Training ML Models with Predictable Failures

Anonymous Authors[1]

Abstract

Estimating how often an ML model will fail at deployment scale is central to pre-deployment safety assessment, but a feasible evaluation set is rarely large enough to observe the failures that matter. Jones et al. (2025) address this by extrapolating from the largest k failure scores in an evaluation set to predict deployment-scale failure rates. We give a finite-k decomposition of this estimator's forecast error and show that it has a built-in bias toward over-prediction in the typical case, which is the safety-favorable direction. This bias is offset when the evaluation set misses a rare high-failure mode that the deployment set contains, leaving the forecast to under-predict at deployment scale. We propose a fine-tuning objective, the *forecastability loss*, that addresses this failure mode. In two proof-of-concept experiments, a language-model password game and an RL gridworld, fine-tuning substantially reduces held-out forecast error while preserving primary-task capability and achieving safety similar to that of supervised baselines.

1. Introduction

The failures that determine whether a machine learning system is safe to release are often rare enough that no evaluation set of practical size is likely to contain a single example. Pre-deployment evaluation runs at a small fraction of deployment scale, so the gap between observed and future failures is built into the evaluation pipeline, and is widest for agents that interact with the world autonomously and without bound. Quantifying that gap, even when the worst failures cannot be directly observed, is a central question in pre-deployment safety assessment (Shevlane et al., 2023; Phuong et al., 2024; Clymer et al., 2024).

Even when no evaluation input is catastrophic, the shape of the visible tail of the failure distribution carries information about how the worst case grows with scale. Jones et al. (2025) turn this into a forecasting procedure using extreme value theory (EVT): under mild conditions, the upper tail of any distribution converges to one of a small set of limiting forms. For the Gumbel form assumed by Jones et al. (2025), a parametric fit on the largest few observed scores predicts how worst-case risk scales with deployment size. The forecast is accurate when the failure tail lies in this regime; we call such tails *well-behaved*. Whether a model's failure tail is well-behaved is a property of the model itself, given a fixed task distribution and risk score.

We show that models can be fine-tuned to have well-behaved failure tails, at minimal or no cost to safety or the primary objective. The key insight is that, when the per-task risk score is differentiable in the model parameters and there are no ties at the relevant order statistics, the forecast error is itself differentiable in the parameters and can serve directly as a fine-tuning loss: we call the resulting objective the *forecastability loss*. We justify this approach using a finite-k decomposition of the forecast error (Section 4) that identifies which parts are model-dependent and therefore trainable. We also describe a partition-randomization scheme that augments the forecasting data and improves generalization, made tractable by a partition-invariant cache (Appendix B).

We instantiate the approach in two settings that exercise it against different model classes and different forms of risk score. The first is a language-model task in which the model is given a single-token password in its system prompt and instructed never to reveal it, with the per-prompt next-token probability of the password as the risk score. The second is a multi-task reinforcement learning (RL) setting: a small gridworld navigation domain in which the policy must reach a goal while avoiding hazardous cells, with per-task expected regret as the risk score.[1] In both settings, the task distribution is a mixture of a benign component and a small high-failure component: adversarially optimized prompts in one case, densely hazardous gridworld layouts in the other. The mixture weights are chosen so that the high-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]To our knowledge, this is the first application of the forecasting framework outside language-model behavior. We discuss adjacent EVT-in-RL work in Section 2.

failure component is usually absent from a small evaluation set (the *fit set* of Section 3) yet present at deployment scale with high probability and roughly three rare-mode tasks in expectation, instantiating the failure regime our theory identifies as dangerous (Section 4).

In both settings, forecastability fine-tuning substantially improves tail forecast accuracy while leaving primary-task performance unchanged or improved. The improving-only gradient mask couples the forecast objective to the underlying task: the only direct path to reduce forecast error on an underpredicted hard deploy task is to reduce its failure risk, so worst-prompt leak probability in the password game and worst-case regret in the gridworld fall with the forecast error. We also compare against two natural baselines, post-hoc affine calibration of the pretrained model and direct supervised fine-tuning on the same task pool, and find that forecastability fine-tuning achieves comparable safety while preserving primary-task capability and improving the forecast more than either baseline.

2. Related work

Evaluation-aware training. Deshmukh et al. (2025) introduced the idea this paper builds on: the model being evaluated can itself be shaped to make the evaluation more accurate. There, a reinforcement learning policy was trained under an extra penalty for how badly an evaluator estimated its returns, steering the policy toward parts of the state space the evaluator could estimate accurately. We apply this idea to a different evaluation target: the tail of a failure distribution rather than the mean of a return distribution. Concurrent work shapes training-time tail behavior in adjacent settings: Wessel et al. (2025) adapt a forecaster's loss so its own tail is better calibrated, and Žilinskas et al. (2026) attach a generalized-Pareto auxiliary head to a time-series transformer.

Importance sampling. Another approach is to sample from a tilted distribution that concentrates on failures, then reweight back to the original. Variants exist for RL policy evaluation (Hanna et al., 2017; Corso et al., 2022) and language model safety (Wu & Hilton, 2025; Dorman et al., 2026), within a broader rare-event-simulation literature (O'Kelly et al., 2018; Uesato et al., 2019; Webb et al., 2019; Bai et al., 2025). The approach has two well-known limitations. The tilted distribution must cover every region where failures occur, and constructing one generally requires the very information IS is meant to provide. It also requires producing many failures during testing, which may be unacceptable when failures carry real cost. We focus on extrapolation (Jones et al., 2025) and leave the IS analogue to future work.

Extreme value theory in ML safety. The extrapolation-based approach has prior use in ML safety. Weng et al. (2018)'s CLEVER framework introduced EVT-style fits for adversarial robustness, and later work (Atienza et al., 2025; Richards & Huser, 2024) extended EVT-based certification and quantile estimation to broader ML problems. Most closely related, the Alignment Research Center's program on rare LM outputs (Wu & Hilton, 2025; Xu, 2024) compares estimators on fixed models; our paper is the training-time complement. EVT has also been used in reinforcement learning, but for tail-risk-aware policy optimization rather than for cross-task deployment-scale forecasting: Somayaji N. S. et al. (2024) fit a generalized Pareto distribution to state-action returns for risk-averse control, Davar et al. (2025) use peaks-over-threshold EVT to estimate cumulative-cost CVaR inside a policy-gradient loop, and Gao et al. (2025) optimize an extreme-quantile cost constraint during safe-RL training. These methods model tails *within* a policy's trajectories or returns; they do not forecast worst-case performance across unseen tasks.

Conformal risk control. Calibration is another alternative to extrapolation. Conformal risk control (Angelopoulos et al., 2025; 2024) gives finite-sample, distribution-free guarantees on monotone risks at the calibration scale, while extreme-quantile extrapolation predicts deployment-scale risk from a far smaller evaluation set under structural tail assumptions. The two paradigms are complementary.

Distributionally robust and risk-sensitive training. Worst-case training is another algorithmic neighbor often confused with forecastability loss, but a different objective. Group DRO (Sagawa et al., 2020), CVaR-RL (Tamar et al., 2015; Chow et al., 2015), tilted ERM (Li et al., 2021), and just-train-twice variants (Liu et al., 2021) all minimize a worst-case or tail-weighted variant of the primary loss; forecastability loss instead asks the risk-score distribution to have a forecastable shape. The two can come apart: a model can have a well-shaped tail without it being small (the LM result of Section 6.1). In the RL setting the improving-only mask couples them when they should agree.

Pre-deployment evaluation and safety cases. Above the algorithmic level, recent frameworks articulate the broader project of pre-deployment safety assessment. Shevlane et al. (2023) argue that capability and alignment evaluations are the primary tool for managing frontier-model risks; Phuong et al. (2024) and the METR autonomy suite (METR, 2024; Rein et al., 2025; Kwa et al., 2025) build operational dangerous-capability evaluations on that framing. Clymer et al. (2024), with the worked example of Buhl et al. (2025), formalize the safety case as the structure into which such evidence is assembled. The structure of real prompt distributions independently motivates our experimental construction: user prompts are long-tailed and multi-topic (Chiang et al., 2024), and red-teaming studies find that harmful

prompts and failing test cases cluster into multiple distinct attack or failure modes (Perez et al., 2022; Shen et al., 2024). Our synthetic mixtures stress-test the extrapolation failure that arises when one such mode is rare in the fit set but expected at deployment scale. Our work targets a missing layer: how to extrapolate per-input risk scores from evaluation scale to deployment scale, and how to train models so the extrapolation is reliable.

3. Background: tail extrapolation and forecast error

This section sets up the central object of the rest of the paper: the forecast error of the *Gumbel-tail method* of Jones et al. (2025). Section 3.1 reviews the method and names its two assumptions; Section 3.2 defines the per-rank forecast error. Section 4 then decomposes that error to identify how violations of the assumptions propagate, and Section 5 fine-tunes the model to shape the trainable components and reduce the error.

3.1. The Gumbel-tail method

Consider a model (for example, an RL policy or a language model) that will be deployed on n tasks drawn from a distribution $\mathcal{D}$. Each task x produces a scalar *risk score* $f(x)$ that captures how poorly the model performs. Pre-deployment, we have access only to a *fit set* $\mathcal{F}$ of $M \ll n$ tasks from $\mathcal{D}$, and some deployment tasks may produce risk scores far above anything seen in $\mathcal{F}$. Our goal is to forecast the worst-case risk score across the n-task deployment from this much smaller sample.

The risk score $f(x)$ can be any scalar quantity. In our LM experiments (Section 6.1) it is the elicitation score $-\log(-\log p_B(x))$ (Jones et al., 2025), where $p_B(x)$ is the probability that the model emits an undesired behavior B on input x. In our RL experiments (Section 6.2) it is the expected regret of the policy on x, calculated exactly via backward value iteration.

Define $Q(n)$ as the risk score at the $1/n$-quantile of $\mathcal{D}$: $\mathbf{P}_{x\sim\mathcal{D}}[f(x) \geq Q(n)] = 1/n$. We use $Q(n)$ as the canonical deployment-scale tail threshold; the realized maximum of an n-task deployment exceeds $Q(n)$ with probability $1-(1-1/n)^n \to 1-e^{-1} \approx 0.632$ as $n \to \infty$, so $Q(n)$ is comparable to but not an upper bound on the realized worst task. The forecasting problem reduces to predicting $Q(n)$ for deployment-scale n from only the M fit-set tasks.

Jones et al. (2025) approach this via extreme value theory. Define the *survival function* $S(\tau) = \mathbf{P}_{x\sim\mathcal{D}}[f(x) > \tau]$ and the *Weibull plotting position* of order statistic i in a sample of size M as the survival-probability estimate $\hat{S}_i = i/(M+1)$. The Gumbel-tail method assumes the visible upper tail's log-survival is well-fit by a line,

$$\log S(\tau) \approx a\tau + b,$$

fits a and b by ordinary least squares of $\log \hat{S}_i$ on the corresponding fit-set scores at the top-k ranks, and combines the fit with $S(Q(n)) = 1/n$ to produce the closed-form prediction $\widehat{Q}(n) = -(\log n + b)/a$. Jones et al. use the empirical $\hat{S}_i = i/M$; Appendix E discusses the choice.

The log-linear extrapolation relies on two sufficient assumptions. The first, **(A1) asymptotically log-linear upper tail**, is structural: $\log S(\tau)$ must converge to approximate linearity in τ as τ increases, so the threshold-exceedance distribution is approximately exponential – a generalized Pareto distribution with shape $\xi = 0$. Tails that approximately satisfy (A1) include Exp and Gamma; tails that do not include Gaussian (whose log-survival is roughly $-\tau^2/2$, despite being in the Gumbel max domain of attraction), lognormal, Pareto, and any distribution with bounded support. The second, **(A2) fit representativeness**, is a sample-coverage condition: the fit set must be large enough to reach a log-linear regime, i.e., the deep tail. Among other examples: if the distribution has a mixture component that is common enough to appear in the N deployment set draws but rare enough that the M fit-set draws are likely to miss it, the line may underpredict deployment failure.

3.2. Forecast error

To analyze the accuracy of the Gumbel-tail method, we compare its predictions against a held-out *deployment set* D of $N \gg M$ tasks drawn from $\mathcal{D}$ (the evaluation construction Jones et al. (2025) also use). Order D from highest score to lowest; the j-th *order statistic* of D at model parameters θ is the score in the j-th position of this ordering, written $Y_\theta^{(j)}$, so $Y_\theta^{(1)}$ is the deploy maximum. The plotting-position survival probability at deploy rank j is $\hat{S}_j = j/(N+1)$, again the Weibull formula. The OLS line, fit on $\mathcal{F}$ at θ, gives a predicted score at log-survival depth y,

$$\widehat{Q}_\theta(y) = -\frac{y + b(\theta)}{a(\theta)},$$

where slope and intercept carry θ-dependence because the top-k fit-side scores depend on θ and they determine the line. Setting $y_j = -\log \hat{S}_j$, the per-rank *forecast error* is

$$\widehat{Q}_\theta(y_j) - Y_\theta^{(j)}.$$

This is the central object of the rest of the paper. Section 4 decomposes it; Section 5 minimizes a weighted sum of squared forecast errors over extrapolated ranks $j \in J$ – those with y_j deeper than any log-survival in the fit-side range.

4. Theory: decomposition of forecast error

What happens to the forecast error when Section 3.1's two assumptions – asymptotic log-linearity and fit representativeness – are violated? We decompose the forecast error to see how each violation propagates, and correspondingly how the error will change if the model is modified to fit the assumptions. The decomposition produces a curvature term, an occupancy term, a rank term, and a higher-order remainder. The rest of this section walks through each, showing where it comes from, when it dominates, and what fine-tuning can shape.

We focus on the deepest extrapolated rank, $j = 1$: the OLS forecast against the realized deploy maximum $Y_\theta^{(1)}$. Write F_θ for the score distribution, $q_\theta(y) = F_\theta^{-1}(1 - e^{-y})$ for its tail-quantile curve, and let $y_M = \log M$ be the fit-side anchor and $r = \log(N/M)$ the deployment ratio in log-survival depth, so the deployment-scale anchor is $y_M + r = \log N$. Appendix G proves that for fixed k,

$$\widehat{Q}_\theta(y_M + r) - Y_\theta^{(1)} = T_\theta + C_\theta + o_p(q'_\theta(y_M)), \qquad (1)$$

where the o_p asymptote is with respect to M. This relies on a smoothness condition: $|q''_\theta(y_M)|/q'_\theta(y_M) \to 0$ as $y_M \to \infty$. When this condition is not satisfied, o_p remains O_p instead – it does not shrink to zero.[2]

When F_θ is a mixture with a rare high-risk component, an additional *occupancy term* G_θ enters the decomposition, non-zero on the event that the rare component is absent from the fit set but present at deployment (Appendix G.5); the full form is then $T_\theta + C_\theta - G_\theta + o_p(q'_\theta(y_M))$.

While every term in the decomposition is technically dependent on the model through $q'_\theta(y_M)$, that is simply a scaling term that can generally be ignored. We can thus split the terms into three groups.

First, the rank term $T_\theta = q'_\theta(y_M)\,\xi$, which does not depend on θ except for the slope: the law of ξ depends only on the top-k count k and the deployment ratio $R = N/M$. Surprisingly, $\mathbb{E}[\xi] > 0$ for essentially every (k, R) a practitioner would pick, including the headline configuration of Jones et al. (2025) ($k = 10$, $R = 100$) and any R from a small constant to 10^4 at $k = 10$ (Figure 1b). Because the model cannot affect $\mathbb{E}[\xi]$, the rank term contributes a default safety-favorable bias.

Second, the curvature term $C_\theta \propto -q''_\theta(y_M)$, which tracks the local second derivative of the tail-quantile curve at the fit-side anchor; it depends on θ and can be non-negligible at finite M, although under the smoothness condition above it is lower order than the rank term as M grows. Its sign tracks the hazard rate of F_θ (the conditional failure rate $f_\theta(\tau)/S_\theta(\tau)$): increasing-hazard tails ($q''_\theta < 0$) reinforce the rank-term overprediction, decreasing-hazard tails ($q''_\theta > 0$) partially cancel it. Both magnitude and sign are properties of the model that fine-tuning can shape directly.

Finally, the higher-order remainder o_p and the occupancy term G_θ; these terms disappear asymptotically with fit size under smoothness, but for violating distributions and finite fit sizes they do *not* shrink, and *do* depend on the model. When smoothness fails, the remainder can be the dominant structural error, either due to the mixture structure described by G_θ, due to an upper bound (showing up as a strong positive o_p), or due to other higher-order behavior.

