MIXTUREPRIOR: ACTION-ROUTED TRANSITION PRIORS FOR LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models train policies on imagined rollouts, in which every future state is sampled from a learned stochastic transition prior. Although this prior is conditioned on the action, a single set of parameters must fit the next-state distribution for every action. A correction that helps one action can therefore hurt another, and such opposing corrections cancel in the shared parameters, so the prior can stall while individual actions remain under-fitted. Our key idea is that the prior should offer several candidate corrections and that the action should decide where each one is used. We propose MIXTUREPRIOR, which predicts multiple categorical transition components from the backbone's action-updated feature and combines them with an action-only gate. Because it replaces only the stochastic prior, MIXTUREPRIOR plugs into recurrent and Transformer-based world models without further changes. We analyze this design through the correction value of a component and show that, when a component helps some actions but hurts others, action-dependent weighting achieves a strictly larger first-order improvement than any fixed mixture. We integrate MIXTUREPRIOR into DreamerV3 and TWISTER and evaluate it on 26 Atari-100k games and 20 DeepMind Control tasks. With everything else fixed, four components improve the aggregate mean over a single-component prior in all four backbone–benchmark combinations. In a trained model, we further find components whose correction value changes sign across actions, which is the condition under which our analysis predicts action routing to help.

1 INTRODUCTION

Latent world models learn a compact model of environment dynamics and train the policy on trajectories imagined in its latent space (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020; 2021; 2025; Burchi & Timofte, 2025), a central approach in model-based reinforcement learning (Chua et al., 2018; Schrittwieser et al., 2020; Hansen et al., 2022; 2024; Moerland et al., 2023). At each imagined step, a stochastic transition prior maps the current latent state and action to a distribution over the next latent state (Micheli et al., 2023; Robine et al., 2023; Zhang et al., 2023; Burchi & Timofte, 2025). During training, a posterior can infer this state from the next observation, but during imagination the prior alone generates the future, so its errors propagate directly into the trajectories from which the policy learns.

Existing world models condition this prior on the action, yet they fit all actions with one shared set of prior parameters (Hafner et al., 2021; Micheli et al., 2023; Robine et al., 2023; Zhang et al., 2023; Burchi & Timofte, 2025). Conditioning lets the prediction depend on the action, but any update to these parameters affects the predictions for all actions at once. Because the correction required by one action need not help another, their learning signals can oppose each other and cancel, analogous to gradient interference in multi-task learning (Chen et al., 2018; Yu et al., 2020; Liu et al., 2021). The prior can thus reach a stationary point at which per-action corrections cancel on average while individual actions could still be predicted better (Section 3.1). Figure 1(a) shows this conflict in a trained TWISTER checkpoint: updating the prior head on transitions from one action lowers that action's held-out KL to the posterior from 35.35 to 27.88, but raises the KL of another action from 33.95 to 35.40.

Our key idea is that the prior should offer several candidate corrections and let the action decide where each one is used. We propose MIXTUREPRIOR, an action-routed mixture prior for latent

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

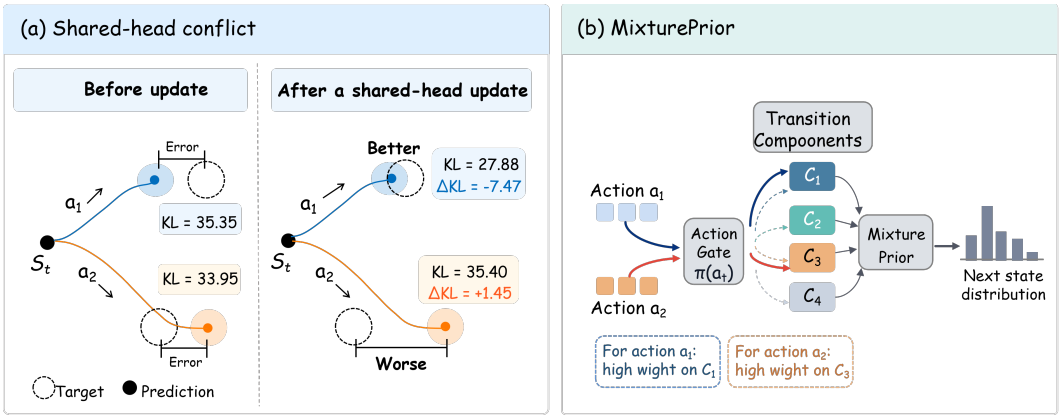

Figure 1: **Shared-prior conflict and the MIXTUREPRIOR mechanism. (a)** In a trained world model, the same shared-prior update can improve the transition prediction for one action while worsening another. **(b)** MIXTUREPRIOR provides multiple transition components and uses action-dependent weights to combine their corrections.

world models (Figure 1(b)). From the backbone's action-updated feature, MIXTUREPRIOR predicts K categorical transition components, and an action-only gate maps the action to weights that mix them (Jacobs et al., 1991; Jordan & Jacobs, 1994; Shazeer et al., 2017). The components determine which corrections are available, and the gate determines where each correction is used. Unlike mixture-of-experts world models that assign experts to separate tasks or objectives (Zhao et al., 2025; Tang et al., 2025), all components predict the same next-state distribution. Because only the stochastic prior changes, MIXTUREPRIOR applies to both recurrent and Transformer-based world models.

We explain why this structure helps through the *correction value* of a component, the first-order reduction in prior-fitting error from moving the prediction for an action toward it. A component mixed in with a fixed weight helps only if its correction value is positive on average over actions (Section 3.2); when it helps some actions and hurts others, action-dependent weighting achieves a strictly larger improvement than any fixed weighting (Section 3.3). For a given total effect, this advantage is largest when benefit and harm balance, which is exactly when a fixed mixture sees no use for the component. Moreover, at a stationary single-component prior whose per-action corrections still conflict, splitting the component and routing the copies by action opens a strict descent direction (Section 3.4). Intuitively, splitting creates two opposite corrections and the gate sends each action to the copy that helps it, so corrections that cancel in a shared head add up instead. Both parts are needed: gating identical copies changes nothing, and splitting without routing helps only if the split itself has negative curvature. The mixture can therefore keep improving where a single head has stalled.

We integrate MIXTUREPRIOR into DreamerV3 and TWISTER and evaluate it on 26 Atari-100k games (Bellemare et al., 2013; Machado et al., 2018) and 20 visual DMC tasks (Tunyasuvunakool et al., 2020). With everything else fixed, four components raise the aggregate mean over a single-component prior in all four backbone–benchmark combinations, by 9.7% and 56.1% on Atari-100k and by 7.2% and 1.4% on DMC for TWISTER and DreamerV3, respectively. MIXTUREPRIOR exceeds the published scores of both backbones in mean and median in all four combinations (Section 5.2). The best number of components varies across tasks (Section 5.3). Checkpoint diagnostics support the analysis: a trained model contains components whose correction value changes sign across actions, and on a trained single-component model, splitting and routing lowers held-out prediction loss by 0.112 against 0.004 for splitting alone, while shuffling the routing removes the gain (Section 5.4).

Our contributions are threefold:

1. We identify a *shared-prior correction conflict* in latent world models, in which corrections needed by different actions cancel in the shared prior parameters.

2. We propose MIXTUREPRIOR, a drop-in action-routed mixture prior, and use the correction value to characterize when components help, when action-dependent weighting beats any fixed mixture, and why splitting and routing escape a stalled single head.
3. We show on two backbones and 46 tasks that four components improve over a matched single-component prior, and verify the action-dependent structure predicted by the analysis on trained checkpoints.

2 RELATED WORK

Latent world models and transition priors. World models learn predictive latent dynamics that support planning or policy learning through imagined trajectories (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020; 2021; 2025), alongside video-prediction models, learned search, and latent model-predictive control (Kaiser et al., 2020; Schrittwieser et al., 2020; Ye et al., 2021; Hansen et al., 2022; 2024; Janner et al., 2019). Transformer-based world models such as IRIS, TWM, STORM, and TWISTER, which add action-conditioned contrastive predictive learning, replace the recurrent sequence model (Micheli et al., 2023; Robine et al., 2023; Zhang et al., 2023; Burchi & Timofte, 2025), and related designs add masked or self-predictive objectives (Seo et al., 2023; Schwarzer et al., 2021). How the stochastic prior is trained has also been refined: DreamerV2 balances the KL between prior and posterior, and DreamerV3 adds free bits and mixes a uniform distribution into the categorical prior (Hafner et al., 2021; 2025). On the conditioning side, action-conditional video prediction injects the action through multiplicative interactions with the encoded state (Oh et al., 2015; Finn et al., 2016). MIXTUREPRIOR is complementary to these advances: it keeps the sequence model and the training losses, and changes how the prior combines candidate predictions for each action.

Mixture models and modular dynamics. Mixtures of experts use learned gates to allocate inputs among specialized predictors (Jacobs et al., 1991; Jordan & Jacobs, 1994), while finite mixture models represent distributions through weighted components (McLachlan & Peel, 2000). Modern mixture-of-experts layers scale conditional routing to large models (Shazeer et al., 2017; Fedus et al., 2022; Riquelme et al., 2021; Puigcerver et al., 2024), and routed or soft-modular networks share parameters across tasks (Rosenbaum et al., 2018; Ma et al., 2018; Yang et al., 2020). In latent-variable models, mixture priors such as the VampPrior make the prior more expressive (Tomczak & Welling, 2018). World models have also used mixtures: the recurrent world model of Ha and Schmidhuber predicts a mixture-density transition (Ha & Schmidhuber, 2018), M3W shares mixture-of-experts dynamics across tasks and agents to reduce interference (Zhao et al., 2025), and MoE-World routes state, reward, and termination predictions to task-specific experts (Tang et al., 2025). In contrast, MIXTUREPRIOR does not decompose tasks or prediction objectives: all components predict the same next-state categorical prior from the same action-updated feature, and the action alone decides how these candidate corrections are combined.

Conflicting updates and parameter splitting. Interference between shared parameter updates has been studied extensively in multi-task and multi-objective learning (Sener & Koltun, 2018): GradNorm balances task gradient magnitudes, PCGrad projects conflicting task gradients, and CAGrad balances average optimization with worst-task local improvement (Chen et al., 2018; Yu et al., 2020; Liu et al., 2021). Architectural growth offers a different response: Net2Net expands a trained network while preserving its function (Chen et al., 2016; Wei et al., 2016), and splitting steepest descent duplicates neurons when the split has negative curvature (Wu et al., 2019). Our setting differs in that actions are not separate tasks but conditioning variables within one transition objective, so we neither alter gradients nor assign fixed experts to subtasks. Instead, we characterize the *correction value* of a component for each action and show that splitting a stationary prior can descend through a cross term between the split and an action-dependent gate, even when the split alone has no negative curvature (Section 3.4).

3 WHY AN ACTION-ROUTED MIXTURE PRIOR?

We analyze the stochastic transition prior alone, following the latent state-space perspective of prior work (Watter et al., 2015; Karl et al., 2017; Lee et al., 2020; Fraccaro et al., 2016; Krishnan et al.,

2017), and hold the data distribution, backbone representation, and posterior fixed; the results describe this prior-fitting subproblem rather than the full world-model and actor–critic optimization.

3.1 Shared Prior Updates Can Cancel across Actions

Let $\mathbf{x} = (\mathbf{h}, \mathbf{a})$, where $\mathbf{h}$ is the action-updated deterministic feature and $\mathbf{a}$ the action, and let $p_{\mathbf{w},i}(\cdot \mid \mathbf{x})$ be the prior for latent factor $i \in \{1, \dots, F\}$, with parameters $\mathbf{w}$ shared across all factors and actions. Training matches this prior to the posterior $q_i(\cdot \mid \mathbf{x}, \mathbf{o}')$ inferred from the next observation $\mathbf{o}'$ (Chung et al., 2015; Karl et al., 2017; Lee et al., 2020). With the predictive posterior target $\bar{q}_i(\cdot \mid \mathbf{x}) = \mathbb{E}_{\mathbf{o}'|\mathbf{x}}[q_i(\cdot \mid \mathbf{x}, \mathbf{o}')]$, the prior-fitting risk for action $\mathbf{a}$ and its average over actions are

$$\mathcal{R}_{\mathbf{a}}(\mathbf{w}) = \mathbb{E}_{\mathbf{x}|\mathbf{a}} \sum_i D_{\mathrm{KL}}\big(\bar{q}_i(\cdot \mid \mathbf{x}) \,\|\, p_{\mathbf{w},i}(\cdot \mid \mathbf{x})\big), \qquad \mathcal{R}(\mathbf{w}) = \mathbb{E}_{\mathbf{a}} \mathcal{R}_{\mathbf{a}}(\mathbf{w}). \tag{1}$$

A small shared update $\mathbf{w} \mapsto \mathbf{w} + \epsilon \mathbf{u}$ changes the risk of each action by

$$\mathcal{R}_{\mathbf{a}}(\mathbf{w} + \epsilon \mathbf{u}) = \mathcal{R}_{\mathbf{a}}(\mathbf{w}) - \epsilon\, C_{\mathbf{a}}(\mathbf{u}) + o(\epsilon), \qquad C_{\mathbf{a}}(\mathbf{u}) := -\nabla_{\mathbf{w}} \mathcal{R}_{\mathbf{a}}(\mathbf{w})^{\top} \mathbf{u}, \tag{2}$$

so the update helps action $\mathbf{a}$ when its *action-conditional improvement* $C_{\mathbf{a}}(\mathbf{u})$ is positive and hurts it when $C_{\mathbf{a}}(\mathbf{u})$ is negative. At a stationary point of the shared risk, every direction satisfies

$$0 = -\nabla_{\mathbf{w}} \mathcal{R}(\mathbf{w})^{\top} \mathbf{u} = \mathbb{E}_{\mathbf{a}}\big[C_{\mathbf{a}}(\mathbf{u})\big]. \tag{3}$$

Stationarity therefore requires only that the per-action improvements cancel on average; individual actions can still have nonzero and opposing improvements, analogous to shared-parameter interference in multi-task optimization (Chen et al., 2018; Yu et al., 2020; Liu et al., 2021). A stationary shared prior thus need not fit each action-conditioned transition well, and Figure 1(a) shows such opposing improvements in a trained checkpoint. Equation (3) holds for any partition of the data; we partition by action because the action is observed at every imagined step, so the prior can use it to decide which correction to apply.

