Theorem 1 extends the classical Hao et al. rate by replacing the fixed design and RE constant with the policy-induced ones: the bound is **correct** at any fixed π, but because π changes during training, so does the guarantee.

B On-Policy Feedback Dynamics

This appendix states the formal coupled dynamics of the on-policy gate–policy feedback loop described informally in Section 3 which discusses the policy-conditioned feature reliance view in detail.

B.1 Coupled gate–policy dynamics

Let the system be defined by three relations that hold at each training step t:

$$\theta_t = \theta(z_{<t},\, \Xi_t) \tag{7}$$

$$\tau_t \sim \pi_{\theta_t} \tag{8}$$

$$\nabla_\alpha \mathcal{L}_t = \mathbb{E}_{\tau\sim\pi_{\theta_t}}\big[\,\nabla_\alpha \ell(\tau;\alpha,\theta_t)\big] \tag{9}$$

Here α is the vector of gate parameters—written α to avoid overloading the policy parameters θ—Ξ_t is the exogenous randomness of the training run (initialization and sampling order), and $\ell(\tau;\alpha,\theta_t)$ is the loss incurred on trajectory τ. The policy θ_t depends on the gate history $z_{<t}$ (7); the trajectory τ_t is sampled on-policy from π_{θ_t} (8); and the gate gradient $\nabla_\alpha\mathcal{L}_t$ is estimated on that very trajectory (9). The key structural property is that the data distribution behind the gradient is **not fixed**—it is itself a function of $z_{<t}$ via the policy. This coupling has no counterpart in supervised learning, where the design matrix is independent of the gate.

B.2 Hard behavioral closure (STG)

When gate u closes (z_u=0) during hard-closing training (STG's constant-variance Gaussian noise), the backbone loses all signal from u. It must then satisfy the task using only surviving gates; the new π_{θ_t} visits u-informative states less often; $\tilde{\Phi}_u$ (Proposition 1) sees less excitation; $\nabla_{\alpha_u}\mathcal{L}_t$ shrinks; the gate has no pressure to re-open u. The cycle restarts with a stronger commitment—more commitment $\rightarrow$ less signal $\rightarrow$ lower gradient $\rightarrow$ deeper lock-in—a positive feedback loop absent from the static-data setting of Lemma 1. Once it is closed, the closing branch of GateKL, $D_{\mathbf{z}}^{\text{close}}(u)$, is zero by construction, so its response must instead be assessed through the reopening branch $D_{\mathbf{z}}^{\text{reopen}}(u)$.

B.3 Soft behavioral closure (HC)

HC can also reach exact z=0 through its stretch–and–clip inference, but its annealed Gumbel noise schedule leads the policy through a more gradual commitment path. Even when z_u drops to zero

late in training, the policy has had more opportunity to partially depend on surviving features, so π_{θ_t} does not fully abandon the u-informative states, $\tilde{\Phi}_u$ does not collapse, and $\nabla_{\alpha_u}\mathcal{L}_t$ does not vanish if the gate remains behaviorally relevant. This mechanism-level account does not by itself imply a particular response profile for already closed gates: that question requires the current-gate GateKL intervention, which opens a closed gate or closes an open gate one at a time, in Section 4.4.

C LIMITATIONS

Our theory (Section 3) linearizes the policy's penultimate-layer representation, which is standard in this line of work but does not capture effects specific to the Transformer policy's nonlinear feature interactions; the rate of Theorem 1 (Appendix A) should be read as governing the exposed dependence on the number of used features rather than as a literal finite-sample guarantee for the trained nonlinear policy. The relaxation-dependent sensitivity mechanism of Proposition 1 is a local-approximation statement; a dedicated gated-mode intervention study across relaxations, as well as comparisons against additional gate mechanisms, is left for future work. The visual-modality test is currently reported at the group level: both STG and HC close the visual gates, and Appendix H confirms that the corresponding no-visual architecture is behaviorally safe, but a fine-grained trajectory analysis of the visual block is left to future work.

