Large Language Models as Generative Bayesian Policies

Anonymous Authors[1]

Abstract

Bayesian optimization is the dominant framework for sample-efficient scientific discovery, providing principled uncertainty estimates that guide expensive experiments. However, its reliance on hand-crafted, domain-specific representations limits its broad applicability. Large language models (LLMs) offer flexibility as many experimental setups can be described in text, but they lack the calibrated uncertainty required in high-stakes scientific applications. Here we introduce a framework that integrates language models and Bayesian optimization at both the representation and the generation level. We train a Gaussian process critic with deep kernel learning, using the LLM as the feature extractor to produce task-adaptive, uncertainty-calibrated representations. The same critic's acquisition function then provides a preference signal that trains the LLM to act as a generative Bayesian policy: candidates favored by the acquisition function become "preferred" completions for preference-based finetuning. The actor learns to generate high-utility candidates directly, thereby amortizing the acquisition search of standard Bayesian optimization into the language model itself. We evaluate on three scientific optimization benchmarks spanning process chemistry, peptide design, and molecular design. The actor's generation distribution shifts progressively toward higher-utility regions across iterations, and the trained model outperforms feature-based Bayesian optimization, LLM-guided optimization, and their union at different levels of integration. These results suggest that calibrated uncertainty can become the training signal that aligns language models with the decision-making demands of experimental discovery.

[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. Introduction

Decision making under uncertainty extends beyond choosing what currently looks best. With incomplete evidence, a rational agent must balance prior beliefs with new observations and act while accounting for what it does not yet know. This challenge appears acutely in expensive scientific optimization where each experiment is costly, feedback is sparse, and the next query must trade off exploration against exploitation (Mervin et al., 2021; Tavazza et al., 2021; Schwaller et al., 2022).

Recent works have shown that large language models (LLMs) can navigate experimental optimization by proposing conditions, suggesting molecular designs, or leveraging scientific context from text (Boiko et al., 2023; Bran et al., 2023; Ramos et al., 2023; MacKnight et al., 2025). These results suggest that LLMs encode useful prior knowledge that can guide experimental search, yet they operate as uncalibrated proposal mechanisms. Raw LLM generation lacks principled probabilistic reasoning under uncertainty (Kalai et al., 2025; Huang et al., 2025), which is especially problematic in high-stakes scientific domains where hallucinations and overconfidence waste experimental resources (Musslick et al., 2025; Ramos et al., 2025).

Bayesian optimization (BO), by contrast, provides a mathematically grounded framework for sequential experimentation through probabilistic surrogates and acquisition functions (Kushner, 1962; 1964; Garnett, 2023; Shahriari et al., 2015). BO's calibrated uncertainty estimates have motivated a body of research at the intersection with LLMs (Ramos et al., 2023; Ranković and Schwaller, 2023; Liu et al., 2024; Mahammadli and Ertekin, 2024; Kristiadi et al., 2024; Ranković and Schwaller, 2025; Cissé et al., 2025; 2026). Yet in these approaches, the LLM typically remains a passive component by proposing candidates, supplying priors, or providing features, while the BO machinery performs the actual uncertainty-aware decision making.

Here we ask: *can we teach an LLM to think like a Bayesian optimizer?* We introduce GGOLLuM (Generative Gaussian Process Optimized LLM), a framework that uses BO as a source of preference supervision for LLM alignment. A deep-kernel GP surrogate (Wilson et al., 2016), in which the LLM serves as the feature extractor, acts as a *critic* that scores candidate experimental conditions via an acquisition

function. These scores produce a preference signal (Christiano et al., 2017; Ouyang et al., 2022): high-acquisition candidates are "preferred" over low-acquisition ones. We then apply Identity Preference Optimization (IPO) (Azar et al., 2024), a stabilized variant of Direct Preference Optimization (DPO) (Rafailov et al., 2023), to train the LLM to internalize these preferences and transform it into a *generative Bayesian policy* that produces promising candidates directly. Unlike standard BO, which requires explicit acquisition maximization over a fixed design space at each iteration, GGOLLuM amortizes this search inside the LLM, enabling direct candidate generation.