3.2 Components Provide Alternative Corrections

The cancellation in Equation (3) arises because one parameter direction must serve every action; multiple transition components relax this constraint by each proposing a different change to the predicted distribution (Rosenbaum et al., 2018; Ma et al., 2018; Yang et al., 2020). To decide when a candidate component r is worth adding to the current prediction p, we perturb the prior's output rather than its parameters, using the interpolation $p_{\epsilon,i} = (1-\epsilon)p_i + \epsilon r_i$ with $\epsilon \in [0, 1]$. Writing $\mathcal{R}_{\mathbf{a}}(p)$ for the risk in Equation (1) evaluated at a prediction p, and omitting the dependence on $\mathbf{x}$, we define the *correction value* of r for action $\mathbf{a}$ as

$$\mathcal{A}_r(\mathbf{a}) := -\frac{d}{d\epsilon} \mathcal{R}_{\mathbf{a}}(p_\epsilon)\Big|_{\epsilon=0} = \mathbb{E}_{\mathbf{x}|\mathbf{a}}\Big[\sum_{i,j} \underbrace{\big(\bar{q}_i(j) - p_i(j)\big)}_{\text{correction required}} \underbrace{\big(r_i(j) - p_i(j)\big)}_{\text{correction supplied}} \Big/\, p_i(j)\Big]. \tag{4}$$

It is positive when the component moves probability where the target requires it, so that mixing in a little of r lowers the error for action $\mathbf{a}$.

Proposition 1 (When an additional component helps)**.** *Assume p and r assign positive probability to every category. Then $\mathcal{R}(p_\epsilon)$ is convex in ϵ with derivative $-\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})]$ at $\epsilon = 0$, so some $\epsilon > 0$ gives $\mathcal{R}(p_\epsilon) < \mathcal{R}(p)$ if and only if $\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})] > 0$.*

Additional components therefore create opportunities for improvement but do not guarantee a better prior; in particular, a component can help some actions and hurt others, $\mathcal{A}_r(\mathbf{a}_1) > 0 > \mathcal{A}_r(\mathbf{a}_2)$.

3.3 Action Gating Recovers Corrections Hidden by Averaging

A fixed mixture weight must then trade the benefit of such a component against its harm, so we let the weight depend on the action, following the principle of conditional routing (Rosenbaum et al., 2018; Ma et al., 2018). For a weight function $0 \le \omega(\mathbf{a}) \le 1$, the routed interpolation and its first-order reduction in prior-fitting risk are

$$p^{\omega}_{\epsilon,i} = p_i + \epsilon\, \omega(\mathbf{a})(r_i - p_i), \qquad \mathcal{G}(\omega) := -\frac{d}{d\epsilon} \mathcal{R}(p^{\omega}_{\epsilon})\Big|_{\epsilon=0} = \mathbb{E}_{\mathbf{a}}\big[\omega(\mathbf{a})\, \mathcal{A}_r(\mathbf{a})\big]. \tag{5}$$

In MixturePrior, ω is realized by the action-only gate (Section 4).

Proposition 2 (Action-selective correction). *For a fixed prediction p and candidate component r, let $(v)_\pm = \max(\pm v, 0)$. The largest first-order improvements achievable with action-dependent weights and with a single weight shared by all actions are $\mathcal{G}_{\text{act}} = \mathbb{E}_{\mathbf{a}}[(\mathcal{A}_r(\mathbf{a}))_+]$ and $\mathcal{G}_{\text{fixed}} = (\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})])_+$, and*

$$\mathcal{G}_{\text{act}} - \mathcal{G}_{\text{fixed}} = \min\Big\{\mathbb{E}_{\mathbf{a}}\big[(\mathcal{A}_r(\mathbf{a}))_+\big],\ \mathbb{E}_{\mathbf{a}}\big[(\mathcal{A}_r(\mathbf{a}))_-\big]\Big\} \geq 0, \tag{6}$$

which is strictly positive whenever the component is useful for some actions and harmful for others, each with nonzero probability.

Components thus determine *which corrections are available*, and the gate determines *where each correction is used*. For a given magnitude $\mathbb{E}_{\mathbf{a}}|\mathcal{A}_r(\mathbf{a})|$, the gap in Equation (6) is largest when benefit and harm are balanced, which is exactly when $\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})] \approx 0$ and a fixed mixture finds the component useless.

3.4 Splitting with Action Routing Escapes a Stationary Single Head

We now show that the improvements that cancel in Equation (3) can themselves produce useful components. From a stationary single-head prior $\mathbf{w}^\star$, we split the head into two copies displaced by $\pm\alpha\mathbf{u}$ and let an action score $s(\mathbf{a})$ set their weights:

$$p_i^{\text{mix}}(\alpha, \beta) = \pi_+(\mathbf{a}; \beta)\, p_{\mathbf{w}^\star + \alpha\mathbf{u}, i} + \pi_-(\mathbf{a}; \beta)\, p_{\mathbf{w}^\star - \alpha\mathbf{u}, i}, \qquad \pi_\pm(\mathbf{a}; \beta) = \tfrac{1}{2}\big(1 \pm \tanh(\beta s(\mathbf{a}))\big). \tag{7}$$

At $(\alpha, \beta) = (0, 0)$ the mixture reproduces the single head exactly; let $\mathcal{R}^{\text{mix}}(\alpha, \beta)$ denote its prior-fitting risk.

Theorem 3 (Split–gate local descent). *Let $\mathbf{w}^\star$ be a stationary point of $\mathcal{R}$ and let $C_{\mathbf{a}}$ be the action-conditional improvement (Equation 2) at $\mathbf{w}^\star$. Then $\mathcal{R}^{\text{mix}}$ has zero gradient at $(\alpha, \beta) = (0, 0)$ and Hessian $\left(\begin{smallmatrix} \kappa & \chi \\ \chi & 0 \end{smallmatrix}\right)$, where $\kappa = \partial_{\alpha\alpha}\mathcal{R}^{\text{mix}}(0, 0)$ and $\chi = -\mathbb{E}_{\mathbf{a}}[s(\mathbf{a})\, C_{\mathbf{a}}(\mathbf{u})]$. If $\chi \neq 0$, this Hessian has a negative eigenvalue for every κ, so $\mathcal{R}^{\text{mix}}$ strictly decreases along its eigenvector for small steps.*

The cross term χ is where routing enters. Because $\mathbb{E}_{\mathbf{a}}[C_{\mathbf{a}}(\mathbf{u})] = 0$ at a stationary point, χ is minus the covariance between the action score and the per-action improvements that cancel in Equation (3). The score $s(\mathbf{a}) = C_{\mathbf{a}}(\mathbf{u})$ gives $\chi = -\mathbb{E}_{\mathbf{a}}[C_{\mathbf{a}}(\mathbf{u})^2]$, which is nonzero whenever some action still has a nonzero improvement along $\mathbf{u}$. Intuitively, splitting creates two opposite corrections, $+\mathbf{u}$ and $-\mathbf{u}$, and the gate sends each action to the copy that helps it, so improvements that cancel under a shared update now add up; an affine gate on action features can realize such a score (Corollary 4, Appendix A.5).

Routing is what removes the dependence on κ. Without it, s is constant, so $\chi = 0$ by Equation (3) and any descent must come from the splitting curvature κ, whose sign is not fixed; shuffling actions relative to scores likewise drives χ toward zero. This distinguishes the mechanism from splitting steepest descent, which splits a unit only when the split itself has negative curvature (Wu et al., 2019). The result holds at any stationary point of the single head, including heads that take the action as input (Appendix A.6 gives an exact example), but it is local and concerns the prior-fitting problem with feature networks fixed. Section 5.4 tests its predictions on a trained checkpoint.

4 World-Model Implementation

MixturePrior replaces the stochastic transition prior of a latent world model with factorized categorical states (van den Oord et al., 2017; Hafner et al., 2021; 2025; Burchi & Timofte, 2025) and keeps the backbone's deterministic dynamics and training interfaces (Figure 2).

Prior parameterization. From its deterministic state $\mathbf{h}_t$, stochastic latent $\mathbf{z}_t$, and action $\mathbf{a}_t$, the backbone computes the action-updated feature $\widetilde{\mathbf{h}}_{t+1} = f_b(\mathbf{h}_t, \mathbf{z}_t, \mathbf{a}_t)$. A shared MLP trunk with separate output blocks maps this feature to KFC logits for K components, F latent factors, and C categories, and each component is normalized over categories. An action-only MLP g_ψ gives mixing weights $\pi_k(\mathbf{a}_t) = \text{softmax}\big(g_\psi(\mathbf{a}_t)/\tau\big)_k$, shared across all F factors; a small temperature τ approaches one component per action, and a large one a uniform mixture (Shazeer et al., 2017; Fedus et al., 2022). The gate output layer is zero-initialized, so training starts from uniform weights

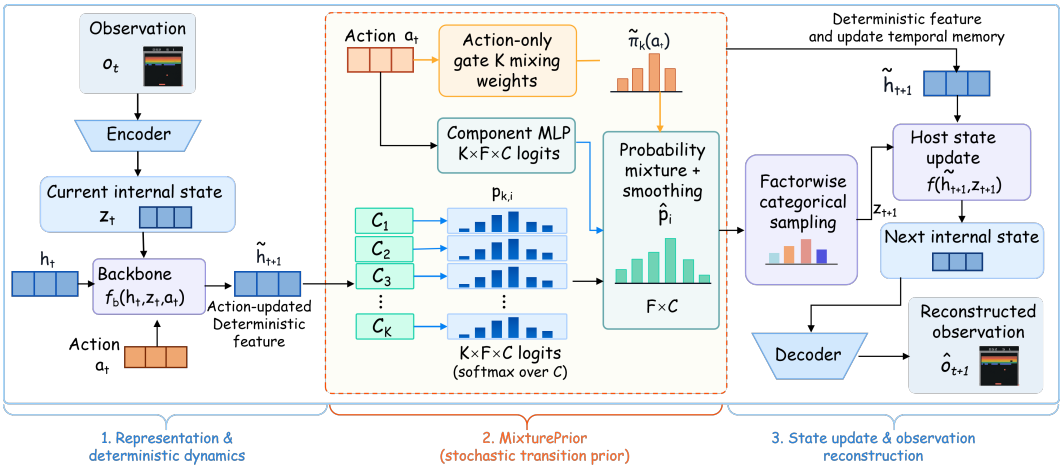

Figure 2: **World-model integration.** The action-updated feature predicts K transition components, and an action-only gate weights their probabilities. Each factor is sampled from the aggregated distribution, and the sampled latents enter the backbone's state update.

for every action. With floored weights $\widetilde{\pi}_k(\mathbf{a}_t) \propto \max\{\pi_k(\mathbf{a}_t), \epsilon_g\}$ and the backbone's smoothing $\eta \in [0, 1)$, the effective prior is

$$\widehat{p}_i(j \mid \mathbf{x}_t) = (1-\eta) \sum_{k=1}^{K} \widetilde{\pi}_k(\mathbf{a}_t) p_{k,i}(j \mid \mathbf{x}_t) + \frac{\eta}{C}, \qquad \mathbf{x}_t = (\widetilde{\mathbf{h}}_{t+1}, \mathbf{a}_t), \tag{8}$$

where $\epsilon_g > 0$ is a numerical floor and $\eta = 0$ removes smoothing. The action-only gate and probability-space mixing match the weights of Proposition 2 and the interpolation of Section 3.2, and since smoothing is affine, the analysis applies to the effective probabilities (Appendix A.7). K and τ are the two hyperparameters varied in our experiments.

Joint training. For posterior marginals q_i inferred from the next observation, the dynamics objective is

$$\mathcal{L}_{\text{dyn}} = \mathbb{E} \max \left\{ \lambda_{\text{free}}, \sum_{i=1}^{F} D_{\text{KL}}(\text{sg}[q_i] \,\|\, \widehat{p}_i) \right\}, \tag{9}$$

where sg stops gradients and $\lambda_{\text{free}} \geq 0$ is the backbone's free-nats threshold (zero without clipping). The representation loss passes no gradient into the prior, and all other objectives and loss weights of the backbone are unchanged. Components and gate are trained jointly with the world model, without component labels, correction-value estimates, or a separate fitting stage. Unlike the analysis in Section 3, training also changes the features and targets, and free-nats clipping removes the dynamics gradient of samples below the threshold (Appendix A.7).

Latent imagination. From a posterior-initialized state, each action updates the deterministic feature, and the next latent is sampled factor by factor from the effective prior,

$$\mathbf{z}_{t+1} \sim \prod_{i=1}^{F} \text{Cat}(\widehat{\mathbf{p}}_i(\mathbf{x}_t)) \,. \tag{10}$$

Because each factor is sampled from its own mixture, no component index is shared across factors. The backbone combines the sampled latent with its deterministic feature and temporal memory to continue the rollout; it runs once per transition, so the added computation is confined to the prior module.

5 Experiments

We ask three questions. **RQ1:** Does MixturePrior improve control across backbones and benchmarks (Section 5.2)? **RQ2:** How does the benefit depend on the number of components

(Section 5.3)? **RQ3:** Do trained models contain the action-dependent corrections that our analysis relies on, and does routing exploit them (Section 5.4)?

5.1 Experimental Setup

We instantiate the same prior interface in the recurrent DreamerV3 (Hafner et al., 2025) and the Transformer-based TWISTER (Burchi & Timofte, 2025), and evaluate each on all 26 Atari-100k games (Bellemare et al., 2013; Machado et al., 2018) and 20 visual DMC tasks (Tunyasuvunakool et al., 2020). Atari-100k allows 100K agent–environment interactions and reports human-normalized score (HNS) (Kaiser et al., 2020; Ye et al., 2021); DMC reports episode return after 1M interactions. We train six settings $(K, \tau) \in \{(1,1),(2,1),(4,1),(8,1),(4,0.5),(4,2)\}$, where K is the number of components and τ the gate temperature, which vary K at $\tau = 1$ and τ at $K = 4$. The $K = 1$ setting uses the same prior interface and training pipeline with a single component, so comparing it with $K > 1$ isolates the effect of the mixture structure. Each backbone–task–setting combination is trained with one seed; we therefore base conclusions on aggregates over tasks and treat single-task differences as descriptive (Henderson et al., 2018; Agarwal et al., 2021).