D DISCUSSION

The central message of this paper is that choosing a differentiable gate mechanism for RL is not just a matter of optimizing sparsity—it is a decision about what kind of policy reliance pattern will emerge. The coupling formalized in Appendix B ((7)–(9)) shows why: what the backbone learns to depend on determines the trajectory distribution, which becomes the data for estimating gate gradients, closing a loop. This loop amplifies the commitment pattern produced by the gate's closure semantics.

All four mechanisms in our study share the same stretch–and–clip inference (Table 2) and can produce exact $z{=}0$, yet their effects on task performance diverge. HC achieves the best completion time among the four ($+0.2\%$ vs. baseline), with STG a close second ($+1.6\%$). CD retains a comparable number of features to STG and HC (80.8% reduction) but degrades by $+12.7\%$. IFE is the most aggressive (89.1% reduction) and the most degraded ($+14.2\%$). These differences in the number of selected features and in completion time motivate, but do not by themselves establish, a difference in selection quality.

The gate-value–GateKL alignment analysis of Section 4.4 supplies the corresponding evidence. Across all maps, gate-value AUC is 0.96 for HC, 0.85 for STG, 0.69 for CD, and 0.59 for IFE. Thus the gate values differ in how well they rank the relied-on gates above the remainder when each gate is toggled around its current value, even when the number of selected features alone is similar. This is a ranking result only: it does not prove that a feature is necessary for the task, that closing it was a mistake, or that changing any gate helps.

Why this matters practically. Standard supervised gate benchmarks evaluate mechanisms by whether they select a correct subset on a fixed test set. Our results indicate that this evaluation is incomplete for RL: gate mechanisms with the same inference-time binarization can produce policies with radically different GateKL response sensitivity. For practitioners deploying gates in on-policy RL systems, the choice among differentiable relaxations carries consequences that a simple count of selected dimensions cannot capture.

The visual-modality test of Appendix H reinforces this message. Both STG and HC independently close a visual feature block that a human designer would consider essential, and architecture-level ablation confirms the features are genuinely dispensable. This demonstrates that the gate's selection is not arbitrary—it discovers redundancy that human design would overlook—but the discussion above shows that **how** it affects policy reliance matters as much as **what** it closes.

E EXTENDED CONCLUSION

We studied gate-based differentiable feature selection as a lightweighting mechanism inside a deep RL policy for a high-dimensional control task with a $4{,}244$-dimensional observation. Three findings emerged.

First, the policy-conditioned feature selection theory of Section 3: feature usefulness is conditioned on the current policy and the trajectory distribution it generates, rather than on a fixed dataset. This explains why a gate decision can only be judged relative to the policy that produced it.

Second, gate mechanism choice matters for RL beyond the selected support. All four mechanisms in our study can produce near-binary gates via stretch–and–clip inference, yet their performance diverges substantially: HC and STG stay within 2% of the no-gating baseline (both removing $\sim$80% of observation dimensions), while CD and IFE degrade by $\sim$10%. The divergence is not explained by sparsity alone—CD is as compact as STG and HC but underperforms—and points to the training dynamics and regularizer structure as the discriminating factors.

Third, the inference-gate GateKL test reveals an additional difference among mechanisms that appear similar under a count-based comparison. It asks whether learned gate values rank larger action-distribution changes when one current gate decision is toggled: open gates are closed and closed gates are opened, with all other gates fixed. The ROC aggregated across all maps gives AUCs of 0.96 for HC, 0.85 for STG, 0.69 for CD, and 0.59 for IFE. This is gate-value–GateKL alignment, not direct evidence that the final gate values preserve task performance, that an open gate is necessary for the task, or that closing a gate was a mistake. The visual-modality test separately shows that the visual block closed by both STG and HC can be removed without worsening task performance in the tested environment.

Taken together, these results argue that deploying differentiable gates in RL requires evaluating what the gate selects, the resulting control performance, and how its gate values align with the response of the jointly trained backbone.