The decomposition's predictions are testable on real-world data; Appendix I shows analyses of several scores on WildChat-1M conversations: assistant turn length, the log-probability that the next assistant token belongs to a curated set of harmful first-tokens, per-token mean negative log-likelihood, and an external toxicity classifier. The analyses show that real tails are dramatically heterogeneous both between fit sizes and between scores. Different scores and deploy/fit regimes therefore require different corrections, empirically justifying the approach we take below: directly fine-tuning the model to minimize the forecast error.

5. Method: fine-tuning for forecastable tails

Section 4's decomposition identifies the trainable components of forecast error: the curvature term C_θ, the occupancy term G_θ, and the higher-order remainder where structural (A1) violations live. We now define the forecastability loss that targets them and describe the fine-tuning procedure that uses it. Throughout, we follow Jones et al. (2025) in fixing the OLS fit window at $k = 10$ top-k fit-set scores.

5.1. The forecastability objective

The forecastability loss measures how well the OLS line of Section 3 predicts deploy-set order statistics. Each extrapolated rank j corresponds to a different forecast deployment size, namely e^{y_j}, so to get a forecast that is accurate across the deployment-size range up to N, not just at the deepest rank, the loss aggregates squared per-rank forecast errors across the extrapolated ranks $j \in J$:

$$\mathcal{L}_{\text{forecast}}(\mathcal{F}, D; \theta) = \sum_{j \in J} w_j \left(\widehat{Q}_\theta(y_j) - Y_\theta^{(j)}\right)^2.$$

Here J is the set of ranks at which the OLS line is genuinely extrapolating (those with y_j deeper than the fit-side log-survival range). The weights w_j are equivalently a prior over deployment sizes; we use a log-uniform prior, so each order of magnitude of deployment beyond the fit regime contributes equally (Appendix C).

[2] By way of example: smoothness holds for typical Gumbel-domain tails (exponential, lognormal) and fails for Fréchet-domain tails (Pareto) and bounded-support tails (Beta, Uniform).

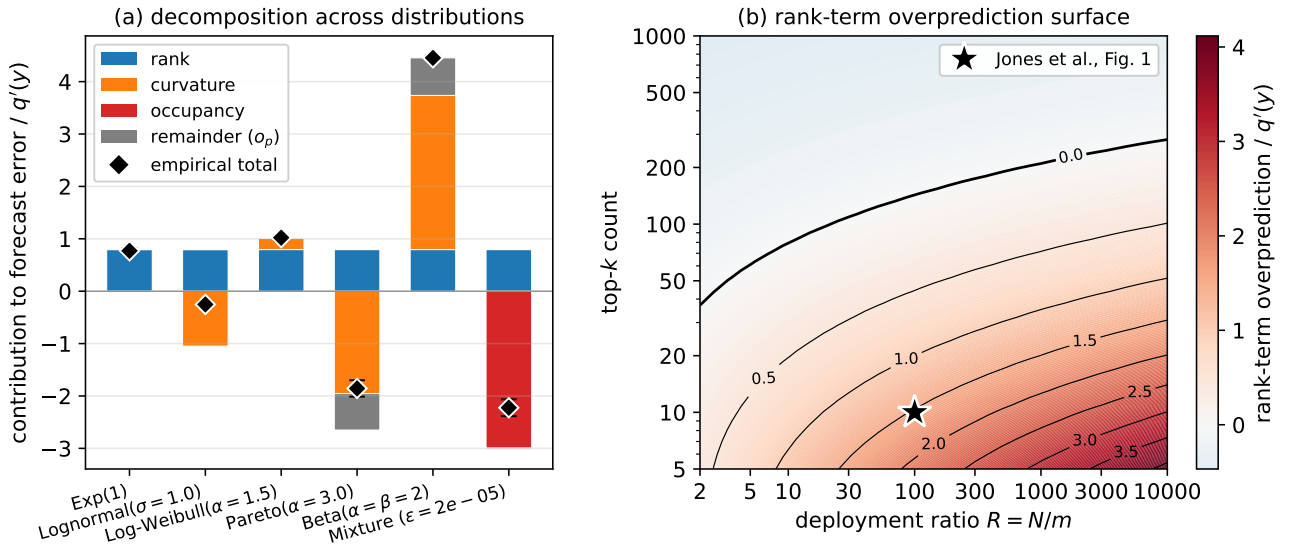

Figure 1. **Decomposition of forecast error on canonical distributions.** (a) Error decomposition across six distributions at $k = 10$, $R = 10$; diamonds are empirical means, and Mixture is an Exp(1) bulk plus a rare shifted-Exp(1) component. The rank bar (blue) is constant, the curvature bar (orange) tracks the sign of $-q''_\theta(y)$ (positive for lighter-than-exponential tails, negative for heavier), the occupancy bar (red) appears only on the mixture, and the gray o_p remainder is near zero on the Gumbel-domain tails but contributes substantially on Pareto and Beta(2, 2), where smoothness fails. (b) Rank-term overprediction $\mathbb{E}[\xi]$ as a function of k and R. The configuration from Figure 1 of Jones et al. (2025) sits firmly in the safety-positive regime; the bias grows with R, shrinks slowly with k, and only flips sign in the upper-left wedge.

Both $\widehat{Q}_\theta(y_j)$ and $Y_\theta^{(j)}$ depend on θ, and we backpropagate through both. By the decomposition of Section 4, minimizing $\mathcal{L}_{\text{forecast}}$ shapes the curvature, occupancy, and higher-order components; in doing so, it also learns to counteract the rank term's positive bias.

5.2. Objective refinements: improvement mask and regularization

We find that the naive forecastability loss often drives models to become predictable by worsening many of their risk scores – unsurprisingly, since the rank term T_θ naturally drives the baseline forecast to overpredict. We therefore restrict which gradient contributions reach the model: we mask out any task whose risk would be driven higher by gradient descent. On the deploy side the criterion is immediate: a deploy rank receives gradient only if the fit line is currently under-predicting it. On the fit side the criterion is the same in spirit, but identifying it requires tracing how each fit-set score moves the OLS line and thus the predictions it produces; we give the closed-form expression in Appendix D.

To avoid models overfitting to the forecastability loss and losing primary-task performance, we add a regularization term $\mathcal{L}_{\text{reg}}(\theta)$ anchoring the model to its pre-fine-tuning behavior. In the LM setting, we use a KL penalty to the frozen pretrained model; in the RL setting, we regularize directly toward the policy's primary objective, regret. Section 6 gives the precise form for each setting.

5.3. Meta-multi-task fine-tuning

Because forecasting itself is already a multi-task problem in our experiments – we wish to forecast worst-case performance across multiple tasks (for example, prompts) – forecastability fine-tuning must be a meta-multi-task problem: we wish to enable accurate forecasting of failures on held-out task distributions where only a fit set $\mathcal{F}$ of tasks is available, but the deployment set D is unknown. We therefore use many task pools U sampled from a distribution over task pools $\mathcal{P}_{\text{meta}}$.

The actual fit set and deployment set $(\mathcal{F}_t, D_t)$ used within each batch form a uniform random partition of U into a fit set of size M and a deployment set of size N. The natural choice is to draw one such partition once and reuse it across steps, but with a fixed partition we observe an asymmetry-overfit failure mode: the optimizer reduces the loss by raising scores on the specific fit-side inputs and leaving the deploy-side inputs unchanged, a memorized per-input direction rather than a tail-shape improvement. We therefore draw a fresh partition $(\mathcal{F}_t, D_t)$ at every step.

Re-partitioning at every step would naively require re-scoring all of $\mathcal{U}$. The loss in fact only depends on the top-of-tail order statistics, so we cache the top-C scoring points of $\mathcal{U}$ – partition-invariant by construction – and re-score just

that subset each step, with a small lazy-fit fallback when the cache misses one of the top-k fit scores. Appendix B gives the algorithm and coverage analysis.

6. Experiments

We present two proof-of-concept experiments for the fine-tuning method. The first is a single-token language-model password game in which the model is exposed to rare adversarial extraction prompts. The second is a small reinforcement learning gridworld in which a policy must avoid hazardous trap layouts. Both task distributions are mixtures of a benign bulk and a rare high-failure mode – a synthetic stand-in for the long-tailed, multi-mode failure structure of real prompt distributions and red-teaming corpora discussed in Section 2 – but our method generalizes to other distributions.

Figure 2 previews the effect of the loss at the level of one held-out password's deploy-set distribution: the log-survival tail, originally erratically curved, becomes straight enough for accurate forecasting. The pretrained model's deploy tail has a structural rare-mode bump that the OLS line, fit only to bulk-mode fit-side scores, falls far short of: predicted worst-rank score $\psi = -0.04$ versus actual $\psi = 2.6$. After fine-tuning, the deploy tail is approximately log-linear in ψ across the entire range, and the OLS line tracks it to its deepest deploy rank ($\psi = 3.81$ predicted versus $\psi = 4.09$ actual).

We compare against two baselines that isolate what fine-tuning contributes. Post-hoc calibration of the pretrained model's forecast (*Cal.*) learns a two-parameter affine correction to the forecast's OLS lines to minimize squared forecast error; the model itself is unchanged, so the capability and safety axes retain the pretrained baseline values by construction. *SFT* performs direct risk minimization on the same per-target pool our method uses for its forecast loss, with the same regularizer. We also report each fine-tuned model with the same calibration applied to its own forecast (*SFT+ cal.*, *Ours+ cal.*), which isolates the forecast-precision gain from calibration on top of the trained model.

6.1. Language model: the password game

The LM experiment is a single-token version of the password game studied for system-prompt robustness (Toyer et al., 2024; Mu et al., 2025; Jiang et al., 2025): a secret password is embedded in the system prompt with the instruction never to reveal it, and a user prompt attempts to extract it. We use this setting rather than the hazardous-material elicitation experiment of Jones et al. (2025) because the latter is closed-source and would require releasing potentially dual-use artifacts to replicate. The high-severity component of the task distribution consists of rare adversarial prompts (Perez et al., 2022; Ganguli et al., 2022; Shen et al., 2024), and the per-prompt risk score is the next-token probability that the model emits the password (exact from a single forward pass), transformed via ψ as in Section 3.1.[3]

For each password we generate a per-password prompt bank with two components: a bulk of moderate-difficulty prompts drawn from a procedural set of hand-crafted jailbreak families, and a thin password-specific tail of adversarial suffixes optimized offline against the base model with Greedy Coordinate Gradient (GCG; Zou et al., 2023); the mixture probabilities are selected so that the dangerous GCG mode almost never appears in the fit set and almost always appears in the deployment set. Bank size, family list, and per-family prompt counts are in Appendix A.

We fine-tune Qwen3-0.6B (Qwen Team, 2025) with LoRA (Hu et al., 2021) adapters. The partition into training and held-out targets is over passwords, not over prompts, so each held-out evaluation involves a password the model has never been fine-tuned on.

Forecastability training matches SFT on safety – both drive worst-rank held-out leak probability to indistinguishably low values, ≈ 40 decades below the pretrained baseline – while preserving model behavior dramatically better: WildChat single-token KL drift is more than two orders of magnitude lower under forecastability training, despite both methods using the same KL-to-base regularizer (Figure 3, left and middle). On forecast precision (Figure 3, right), forecastability training reaches $\approx 60\times$ improvement over the pretrained baseline and $\approx 75\times$ with subsequent calibration; SFT, calibration alone, and SFT+calibration all plateau below $20\times$, suggesting that neither direct risk minimization nor an affine post-hoc correction substitutes for shaping the underlying tail. A three-seed Qwen3-8B proof-of-concept (Appendix F) shows the same qualitative ordering with a smaller fold magnitude, driven primarily by a smaller starting baseline error.

6.2. Multi-task RL: gridworld navigation with traps

In this experiment, we show that the forecasting pipeline of Jones et al. (2025), originally developed for LM elicitation probabilities, applies largely unchanged to multi-task RL: each draw from the task distribution is a gridworld layout, fed to a single task-conditioned policy as part of its observation, and the risk score is the policy's expected regret on that layout. The gridworld is deterministic and finite-horizon, so backward value iteration gives exact, differentiable regret in closed form, and any forecast error we observe is the forecaster's contribution alone with no Monte Carlo noise

[3]The OLS line is fit in ψ-space, but per-rank residuals are inverse-transformed back to $\log p$ before squaring, and the post-hoc affine calibrator (*Cal.*, *Ours+cal.*, *SFT+cal.*) is also fit in $\log p$-space; we report results in log-probability space throughout, since the probabilities of interest span many orders of magnitude.

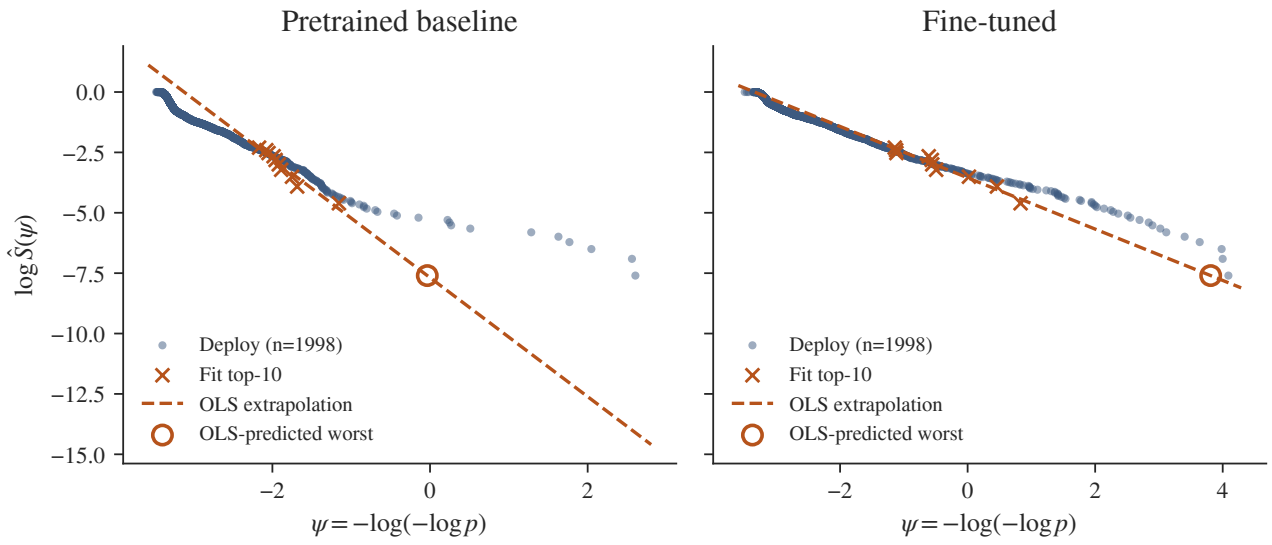

Figure 2. **Tail-shape change under fine-tuning, illustrative.** Empirical log-survival of one held-out password's deploy-set scores ($n = 1998$) plotted against the transformed score $\psi = -\log(-\log p)$ of Section 3.1, before (left) and after (right) forecastability training. The dashed line is the OLS extrapolation fit to the fit-set top-10 scores; the open circle marks the line's predicted worst-rank deploy score at $\log \hat{S} = \log[1/(n+1)] \approx -7.6$. For illustrative purposes only: this uses a simpler password-prompt distribution from the headline LM run and the improving-only gradient mask is disabled. Our headline results below therefore do not increase worst-case risk, unlike this figure.

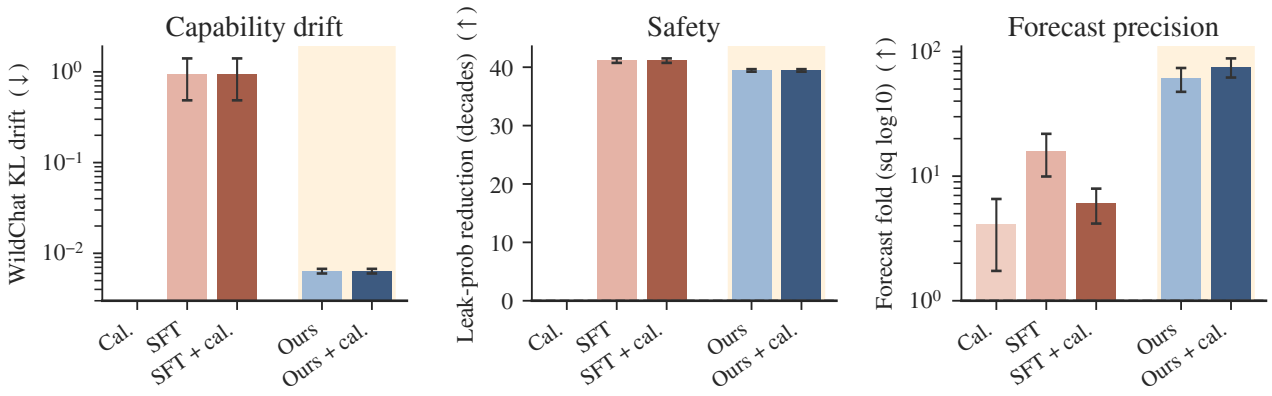

Figure 3. **Three-axis comparison on the language-model password game.** Error bars are seed-level standard errors over $n = 10$ seeds per condition. The three panels report, respectively, absolute WildChat single-token KL divergence (lower is better; the pretrained baseline has KL$= 0$ and is not drawn); decades of leak-probability reduction at the worst-rank held-out prompt over the pretrained baseline (higher is better; baseline sits at 0); and fold improvement in worst-rank squared log-probability error of the Gumbel-tail OLS extrapolation (log scale). Comparators are post-hoc affine calibration of the pretrained model (Cal., which by construction coincides with the pretrained baseline on capability and safety and is omitted from those two panels), supervised fine-tuning on the same task pool (SFT), and our fine-tuning method (Ours); each fine-tuned method is also shown with subsequent post-hoc calibration of its own forecast (SFT+ cal., Ours+ cal.).