5.2 RQ1: Control Performance and Learning Dynamics

Table 1 compares MixturePrior, with K and τ selected per task, against the published scores of each backbone, reporting both mean and median aggregates (Agarwal et al., 2021). It exceeds the published mean and median in all four backbone–benchmark combinations, and the gains are broad rather than driven by a few tasks: it is better on 25 of 26 TWISTER Atari games, 17 of 20 TWISTER DMC tasks, 20 of 26 DreamerV3 Atari games, and all 20 DreamerV3 DMC tasks. Because per-task selection uses the evaluation runs themselves, we also isolate the effect of the structure with a matched comparison. With the backbone, training pipeline, and temperature $\tau = 1$ fixed, replacing the single-component prior with four components raises the aggregate mean in all four combinations (Table 2): from 111.86 to 122.76 HNS (+9.7%) for TWISTER and from 80.46 to 125.62 (+56.1%) for DreamerV3 on Atari-100k, and by 7.2% and 1.4% on DMC. The gain is small only on DreamerV3 DMC, where $K = 4$ is better on 8 of the 20 tasks; on DreamerV3, the fixed $K = 4$ setting alone already exceeds the published mean on both benchmarks (Table 9).

Figure 3 compares learning curves with the backbone's native prior on two representative tasks. On Cartpole Swingup, MixturePrior learns faster with both backbones: it reaches a return of 800 after 84.8K interactions with TWISTER, where the native prior does not reach it within the budget, and after 24K instead of 64K interactions with DreamerV3. On Alien, it achieves a larger area under the return curve with TWISTER but not with DreamerV3 (Appendix D), so faster learning is task dependent.

Table 1: **Retrospective task-wise best-of-six performance on Atari-100k and DMC.** Reference is the published score of the corresponding backbone (TWISTER or DreamerV3). Mean and median are computed across tasks, in human-normalized score for Atari-100k and episode return for DMC. For Atari-100k, HNS is normalized such that 0 corresponds to random performance and 100 to human performance. Rel. Gain is the relative change in the mean. W/T/L counts the tasks on which the selected MixturePrior score is above, equal to, or below the reference. †Five selected runs use temperatures outside the six settings (Table 13).

		Mean		Median			
Backbone	Benchmark	Reference	MixturePrior	Reference	MixturePrior	Rel. Gain	W/T/L
TWISTER	Atari-100k†	162.09	**168.95**	77.26	**91.26**	+4.23%	25/0/1
TWISTER	DMC	801.80	**844.16**	907.60	**936.12**	+5.28%	17/1/2
DreamerV3	Atari-100k	112.40	**153.19**	48.66	**60.72**	+36.29%	20/1/5
DreamerV3	DMC	786.05	**899.13**	861.50	**976.80**	+14.39%	20/0/0

5.3 RQ2: Do Multiple Transition Components Help?

Proposition 1 implies that additional components create correction opportunities without guaranteeing gains, so the benefit need not grow with K. Figure 4 shows that it does not. At $\tau = 1$, the aggregate

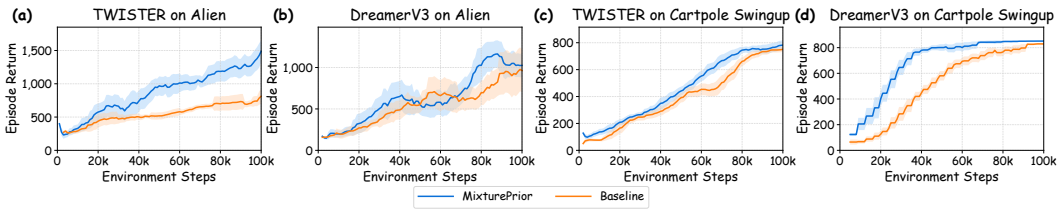

Figure 3: **Representative learning curves.** Episode return versus environment interactions for MIXTUREPRIOR and the backbone's native prior (Baseline) on Alien and Cartpole Swingup. MIXTUREPRIOR uses $(K, \tau) = (4, 0.5)$ for TWISTER, $(4, 2)$ for DreamerV3 on Alien, and $(4, 1)$ for DreamerV3 on Cartpole Swingup. Each curve is a single seed; shaded regions are not across-seed confidence intervals.

mean peaks at $K = 4$ on both Atari-100k settings and at $K = 2$ on TWISTER DMC, and increases up to $K = 8$ only on DreamerV3 DMC; on TWISTER, $K = 8$ even falls below $K = 1$ on both benchmarks. Task-level preferences are more dispersed than the aggregates. On TWISTER Atari-100k, for example, $K = 1$ is best on slightly more games than $K = 4$ (30.8% versus 26.9%), yet $K = 4$ has the higher mean because its wins are larger where they occur, as on Gopher, Boxing, and Freeway. Varying the temperature at $K = 4$, $\tau = 1$ gives the highest mean in three of the four settings, all except TWISTER DMC, where $\tau = 2$ is best. Multiple components are thus useful, but the best number is task dependent; $K = 4$, $\tau = 1$ is the best fixed setting on Atari-100k for both backbones.

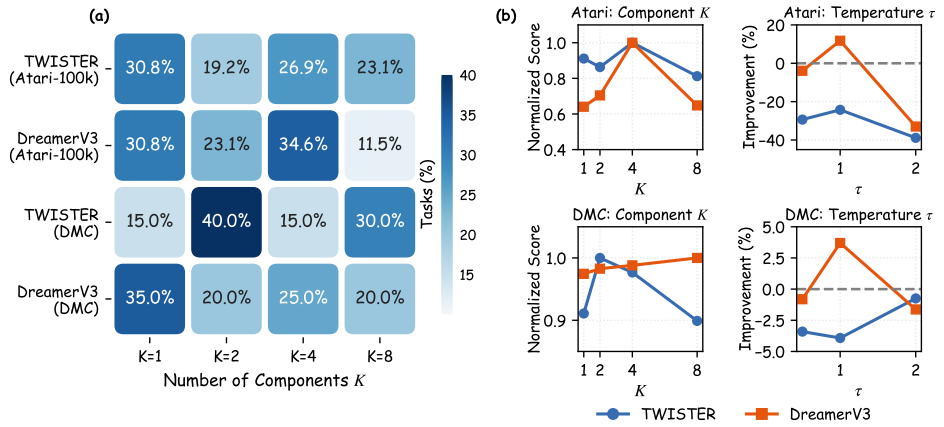

Figure 4: **Sensitivity to the number of components and the temperature.** (a) Share of tasks on which each $K \in \{1, 2, 4, 8\}$ gives the best return at $\tau = 1$. (b) Aggregate mean versus K at $\tau = 1$, normalized by the best K in each setting (left), and relative change of the $K = 4$ mean with respect to the published backbone score versus τ (right).

5.4 RQ3: CAN ACTION-CONDITIONED ROUTING EXPLOIT SELECTIVE CORRECTIONS?

Proposition 2 predicts an advantage for action-dependent weighting when the same component helps some actions and hurts others. Figure 5 shows this structure in a trained Kangaroo checkpoint: component C3 has correction value 13.89 for action 11 but -3.26 and -2.76 for actions 12 and 14, and C2 changes sign in the same way.

Theorem 3 makes sharper predictions, which we test on a trained DreamerV3 Battle Zone checkpoint with $K = 1$. We duplicate its prior into four identical components, which reproduces its held-out KL to numerical precision, and compare five perturbations of this matched starting point on held-out transitions (Table 3; details in Appendix B.4). Consistent with a stationary single head, a common update of the head does not lower the held-out dynamics loss ($+0.014$), and a gate-only update changes nothing because the components are identical. Splitting alone lowers the loss by 0.004,

Table 2: **Matched component count.** Aggregate mean at $\tau{=}1$; $K{=}1$ and $K{=}4$ share the pipeline and differ only in K.

Backbone	Suite	$K{=}1$	$K{=}4$	Change
TWISTER	Atari	111.86	**122.76**	+9.7%
TWISTER	DMC	718.41	**770.37**	+7.2%
DreamerV3	Atari	80.46	**125.62**	+56.1%
DreamerV3	DMC	803.89	**815.13**	+1.4%

Table 3: **Split–gate test** of Theorem 3: best held-out change in dynamics loss for a Battle Zone $K{=}1$ prior copied into four components.

Perturbation	ΔL_{dyn}
Aligned split + gate	$\mathbf{-0.1124}$
Split only	-0.0044
Common-head update	$+0.0141$
Shuffled split + gate	0.0000
Gate only	0.0000

whereas aligned splitting and gating lowers it by 0.112; shuffling the correspondence between split and gate, which drives χ toward zero, removes the gain entirely. On this checkpoint, the benefit therefore comes from the interaction between splitting and routing, as the theorem predicts.

Whether a gate exploits these opportunities in practice is a separate question. On ten trained checkpoints, we freeze the component bank and compare the best fixed mixture with an action-conditioned gate refitted on the same data (Table 5, Appendix C). The refitted gate significantly lowers held-out NLL on five checkpoints across both benchmarks (e.g., Frostbite -1.353, Walker Walk -0.382), is significantly worse on one (Kangaroo), and is indistinguishable on four. The gates learned end-to-end exploit these opportunities much less: shuffling their action inputs changes held-out NLL by at most 0.014, and they trail the best fixed mixture on nine of the ten checkpoints. The learned components thus contain action-dependent corrections that current end-to-end gate training realizes only in part, which we return to in Section 6.

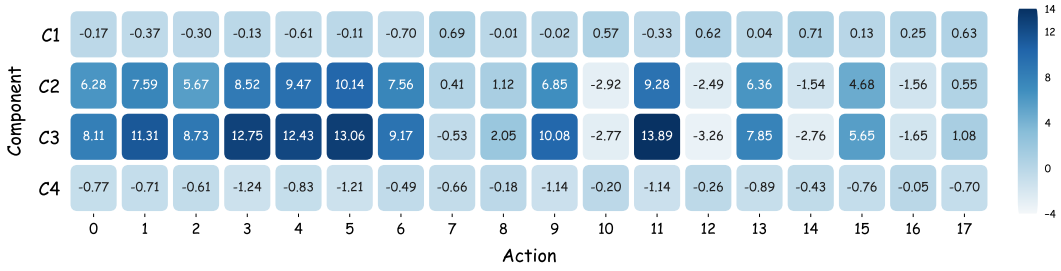

Figure 5: **Action-dependent correction values on Atari Kangaroo.** Each cell reports the empirical correction value $\mathcal{A}_{p_k}(\mathbf{a})$ (Equation 4) for component k and action $\mathbf{a}$, evaluated around the action-conditioned mixture prediction of the trained $K = 4$, $\tau = 1$ checkpoint. Positive values indicate locally beneficial corrections, and negative values indicate harmful ones.

6 Conclusion

We studied the stochastic transition prior of latent world models as a fitting problem in its own right. Because one set of prior parameters serves every state–action context, corrections needed by different actions can cancel, and a stationary shared prior need not fit each action. The correction value of a component makes this precise: extra components enlarge the set of available corrections, but only action-dependent weighting keeps the beneficial ones from being averaged against the harmful ones. MixturePrior follows from this reading and improves over a matched single-component prior on both backbones and benchmarks.

Our analysis fixes features, posterior targets, and the data distribution, so it describes the prior-fitting subproblem rather than the full optimization; free-nats clipping in particular can suppress the descent direction the mechanism supplies. Moreover, a gate refitted on frozen components improves held-out prediction on half of our checkpoints, yet end-to-end trained gates stay close to their action-shuffled counterparts: the corrections are present but not fully exploited, which points to gate-specific training rather than a different parameterization.

Ethics Statement

This work studies world models for reinforcement learning on simulated benchmarks: Atari-100k through the Arcade Learning Environment and the DeepMind Control Suite. It involves no human subjects, personal data, or new data sets. Advances in sample-efficient reinforcement learning could eventually support decision-making systems deployed in the real world, which would require their own safety evaluation; we are not aware of ethical concerns specific to the proposed method.

Reproducibility Statement

Appendix B specifies the prior parameterization, action gate, probability aggregation, smoothing, and backbone configurations, including parameter counts (Appendix B.1). Section 5.1 lists the benchmarks and the six (K, τ) settings, and Appendix E reports the score of every setting on every task together with the published references, so that all aggregates in Section 5 can be recomputed; each setting is trained with one seed. Section 3 and Appendix A give the complete analysis and proofs, and Appendix B.3 describes a script that checks the main derivations numerically. The checkpoint diagnostics are described in Appendices B.4, C, and D.

AI Use Statement

In this work, we used generative AI tools to provide feedback on research methodology and experiments, to interpret results, to formulate and restate mathematical claims, and to assist in the writing of proofs. We have not used generative AI tools to generate synthetic data sets, and translation, data set cleaning and reformatting, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to edit the paper for readability and consistency, to recompute aggregate statistics from the reported per-task scores as a consistency check, and to suggest related work. We have reviewed all AI-assisted work: the authors checked every mathematical statement and proof, recomputed the reported aggregates from the per-task tables, and verified each suggested reference against the original publication. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. doi: 10.1613/JAIR.3912. URL https://mlanthology.org/jair/2013/bellemare2013jair-arcade/.

Maxime Burchi and Radu Timofte. Learning transformer-based world models with contrastive predictive coding. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/148c0aeea1c5da82f4fa86a09d4190da-Abstract-Conference.html.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016. URL https://research.google/pubs/net2net-accelerating-learning-via-knowledge-transfer/.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 794–803, 2018. URL https://proceedings.mlr.press/v80/chen18a.html.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3de568f8597b94bda53149c7d7f5958c-Abstract.html.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/hash/b618c3210e934362ac261db280128c22-Abstract.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://jmlr.org/papers/v23/21-0998.html.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/d9d4f495e875a2e075a1a4a6e1b9770f-Abstract.html.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/208e43f0e45c4c78cafadb83d2888cb6-Abstract.html.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://research.google/pubs/dream-to-control-learning-behaviors-by-latent-imagination/.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://research.google/pubs/mastering-atari-with-discrete-world-models/.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/cf73d57b6dcda32b293df7c2d5341f49-Abstract-Conference.html.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11694. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79. URL https://direct.mit.edu/neco/article/3/1/79/5560/Adaptive-Mixtures-of-Local-Experts.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994. doi: 10.1162/neco.1994.6.2.181. URL https://direct.mit.edu/neco/article/6/2/181/5779/Hierarchical-Mixtures-of-Experts-and-the-EM.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for Atari. In *International Conference on Learning Representations*, 2020. URL https://mlanthology.org/iclr/2020/kaiser2020iclr-model/.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HyTqHL5xg.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 2101–2109, 2017. doi: 10.1609/aaai.v31i1.10779. URL https://ojs.aaai.org/index.php/AAAI/article/view/10779.

Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/08058bf500242562c0d031ff830ad094-Abstract.html.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9d27fdf2477ffbff837d73ef7ae23db9-Abstract.html.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1930–1939, 2018. URL https://doi.org/10.1145/3219819.3220007.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018. doi: 10.1613/jair.5699. URL https://research.google/pubs/revisiting-the-arcade-learning-environment-evaluation-protocols-and-open-problems-for-general-agents/.

Geoffrey McLachlan and David Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2000. doi: 10.1002/0471721182. URL https://onlinelibrary.wiley.com/doi/book/10.1002/0471721182.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023. doi: 10.1561/2200000086. URL https://doi.org/10.1561/2200000086.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/6ba3af5d7b2790e73f0de32e5c8c1798-Abstract.html.

Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/79fea214543ba263952ac3f4e5452b14-Abstract-Conference.html.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TdBaDGCpjly.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ry8dvM-R.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4. URL https://www.nature.com/articles/s41586-020-03051-4.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL https://mlanthology.org/iclr/2021/schwarzer2021iclr-dataefficient/.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/432aca3a1e345e339f35a30c8f65edce-Abstract.html.

Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1332–1344. PMLR, 2023. URL https://proceedings.mlr.press/v205/seo23a.html.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://research.google/pubs/outrageously-large-neural-networks-the-sparsely-gated-mixture-of-experts-layer/.

Cong Tang, Yuang Liu, Yueling Wu, Wence Han, Qian Yin, Xin Zheng, Wenyi Zeng, and Qiuli Zhang. MoE-World: A mixture-of-experts architecture for multi-task world models. *Electronics*, 14(24):4884, 2025. doi: 10.3390/electronics14244884. URL https://www.mdpi.com/2079-9292/14/24/4884.

Jakub M. Tomczak and Max Welling. VAE with a VampPrior. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1214–1223. PMLR, 2018. URL https://proceedings.mlr.press/v84/tomczak18a.html.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. doi: 10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/a1afc58c6ca9540d057299ec3016d726-Abstract.html.

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 564–572. PMLR, 2016. URL https://proceedings.mlr.press/v48/wei16.html.

Lemeng Wu, Dilin Wang, and Qiang Liu. Splitting steepest descent for growing neural architectures. In *Advances in Neural Information Processing Systems*, volume 32, pp. 10655–10665, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/3a01fc0853ebeba94fde4d1cc6fb842a-Abstract.html.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/32cfdce9631d8c7906e8e9d6e68b514b-Abstract.html.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering Atari games with limited data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 25476–25488, 2021. URL https://papers.neurips.cc/paper_files/paper/2021/hash/d5eca8dc3820cad9fe56a3bafda65ca1-Abstract.html.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. STORM: Efficient stochastic transformer based world models for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1182. URL https://papers.neurips.cc/paper_files/paper/2023/hash/5647763d4245b23e6a1cb0a8947b38c9-Abstract-Conference.html.

Zijie Zhao, Zhongyue Zhao, Kaixuan Xu, Yuqian Fu, Jiajun Chai, Yuanheng Zhu, and Dongbin Zhao. Learning and planning multi-agent tasks via an MoE-based world model. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1364. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3a2d96d2eb2902043c2db705ca03e9a2-Abstract-Conference.html.

Appendix for MIXTUREPRIOR

CONTENTS

A Mathematical Details and Proofs **16**

A.1 Predictive Target of Posterior Matching 16
A.2 Conditional Updates and Cancellation 16
A.3 Proof of Proposition 1 17
A.4 Proof of Proposition 2 18
A.5 Split–Gate Local Descent: Proof and Boundaries 18
A.6 Exact Binary Example and the Actual Loss 20
A.7 Smoothing and Free-Nats Clipping 20
A.8 Alternative Interactions and Local Dynamics 22
A.9 Responsibility Gradients 22

B Implementation Details and Verification **22**

B.1 Prior Interface and Effective Distribution 22
B.2 Backbone-Specific Configuration 23
B.3 Numerical Verification 23
B.4 Real-Checkpoint Split–Gate Diagnostic 24

C Additional Routing Diagnostics **24**

C.1 Frozen-Component Routing Controls 24
C.2 Gate-Entropy Analysis 25

D Learning Dynamics and Efficiency Diagnostics **26**

E Complete Control Results **27**

E.1 Fixed Configurations 27
E.2 Per-Task Selected Aggregate 28
E.3 Matched Component-Count Comparison 28
E.4 Structural Single-Head Controls 28
E.5 Complete Task-Level Configuration Results 28

A Mathematical Details and Proofs

A.1 Predictive Target of Posterior Matching

For latent factor i, let $q_i(j \mid \mathbf{x}, \mathbf{o}')$ denote the posterior probability after observing the next observation $\mathbf{o}'$, and define

$$\bar{q}_i(j \mid \mathbf{x}) = \mathbb{E}_{\mathbf{o}'|\mathbf{x}} \left[q_i(j \mid \mathbf{x}, \mathbf{o}') \right]. \tag{11}$$

Consider the expected posterior-to-prior KL

$$\mathcal{L}_{\mathrm{KL}}(p) = \mathbb{E}_{\mathbf{x},\mathbf{o}'} \sum_i D_{\mathrm{KL}} \left(q_i(\cdot \mid \mathbf{x}, \mathbf{o}') \,\|\, p_i(\cdot \mid \mathbf{x}) \right). \tag{12}$$

For one factor,

$$\begin{aligned} &\mathbb{E}_{\mathbf{o}'|\mathbf{x}} D_{\mathrm{KL}} \left(q_i(\cdot \mid \mathbf{x}, \mathbf{o}') \| p_i(\cdot \mid \mathbf{x}) \right) \\ &= \mathbb{E}_{\mathbf{o}'|\mathbf{x}} \sum_j q_i(j \mid \mathbf{x}, \mathbf{o}') \log \frac{q_i(j \mid \mathbf{x}, \mathbf{o}')}{p_i(j \mid \mathbf{x})}. \end{aligned} \tag{13}$$

Insert $\bar{q}_i(j \mid \mathbf{x})$ into the logarithm:

$$\log \frac{q_i}{p_i} = \log \frac{q_i}{\bar{q}_i} + \log \frac{\bar{q}_i}{p_i}. \tag{14}$$

Therefore

$$\begin{aligned} &\mathbb{E}_{\mathbf{o}'|\mathbf{x}} D_{\mathrm{KL}}(q_i \| p_i) \\ &= \mathbb{E}_{\mathbf{o}'|\mathbf{x}} D_{\mathrm{KL}}(q_i \| \bar{q}_i) + \sum_j \mathbb{E}_{\mathbf{o}'|\mathbf{x}}[q_i(j)] \log \frac{\bar{q}_i(j)}{p_i(j)} \\ &= \mathbb{E}_{\mathbf{o}'|\mathbf{x}} D_{\mathrm{KL}}(q_i \| \bar{q}_i) + D_{\mathrm{KL}}(\bar{q}_i \| p_i). \end{aligned} \tag{15}$$

Summing over factors and contexts gives

$$\mathcal{L}_{\mathrm{KL}}(p) = C_q + \mathbb{E}_{\mathbf{x}} \sum_i D_{\mathrm{KL}} \left(\bar{q}_i(\cdot \mid \mathbf{x}) \| p_i(\cdot \mid \mathbf{x}) \right), \tag{16}$$

where

$$C_q = \mathbb{E}_{\mathbf{x},\mathbf{o}'} \sum_i D_{\mathrm{KL}} \left(q_i(\cdot \mid \mathbf{x}, \mathbf{o}') \| \bar{q}_i(\cdot \mid \mathbf{x}) \right) \tag{17}$$

is independent of the prior. Hence minimizing posterior-to-prior KL with fixed posterior targets is equivalent to minimizing the fitting risk $\mathcal{R}(p)$ used in the main text.

A.2 Conditional Updates and Cancellation

Let

$$\mathcal{R}(\mathbf{w}) = \mathbb{E}_{\mathbf{a}} \mathcal{R}_{\mathbf{a}}(\mathbf{w}). \tag{18}$$

Assume that differentiation may be interchanged with expectation. Then

$$\nabla_{\mathbf{w}} \mathcal{R}(\mathbf{w}) = \mathbb{E}_{\mathbf{a}} \left[\nabla_{\mathbf{w}} \mathcal{R}_{\mathbf{a}}(\mathbf{w}) \right]. \tag{19}$$

For any parameter direction $\mathbf{u}$, define

$$C_{\mathbf{a}}(\mathbf{u}) = -\nabla_{\mathbf{w}} \mathcal{R}_{\mathbf{a}}(\mathbf{w})^\top \mathbf{u}. \tag{20}$$

Taylor expansion gives

$$\mathcal{R}_{\mathbf{a}}(\mathbf{w} + \epsilon \mathbf{u}) = \mathcal{R}_{\mathbf{a}}(\mathbf{w}) - \epsilon C_{\mathbf{a}}(\mathbf{u}) + o(\epsilon). \tag{21}$$

Thus $C_{\mathbf{a}} > 0$ means that the update $\epsilon \mathbf{u}$ decreases conditional risk to first order.

At a stationary shared prior $\nabla_{\mathbf{w}} \mathcal{R}(\mathbf{w}) = \mathbf{0}$,

$$\mathbb{E}_{\mathbf{a}}[C_{\mathbf{a}}(\mathbf{u})] = -\mathbb{E}_{\mathbf{a}} \left[\nabla_{\mathbf{w}} \mathcal{R}_{\mathbf{a}}(\mathbf{w}) \right]^\top \mathbf{u} \tag{22}$$

$$= -\nabla_{\mathbf{w}} \mathcal{R}(\mathbf{w})^\top \mathbf{u} = 0. \tag{23}$$

Therefore positive and negative conditional improvements may cancel in the shared objective. The identity does not imply that such cancellation always occurs; it shows that a zero shared update does not require every action-conditional update to vanish.

A.3 Proof of Proposition 1

Let

$$d_i(j \mid \mathbf{x}) = r_i(j \mid \mathbf{x}) - p_i(j \mid \mathbf{x}), \tag{24}$$

so that

$$p_{\epsilon,i} = p_i + \epsilon d_i. \tag{25}$$

Because both p_i and r_i are normalized categorical distributions,

$$\sum_j d_i(j \mid \mathbf{x}) = 0. \tag{26}$$

Assume that there exists $m > 0$ such that

$$p_i(j \mid \mathbf{x}) \geq m, \qquad r_i(j \mid \mathbf{x}) \geq m \tag{27}$$

almost surely. Then every convex interpolation satisfies $p_{\epsilon,i}(j \mid \mathbf{x}) \geq m$.

For fixed $\mathbf{x}$, the relevant KL term is

$$f_{\mathbf{x}}(\epsilon) = \sum_{i,j} \bar{q}_i(j) \log \frac{\bar{q}_i(j)}{p_i(j) + \epsilon d_i(j)}. \tag{28}$$

Its first derivative is

$$f'_{\mathbf{x}}(\epsilon) = -\sum_{i,j} \bar{q}_i(j) \frac{d_i(j)}{p_i(j) + \epsilon d_i(j)}. \tag{29}$$

At $\epsilon = 0$,

$$f'_{\mathbf{x}}(0) = -\sum_{i,j} \frac{\bar{q}_i(j) d_i(j)}{p_i(j)} \tag{30}$$

$$= -\sum_{i,j} \frac{\big(\bar{q}_i(j) - p_i(j)\big) d_i(j)}{p_i(j)}, \tag{31}$$

where the second equality follows from Eq. (26).

Taking conditional expectation given action,

$$-\frac{d}{d\epsilon} \mathcal{R}_{\mathbf{a}}(p_\epsilon)\Big|_{\epsilon=0} = \mathcal{A}_r(\mathbf{a}). \tag{32}$$

Averaging over actions gives

$$\frac{d}{d\epsilon} \mathcal{R}(p_\epsilon)\Big|_{\epsilon=0} = -\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})]. \tag{33}$$

The second derivative is

$$f''_{\mathbf{x}}(\epsilon) = \sum_{i,j} \bar{q}_i(j) \frac{d_i(j)^2}{\big(p_i(j) + \epsilon d_i(j)\big)^2} \geq 0. \tag{34}$$

Hence $\mathcal{R}(p_\epsilon)$ is convex in ϵ.

If

$$\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})] > 0, \tag{35}$$

then $\mathcal{R}'(0) < 0$. By differentiability, a sufficiently small $\epsilon > 0$ satisfies

$$\mathcal{R}(p_\epsilon) < \mathcal{R}(p). \tag{36}$$

Conversely, suppose

$$\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})] \leq 0. \tag{37}$$

Then $\mathcal{R}'(0) \geq 0$. Convexity implies, for every $\epsilon \in [0, 1]$,

$$\mathcal{R}(p_\epsilon) \geq \mathcal{R}(p) + \epsilon \mathcal{R}'(0) \geq \mathcal{R}(p). \tag{38}$$

Thus no positive interpolation amount improves the risk. This proves the proposition.