F COMPLETION-TIME SWEEP: CONVERGED CONFIGURATIONS AND PROTOCOL

Table 3 records, for each method, the converged per-map number of selected features (counted under the mechanism-specific rules of Section 4.1.3), the reduction percentage, completion time, and its change versus the no-gating baseline.

Figure 4 in the main text gives the map-level view of the sensitivity sweep over the number of selected raw features within the $D{=}4{,}244$-dimensional observation. Each colored point is a measured completion time for one method, map, and a shared number of selected features k. At a given sweep point, every map uses the same k; the plot therefore tests whether a single shared k transfers across maps instead of reporting a separate per-map oracle minimum.

The 3×4 panels show that the mean result is not driven by choosing a separate optimum for each map. Across all measured k, HC has the lowest twelve-map mean completion time of the four mechanisms. Its means are 34.3, 34.1, and 34.0 seconds at $k{=}850$, 1060, and 1270, respectively, versus 37.4 seconds for the full-feature reference. At the common analysis point $k{=}1270$, HC retains 29.9% of the features, reduces mean completion time by 3.4 seconds (9.1%), and is faster than the corresponding full-feature reference on all twelve maps.

The map-level curves also delimit the useful compression range. At the extreme sparsity $k{=}210$, HC rises to 35.7 seconds: 10 maps improve, one ties, and one degrades relative to its full-feature reference. The sweep therefore shows a sparsity-dependent, non-monotonic practical trade-off rather than a uniform benefit from removing more features. At the same k, the mechanisms can still produce different completion times, so the number of selected features alone does not explain control performance.

The sweep does not establish why a given k helps or hurts. In particular, it does not measure policy reliance or show that a gate first removes redundant features and then excludes useful ones: each k is produced by the protocol of Section 4.2—a feature-set snapshot taken from the gate evolution at

Table 3: Completion time and the number of selected features across twelve evaluation maps. Each method reports the number of selected features (counted by each mechanism's own L_0 inference rule, Section 4.1.3), reduction percentage, completion time, and relative change versus baseline; the rightmost column reports the cross-map average. Selected-dims and completion-time entries are reported as mean ± std over 3 random seeds; reduction and relative change are cross-seed means (positive: slower than baseline). Methods: IFE (Pham & Cangelosi, 2025), CD (Gal et al., 2017), STG (Yamada et al., 2020), HC (Louizos et al., 2018).