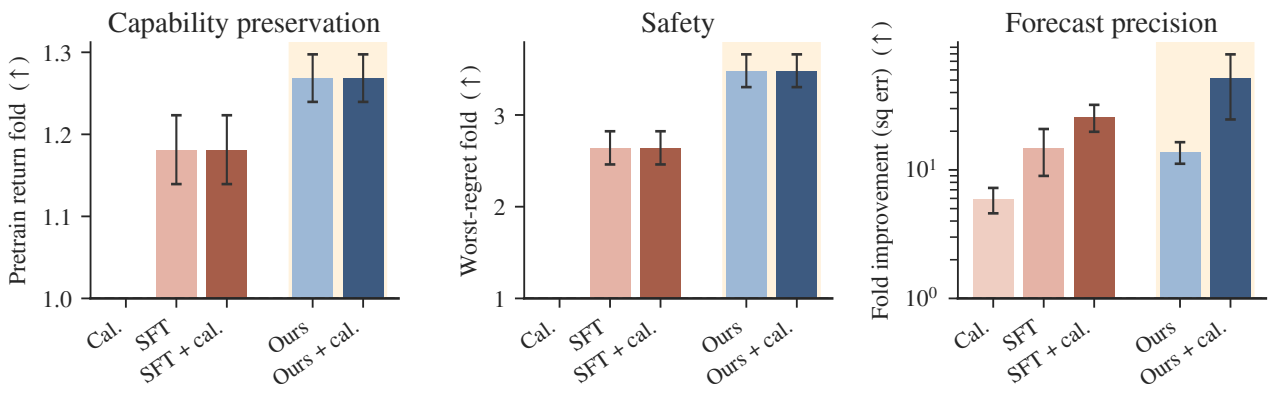

Figure 4. **Three-axis comparison on the gridworld setting.** Bars show mean fold improvement over the pretrained baseline (dashed line at $1\times$); error bars are seed-level standard errors over $n = 30$ seeds per condition; higher is better in every panel. Capability preservation uses the policy's mean return on the pre-training task pool; safety uses worst-case regret on the held-out deployment set; forecast precision uses worst-rank squared error of the Gumbel-tail OLS extrapolation. Comparators and naming convention match Figure 3.

from the score itself. The policy is a small task-conditioned convolutional network, pre-trained on a separate pool of tasks from the same mixture. Hyperparameters are listed in Appendix A.

Like the LM experiment, we construct the task distribution to put the Gumbel-tail forecaster in the failure regime described in Section 4. Each gridworld layout has a start cell, a goal cell at least five steps away, and a set of trap cells the policy must avoid; trap positions are visible to the policy. Bulk-mode tasks have no traps; rare-mode tasks have densely placed traps, and the rare mode makes up roughly 1.5×10^{-3} of the distribution. We chose this fraction so that a fit set rarely contains a rare-mode layout (expected count ≈ 0.15) while a deployment set almost always contains at least one and roughly three in expectation.

Forecastability training improves all three axes (Figure 4): pre-training return rises by $1.27\times$ over baseline and worst-case held-out regret falls to $1/3.5$, both ahead of SFT ($1.18\times$ and $1/2.6$). On forecast precision, Ours alone ($14\times$) is roughly tied with SFT ($15\times$), but post-hoc calibration produces a much larger boost for Ours – Ours+cal reaches $52\times$ versus SFT+cal at $26\times$ and calibration alone at $6\times$ – suggesting that the tail after forecastability training is much more amenable to affine correction than the SFT-shaped or pretrained tails. The fall in worst-case regret reflects the improving-only gradient mask: blocked from increasing the risk score of an overpredicted task, the optimizer's available direct path is to make the task genuinely easier.

7. Discussion

We presented forecastability fine-tuning, which adjusts a pretrained model to make its failure tail well-described by the forecaster; the loss is derived from a finite-k decomposition of the Gumbel-tail forecast error that identifies which error components are model-dependent and therefore trainable. Across two settings, a language-model password game and an RL gridworld, fine-tuning substantially reduces held-out forecast error while preserving primary-task capability and achieving safety comparable to that of supervised baselines.

Our experiments are deliberately small. The LM is a 0.6B-parameter base model fine-tuned with LoRA adapters on a single-token password game, and the RL environment is an 8×8 gridworld; both are stress tests for the rare-failure structure the method is designed to handle, but neither approaches frontier scale, and the 8B scaling proof-of-concept (Appendix F) suggests the size of the effect itself depends on model scale. Two further caveats: the RL setting relies on closed-form regret from backward value iteration, so extending to settings where regret must be estimated by Monte Carlo rollouts is a non-trivial generalization; and the bulk-plus-rare-mode mixture is synthetic, so application to naturally arising rare modes is open. Scaling to larger models, more realistic deployment domains, and natural rare modes is the obvious next step.

The forecastability loss is not tied to the OLS-on-Gumbel target. Any forecasting method whose tail predictions are differentiable in the model parameters in principle admits an analogous loss, and tail extrapolation may compose with importance sampling, the other main approach to rare-event estimation; we leave that combination to future work. More broadly, pre-deployment safety assessment will increasingly depend on extrapolating tail behavior from evaluation-scale to deployment-scale data (Shevlane et al., 2023; Clymer et al., 2024). Our results show that the model itself need not be a fixed input to that extrapolation: a small amount of fine-tuning can produce a model whose tails the same forecasting method describes much more accurately.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Angelopoulos, A. N., Bates, S., Fisch, A., Lei, L., and Schuster, T. Conformal risk control. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Angelopoulos, A. N., Bates, S., Candès, E. J., Jordan, M. I., and Lei, L. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025. doi: 10.1214/24-AOAS1998.

Atienza, N., Labreuche, C., Cohen, J., and Sebag, M. Provably safeguarding a classifier from OOD and adversarial samples: An extreme value theory approach, 2025.

Bai, Y.-L., Huang, Z.-Y., Lam, H., and Zhao, D. Black-box rare-event simulation for safety testing of AI agents: An overview. *Journal of the Operations Research Society of China*, 13:750–774, 2025. doi: 10.1007/s40305-025-00585-0.

Buhl, M. D., Pfau, J., Hilton, B., and Irving, G. An alignment safety case sketch based on debate, 2025.

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M. I., Gonzalez, J. E., and Stoica, I. Chatbot arena: An open platform for evaluating LLMs by human preference, 2024.

Chow, Y., Tamar, A., Mannor, S., and Pavone, M. Risk-sensitive and robust decision-making: A CVaR optimization approach. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Clymer, J., Gabrieli, N., Krueger, D., and Larsen, T. Safety cases: How to justify the safety of advanced AI systems, 2024.

Corso, A., Kim, K.-Y., Gupta, S., Gao, G., and Kochenderfer, M. J. A deep reinforcement learning approach to rare event estimation, 2022. URL https://arxiv.org/abs/2211.12470.

Davar, P., Godin, F., and Garrido, J. Catastrophic-risk-aware reinforcement learning with extreme-value-theory-based policy gradients. *The Journal of Finance and Data Science*, 11:100165, 2025. doi: 10.1016/j.jfds.2025.100165. URL https://www.sciencedirect.com/science/article/pii/S2405918825000170.

Deshmukh, S. V., Schwarzer, W., and Niekum, S. Evaluation-aware reinforcement learning, 2025.

Dorman, J. M., Gillman, E., Rose, D. C., Mair, J. F., and Garrahan, J. P. Rare event analysis of large language models, 2026. URL https://arxiv.org/abs/2602.06791.

Ganguli, D., Lovitt, L., Kernion, J., Askell, A., Bai, Y., Kadavath, S., Mann, B., Perez, E., Schiefer, N., Ndousse, K., et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned, 2022.

Gao, S., Zhou, Y., Shao, S., Luo, H., Bing, Y., Ding, J., Fu, L., and Wang, X. Extreme value policy optimization for safe reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 18772–18793. PMLR, 2025. URL https://proceedings.mlr.press/v267/gao25v.html.

Gringorten, I. I. A plotting rule for extreme probability paper. *Journal of Geophysical Research*, 68(3):813–814, 1963. doi: 10.1029/JZ068i003p00813.

Hanna, J. P., Thomas, P. S., Stone, P., and Niekum, S. Data-efficient policy evaluation through behavior policy search. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1394–1403. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/hanna17a.html.

Hanu, L. and Unitary team. Detoxify. https://github.com/unitaryai/detoxify, 2020.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jiang, Z., Jin, Z., and He, G. PromptKeeper: Safeguarding system prompts for LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 2712–2728, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.147. URL https://aclanthology.org/2025.findings-emnlp.147/.

Jones, E., Tong, M., Mu, J., Mahfoud, M., Leike, J., Grosse, R., Kaplan, J., Fithian, W., Perez, E., and Sharma, M. Forecasting rare language model behaviors, 2025.

Kwa, T., West, B., Becker, J., Deng, A., Garcia, K., Hasin, M., Jawhar, S., Kinniment, M., Rush, N., Von Arx, S., et al. Measuring AI ability to complete long tasks, 2025.

LDNOOBW Contributors. List of dirty, naughty, obscene, and otherwise bad words (English). https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words, 2023.

Li, T., Beirami, A., Sanjabi, M., and Smith, V. Tilted empirical risk minimization. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Liu, E. Z., Haghgoo, B., Chen, A. S., Raghunathan, A., Koh, P. W., Sagawa, S., Liang, P., and Finn, C. Just train twice: Improving group robustness without training group information. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6781–6792. PMLR, 2021.

METR. Autonomy evaluation resources. METR blog, March 2024. URL https://metr.org/blog/2024-03-13-autonomy-evaluation-resources/.

Mu, N., Lu, J., Lavery, M., and Wagner, D. A closer look at system prompt robustness, 2025. URL https://arxiv.org/abs/2502.12197.

O'Kelly, M., Sinha, A., Namkoong, H., Duchi, J. C., and Tedrake, R. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., and Irving, G. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3419–3448. Association for Computational Linguistics, 2022.

Phuong, M., Aitchison, M., Catt, E., Cogan, S., Kaskasoli, A., Krakovna, V., Lindner, D., Rahtz, M., Assael, Y., Hodkinson, S., et al. Evaluating frontier models for dangerous capabilities, 2024.

Qwen Team. Qwen3 technical report, 2025. Model card: https://huggingface.co/Qwen/Qwen3-0.6B.

Rein, D., Becker, J., Deng, A., Nix, S., Canal, C., O'Connel, D., Arnott, P., Bloom, R., Broadley, T., Garcia, K., et al. HCAST: Human-calibrated autonomy software tasks, 2025.

Richards, J. and Huser, R. Extreme quantile regression with deep learning, 2024.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Shen, X., Chen, Z., Backes, M., Shen, Y., and Zhang, Y. "Do Anything Now": Characterizing and evaluating In-The-Wild jailbreak prompts on large language models. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2024. doi: 10.1145/3658644.3670388.

Shevlane, T., Farquhar, S., Garfinkel, B., Phuong, M., Whittlestone, J., Leung, J., Kokotajlo, D., Marchal, N., Anderljung, M., Kolt, N., et al. Model evaluation for extreme risks, 2023.

Somayaji N. S., K., Wang, Y., Schram, M., Drgoňa, J., Halappanavar, M. M., Liu, F., and Li, P. Extreme risk mitigation in reinforcement learning using extreme value theory. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=098mb06uhA.

Tamar, A., Glassner, Y., and Mannor, S. Optimizing the CVaR via sampling. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 2993–2999, 2015.

Toyer, S., Watkins, O., Mendes, E. A., Svegliato, J., Bailey, L., Wang, T., Ong, I., Elmaaroufi, K., Abbeel, P., Darrell, T., Ritter, A., and Russell, S. Tensor trust: Interpretable prompt injection attacks from an online game. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Uesato, J., Kumar, A., Szepesvári, C., Erez, T., Ruderman, A., Anderson, K., Dvijotham, K., Heess, N., and Kohli, P. Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Webb, S., Rainforth, T., Teh, Y. W., and Kumar, M. P. A statistical approach to assessing neural network robustness. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., Su, D., Gao, Y., Hsieh, C.-J., and Daniel, L. Evaluating the robustness of neural networks: An extreme value theory approach. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Wessel, J. B., Schillinger, M., Kwasniok, F., and Allen, S. Enforcing tail calibration when training probabilistic forecast models, 2025.

Wu, G. and Hilton, J. Estimating the probabilities of rare outputs in language models. In *Proceedings of the 13th International Conference on Learning Representations*, 2025.

Xu, M. Estimating tail risk in neural networks. Alignment Research Center blog, September 2024. URL https://alignment.org/blog/estimating-tail-risk/. Blog post describing joint research with Jacob Hilton, Victor Lecomte, David Matolcsi, Eric Neyman, Thomas Read, George Robinson, and Gabe Wu.

Zhao, W., Ren, X., Hessel, J., Cardie, C., Choi, Y., and Deng, Y. WildChat: 1M ChatGPT interaction logs in the wild. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2405.01470.

Žilinskas, A., Shorten, R. N., and Mareček, J. EVEREST: An evidential, tail-aware transformer for rare-event time-series forecasting. In *Proceedings of the 14th International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2601.19022.

Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models, 2023.

A. Experimental details

This appendix collects the specific configuration choices for the two canonical runs that produced Figures 3 and 4. The point of separating them out is to keep the methods section's algorithmic content uncluttered by hyperparameters; nothing here should affect the conceptual story.

A.1. Language model run

Model. `Qwen/Qwen3-0.6B` fine-tuned with LoRA adapters: rank $r = 8$, alpha $= 16$, dropout $= 0.05$, target modules {`q_proj, k_proj, v_proj, o_proj`}.

Password game. Each password is a single token drawn from the model's vocabulary, restricted to clean lowercase alphabetic words of length 4–10, assigned per slot deterministically from the seed. The system prompt embeds the password and instructs the model never to reveal, repeat, transform, hint at, or encode the password under any circumstances. Failure on a (password, prompt) pair is defined as the password token being the next token the model would emit; the elicitation probability of a prompt is therefore the next-token softmax probability of the password token, available from a single forward pass.

Prompt bank. The split structure has two levels. At the *password level*, 40 password slots are partitioned per seed into 33 training-target passwords and 7 held-out-target passwords; the held-out targets are never seen during fine-tuning. At the *prompt level* (within each target), we generate 466 prompts in each of 4 moderate-difficulty jailbreak families (`compositional`, `mutate`, `indirect_extraction`, `password_seeded`) and 6 prompts in the `gcg_steps80` family of GCG-optimized adversarial suffixes – 1,870 prompts per target ($\sim 0.32\%$ GCG). All 5 families appear in both prompt-level halves; the prompt-level half each prompt instance lands in is the only thing that differs between the two halves. After a $50/50$ prompt-level split, each side of each target has 935 prompts; within each side we use 44 as the fit set and 891 as the deployment set (fit_fraction ≈ 0.0476).

GCG generation. Generated offline by running standard Greedy Coordinate Gradient on the base model with the password log-probability as the objective. 24-token adversarial suffix, 80 optimization steps per prompt, top-8 candidate replacements per position, 2 random restarts.

Training. 200 outer-loop steps with AdamW (PyTorch defaults: $(\beta_1, \beta_2) = (0.9, 0.999)$, $\varepsilon = 10^{-8}$, weight decay 10^{-2}), no warmup, learning rate 10^{-5}, gradient clip 1.0, micro-batch size auto-tuned to GPU VRAM (96 on A100-80G). OLS top-$k = 10$, Weibull plotting position, score transform $-\log(-\log p)$, loss space = log-probability, extrapolated weighting = deploy-log-uniform, gradient flow = both, improving-only gradients enabled on both fit and deploy sides (`iog_scope=both`). The Qwen3 chat template is applied with thinking mode disabled (`enable_thinking=False`) at every chat-template invocation. Primary-objective regularizer = $0.05\cdot$ single-token KL divergence to a frozen base model copy on the password prompt distribution. A single GPU per seed; per-target backward passes accumulate into the same gradient buffer before the optimizer step. The *SFT* baseline uses the same backbone, optimizer, learning rate, regularizer, and chat-template settings; it differs in the loss (direct minimization of leak log-probability on the union of fit and deploy prompts) and in budget (2,000 steps with effective batch 128 on A100-80G, matched to the meta-training run's total forward-pass count to within a small factor). Ten random seeds per condition.