A.4 Proof of Proposition 2

For an admissible action-dependent weight $\omega(\mathbf{a}) \in [0,1]$, define

$$p^{\omega}_{\epsilon,i} = p_i + \epsilon\omega(\mathbf{a})(r_i - p_i). \tag{39}$$

Repeating the derivative calculation in Appendix A.3 gives

$$\frac{d}{d\epsilon}\mathcal{R}(p^{\omega}_{\epsilon})\Big|_{\epsilon=0} = -\mathbb{E}_{\mathbf{a}}\left[\omega(\mathbf{a})\mathcal{A}_r(\mathbf{a})\right]. \tag{40}$$

Therefore the first-order risk reduction is

$$\mathcal{G}(\omega) = \mathbb{E}_{\mathbf{a}}\left[\omega(\mathbf{a})\mathcal{A}_r(\mathbf{a})\right]. \tag{41}$$

Action-dependent weighting. For each action individually,

$$\sup_{0\le\omega\le 1} \omega\,\mathcal{A}_r(\mathbf{a}) = \big(\mathcal{A}_r(\mathbf{a})\big)_+. \tag{42}$$

Indeed, if $\mathcal{A}_r(\mathbf{a}) > 0$, the optimum is $\omega = 1$; if $\mathcal{A}_r(\mathbf{a}) < 0$, it is $\omega = 0$. Thus the measurable selector

$$\omega^\star(\mathbf{a}) = \mathbf{1}\left\{\mathcal{A}_r(\mathbf{a}) > 0\right\} \tag{43}$$

attains

$$\mathcal{G}_{\text{act}} = \mathbb{E}_{\mathbf{a}}\left[\big(\mathcal{A}_r(\mathbf{a})\big)_+\right]. \tag{44}$$

Fixed weighting. If the same scalar $c \in [0,1]$ must be used for every action, then

$$\mathcal{G}(c) = c\,\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})]. \tag{45}$$

If the average correction value is positive, the optimum is $c = 1$; otherwise the optimum is $c = 0$. Hence

$$\mathcal{G}_{\text{fixed}} = \left(\mathbb{E}_{\mathbf{a}}[\mathcal{A}_r(\mathbf{a})]\right)_+. \tag{46}$$

Since $(v)_+ = (v + |v|)/2$,

$$\begin{aligned}\mathcal{G}_{\text{act}} - \mathcal{G}_{\text{fixed}} &= \frac{1}{2}\left(\mathbb{E}_{\mathbf{a}}|\mathcal{A}_r(\mathbf{a})| - |\mathbb{E}_{\mathbf{a}}\mathcal{A}_r(\mathbf{a})|\right)\\ &\ge 0,\end{aligned} \tag{47}$$

where the inequality follows from $|\mathbb{E}[X]| \le \mathbb{E}[|X|]$.

For the strictness condition, define

$$M_+ = \mathbb{E}[(\mathcal{A}_r)_+], \qquad M_- = \mathbb{E}[(-\mathcal{A}_r)_+]. \tag{48}$$

Then

$$\mathbb{E}[\mathcal{A}_r] = M_+ - M_-, \qquad \mathbb{E}[|\mathcal{A}_r|] = M_+ + M_-, \tag{49}$$

and therefore

$$\mathcal{G}_{\text{act}} - \mathcal{G}_{\text{fixed}} = \min\{M_+, M_-\}. \tag{50}$$

The inequality is strict exactly when both $M_+ > 0$ and $M_- > 0$, i.e., when the candidate correction is useful for a set of actions of positive probability and harmful for another set of positive probability. Since $M_- = \mathbb{E}[(\mathcal{A}_r)_-]$, this is the closed-form gap in Equation (6), which proves the proposition.

A.5 Split–Gate Local Descent: Proof and Boundaries

This section proves Theorem 3 and states its limits. We keep the notation of Section 3: $\mathcal{R}$ is the single-head prior-fitting risk, $\boldsymbol{w}^\star$ is a stationary point of it, and $\mathcal{R}^{\text{mix}}(\alpha,\beta)$ is the risk of the split path in Equation (7). Write $\boldsymbol{g}(\boldsymbol{a}) = \nabla_{\boldsymbol{w}}\mathcal{R}_{\boldsymbol{a}}(\boldsymbol{w}^\star)$ for the action-conditional gradient, so that $\boldsymbol{g}(\boldsymbol{a})^\top \boldsymbol{u} = -C_{\boldsymbol{a}}(\boldsymbol{u})$ and

$$\chi = \mathbb{E}_{\boldsymbol{a}}\left[s(\boldsymbol{a})\boldsymbol{g}(\boldsymbol{a})^\top \boldsymbol{u}\right] = -\mathbb{E}_{\boldsymbol{a}}\left[s(\boldsymbol{a})C_{\boldsymbol{a}}(\boldsymbol{u})\right], \qquad \kappa = \partial_{\alpha\alpha}\mathcal{R}^{\text{mix}}(0,0).$$

The theorem states that, locally around $(\alpha, \beta) = (0, 0)$,

$$\mathcal{R}^{\text{mix}}(\alpha, \beta) = \mathcal{R}(\boldsymbol{w}^\star) + \frac{1}{2} \begin{pmatrix} \alpha & \beta \end{pmatrix} \boldsymbol{H} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + o(\alpha^2 + \beta^2), \qquad \boldsymbol{H} = \begin{pmatrix} \kappa & \chi \\ \chi & 0 \end{pmatrix}, \tag{51}$$

and that, if $\chi \neq 0$, $\boldsymbol{H}$ has the negative eigenvalue

$$\lambda_- = \frac{\kappa - \sqrt{\kappa^2 + 4\chi^2}}{2} = -\gamma < 0, \qquad \gamma = \frac{\sqrt{\kappa^2 + 4\chi^2} - \kappa}{2} > 0. \tag{52}$$

The result applies at any stationary point of the single head, including heads that take the action as input. It concerns the restricted problem in which the component output blocks and the gate output are trainable and the feature trunks are fixed, and it implies neither a global convergence rate nor a policy improvement (Appendix A.8). The following corollary shows that an affine gate can realize a score with $\chi \neq 0$.

Corollary 4 (Realizable gate-feature direction)**.** *Let*

$$\boldsymbol{\psi}_c = \boldsymbol{\psi} - \mathbb{E}\boldsymbol{\psi}, \qquad \boldsymbol{C} = \mathbb{E}\left[\boldsymbol{\psi}_c \nabla_{\boldsymbol{w}} \ell^\top\right].$$

If $\|\boldsymbol{C}\|_2 > 0$ and the affine gate output can realize $s(\boldsymbol{a}) = \boldsymbol{c}^\top \boldsymbol{\psi}_c(\boldsymbol{a})$, then there exist unit vectors $\boldsymbol{c}$ and $\boldsymbol{u}$ such that

$$|\chi| = |\boldsymbol{c}^\top \boldsymbol{C} \boldsymbol{u}| = \|\boldsymbol{C}\|_2 > 0.$$

Therefore Theorem 3 admits a strict local descent direction realizable by the gate.

A.5.1 Proof of Theorem 3

For a fixed sample and factor, abbreviate $\boldsymbol{p}_0 = \boldsymbol{p}_{\boldsymbol{w}^\star, i}$, $\boldsymbol{d} = D_{\boldsymbol{w}} \boldsymbol{p}_{\boldsymbol{w}^\star, i}[\boldsymbol{u}]$, and $\boldsymbol{e} = D^2_{\boldsymbol{w}} \boldsymbol{p}_{\boldsymbol{w}^\star, i}[\boldsymbol{u}, \boldsymbol{u}]$. The two components have expansions

$$\boldsymbol{p}_{\boldsymbol{w}^\star \pm \alpha \boldsymbol{u}, i} = \boldsymbol{p}_0 \pm \alpha \boldsymbol{d} + \tfrac{1}{2} \alpha^2 \boldsymbol{e} + o(\alpha^2).$$

Writing their probability mixture as their half-sum plus $\frac{1}{2} \tanh(\beta s)$ times their difference gives, at the origin,

$$\begin{aligned} \partial_\alpha \boldsymbol{p}_i^{\text{mix}} &= \partial_\beta \boldsymbol{p}_i^{\text{mix}} = \boldsymbol{0}, & \partial_{\alpha\alpha} \boldsymbol{p}_i^{\text{mix}} &= \boldsymbol{e}, \\ \partial_{\alpha\beta} \boldsymbol{p}_i^{\text{mix}} &= s(\boldsymbol{a}) \boldsymbol{d}, & \partial_{\beta\beta} \boldsymbol{p}_i^{\text{mix}} &= \boldsymbol{0}. \end{aligned} \tag{53}$$

The vanishing first derivatives remove all terms involving the loss Hessian with respect to probabilities from the second-order chain rule. For cross-entropy, $\partial \ell / \partial p_i(j) = -q_i(j)/p_i(j)$. Consequently,

$$\partial_{\alpha\alpha} \mathcal{R}^{\text{mix}} = \kappa, \qquad \partial_{\alpha\beta} \mathcal{R}^{\text{mix}} = \mathbb{E}\left[s(\boldsymbol{a}) \nabla_{\boldsymbol{w}} \ell^\top \boldsymbol{u}\right] = \mathbb{E}_{\boldsymbol{a}}[s(\boldsymbol{a}) \boldsymbol{g}(\boldsymbol{a})^\top \boldsymbol{u}] = \chi,$$

and the remaining pure gate derivative is zero. A finite fixed dataset, finite gate scores, and smooth positive probabilities in a neighborhood suffice for the differentiation steps. For a population risk, assume an integrable envelope for the required first and second derivatives throughout that neighborhood. The same proof works for any twice continuously differentiable probability loss: replace $-q_i(j)/p_i(j)$ in κ by its probability derivative and define $\boldsymbol{g}$ using that loss.

Taylor's theorem gives (51). The characteristic polynomial of $\boldsymbol{H}$ is $\lambda^2 - \kappa\lambda - \chi^2$. If $\chi \neq 0$, its roots have opposite signs, and the negative root is (52). For a unit eigenvector $\boldsymbol{v}_-$, Taylor expansion along $(\alpha, \beta) = t\boldsymbol{v}_-$ yields

$$\mathcal{R}^{\text{mix}}(t\boldsymbol{v}_-) - \mathcal{R}(\boldsymbol{w}^\star) = -\tfrac{1}{2} \gamma t^2 + o(t^2).$$

Choose a neighborhood where the remainder has magnitude at most $\gamma t^2/4$ to obtain the claimed bound.

Stationarity in the expanded head problem. At the duplicated head with a uniform gate, the gradient with respect to either independent component block is $\frac{1}{2} \nabla \mathcal{R}(\boldsymbol{w}^\star)$. All gate gradients vanish because redistributing mass among identical components preserves predictions. Thus a stationary single head embeds as a stationary point when the independent component output blocks and the affine gate output are trainable, with both feature trunks frozen. The realizable path in (7) has negative second variation there, so the Hessian of that expanded problem has a negative direction. This establishes a strict saddle of this restricted problem. It does not assert stationarity with respect to an unfrozen backbone, encoder, shared component trunk, or other training objectives. If $\boldsymbol{w}^\star$ is not stationary, the antisymmetric slice still has zero first derivatives and the same expansion, but the common component direction can already descend.

Splitting curvature is not single-head curvature. Let $\boldsymbol{H}_{\text{single}} = \nabla^2 \mathcal{R}(\boldsymbol{w}^\star)$. Differentiating cross-entropy along $\boldsymbol{w}^\star + t\boldsymbol{u}$ gives

$$\kappa = \boldsymbol{u}^\top \boldsymbol{H}_{\text{single}} \boldsymbol{u} - \mathbb{E} \sum_{i,j} q_i(j) \left(\boldsymbol{u}^\top \nabla_{\boldsymbol{w}} \log p_{\boldsymbol{w}^\star, i}(j) \right)^2 . \tag{54}$$

Even if $\boldsymbol{H}_{\text{single}}$ is positive semidefinite, κ need not be positive. This is why uniform splitting can lower loss without the action-dependent interaction. Without routing, s is constant, so $\chi = s\, \mathbb{E}_{\boldsymbol{a}}[\boldsymbol{g}(\boldsymbol{a})^\top \boldsymbol{u}] = 0$ at a stationary point and only κ remains; this is the split-only condition in Table 3. The action mechanism is the additional cross term, not a claim that all other splitting directions fail.

A.5.2 Proof of Corollary 4

By conditioning on $\boldsymbol{a}$ and substituting $s = \boldsymbol{c}^\top \boldsymbol{\psi}_c$,

$$\chi = \boldsymbol{c}^\top \mathbb{E}[\boldsymbol{\psi}_c \nabla_{\boldsymbol{w}} \ell^\top] \boldsymbol{u} = \boldsymbol{c}^\top \boldsymbol{C} \boldsymbol{u}.$$

The variational characterization of the largest singular value yields $\max_{\|\boldsymbol{c}\|_2 = \|\boldsymbol{u}\|_2 = 1} |\boldsymbol{c}^\top \boldsymbol{C} \boldsymbol{u}| = \|\boldsymbol{C}\|_2$. The affine gate output realizes $\pm\tau\beta \boldsymbol{c}^\top (\boldsymbol{\psi} - \mathbb{E}\boldsymbol{\psi})$ by scaling its weights and adjusting its bias. Thus the direction is an actual parameter path in the supplied gate, with its hidden layer fixed. For finite actions, the conditional-score construction in the main text follows by direct substitution, subject to representability. At a stationary head, replacing a score by its centered version leaves χ unchanged, because $\mathbb{E}\nabla\ell = \boldsymbol{0}$. Away from stationarity, centering excludes a contribution from the common gradient; this is useful in checkpoint diagnostics.

A.6 Exact Binary Example and the Actual Loss

Let $\boldsymbol{f} = (1, \xi, a)^\top$ and $q = 1/2 + \delta\xi a$. For the single head, $\nabla_{\boldsymbol{w}} \ell = (\sigma(\boldsymbol{w}^\top \boldsymbol{f}) - q)\boldsymbol{f}$. At $\boldsymbol{w} = \boldsymbol{0}$, averaging over the four equiprobable contexts gives a zero gradient, while conditioning on a gives $(0, -\delta a, 0)^\top$. For every finite $\boldsymbol{w}$, the Hessian is

$$\mathbb{E}\left[\sigma(\boldsymbol{w}^\top \boldsymbol{f})(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{f}))\boldsymbol{f}\boldsymbol{f}^\top\right].$$

It is positive definite because the four design rows span $\mathbb{R}^3$ and all weights are positive. The loss is strictly convex, and its stationary point is the unique global minimum. At zero the Hessian is $\boldsymbol{I}_3/4$ and every prediction is $1/2$.