Method	Metric	Evaluation map												
		map-1	map-2	map-3	map-4	map-5	map-6	map-7	map-8	map-9	map-10	map-11	map-12	Avg
Baseline	Selected dims	4244 ±0												4244 ±0
	Reduction	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	Time (s)	42.2 ±0.41	41.3 ±0.35	35.9 ±0.40	35.3 ±0.34	38.8 ±0.29	41.0 ±0.35	32.3 ±0.28	31.1 ±0.22	36.9 ±0.50	30.7 ±0.22	21.8 ±0.28	24.5 ±0.34	34.3 ±0.30
	Δ (vs base)	–												–
IFE	Selected dims	499 ±16	474 ±4	432 ±14	440 ±12	464 ±15	485 ±13	476 ±11	478 ±8	460 ±10	460 ±8	442 ±12	454 ±11	464 ±9
	Reduction	88.2%	88.8%	89.8%	89.6%	89.1%	88.6%	88.8%	88.7%	89.2%	89.2%	89.6%	89.3%	89.1%
	Time (s)	47.2 ±0.37	47.6 ±0.31	38.8 ±0.35	37.7 ±0.29	43.9 ±0.24	42.6 ±0.52	37.7 ±0.23	37.3 ±0.50	42.5 ±0.46	31.7 ±0.49	29.8 ±0.30	29.3 ±0.24	38.8 ±0.31
	Δ (vs base)	+11.9%	+15.3%	+8.1%	+6.8%	+13.1%	+3.9%	+16.7%	+19.9%	+15.2%	+3.3%	+36.7%	+19.6%	+14.2%
CD	Selected dims	887 ±31	830 ±25	776 ±20	759 ±23	836 ±21	844 ±17	841 ±13	822 ±17	796 ±12	828 ±9	760 ±19	773 ±16	813 ±13
	Reduction	79.1%	80.4%	81.7%	82.1%	80.3%	80.1%	80.2%	80.6%	81.2%	80.5%	82.1%	81.8%	80.8%
	Time (s)	45.6 ±0.32	50.3 ±0.38	38.4 ±0.30	36.8 ±0.24	41.1 ±0.52	42.7 ±0.25	36.4 ±0.51	35.8 ±0.46	42.0 ±0.52	31.7 ±0.45	30.0 ±0.26	29.2 ±0.21	38.3 ±0.49
	Δ (vs base)	+8.1%	+21.8%	+7.0%	+4.3%	+5.9%	+4.1%	+12.7%	+15.1%	+13.8%	+3.3%	+37.6%	+19.2%	+12.7%
STG	Selected dims	802 ±9	755 ±26	686 ±20	684 ±17	774 ±23	782 ±19	785 ±15	769 ±19	769 ±15	779 ±12	738 ±24	751 ±21	756 ±18
	Reduction	81.1%	82.2%	83.8%	83.9%	81.8%	81.6%	81.5%	81.9%	81.9%	81.7%	82.6%	82.3%	82.2%
	Time (s)	43.3 ±0.36	41.4 ±0.30	36.2 ±0.35	35.6 ±0.29	41.5 ±0.24	42.0 ±0.30	31.4 ±0.22	32.2 ±0.50	36.9 ±0.45	31.1 ±0.50	22.7 ±0.34	24.4 ±0.29	34.9 ±0.25
	Δ (vs base)	+2.6%	+0.2%	+0.8%	+0.9%	+7.0%	+2.4%	−2.8%	+3.5%	−0.1%	+1.3%	+4.0%	−0.4%	+1.6%
HC	Selected dims	735 ±17	676 ±19	612 ±14	634 ±12	675 ±15	681 ±12	684 ±9	667 ±6	666 ±9	654 ±6	637 ±17	650 ±21	664 ±18
	Reduction	82.7%	84.1%	85.6%	85.1%	84.1%	84.0%	83.9%	84.3%	84.3%	84.6%	85.0%	84.7%	84.4%
	Time (s)	42.9 ±0.28	41.3 ±0.22	36.3 ±0.27	35.3 ±0.21	40.2 ±0.49	41.6 ±0.44	31.2 ±0.48	31.8 ±0.42	36.8 ±0.37	30.7 ±0.42	21.2 ±0.38	24.3 ±0.33	34.5 ±0.29
	Δ (vs base)	+1.7%	+0.0%	+1.1%	+0.0%	+3.6%	+1.5%	−3.4%	+2.3%	−0.3%	+0.0%	−2.8%	−0.8%	+0.2%

the moment approximately k features remain, followed by a 24-hour from-scratch retraining with only those features—so the sweep is not a temporal pruning trajectory of one gate: adjacent k points restart from random initialization. The sweep reports measured means without seed-level uncertainty estimates, so we do not claim statistical significance for the differences above.

G Extended related work

This appendix carries the full versions of the related-work discussions that the main text summarizes: feature selection **for** RL (Section 2) and lightweighting of deep RL agents, together with the wrapper-style line that uses RL **for** feature selection and the occlusion-based feature-importance line that our pruning signal builds on.

G.1 Feature selection for reinforcement learning

A first line treats feature selection as part of value-function approximation, aiming to reduce variance and sample complexity. Classical analyses study which features make linear value-function approximation well-posed (Parr et al., 2008) and add sparsity-inducing penalties to temporal-difference learning: L_1-regularized LSTD (Kolter & Ng, 2009), Bayesian basis selection (Geramifard et al., 2012), broader regularized feature selection in RL (Painter-Wakefield & Parr, 2015), and, most recently, non-convex regularizers within the same LSTD-style policy-evaluation framework (Suzuki & Slavakis, 2025). Others select or compress bases through low-rank/transition structure (Behzadian & Petrik, 2019) or mutual information between state and reward (Hachiya & Sugiyama, 2010). More

Table 4: Removing the visual block at a fixed backbone. Both rows share the same backbone and differ only in whether the 1024-dimensional visual input is present.