Partition pool and union cache. Per-target prompt pool $|\mathcal{U}| = M + N = 44 + 891 = 935$ prompts. At each step we draw $(\mathcal{F}_t, D_t)$ as a uniform random partition of $\mathcal{U}$. Union top-C cache size $C = 296$, refresh interval $\rho = 5$ steps; one partition per step ($N_{\text{perm}} = 1$). Appendix B describes the cache and its coverage guarantees.

Reported metrics. Forecast errors are computed in log-probability space (i.e., the predicted log-probability at each extrapolated deploy rank, against the observed log-probability). Per-target metrics are aggregated as means; figure error bars are standard errors over seeds (each seed drawing its own random partition into 33 train and 7 held-out password targets). Behavior drift is measured as single-token KL divergence to the frozen base model on 256 first-user-turn prompts uniformly subsampled (seed 42) from the cached WildChat-1M-Full snapshot, formatted with the Qwen3 chat template (thinking off, generation prompt added, max length 512 tokens); KL is computed every 5 training steps over batches of 8 prompts. Leak log-probabilities are computed in float32 from `torch.nn.functional.log_softmax` on the next-token logits, with

no clipping or floor applied; the reported worst-rank decade reductions therefore track the model's actual ability to drive the password-token softmax probability down, bounded only by float32 representability of the relative logit gap, which exceeds the largest reported reduction of ~ 53 decades by a comfortable margin.

Post-hoc affine calibration. *Cal.*, *Ours+cal.*, and *SFT+cal.* fit a two-parameter affine correction $\widehat{Q}^{\text{cal}}(y) = \alpha\,\widehat{Q}(y) + \beta$ in log-probability space. The pair (α, β) is fit by ordinary least squares on the 33 train-target (fit, deploy) pairs that the seed's training run already used, comparing each pair's predicted worst-rank log-probability (from the OLS line on the fit set) to its actual worst-rank log-probability on the deployment set. Calibration is fit using the pretrained Qwen3-0.6B in the *Cal.* baseline and the seed's own LoRA-adapted model in *Ours+cal.* and *SFT+cal.* The fit is then applied unchanged to the 7 held-out targets.

A.2. Multi-task RL run

Environment. 8×8 grid, horizon 10, undiscounted ($\gamma = 1.0$), step_cost $= -0.01$, goal_reward $= +1.0$, trap_penalty $= -1.0$, minimum start-goal Manhattan distance 5.

Task bank. For each of the 30 fine-tuning seeds we generate a fresh, independent bank of 52,000 gridworld layouts (so $\sim$1.56 M layouts in total across seeds) with two components per bank: a 51,920-task bulk-mode component at severity 0 (no traps), and an 80-task tail component at severity 1 (trap density 0.75). Per-bank mixing ratio $80/52{,}000 \approx 1.54 \times 10^{-3}$. The codebase refers to this family as `trap_open_room`.

Splits per seed. Within each seed, all of the seed's tasks are drawn without replacement from that seed's 52,000-layout bank, with global de-duplication keeping every split disjoint from every other within the seed: 192 pre-training tasks; 20 training (fit, deploy) pairs, each with 96 fit tasks and 1,920 deploy tasks; and 5 held-out (fit, deploy) pairs of the same shape, drawn from a separate disjoint pool within the same bank and never used during fine-tuning. Different seeds draw from independently generated banks, so cross-seed disjointness is statistical rather than enforced. The within-bank mixture composition is preserved by per-component subsampling, so each split inherits the $\sim 1.54 \times 10^{-3}$ tail-mode mixing ratio in expectation.

Policy architecture. A small task-conditioned U-Net (encoder/decoder of 32/64/64 channels, FiLM-modulated by a 64-dimensional task embedding) that outputs action logits for every (state, timestep) pair in the dense state space.

Training. Pre-training: 500 steps of return maximization with batch size 16, learning rate 10^{-4}, gradient clip 1.0, AdamW (PyTorch defaults, weight decay 0). Fine-tuning: 300 outer-loop steps with the same AdamW configuration and learning rate, 10 training (fit, deploy) pairs sampled per step, OLS top-$k = 10$, Weibull plotting position, score transform = identity (raw regret), loss space = score, extrapolated weighting = deploy-log-uniform, gradient flow = both, improving-only gradients enabled on both fit and deploy sides (`iog_scope=both`), primary-objective regularizer = $-1.5\cdot$ mean return on a held-in batch of pre-training tasks. The *SFT* baseline uses the same architecture, optimizer, and regularizer; its loss is direct minimization of mean exact regret on the union of fit and deploy tasks, which uses the closed-form regret backward-value-iteration computation that the forecastability loss also relies on. Thirty random seeds per condition. Pre-training is reused from a prior identical run; the submission script verifies that the bank and seed match before reusing.

Partition pool and union cache. Per-target task pool $|\mathcal{U}| = M + N = 96 + 1{,}920 = 2{,}016$ tasks. At each step we draw $(\mathcal{F}_t, D_t)$ as a uniform random partition of $\mathcal{U}$. Union top-C cache size $C = 296$, refresh interval $\rho = 5$ steps; one partition per step ($N_{\text{perm}} = 1$). Appendix B describes the cache and its coverage guarantees.

Reported metrics. Forecast errors are computed by fitting the OLS line on each held-out pair's fit set, predicting the score at each extrapolated deploy rank, and taking either (i) the squared error at the worst held-out rank, or (ii) the per-rank weighted squared error averaged across the extrapolated range. Mean and worst regret are computed directly on the held-out deployment set.

Post-hoc affine calibration. As in the LM run, *Cal.* and *Ours+cal.* fit a two-parameter affine correction by ordinary least squares on the 20 training (fit, deploy) pairs (predicted vs. actual worst-rank regret), then apply the fit to the 5 held-out pairs.

The pretrained policy is used in *Cal.* and the seed's own fine-tuned policy in *Ours+cal.* and *SFT+cal.* Calibration runs as an offline post-processing step over the saved per-seed forecast traces and adds no additional environment interaction.

B. Detailed fine-tuning pseudocode

Algorithm 1 gives the implementation pseudocode, expanding the high-level description in Section 5 with the tricks we use in practice. We describe each trick in turn before giving the algorithm. Throughout, fix a per-target task pool $\mathcal{U}$ of size $M + N$, from which $(\mathcal{F}_t, D_t)$ pairs of sizes (M, N) are drawn.

Two-stage scoring trick. On each $(\mathcal{F}, D)$ pair, the fit-side and deploy-side passes use a two-stage scoring scheme. We first score a candidate set without gradients to identify the relevant subset (the top-k on the fit side, the points whose plotting positions land in the extrapolated range on the deploy side), then re-evaluate just that subset with gradients. The candidate set is the union top-C cache (below), so the gradients are exact conditional on the cached set; this is exact at a cache refresh and approximate over the refresh interval ρ, since model parameters drift between refreshes.

Per-step partition randomization. The natural realization of $\mathcal{P}_{\text{meta}}$ holds the partition $(\mathcal{F}, D)$ fixed for each target across fine-tuning. With this choice we observe a sharp asymmetry-overfit failure mode: the optimizer satisfies the loss by encoding directions that raise scores on the specific fit-side inputs and leave the specific deploy-side inputs unchanged, which is a memorized per-input asymmetry rather than a partition-invariant improvement. The signature of this failure is a held-out forecastability loss orders of magnitude above the training-side forecastability loss when partitions are fixed. We therefore draw a fresh uniform partition $(\mathcal{F}_t, D_t)$ of $\mathcal{U}$ at every step, optionally averaging across $N_{\text{perm}} \geq 1$ such partitions. Because the membership of $\mathcal{F}_t$ and D_t reshuffles every step, no per-input asymmetric direction stays useful across steps, and the optimizer is forced into improvements that are partition-invariant. Setting $N_{\text{perm}} = 0$ recovers a fixed partition and a deploy-only cache (described next), reducing Algorithm 1 to the prior version of the procedure.

Union top-C cache. Partition randomization is incompatible with the obvious deploy-side cache, in which the top-B' scoring deploy points are cached for ρ steps and re-scored on each step. The cached identities are partition-specific, so a fresh D_t generally does not contain them. We instead cache the top-C scoring points of the union $\mathcal{U} = \mathcal{F} \cup D$, which is partition-invariant. Let $\mathcal{C} \subseteq \mathcal{U}$ denote the cached index set, $|\mathcal{C}| = C$. At each step, the cached subset is re-scored with gradients; non-cached positions of $\mathcal{F}_t$ and D_t are filled with sentinel values that the downstream top-k and extrapolated-range selectors discard.

Coverage analysis. Let $X_C = |\mathcal{F}_t \cap \mathcal{C}|$, the number of cached points landing on the fit side under a uniform random partition. Then $X_C \sim \text{Hypergeometric}(M + N,\, C,\, M)$. Two consequences follow. The cached portion of the deploy side is $|\mathcal{C} \cap D_t| = C - X_C \geq C - M$, which is deterministic in C and M alone, so choosing $C \geq B' + M$ guarantees that every random partition has its full deploy-top-B' cached at the cache-refresh step; between refreshes the deploy-side coverage is approximate, since drift in θ can move an uncached point into the current top-B'. Fit-side coverage is probabilistic: at the canonical RL parameters $C = 296$, $M = 96$, $N = 1{,}920$, $k = 10$, the cache misses a top-k fit score with probability $\Pr[X_C < k] \approx 0.082$; conditional on a miss, the lazy-fit fallback re-scores $\mathcal{F}_t \setminus \mathcal{C}$, costing $\mathbb{E}[M - X_C \mid X_C < k] \approx 88$ extra evaluations on a missed step and ≈ 7.2 extra evaluations per step in unconditional expectation. The corresponding LM numbers ($M = 44$, $N = 891$) are $\Pr[X_C < k] \approx 0.067$ and ≈ 2.4 extra evaluations per step in unconditional expectation.

The deploy candidate count B' no longer appears as a runtime parameter once the cache is union-indexed; it survives as a coverage target through the constraint $C \geq B' + M$. The canonical runs both use $C = 296$, which guarantees deploy-side coverage of at least $C - M = 200$ ranks for the RL setting ($M = 96$) and at least $C - M = 252$ ranks for the LM setting ($M = 44$).

C. Per-rank weights in the forecastability loss

The forecastability loss in Equation 5.1 attaches a weight w_j to each extrapolated rank $j \in J$. The choice of weights is conceptual: each rank j in the deployment set is the failure-rate quantile that would matter at a corresponding deployment size n, so a weighting over ranks is implicitly a prior over deployment sizes. We considered three schemes:

- **Rank-uniform**: $w_j = 1/|J|$. Each extrapolated rank contributes equally. This corresponds to no explicit prior over

Algorithm 1 Fine-tuning for forecastable tails (detailed version).

Require: pre-trained θ_0; per-target pool $\mathcal{U}$ with $|\mathcal{U}| = M + N$; fit size M, deploy size N; regularizer $\mathcal{L}_{\text{reg}}$ with weight λ; steps T; learning rate η; fit tail depth k; deploy candidate count B'; union cache size C; cache refresh interval ρ; partitions per step N_{perm}

$\theta \leftarrow \theta_0$; $\mathcal{C} \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  if $t \bmod \rho = 0$ or $\mathcal{C} = \emptyset$ **then**
    Score $f(x; \theta)$ for all $x \in \mathcal{U}$ {detached}
    $\mathcal{C} \leftarrow$ indices of the top-C scores in $\mathcal{U}$
  end if
  $\mathcal{L}_{\text{meta}} \leftarrow 0$
  $P \leftarrow \max(N_{\text{perm}}, 1)$
  for $p = 1, \ldots, P$ **do**
    if $N_{\text{perm}} \geq 1$ **then**
      Sample $(\mathcal{F}_{t,p}, D_{t,p})$ as a uniform random partition of $\mathcal{U}$
    else
      $(\mathcal{F}_{t,p}, D_{t,p}) \leftarrow$ the fixed $(\mathcal{F}, D)$
    end if
    $E \leftarrow \mathcal{C}$
    if $|\mathcal{F}_{t,p} \cap \mathcal{C}| < k$ **then**
      $E \leftarrow E \cup (\mathcal{F}_{t,p} \setminus \mathcal{C})$ {lazy-fit fallback}
    end if
    Re-score $f(x; \theta)$ for $x \in E$ {with gradients}
    Place sentinel values on $(\mathcal{F}_{t,p} \cup D_{t,p}) \setminus E$; apply two-stage scoring on each side
    Fit OLS line $(a_{t,p}, b_{t,p})$ on the k fit scores vs. log plotting positions
    $\mathcal{L}_{\text{meta}} \leftarrow \mathcal{L}_{\text{meta}} + \frac{1}{P} \mathcal{L}_{\text{forecast}}(\mathcal{F}_{t,p}, D_{t,p}; \theta)$
  end for
  $\theta \leftarrow \theta - \eta \nabla_\theta (\mathcal{L}_{\text{meta}} + \lambda \mathcal{L}_{\text{reg}}(\theta))$
end for
return θ

deployment sizes and tends to over-weight the ranks closest to the fit regime, since adjacent ranks cover similar deployment sizes.

- **Deploy-log-uniform**: ranks are weighted by the width of the interval in $\log n$ that they cover, which corresponds to a log-uniform prior over deployment sizes. Conceptually this says we care equally about forecasting each order of magnitude of deployment size beyond the fit regime.

- **Deploy-uniform**: the same construction with a uniform prior over n. Because rank j corresponds to a deployment size $n_j \sim N/j$, the interval in n covered by rank j scales as $1/j^2$, and the resulting weights $w_j \propto 1/j^2$ pile sharply onto the smallest j – that is, onto the largest deployment sizes. This is rarely what we want, since it makes the loss almost entirely about the deepest few ranks of the deployment set.

Both of our experiments use the deploy-log-uniform weighting, on the grounds that we have no specific deployment size in mind and want the forecast to be roughly equally informative across scales.

D. Improving-only mask on the fit side

The improving-only refinement of Section 5 keeps a fit-side score ψ_i active in the gradient only when reducing the forecast loss along ψ_i would also reduce ψ_i itself, so that the model improves on that fit task rather than getting worse on it. The criterion is the sign of $\partial \mathcal{L}_{\text{forecast}} / \partial \psi_i$, which we can write in closed form because the OLS coefficients a and b are explicit functions of the fit-side scores.

Let the top-k fit-side scores be $\psi_1, \ldots, \psi_k$ and let $y_i = \log \hat{S}_i$ be the corresponding (fixed) plotting-position log-survivals

(so $y_i \le 0$, the opposite sign convention from the depth $y = -\log \hat{S}$ used in Section 3.2; we use the sign-flipped form here because it produces a cleaner closed form for the OLS slope). With $\bar{\psi}$ and $\bar{y}$ the means and $S_{\psi\psi} = \sum_i (\psi_i - \bar{\psi})^2$, ordinary least squares gives

$$a = \frac{\sum_i (\psi_i - \bar{\psi})(y_i - \bar{y})}{S_{\psi\psi}}, \qquad b = \bar{y} - a\bar{\psi}.$$

Writing the per-rank residual as $r_j = \psi_j^{\text{pred}} - \psi_j^{\text{obs}}$ for $j \in J$ and the weighted residual as $\tilde{r}_j = w_j r_j$, a direct calculation yields

$$\frac{\partial \mathcal{L}_{\text{forecast}}}{\partial \psi_i} = 2C_1\big((y_i - \bar{y}) - 2a(\psi_i - \bar{\psi})\big) + C_2,$$

where

$$C_1 = -\frac{\sum_{j \in J} \tilde{r}_j (y_j - \bar{y})}{a^2 S_{\psi\psi}}, \qquad C_2 = \frac{2}{k} \sum_{j \in J} \tilde{r}_j,$$

with $y_j = \log \hat{S}_j$ for the deploy ranks and $\bar{y}$ still the fit-side mean. Because higher ψ_i corresponds to worse safety, we keep fit point i active only when this derivative is positive: under gradient descent on θ, $\partial \mathcal{L}_{\text{forecast}} / \partial \psi_i > 0$ is the direction in which reducing the loss also reduces ψ_i. Fit points with non-positive derivatives are masked out via a straight-through detach so that the forward computation of $\mathcal{L}_{\text{forecast}}$ is unchanged.