Using $\sigma(r) = (1 + \tanh(r/2))/2$ and the oddness of $\tanh$, the routed two-component prediction is

$$p_{\text{mix}}(1 \mid \xi, a; u, v) = \frac{1}{2} + \frac{\xi a}{2} \tanh\left(\frac{u}{2}\right) \tanh\left(\frac{v}{2}\right). \tag{55}$$

For $u = v = 2\,\operatorname{artanh}\sqrt{2\delta}$, the prediction equals q at all four contexts. The optimal cross-entropy is the binary entropy of $1/2 + \delta$. At $\delta = 0.3$, this is 0.5004024235, compared with $\log 2 = 0.6931471806$ for the single head, a difference of 0.1927447570 nats. Near the origin, $p_{\text{mix}} - 1/2 = \xi a\, uv/8 + O((|u| + |v|)^4)$. The derivative of binary cross-entropy with respect to probability at $1/2$ is $-4\delta\xi a$, giving

$$\mathcal{L}(u, v) = \log 2 - \frac{\delta}{2} uv + O((|u| + |v|)^4), \qquad \nabla^2 \mathcal{L}(0, 0) = \begin{pmatrix} 0 & -\delta/2 \\ -\delta/2 & 0 \end{pmatrix}.$$

The two tied scalar parameters here illustrate a different interaction structure; they do not imply that the actual network has fewer parameters than the native prior.

A.7 Smoothing and Free-Nats Clipping

The categorical wrapper uses $S_\eta(\boldsymbol{p}) = (1 - \eta)\boldsymbol{p} + (\eta/C)\boldsymbol{1}$. Since this map is affine,

$$S_\eta\left(\sum_k \pi_k \boldsymbol{p}_{k,i}\right) = \sum_k \pi_k S_\eta(\boldsymbol{p}_{k,i}). \tag{56}$$

Theorem 3 therefore applies directly to the effective smoothed component probabilities, with the same fixed effective posterior used by the loss. Both $\boldsymbol{g}$ and κ must be computed for that effective

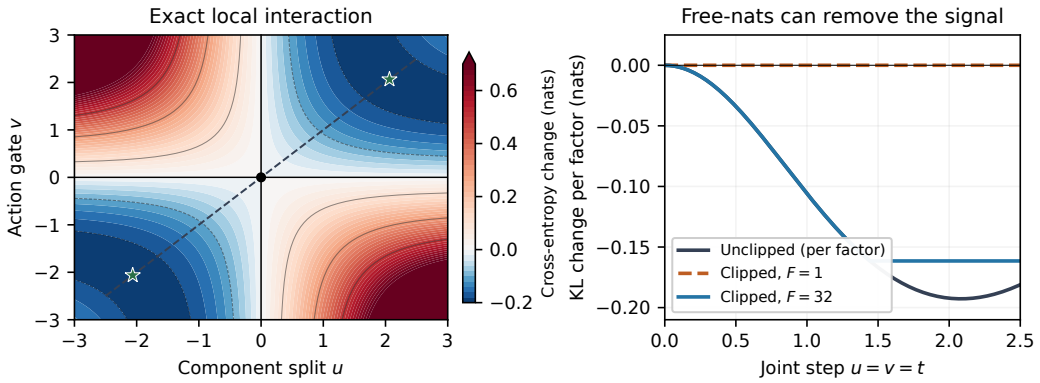

Figure 6: **Exact analytical example; no benchmark training.** Left: cross-entropy minus the optimal single-head cross-entropy on the four binary contexts at $\delta = 0.3$. Both coordinate axes remain at zero loss difference, and the aligned diagonal descends. Right: untruncated and one-nat-clipped KL along $u = v = t$, using the implemented smoothing $\eta = 0.01$. Changes are normalized per factor after clipping the summed KL. The one-factor clipped loss is locally flat; 32 identical binary factors start above the threshold and retain descent. The repeated binary construction illustrates the loss reduction, not the 32-class benchmark distribution.

objective. An unsmoothed nonzero χ does not automatically certify a nonzero χ for a changed loss. In the toy, smoothing multiplies the interaction in (55) by $1 - \eta$; exact fit requires $2\delta/(1 - \eta) < 1$.

Let $D_n(\boldsymbol{w}) = \sum_i D_{\mathrm{KL}}(\boldsymbol{q}_{n,i} \| S_\eta(\boldsymbol{p}_{\boldsymbol{w},n,i}))$ and let the empirical dynamics objective be $N^{-1} \sum_n \max\{D_n, \lambda_{\mathrm{free}}\}$. Clipping is applied to the summed factor KL for each sample, before sample averaging.

Proposition 5 (Local active-set version)**.** *For a fixed finite dataset, suppose the duplicated starting point has a strict margin* $\min_n |D_n(\boldsymbol{w}^\star) - \lambda_{\mathrm{free}}| > 0$. *Put* $m_n = \mathbb{I}\{D_n(\boldsymbol{w}^\star) > \lambda_{\mathrm{free}}\}$. *Then the clipped objective has expansion* (51) *with*

$$\chi_{\mathrm{act}} = \frac{1}{N} \sum_n s(\boldsymbol{a}_n) m_n \nabla_{\boldsymbol{w}} D_n^\top \boldsymbol{u}, \qquad \kappa_{\mathrm{act}} = -\frac{1}{N} \sum_{n,i,j} m_n \frac{q_{n,i}(j)}{p^{\mathrm{eff}}_{n,i}(j)} D^2_{\boldsymbol{w}} p^{\mathrm{eff}}_{n,i}(j)[\boldsymbol{u}, \boldsymbol{u}]. \tag{57}$$

If $\chi_{\mathrm{act}} \neq 0$*, it has a negative-curvature direction. If the single head is stationary for this clipped objective, its duplicate is a strict saddle of the corresponding expanded head problem. If all samples are strictly inactive, the clipped objective is locally constant.*

Proof. Continuity and the finite positive margin give a common neighborhood in which all activity indicators remain fixed. There the objective is $N^{-1} \sum_n [m_n D_n + (1 - m_n)\lambda_{\mathrm{free}}]$. Apply Theorem 3 to the active weighted losses; inactive terms are constant. □

At the threshold, the maximum is not differentiable, so this Hessian argument is not asserted. A population version needs a uniform activity margin almost surely and the derivative envelope; a finite-sample statement does not establish such a population margin. The positive dynamics-loss coefficient simply scales the derivatives.

For the one-factor $\delta = 0.3$ toy, the KL at the single optimum is $0.1927447570 < 1$. Thus, the implemented one-nat clipping makes its entire local loss flat, despite the untruncated cross-entropy saddle. Replicating this target over $F = 32$ independent binary factors gives initial summed KL $6.1678322247 > 1$ at every context. With $\eta = 0.01$, the mixed Hessian entry in (u, v) is $-F\delta(1 - \eta)/2$ on this active neighborhood. Our implementation reproduces this behavior for $K = 2$ and $K = 4$ (Appendix B). For trained checkpoints, we therefore evaluate the effective clipped loss directly (Appendix B.4).

A.8 Alternative Interactions and Local Dynamics

For affine component logits $\boldsymbol{l}_\pm = \boldsymbol{l}_0 \pm \alpha \boldsymbol{l}_{\boldsymbol{u}}$, action-dependent logit mixing gives

$$\boldsymbol{l}_{\text{mix}} = \pi_+ \boldsymbol{l}_+ + \pi_- \boldsymbol{l}_- = \boldsymbol{l}_0 + \alpha \tanh(\beta s(\boldsymbol{a})) \boldsymbol{l}_{\boldsymbol{u}}.$$

At the origin, its pure second derivatives vanish, and its mixed derivative is $s\boldsymbol{l}_{\boldsymbol{u}}$. After softmax and the same smooth loss, the Hessian on this slice is $\left(\begin{smallmatrix} 0 & \chi \\ \chi & 0 \end{smallmatrix}\right)$, where χ uses the corresponding single-head gradient. This is an explicit counterexample to a claim that the interaction mechanism uniquely favors probability mixing. A nonlinear single head able to represent the required interaction is also outside the toy's failing affine class. The AC-Single and PM-Single controls (Appendix E.4) test particular single-head designs and do not rule out every alternative.

Local instability is not a global convergence rate. If gradient descent is restricted to the two slice coordinates $\boldsymbol{r} = (\alpha, \beta)^\top$ with step size $\eta_{\text{gd}} > 0$, its linearization at the origin is

$$\boldsymbol{r}_{t+1} = (\boldsymbol{I}_2 - \eta_{\text{gd}} \boldsymbol{H}) \boldsymbol{r}_t + o(\|\boldsymbol{r}_t\|_2).$$

A perturbation in $\boldsymbol{v}_-$ is multiplied, to first order, by $1 + \eta_{\text{gd}}\gamma > 1$. Exactly zero remains zero. The slice depends on the scaling of component and gate directions, including temperature; its eigenvalue is not numerically the smallest eigenvalue of the full network Hessian. Nor is full-parameter Adam equivalent to gradient descent in these two coordinates. The calculation therefore describes local instability under a specified parameterization, not a speed ratio, global convergence rate, or policy learning guarantee. Section 5.2 and Appendix D separately discuss observable learning-speed evidence.

A.9 Responsibility Gradients

For positive normalized components and gate weights, let $r_{i,k}(j) = \pi_k p_{k,i}(j)/p_i(j)$ and $\rho_k = F^{-1} \sum_{i,j} q_i(j) r_{i,k}(j)$. The scalar $r_{i,k}(j)$ is the responsibility of component k for class j in factor i; $\boldsymbol{\rho}$ averages responsibilities under the observed posterior. For component logits $v_{k,i}(j)$ and pre-temperature gate logits b_k,

$$\frac{\partial \ell}{\partial b_k} = \frac{F}{\tau}(\pi_k - \rho_k), \qquad \frac{\partial \ell}{\partial v_{k,i}(j)} = p_{k,i}(j) \sum_l q_i(l) r_{i,k}(l) - q_i(j) r_{i,k}(j). \tag{58}$$

Indeed $\partial \ell / \partial \log \pi_k = -F\rho_k$ and $\partial \log \pi_l / \partial b_k = (\mathbb{I}\{l = k\} - \pi_k)/\tau$. The component derivative follows from $\partial \log p_{k,i}(l)/\partial v_{k,i}(j) = \mathbb{I}\{l = j\} - p_{k,i}(j)$. For smoothed components, define responsibilities using their effective probabilities; the gate formula remains valid, while the component-logit formula additionally differentiates through smoothing. Clipping multiplies these per-sample gradients by the active indicator away from the threshold. At identical components $r_{i,k}(j) = \pi_k$, giving $\rho_k = \pi_k$ and a zero gate update, as required by the splitting construction.

B Implementation Details and Verification

This appendix records the implementation details needed to reproduce the MixturePrior prior interface and the controlled mechanism checks used in the main text. The emphasis is on the component parameterization, action-only gating, probability-space aggregation, and checkpoint-level verification.

B.1 Prior Interface and Effective Distribution

Gate flooring: exact effective distribution. Put $v_k = \max\{\pi_k, \epsilon\}$ and $Z = \sum_k v_k$. The returned log-sum-exp logits exponentiate to $\sum_k v_k c_{k,i}(j)$, which sums to Z over classes. Categorical normalization therefore produces $\sum_k \tilde{\pi}_k c_{k,i}(j)$ with $\tilde{\pi}_k = v_k/Z$. This establishes the effective weights in Equation (8). Away from the boundaries $\pi_k = \epsilon$, define responsibilities using $\widetilde{\boldsymbol{\pi}}$, and let $\tilde{\rho}_k$ be their posterior average. For pre-temperature gate logits b_k, the exact gate derivative becomes

$$\frac{\partial \mathcal{L}}{\partial b_k} = \frac{F}{\tau} \sum_\ell (\tilde{\pi}_\ell - \tilde{\rho}_\ell) \mathbb{I}\{\pi_\ell > \epsilon\} (\mathbb{I}\{\ell = k\} - \pi_k). \tag{59}$$

Table 4: Implementation of the MIXTUREPRIOR prior interface.

Item	Implementation
Component network	One shared linear–ELU–linear network maps the action-updated feature to all KFC component logits; the components do not have separate feature networks.
Action-only gate	The action passes through one hidden ELU layer and a zero-initialized output layer, followed by a softmax with temperature τ. The same gate weights are used for all F factors.
Probability aggregation	Each component is normalized over classes, and the components are combined by log-sum-exp with the log gate weights; gate probabilities are floored at 10^{-8}.
Factorized distribution	After aggregation, the prior is a product of F conditionally independent one-hot categorical distributions.
Smoothing	The categorical wrapper applies S_η with $\eta = 0.01$ after categorical normalization.
TWISTER width	The default TWISTER configuration uses deterministic width 512 with $F = C = 32$.
Dynamics target (TWISTER)	The posterior is detached for the prior KL; the factor KL is summed and clipped at one nat before averaging over batch and time. Dynamics and representation loss weights are 0.5 and 0.1.
Execution paths (TWISTER)	The same transition function is used for observation processing and imagination and passes the clipped previous action to the gate; after a reset, the previous action is zero.

This follows by differentiating the loss through the normalized positive weights and using $\partial \log v_\ell / \partial b_k = \mathbb{I}\{\pi_\ell > \epsilon\}(\mathbb{I}\{\ell = k\} - \pi_k)/\tau$. When no floor is active, all indicators are one, $\widetilde{\boldsymbol{\pi}} = \boldsymbol{\pi}$, and the differences sum to zero, reducing the expression to (58). The local splitting construction starts from uniform positive gates. On a finite dataset, its gate floor stays inactive for sufficiently small perturbations.

Smoothing and clipping. Appendix A.7 shows that clipping can remove an otherwise valid descent direction. The checkpoint diagnostic in Appendix B.4 therefore reports the effective (smoothed and clipped) dynamics loss together with the unclipped KL.

Single-head nesting and parameter count. Duplicating one component distribution in every branch recovers it exactly for every normalized gate, including floored gates, so the $K = 1$ setting is nested in the mixture. This nesting holds for the shared component architecture, not for native priors of a different width. For TWISTER ($D_h = 512$, $F = C = 32$, and 256 hidden units in the component network and the gate), the analytic parameter count of the $K = 4$ prior is $1{,}185{,}284 + 256D_a$ for action dimension D_a; the $K = 1$ component network has 394,496 parameters without the gate, and the native affine projection with bias has $(512 + 1) \times 1024 = 525{,}312$. The $K = 1$ versus $K = 4$ comparison in Section 5.2 therefore also changes the parameter count; Appendix E.4 reports a parameter-matched single-head control.