ID	Visual	Time(s)
M5	yes	39.83
M7′	no	39.60

recently, Hao et al. (2021) prove that **sparse** feature selection improves the sample efficiency of batch RL under a group-sparse linear model, giving minimax rates that we build on in our analysis; Liu et al. (2015) survey this high-dimensional batch-RL setting, and high-dimensional **action** selection has been studied with false-discovery-rate control (Zhang & Cai, 2025). These works largely assume **linear** approximation and a **fixed** feature basis; our setting is a deep, non-linear policy whose feature dependence **co-evolves** with training.

G.2 Reinforcement learning for feature selection

These methods cast "select/skip a feature" as a sequential decision (Hazrati Fard et al., 2013) and optimize a downstream metric with DQN-style agents (Rasoul et al., 2021) or two-stage pipelines that pre-screen with recursive feature elimination before a DQN refines the subset (Ren et al., 2022). Multi-agent formulations assign one agent per feature (Liu et al., 2019), improve the state representation with a decision tree in the loop (Fan et al., 2021), jointly select features and instances (Fan et al., 2022), target multiple objectives (Hellali et al., 2024), or integrate the search into automated pipelines (Sanyasipura Nagaraju, 2025). These methods select features **for a separate supervised model**; in contrast, we select the observation features **of the RL agent itself**, where each subset evaluation would otherwise require re-training a costly policy—motivating our use of an end-to-end differentiable gate and proxy-reward baselines rather than per-subset retraining.

G.3 Lightweighting deep RL agents

Deep RL policies often require compact, low-latency models, yet dominant lightweighting methods compress the **network** rather than select the agent's **features**. Policy distillation compresses a large teacher into a small student (Rusu et al., 2016; Furlanello et al., 2018); pruning-quantization pipelines (Han et al., 2016) and architecture transforms (Chen et al., 2016) shrink weights and inference cost; and compact representations lower the effective state dimensionality via successor features (Barreto et al., 2017) or bisimulation abstractions (Kemertas & Aumentado-Armstrong, 2021). All take the observation as fixed and optimize a cheaper internal mapping, while the usual feature-side alternative—hand-engineered feature reduction—is brittle and task-specific. The closest feature-side precedent is automatic noise filtering with dynamic sparse training (Grooten et al., 2023), which prunes input-layer connectivity with binary masks updated by discrete prune-and-regrow rules; we instead train the gate itself by gradient through differentiable L_0 relaxations and select raw observation features end-to-end with the policy under matched sparsity levels.

G.4 Feature importance and interpretability in deep RL

Our pruning signal builds on occlusion- and saliency-based attribution for deep policies (Pham & Cangelosi, 2025), where masking a feature and measuring the change in the policy distribution estimates feature importance. We make this signal **dynamic** (tracked across training) and connect its collapse on not-yet-learned features to a statistical identifiability condition, which to our knowledge has not been formalized in prior occlusion-based selection.

H The Raycast Block as a Removable Visual Modality

The gated observation has $4{,}244$ feature-level gates, including a 32×32 single-channel visual block with one gate per pixel ($1{,}024$ gates). At inference the visual block's gates are all closed ($z{=}0$); this appendix examines what they decide and whether that decision is safe to act on.

Both STG and HC close the visual gates, so the learned selection closes an independently encoded visual modality.

We verify at the architecture level that acting on this closed gate is safe: Table 4 shows that a policy with the visual block removed matches the completion time of one that keeps it (statistically indistinguishable) while using substantially lower compute. Quantitatively, replacing the visual features with no visual channel (M5→M7′ in Table 4) changes completion time by only $-0.23\,\mathrm{s}$ ($39.83 \rightarrow 39.60\,\mathrm{s}$, marginally **faster**). The marginally faster M7′ suggests that the visual channel acts as a mild noise source for this policy, and closing the gate removes that interference; the closed gate is thus a signal that the modality can be removed, not merely a mask that looks reasonable.