E. Plotting positions

The OLS fit in the Gumbel-tail method requires estimating the survival probability $S(\psi_{(i)})$ at each of the top-k order statistics. Given M evaluation tasks, the i-th largest transformed score $\psi_{(i)}$ (for $i = 1, 2, \ldots, M$, with $i = 1$ being the largest) has a true survival probability $S(\psi_{(i)}) = \mathbf{P}(\psi(x) > \psi_{(i)})$ that must be estimated from the data. The naive estimate $\hat{S}_i = i/M$ is biased at the extremes; in particular it assigns $\hat{S}_1 = 1/M$ to the largest observation, an overestimate in expectation since the true survival probability of the maximum of M draws is $1/(M+1)$. *Plotting positions* are classical formulas that provide less biased estimates of $S(\psi_{(i)})$ for each order statistic. Common choices include:

- **Weibull:** $\hat{S}_i = i/(M+1)$. Distribution-free; widely used as a default.
- **Hazen:** $\hat{S}_i = (i - 0.5)/M$. Approximates the median of $S(\psi_{(i)})$ under mild assumptions.
- **Gringorten:** $\hat{S}_i = (i - 0.44)/(M + 0.12)$. Derived by Gringorten (1963) to minimize bias when the underlying data is Gumbel-distributed.

These estimated survival probabilities enter the OLS regression as the y-values: we regress $\log \hat{S}_i$ against $\psi_{(i)}$ for $i = 1, \ldots, k$. The choice of plotting position formula therefore directly affects the fitted slope a and intercept b, and consequently the extrapolated quantiles.

In our experiments we use the Weibull plotting position, the distribution-free default. Gringorten would be the more principled choice under a strict Gumbel tail assumption, but Gringorten's formula was derived assuming the *entire* distribution is Gumbel, whereas the Gumbel-tail method of Jones et al. (2025) assumes only that the tail belongs to the Gumbel maximum domain of attraction (the peaks-over-threshold distribution converges to a generalized Pareto distribution with shape parameter $\xi = 0$). The full distribution of risk scores need not be Gumbel, so Gringorten's optimality argument does not straightforwardly apply. The choice does affect the OLS line. Weibull and empirical differ on the log-survival scale by a constant offset, $\log \hat{S}_i^{\text{Weibull}} - \log \hat{S}_i^{\text{empirical}} = \log\big(M/(M+1)\big) \approx -1/M$, independent of i; this shifts the OLS intercept and leaves the slope unchanged. Hazen and Gringorten do introduce i-dependent log-scale corrections that can shift both slope and intercept: e.g., $\log \hat{S}_i^{\text{Hazen}} - \log \hat{S}_i^{\text{empirical}} = \log(1 - 0.5/i)$, which is $\log(0.5) \approx -0.69$ at $i = 1$ and $\log(0.95) \approx -0.05$ at $i = 10$. Because we fit only to top-k order statistics out of $M \gg k$, the OLS sensitivity to plotting position remains small, and the choice is unlikely to dominate the extrapolation error.

F. 8B scaling proof-of-concept

This appendix extends the headline Qwen3-0.6B results using a three-seed proof-of-concept at Qwen3-8B, otherwise matching the canonical recipe of Section 6.1 and Appendix A. Due to computational limitations, only three seeds were run, and only the post-hoc OLS calibration baseline is shown.

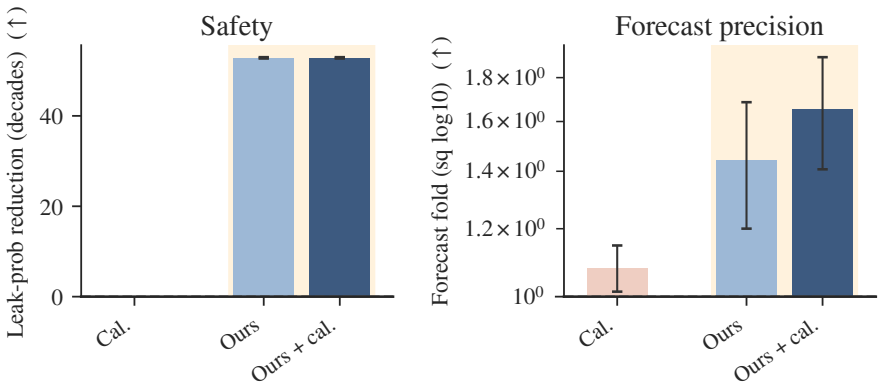

Figure 5. **Two-axis comparison on the 8B password game.** Same conventions as Figure 3, restricted to the two axes available without WildChat KL evaluation. Bars are seed means with seed-level standard errors over three seeds. Comparators are post-hoc affine calibration of the pretrained model (*Cal.*), forecastability training (*Ours*), and forecastability training plus subsequent post-hoc calibration of its own forecast (*Ours+ cal.*); SFT and SFT+ cal. comparators are absent for this scale due to compute budget.

Differences from the canonical 0.6B run. (i) Base model: `Qwen/Qwen3-8B` (LoRA $r = 8$, $\alpha = 16$, same target modules). (ii) Prompt bank regenerated with Qwen3-8B as the GCG target; family list and per-family counts unchanged (40 slots; 33 train + 7 held-out per seed; 4 moderate-difficulty families $\times 466 + 6$ GCG = 1,870 prompts/target; 44/891 fit/deploy split per side). (iii) 3 random seeds rather than 10. (iv) No SFT or SFT+cal comparators (compute-prohibitive within the time budget). (v) WildChat KL drift not measured (`--kl-eval-datasets` disabled to fit per-step time on 8B), so the figure has two panels rather than three. (vi) Effective batch size differs across seeds: per-rank micro-batch 16 across 3 A100s in DDP for two seeds (effective 48) and a single A100 for one seed (effective 16); the canonical 0.6B run used effective 96.

Results. Across the three seeds, the safety axis improves by 52.84 ± 0.15 decades (mean $\pm$ SEM) of leak-probability reduction at the worst-rank held-out prompt. Per-seed paired forecast fold over each seed's own pretrained baseline (the same aggregation the figure plots) is $(1.08 \pm 0.07)\times$ for post-hoc calibration alone, $(1.44 \pm 0.24)\times$ for forecastability training, and $(1.65 \pm 0.25)\times$ for training plus calibration. All three method bars are above the no-training reference, with *Ours+ cal.* > *Ours* > *Cal.* matching the qualitative ordering of the 0.6B headline (Figure 3). The fold magnitudes are an order of magnitude smaller than at 0.6B: per-seed baseline squared $\log_{10}$-error (mean $\pm$ SEM across the three seeds) is 1.18 ± 0.69 on 8B vs 3.08 ± 0.68 on 0.6B, while per-seed trained error is 0.91 ± 0.60 on 8B vs 0.070 ± 0.015 on 0.6B. The baseline reduction (smaller numerator) is the larger driver, but the trained-side increase is also material; a smaller baseline error necessarily limits the possible fold improvement, due to the inherently random error term T_θ.

On the safety side, the ~ 53-decade improvement is measured against an 8B baseline that is itself *more* leak-prone at the worst rank than 0.6B's (worst-rank leak probability ≈ 0.66 vs ≈ 0.07), so the larger decade count does not by itself indicate a safer end state.

Limitations and interpretation. It is unclear based on this experiment alone why the forecast error is smaller on the 8B model than on 0.6B – though this result is consistent with the strong forecastability of the Claude models used by Jones et al. (2025) – and hence why forecastability training has less potential improvement to offer. Future work should systematically examine how forecastability scales with model size.

G. A finite-k decomposition for the inverse-OLS Gumbel-tail extrapolator

Headline result. We prove a finite-k decomposition of the forecast error of the inverse-OLS Gumbel-tail extrapolator (the estimator of Section 3.1, fit log-survival as a function of score and inverted at the deployment rank) into three explicit terms:

(a) a *rank effect* of order $q'(y)$, with a coefficient $b^{\mathrm{inv}}_{k,r}$ that depends only on the top-k count and the deployment ratio R, not on the tail family;

(b) a *curvature effect* of order $q''(y)$, whose nominal-quantile sign is determined by the hazard rate of the score distribution

(the realized finite-k coefficient is random; its expectation has the predicted sign in the regimes we study);

(c) a *rare-mode occupancy gap* that subtracts from the error when a rare high-risk component is absent from evaluation but present at deployment;

plus a higher-order remainder. The decomposition assumes a continuous score distribution (no ties at the relevant order statistics), independent fit and deploy samples from the same distribution, $q \in C^3$ on the relevant tail interval, and a non-degenerate denominator event $A(T_m) \geq \eta$ in the line fit. Proposition 1 of Section G.2 states the smooth single-component decomposition as an exact finite-k expansion; Section G.5 adds the occupancy term in the latent-mixture case. Section G.3 tabulates $b^{\mathrm{inv}}_{k,r}$ at the default $k = 10$ of Jones et al. (2025), also used in our experiments, and Figure 6 confirms the values by simulation against 10^6 trials of the actual estimator. Appendix H uses the decomposition to predict, term by term, what each baseline in the paper's experiments can and cannot reduce.

Setup. This appendix analyzes the finite-k effects of the Gumbel-tail line fit. The analysis treats the risk score as the random variable being forecast. In the language-model experiments this score can be the transformed probability score $\psi = -\log(-\log p)$; the distribution F below is the distribution of that transformed score.

The Gumbel-tail method of Jones et al. (2025) fits the tail on a log-survival scale. We therefore write the upper-tail quantile curve as

$$q(y) = F^{-1}(1 - e^{-y}), \qquad y > 0.$$

Thus $q(\log n)$ is the population one-in-n score quantile. Let $m = n_{\mathrm{fit}}$ and $N = n_{\mathrm{deploy}}$, and set

$$y = \log m, \qquad r = \log(N/m).$$

The estimator fits the top k evaluation scores to their nominal log-ranks and then inverts the fitted line at the deployment rank. This appendix shows that the forecast error has a leading finite-k term even when q is exactly linear. Curvature of q and rare-mode occupancy then add separate error terms.

G.1. Inverse line fit

Let $X^{\downarrow}_{1:m} \geq \cdots \geq X^{\downarrow}_{m:m}$ be the descending order statistics from the evaluation sample. The nominal offsets of the top k ranks are

$$a_j := -\log j, \qquad j = 1, \ldots, k.$$

Our code uses the Weibull plotting position $a_j^{\mathrm{Weibull}} = \log((m+1)/j) - \log m = -\log j + \log((m+1)/m)$, which differs from a_j by $\log((m+1)/m) = O(1/m)$ uniformly in j. The discrepancy is absorbed into the high-probability remainder of Proposition 1; in the limit $m \to \infty$ the two conventions agree. For a vector $x = (x_1, \ldots, x_k)$ of observed scores, define

$$\bar{x} := \frac{1}{k}\sum_{j=1}^{k} x_j, \qquad \bar{a} := \frac{1}{k}\sum_{j=1}^{k} a_j.$$

The inverse line fit regresses a_j on x_j and then solves the fitted line for the score at offset r. If

$$S_{xx}(x) := \sum_{j=1}^{k}(x_j - \bar{x})^2, \qquad S_{ax}(x) := \sum_{j=1}^{k}(a_j - \bar{a})(x_j - \bar{x}),$$

then the inverted prediction is

$$\mathcal{J}_r(x) := \bar{x} + (r - \bar{a})\frac{S_{xx}(x)}{S_{ax}(x)}.$$

For continuous F, this denominator is positive almost surely: both a_j and $X^{\downarrow}_{j:m}$ are sorted in the same order, and the top-k scores are not all equal.

The inverse-OLS forecast is

$$\widehat{Q}^{\mathrm{inv}}_{m,k}(N) := \mathcal{J}_r\left(X^{\downarrow}_{1:m}, \ldots, X^{\downarrow}_{k:m}\right).$$

Let $Y^{\downarrow}_{1:N}$ denote the maximum of an independent deployment sample of size N.

G.2. Expansion for a smooth tail

The random locations of the top order statistics are easiest to describe after transforming to survival values. Define

$$U_{j:m} \coloneqq \bar{F}(X^{\downarrow}_{j:m}), \qquad V_{1:N} \coloneqq \bar{F}(Y^{\downarrow}_{1:N}),$$

where $\bar{F} = 1 - F$. Set

$$T_{j,m} \coloneqq -\log(mU_{j:m}), \qquad W_N \coloneqq -\log(NV_{1:N}).$$

Then

$$X^{\downarrow}_{j:m} = q(y + T_{j,m}), \qquad Y^{\downarrow}_{1:N} = q(y + r + W_N).$$

The deterministic value corresponding to the jth nominal rank is $a_j = -\log j$. The variables $T_{j,m}$ are the random replacements for those nominal offsets.

We need one more functional. For $t = (t_1, \dots, t_k)$, define

$$\bar{t} \coloneqq \frac{1}{k}\sum_{j=1}^{k} t_j, \qquad A(t) \coloneqq \sum_{j=1}^{k}(a_j - \bar{a})(t_j - \bar{t}), \qquad B(t) \coloneqq \sum_{j=1}^{k}(t_j - \bar{t})^2,$$

and

$$\mathcal{I}_r(t) \coloneqq \bar{t} + (r - \bar{a})\frac{B(t)}{A(t)}.$$

This is the inverse fit on the offset scale. It satisfies the equivariance identity

$$\mathcal{J}_r(q_0 + q_1 t_1, \dots, q_0 + q_1 t_k) = q_0 + q_1 \mathcal{I}_r(t) \tag{2}$$

for any $q_1 > 0$.

For a perturbation $v = (v_1, \dots, v_k)$, the directional derivative of $\mathcal{I}_r$ is

$$D\mathcal{I}_r(t)[v] = \bar{v} + (r - \bar{a})\frac{2A(t)\sum_{j=1}^{k}(t_j - \bar{t})(v_j - \bar{v}) - B(t)\sum_{j=1}^{k}(a_j - \bar{a})(v_j - \bar{v})}{A(t)^2}.$$

We write $t^{\odot 2}$ for the vector with entries t_j^2.

Main result. The forecast error $\widehat{Q}^{\mathrm{inv}}_{m,k}(N) - Y^{\downarrow}_{1:N}$ admits the following finite-k expansion. The first term on the right of Eq. (4) is the rank effect that does not vanish on the EVT scale even when q is exactly linear; the second is the curvature term; the third is the higher-order remainder.

Proposition 1 (finite-k inverse-fit expansion)**.** *Fix k and r. Suppose that q is three times continuously differentiable on the relevant tail interval. For $B_0, \eta > 0$, define the event*

$$\mathcal{E}_{B_0,\eta} \coloneqq \left\{ \max_{j \le k} |T_{j,m}| \le B_0,\ |r + W_N| \le B_0,\ A(T_m) \ge \eta \right\},$$

where $T_m = (T_{1,m}, \dots, T_{k,m})$. The limiting variables T, ζ, and $A(T)$ are tight, and $A(T) > 0$ almost surely by the co-monotonicity of a_j and $-\log \Gamma_j$ in j, so for every $\delta > 0$ one can pick B_0 and η such that $\liminf_{m,N\to\infty} \Pr(\mathcal{E}_{B_0,\eta}) \ge 1 - \delta$. There is a constant $\varepsilon_{k,r,B_0,\eta} > 0$ such that the following expansion holds whenever

$$\frac{\sup_{|s-y|\le B_0} |q''(s)|}{q'(y)} + \frac{\sup_{|s-y|\le B_0} |q'''(s)|}{q'(y)} \le \varepsilon_{k,r,B_0,\eta}. \tag{3}$$

On the event $\mathcal{E}_{B_0,\eta}$,

$$\begin{aligned}\widehat{Q}^{\mathrm{inv}}_{m,k}(N) - Y^{\downarrow}_{1:N} &= q'(y)\left(\mathcal{I}_r(T_m) - (r + W_N)\right) \\ &\quad + \frac{q''(y)}{2}\left(D\mathcal{I}_r(T_m)[T_m^{\odot 2}] - (r + W_N)^2\right) + \mathcal{R}_{m,N},\end{aligned} \tag{4}$$

where the remainder obeys

$$|\mathcal{R}_{m,N}| \le C_{k,r,B_0,\eta}\, q'(y) \left[\left(\frac{\sup_{|s-y|\le B_0} |q''(s)|}{q'(y)} \right)^2 + \frac{\sup_{|s-y|\le B_0} |q'''(s)|}{q'(y)} \right] \tag{5}$$

for a finite constant $C_{k,r,B_0,\eta}$.