B.2 BACKBONE-SPECIFIC CONFIGURATION

DreamerV3 configuration. For DreamerV3 with $K = 4$ and $\tau = 1$ on Atari-100k, the stochastic state uses $F = 32$ factors with $C = 64$ categorical classes, the deterministic state has width 8192, the gate uses a 256-unit hidden layer, the free-nats threshold is 1.0, and categorical smoothing uses $\eta = 0.01$. The prior has four separate component output blocks and an action-gating network.

B.3 NUMERICAL VERIFICATION

The script `verification/check_action_gradient_theory.py` runs 42 float64 autograd and finite-difference checks of the Hessian, invariances, descent directions, exact toy example, gate-feature criterion, and logit-mixing counterexample. It uses the same prior module as our experiments and covers $K \in \{2, 4\}$, $\tau \in \{0.5, 1, 2\}$, smoothing, and active and inactive free-nats cases; its JSON report lists tolerances and errors. These checks use controlled parameters rather than trained checkpoints.

B.4 Real-Checkpoint Split–Gate Diagnostic

This diagnostic tests Theorem 3 on a trained DreamerV3 Battle Zone checkpoint with a single transition component. We hold the saved posterior targets and deterministic features fixed and duplicate the trained $K = 1$ prior into four identical component heads.

Clone validation. Before perturbation, the duplicated mixture reproduces the original single-head prediction to numerical precision. The maximum probability difference is 2.92×10^{-6}, the mean probability difference is 1.19×10^{-8}, and the held-out posterior-target KL is unchanged. The duplicated model therefore provides a matched starting point for isolating the interaction between component differentiation and action-dependent routing.

Perturbations and predictions. We compare five perturbations of this starting point on held-out transitions over a common step-size grid, and Table 3 in Section 5.4 reports the best held-out change for each. The best step sizes are 3×10^{-4} for aligned splitting and routing and for splitting alone, and 1×10^{-4} for the common-head and gate-only updates; for the shuffled perturbation, no step on the grid lowered the loss. The unclipped KL changes are -0.1293 (aligned), -0.0162 (split only), -0.0034 (common head), and zero for the other two. Theorem 3 predicts their ordering. A common update of the shared head should not help, because the trained head is approximately stationary. A gate-only update cannot change the prediction, because the components are identical. Splitting without routing sets $\chi = 0$ and can help only through the splitting curvature κ. Aligned splitting and routing makes $\chi \neq 0$ and should give the largest decrease, whereas shuffling the correspondence between the split direction and the gate should drive χ back toward zero.

The results follow the predicted ordering. The common-head update increases the effective dynamics loss ($+0.0141$), the gate-only and shuffled perturbations leave it unchanged, splitting alone lowers it by 0.0044, and aligned splitting and routing lowers it by 0.1124, about 25 times more. On this checkpoint, the gain therefore comes from the interaction between splitting and routing rather than from splitting alone.

C Additional Routing Diagnostics

This appendix collects the routing analyses that extend the main-text checkpoint diagnostics. We first report frozen-component controls and bootstrap uncertainty, and then show an additional gate-entropy analysis.

C.1 Frozen-Component Routing Controls

This section gives the details and controls for the frozen-component routing experiment in Section 5.4. For each of ten trained checkpoints, we freeze the backbone, the posterior targets, and the component predictions, fit the best action-independent weight vector (Global), and refit an action-conditioned gate on the same data (Action Gate). Table 5 reports their held-out NLL, with Δ_{refit} denoting Action Gate minus Global. Table 6 adds two controls: the gate stored in the checkpoint (Original) and the same gate with actions shuffled (Shuffled), with

$$\Delta_{\text{orig}} = \mathcal{L}_{\text{original gate}} - \mathcal{L}_{\text{fitted global}}, \tag{60}$$

so that $\Delta_{\text{orig}} < 0$ indicates that the original end-to-end learned gate outperforms the optimized fixed mixture. It also reports 95% bootstrap intervals for Δ_{refit} from 1,000 resamples.

Table 5: **Action routing versus optimized fixed mixing.** The component bank is frozen; lower held-out NLL is better. Negative Δ favors the refitted action gate. Bold marks a 95% bootstrap interval for Δ entirely below zero; the Kangaroo interval lies entirely above zero (Table 6).

Benchmark	Task	Global	Action Gate	Δ
Atari	Kangaroo	117.187	117.824	+0.636
Atari	Frostbite	132.233	130.879	**-1.353**
Atari	Up N Down	151.103	151.145	+0.042
Atari	Gopher	157.753	157.606	**-0.147**
Atari	Amidar	194.195	194.145	-0.050
DMC	Walker Walk	136.394	136.012	**-0.382**
DMC	Cartpole Swingup Sparse	75.540	75.540	**-0.0001**
DMC	Reacher Hard	75.499	75.499	+0.0004
DMC	Cartpole Swingup	61.798	61.788	**-0.010**
DMC	Cartpole Balance Sparse	69.919	69.920	+0.0007

Table 6: **Additional controls for the frozen-component routing diagnostic.** Original denotes the action-conditioned gate stored in the trained checkpoint, while Shuffled breaks the correspondence between actions and the original gate outputs. Δ_{orig} is Original minus the fitted-global result reported in Table 5; negative values favor the original action-conditioned gate. The final column reports the 95% bootstrap interval for Δ_{refit} from Table 5.

Benchmark	Task	Original Gate	Shuffled Gate	Δ_{orig}	95% CI for Δ_{refit}
Atari	Kangaroo	116.879	116.883	-0.308	$[+0.575, +0.769]$
Atari	Frostbite	135.985	135.987	+3.752	$[-2.116, -0.999]$
Atari	Up N Down	163.698	163.700	+12.596	$[-0.020, +0.107]$
Atari	Gopher	160.738	160.725	+2.985	$[-0.204, -0.075]$
Atari	Amidar	205.997	206.011	+11.802	$[-0.098, +0.796]$
DMC	Walker Walk	139.973	139.972	+3.579	$[-0.445, -0.337]$
DMC	Cartpole Swingup Sparse	77.660	77.660	+2.119	$[-0.0003, -0.0001]$
DMC	Reacher Hard	77.273	77.273	+1.774	$[-0.0019, +0.0028]$
DMC	Cartpole Swingup	62.638	62.638	+0.840	$[-0.014, -0.007]$
DMC	Cartpole Balance Sparse	70.636	70.635	+0.717	$[-0.0009, +0.0023]$

The refitted gate lowers held-out NLL below the fitted global mixture on six of the ten checkpoints, and its bootstrap interval lies entirely below zero on five: Frostbite, Gopher, Walker Walk, Cartpole Swingup Sparse, and Cartpole Swingup. Kangaroo is the only checkpoint whose interval lies entirely above zero; there the refitted gate is also worse than the original gate, so the refit does not always recover the best available gate.

The original gates exploit these opportunities much less. They outperform the fitted global mixture only on Kangaroo, and on all ten checkpoints their held-out NLL changes by at most 0.014 when actions are shuffled. Even on Kangaroo, the shuffled gate performs almost identically (116.883 versus 116.879), so its advantage over the global mixture does not come from action dependence.

Together, these controls separate two questions. The refitted gate tests whether the frozen components contain corrections that action-dependent weighting can exploit, which holds on half of the checkpoints across both benchmarks. The original gate tests whether end-to-end training already exploits them, which it does not on these checkpoints.

C.2 Gate-Entropy Analysis

Figure 7 tracks the normalized entropy of the gate weights during training on Cartpole Swingup. The gate moves away from its uniform initialization (entropy 1) within the first 10K interactions, and it concentrates more at lower temperature and, for $K = 2$, at times almost onto a single component. Entropy measures how concentrated the weights are, not whether they depend on the action; together

with the shuffle control above, it suggests that the learned gates mainly reweight the components in the same way for all actions.

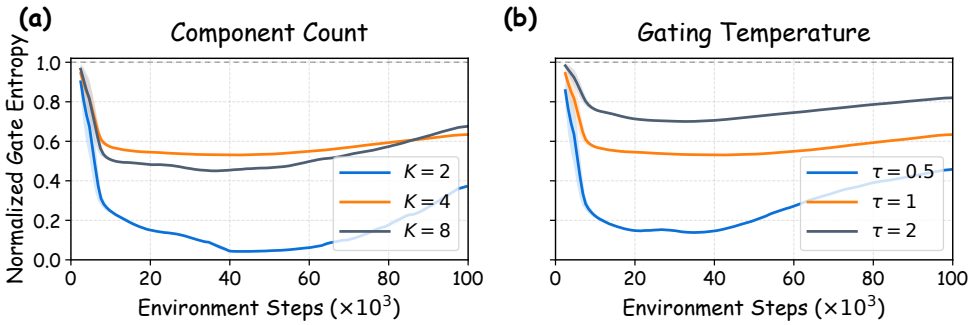

Figure 7: **Gate entropy on Cartpole Swingup.** Normalized gate entropy across component counts at $\tau = 1$ (left) and temperatures at $K = 4$ (right). Entropy records weight concentration and does not identify a component's predictive role.

D Learning Dynamics and Efficiency Diagnostics

We use saved training logs to provide exploratory diagnostics of control-learning dynamics and update cost for the representative runs in Figure 3, comparing MixturePrior with the backbone's native prior. Only one training seed is available for each method–task pair, so these measurements are descriptive and do not support claims of statistical significance or universal convergence speed.

Return versus interactions. For a task and fixed method/configuration, let R_ℓ denote the recorded return at interaction count t_ℓ. We summarize the learning trajectory using the interval-normalized trapezoidal area

$$\mathrm{AUC}_R = \frac{1}{B - t_0} \sum_{\ell=1}^{m} (t_\ell - t_{\ell-1}) \frac{R_\ell + R_{\ell-1}}{2}, \tag{61}$$

and the first recorded crossing of a return threshold ζ,

$$T_\zeta = \min\{t_\ell : R_\ell \geq \zeta\}. \tag{62}$$

The thresholds are chosen post hoc for this exploratory diagnostic: $\zeta = 1000$ for Alien and $\zeta = 800$ for Cartpole Swingup. For DreamerV3, logged Atari and DMC steps are converted to the interaction convention used in the corresponding learning curves by dividing the logged step by four. Runs that do not cross the threshold are reported as right-censored rather than assigning the final evaluation step as a crossing time.

For Cartpole Swingup, AUC_R is computed over the common interval from approximately $4{,}000.25$ to $96{,}096$ interactions. For Alien, the reported values approximately correspond to the common interval from $2{,}100$ to $100{,}000$ interactions. Because the reported values depend slightly on the exact interval endpoints, in particular for DreamerV3 MixturePrior on Alien, we treat all AUC_R values as exploratory summary statistics.

Prediction losses. The saved logs also provide final image-prediction and reward-prediction losses. These are endpoint training losses rather than measurements on a common held-out validation set. We therefore report them only as descriptive fitting diagnostics and do not use them to claim faster predictive convergence.

Update time. For TWISTER, milliseconds per update are estimated from TensorBoard wall-clock time and logged training-step counts. For DreamerV3, milliseconds per update are approximated from `fps/train`, using batch size 16 and sequence length 64. Because these quantities are reconstructed from different logging mechanisms, timing comparisons are made only within the same backbone.

Table 7: **Exploratory learning-speed diagnostics for TWISTER.** AUC_R summarizes return over the recorded interaction interval, and T_ζ is the first recorded threshold crossing with $\zeta = 1000$ for Alien and $\zeta = 800$ for Cartpole Swingup. Prediction losses are training-log endpoints. The native prior on Cartpole Swingup does not reach $\zeta = 800$ and is therefore reported as right-censored. All measurements use a single training seed.

Task	Method	Seeds	AUC_R	T_ζ	Image loss	Reward loss	ms/update
Alien	Native prior	1	588.40	14,700	1.86366	0.05924	690.07
Alien	MIXTUREPRIOR ($K = 4, \tau = 0.5$)	1	962.78	15,200	2.49656	0.04213	881.76
Cartpole Swingup	Native prior	1	446.14	$> 99{,}800$	0.42063	0.46316	1625.33
Cartpole Swingup	MIXTUREPRIOR ($K = 4, \tau = 0.5$)	1	508.75	84,800	0.40992	0.48532	1421.73

Table 8: **Exploratory learning-speed diagnostics for DreamerV3.** AUC_R summarizes return over the recorded interaction interval, and T_ζ is the first recorded threshold crossing with $\zeta = 1000$ for Alien and $\zeta = 800$ for Cartpole Swingup. Prediction losses are training-log endpoints. Update time is approximated from `fps/train` using batch size 16 and sequence length 64. All measurements use a single training seed.

Task	Method	Seeds	AUC_R	T_ζ	Image loss	Reward loss	ms/update
Alien	Native prior	1	691.56	16,341	1.16225	0.04003	651.87
Alien	MIXTUREPRIOR ($K = 4, \tau = 2$)	1	613.98	29,706	1.18229	0.05161	889.16
Cartpole Swingup	Native prior	1	565.35	64,061	0.12054	0.54603	662.52
Cartpole Swingup	MIXTUREPRIOR ($K = 4, \tau = 1$)	1	742.73	24,022	0.15116	0.56982	645.32

Observed learning dynamics. The control-learning effects are heterogeneous across tasks and backbones. For TWISTER on Alien, MIXTUREPRIOR achieves a substantially larger return-curve area than the native prior (962.78 versus 588.40), while the two runs reach the exploratory return threshold at similar interaction counts (15,200 versus 14,700). On Cartpole Swingup, MIXTUREPRIOR increases AUC_R from 446.14 to 508.75 and reaches $\zeta = 800$ at 84,800 interactions, whereas the native prior does not cross this threshold within the available training budget.

For DreamerV3, MIXTUREPRIOR improves the Cartpole Swingup AUC_R from 565.35 to 742.73 and reaches the exploratory threshold earlier (24,022 versus 64,061 interactions). Alien shows the opposite pattern: the native prior has both a larger reported AUC_R and an earlier threshold crossing than the selected MIXTUREPRIOR configuration. The available runs therefore show task-dependent differences in control-learning dynamics rather than a universal acceleration.

The endpoint image and reward losses likewise do not show a uniform advantage for either method. Because these quantities are training-log endpoints rather than held-out predictive losses, they do not establish faster world-model fitting. Update cost is also task and backbone dependent: MIXTUREPRIOR is more expensive per update on Alien for both backbones, while on Cartpole Swingup, its estimated update time is lower than the native prior for both TWISTER and DreamerV3.