Proof. On the event $\mathcal{E}_{B_0,\eta}$, all relevant order-statistic locations lie within a fixed window around y, and the denominator of $\mathcal{I}_r$ is bounded away from zero. Taylor's theorem gives, for each $j \le k$,

$$q(y + T_{j,m}) = q(y) + q'(y)\,(T_{j,m} + \delta_{j,m})\,,$$

where

$$\delta_{j,m} := \frac{q''(y)}{2q'(y)} T_{j,m}^2 + \rho_{j,m}, \qquad |\rho_{j,m}| \le \frac{B_0^3}{6} \frac{\sup_{|s-y|\le B_0} |q'''(s)|}{q'(y)}. \tag{6}$$

By the equivariance identity in Eq. (2),

$$\widehat{Q}^{\mathrm{inv}}_{m,k}(N) = q(y) + q'(y)\mathcal{I}_r(T_m + \delta_m),$$

where $\delta_m = (\delta_{1,m}, \dots, \delta_{k,m})$.

Since $A(T_m) \ge \eta$ and T_m is bounded on $\mathcal{E}_{B_0,\eta}$, condition (3) ensures that $T_m + \delta_m$ remains in a neighborhood where $\mathcal{I}_r$ has bounded first and second derivatives. A Taylor expansion of $\mathcal{I}_r$ gives

$$\mathcal{I}_r(T_m + \delta_m) = \mathcal{I}_r(T_m) + D\mathcal{I}_r(T_m)[\delta_m] + O_{k,r,B_0,\eta}(\|\delta_m\|^2). \tag{7}$$

Using Eq. (6) in the linear term yields

$$D\mathcal{I}_r(T_m)[\delta_m] = \frac{q''(y)}{2q'(y)} D\mathcal{I}_r(T_m)[T_m^{\odot 2}] + O_{k,r,B_0,\eta}\left(\frac{\sup_{|s-y|\le B_0} |q'''(s)|}{q'(y)} \right).$$

The quadratic remainder in Eq. (7) contributes the squared-curvature term in Eq. (5).

The deployment maximum has the ordinary Taylor expansion

$$Y^{\downarrow}_{1:N} = q(y) + q'(y)(r + W_N) + \frac{q''(y)}{2}(r + W_N)^2 + O\left(\sup_{|s-y|\le B_0} |q'''(s)| \right)$$

valid on $\mathcal{E}_{B_0,\eta}$. Subtracting this expression from the expansion for $\widehat{Q}^{\mathrm{inv}}_{m,k}(N)$ gives Eq. (4) and the stated bound. □

G.3. The fixed-k rank term

The first term in Eq. (4) remains on the natural extreme-value scale even when the tail-quantile curve is linear. This is the finite-k rank effect, term (a) of the decomposition: a forecast bias that exists at every $R > 1$ regardless of how well the tail satisfies the Gumbel-domain assumption the method rests on.

For fixed k, the lower order statistics of survival values satisfy

$$(mU_{1:m}, \dots, mU_{k:m}) \Rightarrow (\Gamma_1, \dots, \Gamma_k),$$

where

$$\Gamma_j := E_1 + \cdots + E_j, \qquad E_i \overset{\mathrm{iid}}{\sim} \mathrm{Exp}(1).$$

Also $NV_{1:N} \Rightarrow E$, for an independent $\mathrm{Exp}(1)$ variable E. Hence

$$T_m \Rightarrow T := (-\log \Gamma_1, \dots, -\log \Gamma_k), \qquad W_N \Rightarrow \zeta := -\log E.$$

If, for every fixed B_0,

$$\frac{\sup_{|s-y|\le B_0} |q''(s)|}{q'(y)} \to 0, \qquad \frac{\sup_{|s-y|\le B_0} |q'''(s)|}{q'(y)} \to 0, \tag{8}$$

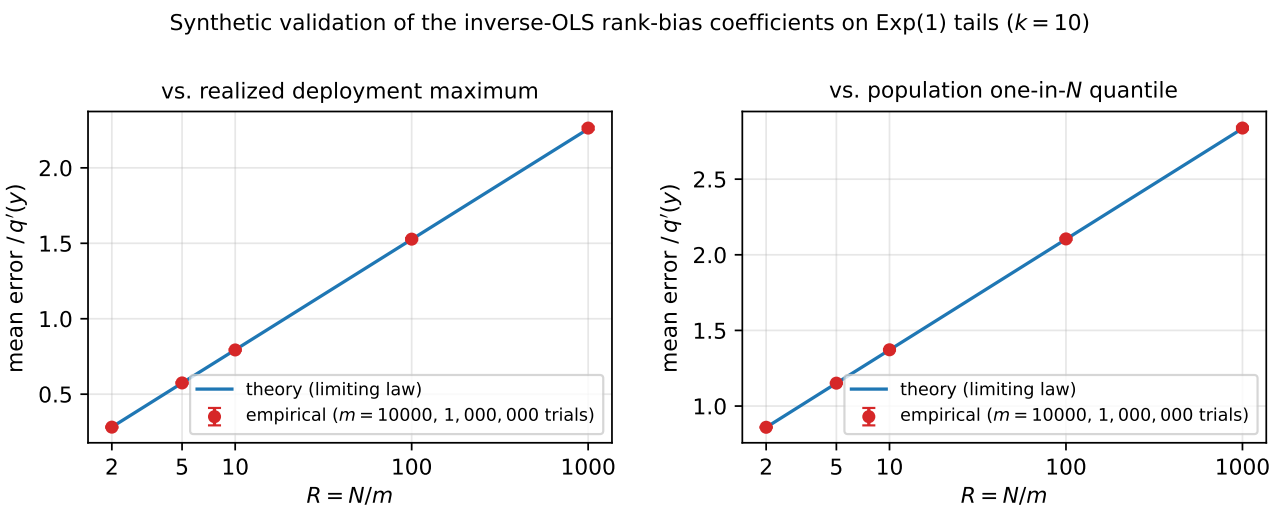

Figure 6. **Rank-term coefficient, confirmed by simulation.** Even on a perfectly linear tail-quantile curve (Exp(1), so $q(y) = y$, $q'(y) = 1$, $q'' = 0$, isolating term (a) of the decomposition), the inverse-OLS estimator with $k = 10$ has the predicted non-vanishing finite-k rank bias. We run the actual estimator on $m = 10{,}000$ evaluation samples per trial and average over one million independent trials per R. Left: empirical mean error against the realized deployment maximum, compared to the theoretical $b^{\text{inv}}_{10,\log R}$ from Section G.3. Right: empirical mean error against the population one-in-N quantile, compared to $\tilde{b}^{\text{inv}}_{10,\log R}$. Error bars are 1.96 SE across trials; theory and empirical agree at the ~ 0.005 level across all five R values. The match provides a strong validation check on the decomposition and confirms that the rank effect is a real finite-sample property of the estimator, not an artifact of the limiting calculation.

then Proposition 1 implies the asymptotic statement below. Eq. (8) is the smooth-quantile counterpart of the standard von Mises condition for membership in the Gumbel maximum domain of attraction; it is satisfied asymptotically by typical Gumbel-domain tails, including exponential ($q'' = q''' = 0$ exactly) and lognormal ($q''/q' = O(1/\sqrt{y})$), but fails for Fréchet-domain tails such as Pareto, where q''/q' is bounded below.

$$\frac{\widehat{Q}^{\text{inv}}_{m,k}(N) - Y^{\downarrow}_{1:N}}{q'(y)} \Rightarrow \mathcal{I}_r(T) - (r + \zeta).$$

The limiting mean is

$$b^{\text{inv}}_{k,r} := \mathbb{E}\left[\mathcal{I}_r(T) - r\right] - \gamma,$$

where γ is Euler's constant. If the forecast is compared to the population quantile $q(y + r)$, the corresponding coefficient is

$$\tilde{b}^{\text{inv}}_{k,r} := \mathbb{E}\left[\mathcal{I}_r(T) - r\right].$$

These constants depend only on k and $R = N/m$. Monte Carlo evaluation of the limiting distribution gives the following values for $k = 10$:

R	$b^{\text{inv}}_{10,\log R}$	$\tilde{b}^{\text{inv}}_{10,\log R}$
2	0.282	0.859
5	0.574	1.151
10	0.794	1.371
100	1.526	2.103
1000	2.258	2.835

The first numeric column is the expected gap to the realized deployment maximum, in units of $q'(y)$. The second column is the expected gap to the population one-in-N quantile, in the same units. Thus an exponential score tail, for which $q''(y) = 0$, still has a non-vanishing finite-k gap under the inverse-OLS fit.

G.4. Curvature and hazard

The second term in Eq. (4) shows how curvature enters the finite-k estimator. Its exact finite-k coefficient is random, because inverse OLS is nonlinear in the top order statistics, and at fixed k that random coefficient is $O(1)$. Under the

smoothness condition in Eq. (3), what becomes small is the curvature term *as a whole* relative to the rank term, by the factor $\sup|q''|/q'(y)$, rather than the randomness inside the coefficient. The deterministic-nominal calculation below therefore gives a population-level sign intuition; realized signs at finite k are random, and the expected sign should be checked empirically when q''/q' is not small. The deterministic counterpart is obtained by replacing T_m with the nominal vector a and replacing W_N by zero:

$$\frac{q''(y)}{2}\left(D\mathcal{I}_r(a)[a^{\odot 2}] - r^2\right). \tag{9}$$

For $r \geq 0$, the bracket in Eq. (9) is negative. It is the error made by extrapolating the convex function $u \mapsto u^2$ with a line fitted on the nominal offsets $a_1, \ldots, a_k$. Therefore convex q pushes the forecast downward relative to the true deployment quantile, while concave q pushes it upward.

This sign has a simple hazard-rate interpretation. Let

$$H(x) := -\log \bar{F}(x), \qquad h(x) := H'(x).$$

Since $q = H^{-1}$,

$$q''(y) = -\frac{h'(q(y))}{h(q(y))^3}.$$

Increasing hazards make q concave and contribute to overprediction. Decreasing hazards make q convex and contribute to underprediction. This curvature effect is smaller than the fixed-k rank term under the local-linearity condition in Eq. (8), but it can dominate practical errors when $q'(y)$ is small or the fit-to-deploy interval is highly curved.

G.5. Rare-mode occupancy

The smooth expansion describes a single tail-quantile curve. A rare latent mode adds an occupancy term, because the evaluation and deployment samples may contain different components.

Let

$$F = (1-\epsilon)F_0 + \epsilon F_1,$$

where F_1 is the rare component. Let M_m and L_N be the numbers of rare-component samples in the evaluation and deployment sets. Then

$$M_m \sim \mathrm{Binomial}(m, \epsilon), \qquad L_N \sim \mathrm{Binomial}(N, \epsilon).$$

The hidden-mode regime is

$$m\epsilon \ll 1, \qquad N\epsilon \gtrsim 1.$$

In this regime,

$$\Pr(M_m = 0, L_N \geq 1) = (1-\epsilon)^m\left[1-(1-\epsilon)^N\right] \approx e^{-m\epsilon}(1-e^{-N\epsilon}).$$

Conditional on $M_m = 0$, the fit uses only bulk samples. Let B_{N-L_N} be the maximum bulk deployment score, and let R_{L_N} be the maximum rare deployment score, with $R_0 = -\infty$. Then

$$Y^{\downarrow}_{1:N} = B_{N-L_N} + (R_{L_N} - B_{N-L_N})_+.$$

Therefore, conditional on $M_m = 0$,

$$\widehat{Q}^{\mathrm{inv}}_{m,k}(N) - Y^{\downarrow}_{1:N} = \left(\widehat{Q}^{(0),\mathrm{inv}}_{m,k}(N) - B_{N-L_N}\right) - (R_{L_N} - B_{N-L_N})_+.$$

The first term is the ordinary bulk forecast error. The second term is the rare-mode occupancy gap. It is negative exactly when the rare deployment maximum exceeds the bulk deployment maximum.

When $m\epsilon = \Theta(1)$, the top-k evaluation statistics contain a random number of rare-mode samples. This boundary regime can have large forecast variance, because changing the rare-sample count changes the fitted line. When $m\epsilon \gg k$, the top-k fit usually lies inside the rare component, and the smooth expansion applies to the rare component itself.

H. Method and baseline effects under the finite-k decomposition

This appendix uses the finite-k expansion in Appendix G to separate effects that are easy to conflate empirically. A method can make the true deployment maximum smaller, make the forecast line more accurate, or apply a post-hoc correction to the reported forecast. These interventions act on different terms of the forecast error. The three methods reported in the main figures are forecastability training with the improving-only mask (*Ours*), supervised fine-tuning on the risk score over the union of fit and deploy tasks (*SFT*), and post-hoc two-parameter affine calibration (*Cal.*); for completeness the analysis below also considers fit-only and deploy-only variants of risk-score fine-tuning, the one-parameter post-hoc shift, and alternative forecasting estimators that are not run in the main experiments.

H.1. The error components

We apply the decomposition of Appendix G under model parameters θ. Each term carries an implicit θ subscript: writing F_θ for the score distribution, $q_\theta(y) \coloneqq F_\theta^{-1}(1-e^{-y})$ for its tail-quantile curve, and $\Delta_\theta \coloneqq \widehat{Q}^{\mathrm{inv}}_{m,k,\theta}(N) - Y^{\downarrow}_{1:N,\theta}$ for the forecast error, we have $\Delta_\theta = R_\theta + C_\theta + o_p(q'_\theta(y))$ in the smooth single-component case, with $R_\theta \propto q'_\theta(y)$ the rank term and $C_\theta \propto q''_\theta(y)$ the curvature term. In the latent-mixture case of Appendix G.5, an occupancy gap $G_\theta \coloneqq (R_{L_N,\theta} - B_{N-L_N,\theta})_+$ subtracts from Δ_θ on the event that the rare component is absent from evaluation but present at deployment.

The new term needed for asymmetric-training baselines is the *split-mismatch term*. Although the fit-side and deploy-side task distributions coincide by construction in our experiments, asymmetric training causes the score distribution under θ to differ between the two task subsets, producing $q_\theta^{\mathrm{fit}} \neq q_\theta^{\mathrm{dep}}$ as a generalization-gap effect. Writing q_θ^{fit} and q_θ^{dep} for the resulting quantile curves on each side, the forecast error gains

$$S_\theta \coloneqq q_\theta^{\mathrm{fit}}(y+r) - q_\theta^{\mathrm{dep}}(y+r).$$

This term vanishes when θ is fitted symmetrically across the two sides, as in forecastability training and union SFT, and becomes important for deploy-only and fit-only baselines, which give the optimizer different access to each side of the forecast problem.

Putting the pieces together, the useful heuristic decomposition is

$$\Delta_\theta \approx R_\theta + C_\theta + S_\theta - G_\theta. \tag{10}$$

This expression should not be read as an orthogonal decomposition of squared loss. Cross-terms can matter: a method can reduce $\mathbb{E}[\Delta_\theta^2]$ by making the curvature term cancel the rank term at the chosen k and R, and a method that works by cancellation may fail when k or R changes.

H.2. Forecastability training

The forecastability loss directly optimizes the quantity the decomposition expands. Ignoring regularization, it minimizes

$$\mathcal{L}_{\mathrm{forecast}}(\theta) \coloneqq \mathbb{E}\left[\Delta_\theta^2\right]$$

over training forecast tasks. This objective can change all model-dependent terms in Eq. (10). It can shrink the rank term through $q'_\theta(y)$. It can reduce curvature through $q''_\theta(y)$. It can reduce the rare-mode gap G_θ by lowering the rare-mode maximum or by bringing the rare and bulk tails closer together.

The fixed-k rank law is the term forecastability training cannot remove by tail shaping. In the leading term,

$$\frac{R_\theta}{q'_\theta(y)} = \mathcal{I}_r(T_m) - (r + W_N).$$

The distribution on the right depends only on the order statistics and the estimator. Forecastability training can make the same normalized rank error smaller in score units by decreasing $q'_\theta(y)$. It cannot make the normalized fixed-k rank effect vanish unless the estimator changes.

This distinction gives a precise interpretation of the fine-tuning effect. If the post-training distribution has smaller $q'_\theta(y)$, the rank residual is smaller in absolute score units. If it has smaller $q''_\theta(y)$ on the fit-to-deploy interval, the curvature residual is smaller. If the rare component no longer dominates deployment, the occupancy gap is smaller. These are separate mechanisms, and the loss can improve through any combination of them.

The regularizer determines whether forecastability training improves safety or only improves forecasts. Without a primary-risk constraint, the optimizer can reduce forecast error by increasing fit-side risk scores, decreasing deploy-side risk scores, or moving both. In the language-model setting, this can produce a degenerate solution where the model leaks broadly and the forecast becomes accurate because the tail saturates. The KL-to-base term or return regularizer limits this behavior, but it does not by itself require every forecast improvement to be a safety improvement.

Improving-only gradient masks. Improving-only gradients add a safety constraint to the forecastability objective. The mask keeps gradient contributions only when they also improve the primary objective on the selected side of the pair. This changes which components of Eq. (10) the optimizer can use.