E COMPLETE CONTROL RESULTS

This appendix collects the aggregate configuration results, matched component-count comparison, structural controls, and complete task-level scores underlying the control experiments in Section 5.2. The per-task tables are placed last.

E.1 FIXED CONFIGURATIONS

Performance depends substantially on the configuration, and increasing K or sharpening the gate does not monotonically improve it. The best fixed setting is $(4, 1)$ on Atari-100k for both backbones, $(4, 2)$ for TWISTER on DMC, and $(8, 1)$ for DreamerV3 on DMC. Table 1 therefore reports a per-task selection over the settings, and Table 2 the matched $(1, 1)$ versus $(4, 1)$ comparison. For TWISTER on Atari-100k, the fixed-setting means use the runs at the displayed settings, not the five runs marked † in Table 13.

Table 9: Mean performance for all six fixed mixture-prior configurations. Atari reports HNS and DMC reports episode return.

Backbone	Benchmark	$(4, 1)$	$(1, 1)$	$(2, 1)$	$(8, 1)$	$(4, .5)$	$(4, 2)$
TWISTER	Atari	122.76	111.86	106.00	99.70	114.53	99.11
TWISTER	DMC	770.37	718.41	788.70	709.04	774.46	795.80
DreamerV3	Atari	125.62	80.46	88.48	81.41	108.01	75.33
DreamerV3	DMC	815.13	803.89	810.68	824.93	779.72	773.26

Table 10: Per-task selected aggregates: the best of the six settings for each task, compared with the published backbone scores (Reference). Mean gain is the relative change in the aggregate mean, and W/T/L counts the tasks on which the selected score is above, equal to, or below the reference. †Includes five runs at temperatures outside the six settings (Table 13).

		Mean		Mean gain	Median		
Backbone	Suite	Reference	Selected	(%)	Reference	Selected	W/T/L
TWISTER	Atari	162.09	168.95†	4.23†	77.26	91.26†	25/0/1†
TWISTER	DMC	801.80	844.16	5.28	907.60	936.12	17/1/2
DreamerV3	Atari	112.40	153.19	36.29	48.66	60.72	20/1/5
DreamerV3	DMC	786.05	899.13	14.39	861.50	976.80	20/0/0

E.2 Per-Task Selected Aggregate

Per-task selection uses the evaluation runs themselves, so these aggregates are optimistic summaries of what the six settings can reach rather than the result of a fixed method. Table 10 repeats Table 1 for reference.

E.3 Matched Component-Count Comparison

Table 2 in Section 5 reports the matched comparison: at $\tau = 1$, the $K = 4$ mean exceeds the $K = 1$ mean in all four backbone–benchmark settings.

Because both settings share the prior interface and training pipeline, this comparison isolates the number of components more closely than a comparison with published scores. It does not separate the structure from capacity, because $K = 4$ has more parameters than $K = 1$ (Appendix B), and $K = 1$ retains the component network, so it is not identical to the native prior. Per task, $K = 4$ is better than $K = 1$ on 11 of the 20 DMC tasks for TWISTER and on 8 of the 20 for DreamerV3.

E.4 Structural Single-Head Controls

We report two single-head controls on TWISTER. AC-Single is an action-conditioned single-head prior; it reaches a mean HNS of 100.07 on Atari-100k and a mean return of 738.04 on DMC (Table 11), below $K = 4$ on both benchmarks (122.76 and 770.37) and below the published TWISTER scores. PM-Single is a single-head prior whose parameter count is matched to the $(K, \tau) = (4, 1)$ mixture. On six Atari games (Table 12), the mixture has the higher mean (183.84 versus 158.87) but is better on only three games, and it is slightly below the published TWISTER mean on these games. These controls test particular single-head designs and do not establish an advantage independent of capacity; broader capacity-matched controls are left for future work.

E.5 Complete Task-Level Configuration Results

For completeness, Tables 13–16 report the published backbone score (Official) and the scores of all six settings

$$(K, \tau) \in \{(4, 1), (1, 1), (2, 1), (8, 1), (4, 0.5), (4, 2)\}$$

for every task. Bold marks the best MixturePrior score per task, which enters Table 1. For five TWISTER Atari games, the selected run used a temperature outside the six settings; its score is

Table 11: Action-conditioned single-head control (AC-Single) on TWISTER. Reference is the published TWISTER score.

Suite	Metric	Reference	AC-Single
Atari	Mean HNS	162.09	100.07
Atari	Median HNS	77.26	35.17
DMC	Mean return	801.80	738.04
DMC	Median return	907.60	884.97

Table 12: Parameter-matched single-head control (PM-Single) on six TWISTER Atari games (HNS). Reference is the published TWISTER score; MIXTUREPRIOR uses $(K, \tau) = (4, 1)$.

Game	Reference	PM-Single	MIXTUREPRIOR
Assault	95.96	134.92	70.52
Battle Zone	21.71	24.09	26.53
Crazy Climber	283.84	302.99	244.16
Frostbite	5.62	6.76	7.92
Krull	678.31	454.79	747.58
Qbert	22.82	29.65	6.33
Mean	184.71	158.87	183.84

shown in the column of the nearest setting and marked †, and the exact setting is given in the caption. The fixed-setting means in Tables 9 and 2 use the runs at the displayed settings, not the † runs.

Table 13: **Complete task-level results for TWISTER on Atari-100k.** Official reports the published TWISTER score. Bold denotes the best MixturePrior score for each task. Dagger-marked entries use nearby exploratory temperatures grouped under the nearest displayed configuration: Hero $(K, \tau) = (1, 0.8)$, Breakout $(4, 1.2)$, Gopher $(4, 0.95)$, Chopper Command $(4, 1.8)$, and Demon Attack $(1, 0.8)$. Fixed-setting means use the runs at the displayed settings instead of the † entries.

Task	Official	$(4, 1)$	$(1, 1)$	$(2, 1)$	$(8, 1)$	$(4, 0.5)$	$(4, 2)$
Alien	970	1101.8	955.2	1030.2	887	**1455.6**	588.5
Amidar	184	196.4	168.24	172.94	101.48	**203.6**	145.4
Seaquest	532	398.2	365	377.6	424.4	396.4	**653.2**
Pong	20	17.48	7.66	**20.19**	17.15	18.93	18.22
Asterix	1306	1114	928.5	981	1185	1099.5	**1308**
Battle Zone	9920	11600	4270	8480	**19570**	8210	8800
Hero	8773	7670	**13466.6**†	7560	7666.75	7657.5	7515.75
Qbert	3197	1005	4111.25	1631	4165.75	**4541**	4421.5
Breakout	35	**35.06**†	15.2	12.65	13.18	9.67	10.9
Boxing	88	82.46	62.32	77.21	72.13	**94.6**	73.31
Frostbite	305	403.2	317.3	359.2	**2663.6**	279.9	2004.8
Krull	8839	**9578.4**	9516.5	9414.1	8923.8	9471.1	7197.1
Road Runner	17832	9404	**17955**	15342	10411	15000	10909
James Bond	573	332.5	**637.5**	418.5	322.5	456.5	610.5
Ms Pacman	2206	1270.4	2116	2039.6	1957.1	**4449**	2068.8
Crazy Climber	81880	71939	67261	71243	**95781**	84525	69156
Gopher	22234	**13252.6**†	1713.8	848	858.6	994.2	3262.8
Chopper Command	910	359	486	684	537	518	**976**†
Freeway	32	**33.51**	0.01	0	0	0.02	0
Up N Down	7068	5308.8	5274	**8267.9**	4171.6	2995.4	4065.1
Bank Heist	942	780.1	**1243.5**	383.8	817.6	366.8	388.5
Demon Attack	289	187.75	**456.1**†	259.85	199.85	195.1	211.35
Private Eye	1608	76.04	**4814.58**	439.05	4417.01	4738.3	4490.63
Kung Fu Master	23442	26541	26863	29914	13802	19673	**29921**
Kangaroo	6016	2510	**6884**	2588	966	2751	3363
Assault	721	588.82	698.52	1079.38	737.91	**1288.31**	580.65

Table 14: **Complete task-level results for TWISTER on DMC.** Official reports the published TWISTER score. Bold denotes the best MixturePrior result among the six configurations.

Task	Official	$(4, 1)$	$(1, 1)$	$(2, 1)$	$(8, 1)$	$(4, 0.5)$	$(4, 2)$
Acrobot Swingup	239.4	60.02	64.03	69.07	72.01	73.78	**138.23**
Ball In Cup Catch	966.8	958	**973.8**	946.2	969.1	968.2	965
Cartpole Balance	997.9	999.46	998.91	**999.48**	999.15	999.14	999.30
Cartpole Balance Sparse	1000	**1000**	998.2	**1000**	**1000**	**1000**	**1000**
Cartpole Swingup	819.2	863.73	832.26	836.86	**874.92**	860.03	840.19
Cartpole Swingup Sparse	735	837.4	27.7	778.2	**859.17**	834.42	827.4
Cheetah Run	694	625.89	754.66	661.30	**780.84**	654.21	671.08
Finger Spin	976.3	675.4	977.9	**984.7**	983.3	685.5	681.1
Finger Turn Easy	923.7	968	779.5	731.1	659.3	951.8	**972.4**
Finger Turn Hard	910.3	957.2	959.6	971.6	768.1	**975.7**	970.8
Hopper Hop	313.9	319.57	303.92	**457.50**	326.19	342.62	279.90
Hopper Stand	932	933.36	931.09	933.80	939.88	940.11	**940.73**
Pendulum Swingup	832.1	825.4	**847.9**	815.7	0	816.2	835.1
Quadruped Run	652.1	550.57	468.41	601.94	478.24	542.64	**868.37**
Quadruped Walk	904.9	651.85	868.56	903.38	510.03	919.59	**931.52**
Reacher Easy	933.1	968.6	870.5	965.1	**979.6**	969.9	974.5
Reacher Hard	565.8	**660.3**	179.4	568.2	381	580.2	387.7
Walker Run	711.2	662.71	590.64	605.36	658.96	448.01	**685.09**
Walker Stand	976.9	974.43	**985.80**	977.25	982.02	976.91	981.60
Walker Walk	951.3	915.58	955.45	**967.25**	958.91	950.34	965.95

Table 15: **Complete task-level results for DreamerV3 on Atari-100k.** Official reports the published DreamerV3 score. Bold denotes the best MixturePrior result among the six configurations.

Task	Official	$(4, 1)$	$(1, 1)$	$(2, 1)$	$(8, 1)$	$(4, 0.5)$	$(4, 2)$
Alien	959	1070	1120	510	**1310**	1000	1060
Amidar	139	**243**	214	199	147	233	190
Asterix	932	600	850	**1150**	500	950	350
Battle Zone	12250	12000	8000	16000	14000	14000	**17000**
Pong	18	-9	**-3**	**-3**	-10	-9	-7
Hero	11161	13365	7665	13345	13355	7645	**13555**
Qbert	3405	775	1275	1000	1200	4675	**7925**
Crazy Climber	97190	**96100**	38100	72100	40900	40300	47500
Ms Pacman	1327	2550	1500	2180	**4140**	1920	3110
Up N Down	7600	**40590**	4010	10600	29800	37380	3610
Assault	706	693	724	672	682	630	**745**
Bank Heist	649	70	**480**	110	100	280	70
Boxing	78	86	**89**	64	86	83	51
Krull	7782	**10330**	7640	7970	5030	8160	3790
Seaquest	618	**1000**	400	920	380	**1000**	420
Chopper Command	420	900	1600	1300	1200	**2500**	1300
Demon Attack	303	50	90	370	40	**1480**	235
James Bond	445	250	150	300	300	500	**650**
Kung Fu Master	21420	8700	25800	21880	13100	26200	**27400**
Frostbite	909	**3180**	290	440	300	450	440
Breakout	31	8	4	9	4	**12**	2
Freeway	0	**0**	**0**	**0**	**0**	**0**	**0**
Gopher	3730	1880	800	**3020**	400	**3020**	920
Kangaroo	4098	5200	600	4400	200	1800	**8000**
Road Runner	15565	**23400**	8900	1100	15500	10500	10300
Private Eye	882	3982	100	4248	**13500**	4211	100

Table 16: **Complete task-level results for DreamerV3 on DMC.** Official reports the published DreamerV3 score. Bold denotes the best MixturePrior result among the six configurations.

Task	Official	$(4, 1)$	$(1, 1)$	$(2, 1)$	$(8, 1)$	$(4, 0.5)$	$(4, 2)$
Acrobot Swingup	229	**524.35**	373.64	272.22	481.94	391.97	360.95

Task	Official	$(4, 1)$	$(1, 1)$	$(2, 1)$	$(8, 1)$	$(4, 0.5)$	$(4, 2)$
Ball In Cup Catch	972	992	**994**	991	954	992	990
Cartpole Balance	993	995.4	**996.74**	995.25	994.74	993.58	992.66
Cartpole Balance Sparse	964	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
Cartpole Swingup	861	854.26	**868.77**	853.15	850.1	847.75	853.49
Cartpole Swingup Sparse	759	**794**	138	0	568	187	0
Cheetah Run	836	814.41	823.19	862.69	**886.06**	723.95	846.46
Finger Spin	589	980	983	983	**994**	980	716
Finger Turn Easy	878	923.7	837.64	**1000**	790	**1000**	943
Finger Turn Hard	904	902.42	902.71	**982**	955	926	962
Hopper Hop	227	294.16	**379.36**	286.37	0	341.51	293.88
Hopper Stand	903	898.02	912.71	919.69	910.85	**919.98**	847.28
Pendulum Swingup	744	**889**	424	635	632	0	0
Quadruped Run	617	844.59	906.85	909.30	823.90	851.55	**916.99**
Quadruped Walk	811	**973.50**	872.06	859.78	921.72	942.38	962.34
Reacher Easy	951	981	968	971	988	962	**1000**
Reacher Hard	862	945	901	967	972	978	**986**
Walker Run	684	632.57	823.23	813.81	817.20	644.72	**899.55**
Walker Stand	976	84.17	**998.11**	947.93	997.72	981.72	984.20
Walker Walk	961	**980.09**	974.89	964.48	961.29	930.27	910.42