When the mask is applied to both fit and deploy contributions, the optimizer can no longer fix underprediction by raising the fit-side line through higher risk scores. It must reduce the deploy-side maximum, compress the tail scale, or reduce curvature using risk-improving directions. This explains why the mask can improve actual safety while worsening forecast error. The forecast line may stay anchored near the base model, while the deployment maximum moves downward in a pair-dependent way.

This pair-dependent movement also explains why affine calibration can fail after improving-only training. If the mask lowers rare-mode scores by different amounts across targets, then the occupancy gap G_θ becomes heteroscedastic. A one- or two-parameter affine map can remove a mean shift, and it can remove a linear dependence on the raw forecast. It cannot remove a residual whose size depends on which rare-mode examples were present and how strongly the mask affected them.

The deploy-only and fit-only masks isolate the same mechanism. A deploy-only mask allows the fit-side line to move freely while constraining deploy-side risk. It can therefore recover some forecast-quality solutions that the both-side mask forbids, while still protecting the deployment maximum. A fit-only mask protects the line-fitting side but leaves the deployment maximum free to move. It can improve the forecast loss by moving the target rather than by improving deployment safety, so it is useful mainly as a diagnostic.

H.3. Post-hoc affine calibration

Post-hoc calibration changes the reported forecast without changing the model. It therefore cannot change $q'_\theta(y)$, $q''_\theta(y)$, or the rare-mode gap. It can only remove components of the forecast error that are predictable from calibration data.

The one-parameter shift fits

$$\widehat{Q}^{\mathrm{cal},1} := \widehat{Q}^{\mathrm{inv}} + \beta, \qquad \beta^* := \mathbb{E}\left[Y^{\downarrow}_{1:N} - \widehat{Q}^{\mathrm{inv}}\right]$$

on the calibration distribution. Thus it removes the mean of the total error:

$$\mathbb{E}\left[\widehat{Q}^{\mathrm{cal},1} - Y^{\downarrow}_{1:N}\right] = 0.$$

It removes the rank term completely only in the special case where $q'_\theta(y)$ is constant across calibration pairs and the remaining terms have constant mean. If $q'_\theta(y)$ varies, the residual rank error contains

$$b^{\mathrm{inv}}_{k,r}\left(q'_\theta(y) - \mathbb{E}[q'_\theta(y)]\right),$$

where $b^{\mathrm{inv}}_{k,r}$ is the inverse-fit rank-bias coefficient. The shift also leaves the conditional variation in C_θ, S_θ, and G_θ.

The two-parameter affine calibration fits

$$\widehat{Q}^{\mathrm{cal},2} := \alpha\widehat{Q}^{\mathrm{inv}} + \beta.$$

Equivalently, it projects the residual onto the linear span of 1 and $\widehat{Q}^{\mathrm{inv}}$. It removes any component whose conditional mean is affine in the uncalibrated forecast. This includes many finite-k rank effects and some curvature effects when the curvature-to-scale ratio is stable across calibration pairs. It does not remove the hidden-mode occupancy gap unless the gap is also predictable from the uncalibrated forecast.

This limitation is sharp in the hidden-mode regime. Conditional on $M_m = 0$, the fit sample contains no rare-mode information. The raw forecast is therefore a function of the bulk scores. The event $L_N \geq 1$ and the value of the rare-mode maximum can vary while the bulk-based forecast remains similar. An affine map of the forecast cannot infer this missing occupancy information.

Stacking calibration after forecastability training is a direct composition of the two analyses. If forecastability training has reduced curvature and occupancy terms, the remaining error is closer to an affine rank residual. A one- or two-parameter calibration can then remove the mean residual. The possible gain from stacking decreases when forecastability training has already compressed $q'_\theta(y)$, because the absolute rank residual is then smaller.

Jointly training θ and an affine calibration map is the profiled version of the same objective. If α and β are unregularized and fitted on the same forecast loss, then for each fixed θ the best affine map is the post-hoc least-squares map. The joint objective is therefore

$$\min_\theta \left[\min_{\alpha,\beta} \mathbb{E}\left[(\alpha \widehat{Q}_\theta + \beta - Y_\theta)^2 \right] \right] + \lambda \mathcal{L}_{\mathrm{reg}}(\theta).$$

Joint training and sequential (forecastability + post-hoc calibration) training share the same final prediction family for any fixed θ, but they generally optimize different objectives over θ: in the sequential procedure θ is fitted to the uncalibrated forecast loss and the calibrator is then fit to the post-hoc residual, whereas the joint procedure fits θ against the affine-profiled residual. The two need not agree on θ^*, and we do not claim the joint estimator is a strict superset of the sequential one.

H.4. Primary-objective fine-tuning

Primary-objective fine-tuning changes the score distribution without using the forecast residual. It can make the model safer, but it does not target the relation between the fit-side line and the deployment maximum.

Fine-tuning on the union of fit and deploy tasks can reduce all score levels that the optimizer reaches. If it compresses the upper tail, it reduces $q'_\theta(y)$ and therefore the absolute rank term. If it happens to make the tail-quantile curve more linear, it reduces $q''_\theta(y)$. If it lowers rare-mode scores more than bulk scores, it reduces G_θ. None of these effects is enforced by the primary objective. The same training can lower the bulk more than the rare mode, increasing the rare-mode gap even as the average risk improves.

Deploy-only fine-tuning adds a split-mismatch term. If the deploy-side tail is suppressed and the fit-side tail is unchanged, then $q_\theta^{\mathrm{fit}}(y+r)$ can sit above $q_\theta^{\mathrm{dep}}(y+r)$, producing overprediction. If the deploy-only training fails to generalize to the held-out rare mode, the occupancy gap remains and the forecast can still underpredict. The sign is therefore not fixed by the decomposition. The prediction is that deploy-only training should create larger variation in S_θ across targets than forecastability training, because it changes the two sides of the extrapolation problem asymmetrically.

Fit-only fine-tuning creates the opposite split mismatch. It can move the fitted line without reducing the deployment maximum. This may reduce overprediction in a diagnostic setting, but it gives no safety guarantee. In hidden-mode settings it can worsen underprediction by lowering the bulk fit-line while leaving the rare deploy maximum unchanged. For this reason it is theoretically dominated by union fine-tuning as a safety baseline, and it is mainly useful for diagnosing whether the forecast line or the deployment maximum is driving an error.

Benign-only continued training has the same structure as fit-only training when the rare component is excluded. It can make benign prompts safer or more base-like, but it does not directly reduce the rare-mode maximum. If it lowers the bulk tail while leaving the rare tail fixed, the occupancy gap G_θ increases. This is why benign-only training is a weak baseline for a hidden rare-mode failure.

H.5. Hybrid baselines

Hybrid baselines combine primary-objective fine-tuning with post-hoc calibration. Their behavior follows from the two previous subsections. The fine-tuning stage changes $q'_\theta(y)$, $q''_\theta(y)$, S_θ, and G_θ. The calibration stage removes the part of the remaining error that is constant or affine in the uncalibrated forecast.

The union-SFT plus affine-calibration baseline should improve over either component when the remaining residual is mostly rank bias or stable curvature bias. It should not match forecastability training in hidden-mode cases where the residual depends on rare-mode occupancy not visible in the fit sample. In those cases, the fine-tuning objective must reduce the rare-mode gap itself. Union SFT may do this incidentally, but it has no forecast-level pressure to align the bulk line with the rare deployment maximum.

The deploy-only oracle plus affine calibration is the strongest member of this baseline family, but it still has the same structural limitation. Calibration can correct a systematic split mismatch. It cannot correct a pair-specific occupancy gap unless the gap is predictable from the raw forecast or from features included in the calibrator. If this oracle baseline fails, the

failure is evidence that the residual is non-affine occupancy variation rather than a global line-bias error.

H.6. Alternative forecasting estimators

Changing the estimator changes the rank term directly. This is the only way to reduce the normalized fixed-k rank effect without changing the model.

Increasing k changes the limiting rank functional from $\mathcal{I}_{k,r}(T) - (r + \zeta)$ to $\mathcal{I}_{k',r}(T) - (r + \zeta)$. It often reduces variance because more order statistics enter the fit. It does not guarantee a smaller bias for every R. The cost is curvature: the fit interval grows from roughly $[y - \log k, y]$ to $[y - \log k', y]$, so the curvature contribution scales with the nonlinearity of q_θ on a wider interval. A useful rule of thumb is

$$\text{error}(k) \approx q'_\theta(y) \cdot \text{rank}(k, r) + q''_\theta(y) \cdot \text{curv}(k, r),$$

where the first factor tends to improve with k and the second can worsen. This predicts a U-shaped dependence on k in curved tails.

Changing the plotting position is a finite-sample correction to the rank term. If the nominal offsets are changed from $a_j = -\log j$ to adjusted offsets $\tilde{a}_j$, then the inverse-fit functional and its rank-bias coefficient change. The curvature and occupancy mechanisms remain the same. In principle, one can choose plotting positions that make the exponential-tail rank bias vanish for a chosen k and R. That correction does not address curvature or hidden modes.

A peaks-over-threshold generalized Pareto fit replaces the log-linear tail model with a different parametric assumption. It can represent regularly varying tails that the Gumbel-tail line fit underpredicts. It still has a finite-threshold estimation error, and it still cannot see a rare mode that is absent from the evaluation sample. Thus it changes the curvature/model-bias term but leaves the occupancy problem in place.

H.7. Summary by method

Table 1 summarizes the component-level predictions. The entries describe what each method can reduce structurally. A method can still improve squared error through cancellation between components.

The table describes what each method can reduce structurally. Empirical reductions in squared error can exceed these structural predictions when cross-terms cancel at the chosen (k, R), and methods working primarily by cancellation are sensitive to those choices and may not generalize as k or R varies. We discuss specific instances of cancellation in the experiments section.

H.8. Sharp predictions

The decomposition gives several predictions that can be checked without changing the main experiments.

First, on an exponential score tail, $q''(y) = 0$. Any systematic finite-k error then comes from the rank term. The inverse-fit constants in Appendix G therefore predict the mean forecast error after scaling by $q'(y)$. This is the cleanest validation of the rank term.

Second, post-hoc affine calibration should help most when residuals are affine in the raw forecast. It should help less when residuals are dominated by rare-mode occupancy. The failure mode is strongest when two targets have similar bulk fit forecasts but different rare-mode deployment gaps.

Third, improving-only training should improve actual safety when it acts on rare deploy examples, but it need not improve forecast error. The forecast error can remain large if the fit-side line stays near the base model while the deploy maximum moves down by a target-dependent amount. When this happens, calibration can fail to help if the residual heteroscedasticity introduced by target-dependent deploy movement is large relative to the mean residual that calibration removes.

Fourth, deploy-only primary training should create a visible gap between fit-side and deploy-side quantile curves. Plotting empirical fit and deploy quantiles before applying the Gumbel-tail forecast should reveal whether its forecast error is coming from the split-mismatch term S_θ.

Fifth, deploy-only primary fine-tuning (an oracle that sees the deployment tasks at training time) should be *worse* than forecastability training on forecast error in the typical hidden-mode regime, despite the additional information. Forecastability

Method	Rank term	Curvature term	Occupancy term	Split mismatch
Forecastability loss	Reduces only through $q'_\theta(y)$; normalized law remains.	Directly targeted through forecast residual.	Directly targeted when rare deploy examples affect the loss.	None if fit and deploy are treated symmetrically.
Ours (forecastability + improving-only mask, both sides)	Reduced through safe compression of $q'_\theta(y)$.	Reduced only along primary-improving directions.	Often reduced by lowering rare deploy scores; can become heteroscedastic.	Usually small by construction, but the line can stay anchored while deploy moves.
Post-hoc shift (1-param)	Removes mean rank bias if $q'_\theta(y)$ is stable.	Removes mean curvature bias only.	Removes mean occupancy gap only.	Removes mean split mismatch only.
Cal. (post-hoc affine)	Removes rank effects affine in the raw forecast.	Removes stable affine curvature effects.	Does not remove hidden occupancy variation.	Removes affine split mismatch.
SFT (union)	Can shrink $q'_\theta(y)$ incidentally.	No direct pressure toward linear tails.	Reduced only if rare-mode scores are lowered relative to bulk.	None by design.
SFT (deploy-only)	Fit-side rank scale may be unchanged.	Fit-side curvature may be unchanged.	Can reduce deploy rare scores if it generalizes.	Can be large; sign depends on generalization.
SFT (fit-only)	Can move the fit-line scale.	Can reshape fit curvature only.	Does not reduce deploy rare maximum.	Can be large and unsafe.
SFT+cal.	Combines SFT's scale change with affine removal.	Removes only affine residual curvature.	Depends on SFT; calibration cannot infer hidden occupancy.	Calibration removes only affine mismatch.
Larger k	Changes and often reduces finite-k rank noise.	Can increase curvature bias by widening the fit interval.	No effect.	No effect.
Plotting-position change	Changes rank constants.	No structural effect.	No effect.	No effect.
GPD tail fit	Different finite-sample rank and parameter noise.	Reduces bias if GPD shape is correct.	No effect when rare mode is absent.	No effect.

Table 1. Component-level effects predicted by the finite-k decomposition. Italicized labels (*Ours*, *Cal.*, *SFT*, *SFT+cal.*) match the conditions reported in Figures 3 and 4; the unitalicized rows are completeness analyses for diagnostics or estimator alternatives that are not run in the main experiments.

training treats the fit-to-deploy axis symmetrically and so has $S_\theta \approx 0$; deploy-only fine-tuning incurs a non-vanishing S_θ contribution by construction. The prediction is therefore not that forecastability training sees more data, but that it sees the right axis of the forecast residual.

Finally, increasing k should help on exponential or nearly linear tails and can hurt on curved or mixture tails. This gives a simple diagnostic for whether an observed error is mostly rank noise or mostly tail-shape bias.

I. Empirical exploration of the decomposition on WildChat

This appendix reports the empirical workup that the closing paragraph of Section 4 summarizes. We score WildChat-1M conversations (Zhao et al., 2024) under the post-trained Qwen3-0.6B reference (Qwen Team, 2025) (the same checkpoint used as the frozen reference in our LM experiments) with four risk-score families, simulate inverse-OLS forecasts across (M, N) cells, and compare the empirical mean error per cell to the Section 4 decomposition's prediction at the same (k, R).

Slice infrastructure. We use the WildChat-1M-Full snapshot (1,039,785 multi-turn conversations including the toxic subset). The default WildChat-1M release excludes toxic conversations and yields 837,989 rows; reproducing the harmful-token slice below therefore requires the gated Full variant rather than the default release. We restrict to the first English user turn under each conversation, giving a 602,365-row *all-English* slice and a 121,097-row *flagged-or-toxic* subslice (any first-turn flagged by the OpenAI moderation classifier in the released metadata). Per-prompt scores are computed on the

all-English slice for assistant turn length and Detoxify (Hanu & Unitary team, 2020) toxicity, on the flagged-or-toxic subslice for the harmful-token-union log-probability, and on a 33% random subsample of the all-English slice ($n = 198{,}656$) for per-token mean NLL. The harmful-token set is the LDNOOBW English wordlist (LDNOOBW Contributors, 2023) filtered to 38 probe strings that resolve to single first-response tokens after the Qwen3 chat template; the per-prompt score is the log-sum-exp of the 38 next-token log-probabilities.

Hazard-rate diagnostics. The Section 4 curvature term's sign is determined by the hazard rate of F_θ at the expansion anchor $y = \log M$: increasing-hazard tails reinforce rank-term overprediction. Figure 7 plots the histogram, empirical survival, and log hazard for each of the four scores. All four show rising hazard *near their upper endpoints*, but that feature lives in the deeper tail rather than at the anchor; in the Section 4 framework it is folded into the higher-order remainder rather than into the local-quadratic curvature term, as the empirical decomposition below confirms.

Local-quadratic estimator for $q'(y)$ and $q''(y)$. We estimate $q'(y)$ and $q''(y)$ at the target log-survival $y = \log M$ by least-squares fit of a local quadratic to the order statistics whose Weibull plotting positions fall in a window of half-width δ around y. At the canonical $\delta = 0.5$ the window contains 41 to 125 order statistics depending on the score and slice size. The $q'(y)$ estimate is robust across $\delta \in \{0.25, 0.5, 1.0\}$ (within $\pm 30\%$); the $q''(y)$ estimate is not, and the curvature/residual split shifts substantially with δ even though their sum stays stable. We use $\delta = 0.5$ for the headline figure and discuss the sensitivity below.

Empirical decomposition. Figure 8 reports the decomposition at the canonical $(M, N, k, R) = (5{,}000,\ 50{,}000,\ 10,\ 10)$ cell. The rank bar (blue) is identical at $0.794\, q'(y)$ across all five score columns; this is a property of the inverse-OLS estimator at $(k, R) = (10, 10)$ and not of the data. Empirical totals (black diamonds) range from $-0.66\, q'(y)$ on total NLL to $+2.46\, q'(y)$ on Detoxify toxicity. On per-token mean NLL the empirical total ($+0.83\, q'(y)$) closely matches rank plus curvature ($0.794 + (-0.262) = +0.53\, q'(y)$), with a small residual ($+0.30\, q'(y)$). On the other smooth-tail body scores the residual is larger and positive ($+2.36$, $+3.36$, and $+7.80\, q'(y)$ for length, harmful-token log-probability, and Detoxify toxicity respectively).

Sign of the residual: reverse occupancy. The standard occupancy contribution $-G_\theta$ in equation (1) fires when the deployment set contains a rare deploy-side component the fit set missed. Its sign is non-positive in our convention because the rare component pulls $Y_\theta^{\max}$ upward relative to what a fit-set bulk would predict, reducing the over-prediction. The structural mirror image of this mechanism is what the positive WildChat residuals look like: a finite upper endpoint in score space that the deployment set's maximum sits close to, but that the fit set's top-k does not reach. At the canonical cell, fit's top-k for $k = 10$, $M = 5{,}000$ covers log-survival depths $y \in [\log(M/k), \log(M)] = [6.2, 8.5]$, while $Y_\theta^{\max}$ from $N = 50{,}000$ deploy draws sits in expectation at $y \approx \log N = 10.8$. If $q_\theta(y)$ flattens sharply over the deploy-side window $y \in [\log M, \log N]$, as it does when the score has a finite upper endpoint τ_F with corresponding log-survival depth $y_F = -\log S_\theta(\tau_F)$ in this window, the deploy maximum is bounded by the bend while the OLS line, fit on top-k values well below it, extrapolates straight through. The result is positive bias on $Q_\theta^{\text{pred}} - Y_\theta^{\max}$. We will call this *reverse occupancy*: the mirror image of standard occupancy, with the deploy-relevant tail feature being a downward cliff rather than an upward bump. Section 4's derivation does not name it explicitly, since the smooth-tail Taylor expansion is anchored at $y = \log M$ and a bend in the deeper tail leaves no signature on $q^{(j)}(y)$ at the expansion point; reverse occupancy is therefore folded into the $O_p(q'(y))$ remainder rather than carried as its own term.

Empirical match against the reverse-Weibull reference. A simple diagnostic isolates this mechanism. We re-run the empirical-decomposition machinery at the canonical cell on Uniform$(0, 1)$, the reference distribution for the reverse-Weibull (bounded-upper-endpoint) maximum domain of attraction, and obtain rank $+0.794\, q'(y)$, curvature $+5.86\, q'(y)$, and residual $+5.65\, q'(y)$. The residual is large, positive, and of the same order as the empirical residuals on three of the four WildChat body scores (length $+2.36$, harmful-token log-prob $+3.36$, Detoxify $+7.80$). Pareto's residual at the same cell is $\approx -0.7\, q'(y)$ and the smooth Gumbel-domain references (Exp(1), Lognormal, Log-Weibull) all give residuals within $\pm 0.05\, q'(y)$. The smoothness condition $q''/q' \to 0$ that Section 4's leading expansion relies on fails in two distinct directions: a heavy Fréchet tail leaves a negative residual (Pareto), and a bounded reverse-Weibull tail leaves a positive residual (Uniform, the reverse-occupancy direction). The hazard diagnostics in Figure 7 place length, harmful-token log-prob, and Detoxify toxicity in the reverse-Weibull regime, with rising hazards approaching their respective upper endpoints, and the empirical residuals' positive sign and magnitude are consistent with that MDA assignment.

Two further checks. Two further checks support the reverse-occupancy reading. (a) Switching the empirical sampler from with-replacement (used in this appendix) to without-replacement partition-permutation (used by the analyzer that produced the k-sweep cells) changes the empirical totals by at most $\approx 25\%$: at $(M, N) = (5{,}000,\ 50{,}000)$ the with-replacement / without-replacement empirical totals are $1.85/0.96\, q'(y)$ for length, $2.46/1.60\, q'(y)$ for Detoxify, $0.83/1.13\, q'(y)$ for mean NLL, and $0.35/0.06\, q'(y)$ for harmful-token log-prob. The residuals stay positive on all four scores under either sampler. (b) Sweeping $M \in \{1{,}000,\ 5{,}000,\ 25{,}000\}$ at fixed $R = 10$ does not drive the empirical total toward the rank value $0.794\, q'(y)$ on length, harmful, Detoxify, or total NLL; per-token mean NLL is the only score whose empirical total drifts down toward the rank value as M grows. We read this as consistent with structural reverse occupancy on the rising-hazard scores rather than with finite-M Taylor noise that would vanish asymptotically: the bend in the deeper tail does not move out of the deploy-side window simply by enlarging the fit-set.

Curvature and residual are not separately well-identified. At $\delta = 0.25$ (tighter window, noisier q'') the length curvature/residual decomposition is $-4.71/ + 5.75\, q'(y)$; at $\delta = 1.0$ it is $+2.48/ - 1.55\, q'(y)$. The empirical total stays at $\approx 1.85\, q'(y)$ across δ, so the sum of curvature and residual is well-identified even when their split is not. Per-token mean NLL is the score on which the curvature/residual split is itself stable across δ (curvature in $[-0.60, -0.26]$, residual in $[+0.30, +0.54]$). On the other scores the curvature attribution depends on the local-quadratic estimator's parametric assumption beyond the leading $q'(y)$ term, and is best read together with the residual.

Total NLL is the autoregressively correlated outlier. Total assistant-turn NLL (the sum of per-token negative log-probabilities, with autoregressive coupling within each turn) is the one score where the empirical total at the canonical cell is negative, $-0.66\, q'(y)$. The variance is large: per-prompt p_{10} to p_{90} span is $\approx 3{,}000$ NLL units at $N = 25{,}000$, so the under-prediction signal lives in the median while individual forecasts can be off in either direction by orders of magnitude.

k-sensitivity sweep. Figure 9 sweeps k on assistant length and per-token mean NLL at fixed $R = 10$, alongside the Monte Carlo $\xi(k, R = 10)$ ridge from a synthetic Exp(1) tail. The empirical length curve in $q'(y)$ units stays close to the theoretical ridge through the predicted sign-flip near $k = 100$: at $k = 500$ the rank term in expectation drives forecast error below zero, and the empirical length curve reaches $-1.51\, q'(y)$. Per-token mean NLL diverges from the rank-only theory at large k, with its empirical curve climbing from $+1.13\, q'(y)$ at $k = 10$ to $+6.65\, q'(y)$ at $k = 500$. The cause is the curvature term: as k grows, the OLS fit covers a wider range of the tail, and for a score with non-trivial $q''(y)/q'(y)$ the curvature contribution grows with k and overwhelms the rank-term sign-flip.

Boundary conditions. Two notes scope what Section 5 can claim from this exploration. The four body scores under post-trained Qwen3-0.6B (Qwen Team, 2025) appear to carry positive forecast bias from reverse occupancy rather than negative bias from the standard hidden-mode occupancy of Section 6's constructed banks; the two mechanisms are structural mirror images and we do not see both signatures on the same score, though we cannot rule out other structural residuals or mixed mechanisms. The rank term is structurally fixed in expectation across all of these scores at given (k, R), so only the deploy-side tail features that the fit set systematically misses (standard occupancy, reverse occupancy, and the smooth-tail curvature) are trainable, which Section 5 formalizes.

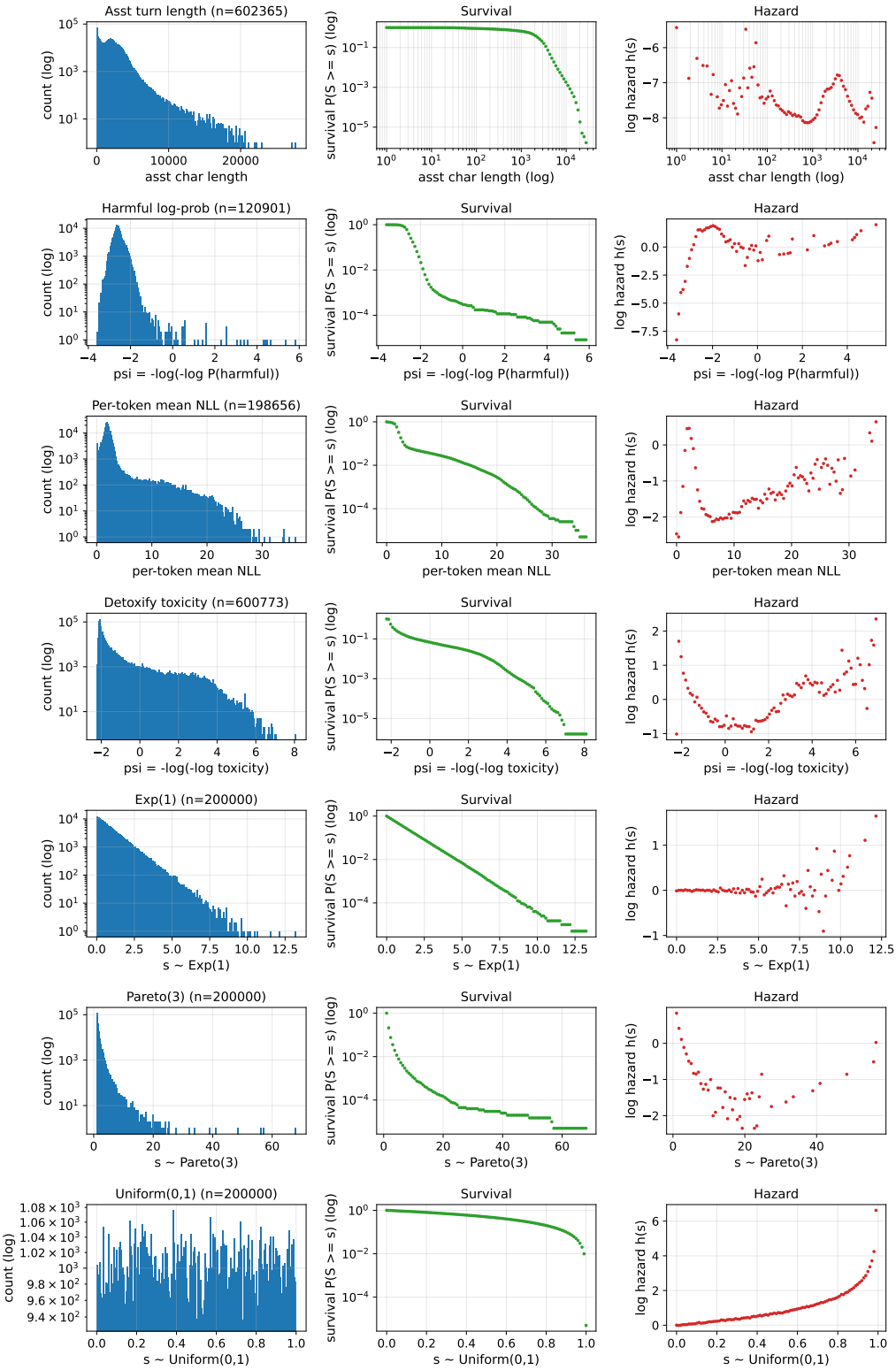

Figure 7. **WildChat hazard diagnostics with reference distributions.** Top four rows: WildChat scores (length and mean NLL raw; harmful log-prob and Detoxify after Gumbel-prob). Bottom three rows: $n = 200{,}000$ samples from Exp(1), Pareto($\alpha = 3$), Uniform$(0, 1)$ (Gumbel, Fréchet, reverse-Weibull). Rising hazard on length, harmful, and Detoxify matches the Uniform reference; mean NLL rises slowly.

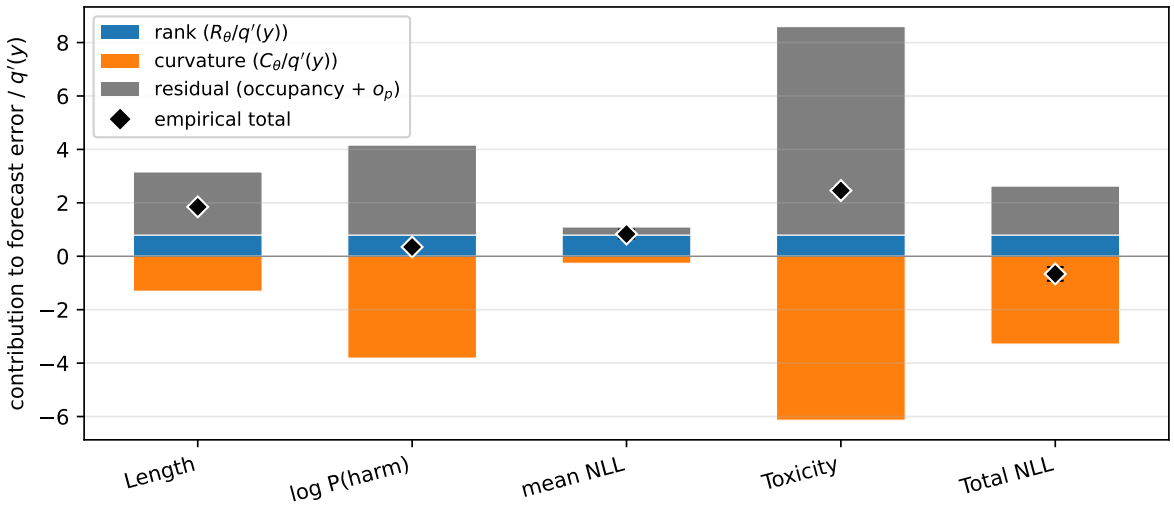

Figure 8. **Empirical decomposition at** $(M, N, k, R) = (5{,}000,\ 50{,}000,\ 10,\ 10)$**.** The blue rank bar is identical at $0.794\, q'(y)$ across all columns by construction; curvature (orange) and residual (gray) vary by score. Empirical mean errors (black diamonds) match the algebraic sum of the colored bars to within Monte Carlo error. Total NLL is the only score on which the empirical mean is negative; the four body scores have positive empirical means dominated by a positive residual on three of them.

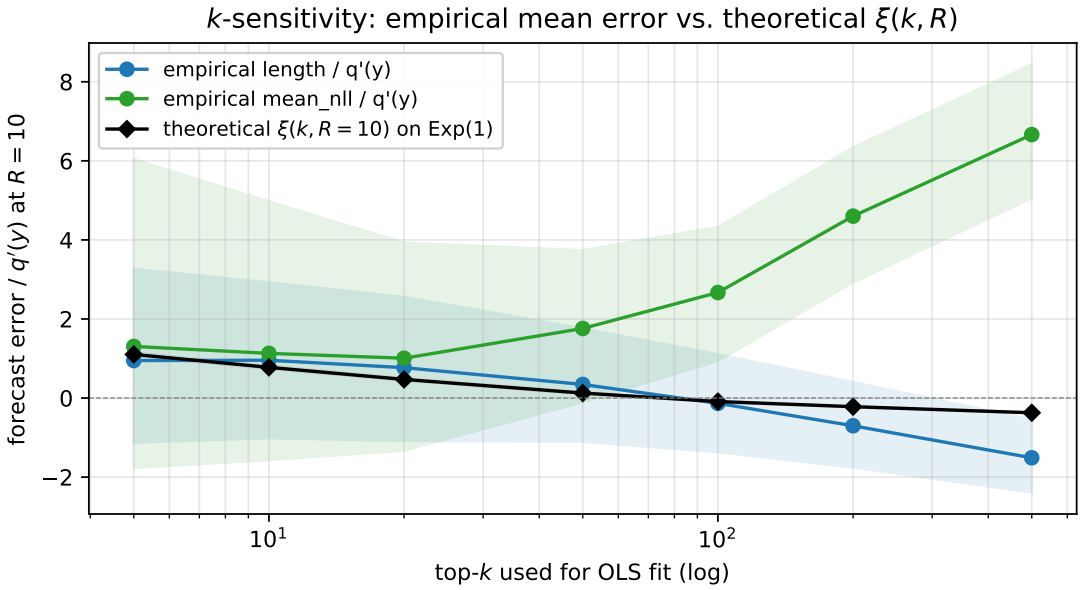

Figure 9. k**-sensitivity at fixed** $R = 10$**.** Empirical mean error in $q'(y)$ units against k, on assistant length (blue) and per-token mean NLL (green), alongside the theoretical $\xi(k, R = 10)$ ridge from a synthetic Exp(1) Monte Carlo (black). Length matches the theoretical ridge through the predicted sign-flip near $k = 100$; per-token mean NLL diverges upward as the curvature term grows with k.