The closest related work is ToSFiT (Menet et al., 2025), which finetunes an LLM via a variational policy-gradient objective using a GP-based advantage. GGOLLuM differs in how the surrogate signal enters training: we use off-policy preference pairs rather than an on-policy policy gradient, so the surrogate informs the actor's loss through pairwise winner/loser labels. The framework is also acquisition-agnostic as any scalar score can replace the choice of acquisition. To isolate this training-signal comparison, our TOSFIT reimplementation uses the same deep-kernel GP as GGOLLuM.

We evaluate GGOLLuM on three benchmarks spanning three representation regimes: mixed-variable reaction optimization (catechol rearrangement reactions) (Boyne et al., 2025), molecular design (dopamine receptor D2 affinity), and sequence-based biological design (antimicrobial peptide optimization). Against a ladder of seven methods that isolate the contributions of (i) LLM-generated versus random candidate pools, (ii) classical versus deep-kernel GPs, and (iii) policy-gradient versus preference-based adaptation, GGOLLuM outperforms all baselines on every benchmark, retains its advantage under batch acquisition and progressively improves generation quality over optimization iterations, displaying the behavioural signature of the actor internalizing Bayesian feedback.

2. A generative policy shaped by a Bayesian critic

GGOLLuM operates in iterative rounds of Bayesian optimization. Each iteration involves five phases: (1) the actor LLM generates a pool of $N{=}16$ candidates from its current IPO-updated distribution; (2) the GP critic computes an acquisition score for each candidate; (3) the top-b candidates are evaluated on the true objective; (4) the full pool is split into winners/losers by acquisition rank and converted into confidence-weighted preference pairs; (5) the actor is finetuned on the preference pairs via IPO, updating its LoRA adapter (Hu et al., 2022) to prefer high-acquisition candidates.

In this architecture the actor and the critic share the same base LLM but carry different roles. The actor adapter is trainable and accumulates across iterations. It is finetuned via IPO on acquisition-guided preference pairs and is active during candidate generation. The critic adapter is reset and retrained each iteration; it is finetuned jointly with the GP via marginal likelihood and is active when computing embeddings for the GP kernel. The base model serves as the reference policy π_{ref} for IPO.

3. Background

Bayesian optimization with deep kernels. Standard BO fits a Gaussian Process (GP) surrogate (Rasmussen, 2003) $f \sim \mathcal{GP}(\mu, k)$ to observed data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ and selects the next experiment by maximizing an acquisition function $\alpha(\mathbf{x})$. Deep kernel learning (Wilson et al., 2016) replaces the fixed kernel with $k(\phi(\mathbf{x}), \phi(\mathbf{x}'))$ where ϕ is a neural network. In our case, an LLM constitutes the deep kernel that maps textual descriptions of experimental conditions to embeddings. The LLM and GP are trained jointly by maximizing the GP marginal likelihood, yielding the GOLLuM critic (Ranković and Schwaller, 2025).

Acquisition functions. Given a fitted GP with posterior mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$, the acquisition function $\alpha(\mathbf{x})$ trades exploration against exploitation. We use Upper Confidence Bound (Auer, 2002; Srinivas et al., 2009) throughout: $\alpha_{\text{UCB}}(\mathbf{x}) = \mu(\mathbf{x}) + \sqrt{\beta_{\text{UCB}}}\,\sigma(\mathbf{x})$ with $\beta_{\text{UCB}} = 4$. For batched acquisition ($b{>}1$), the GP-based baselines use qUCB—the sequential-greedy Monte-Carlo extension that jointly selects b points from a candidate pool while accounting for within-batch correlations (Balandat et al., 2020). GGOLLuM does not use qUCB but takes the first-b valid novel candidates in the actor's generation order, relying on the actor's adapted distribution to produce a diverse high-utility batch (Section 7).

Direct preference optimization. DPO (Rafailov et al., 2023) trains a policy π_θ to prefer "chosen" over "rejected" completions without explicit reward modeling. Given preference pairs (y_w, y_l) with prompt x, the loss is

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log \sigma\left(\beta\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right)\right], \tag{1}$$

where π_{ref} is a reference policy and β controls the KL penalty.

4. Confidence-weighted preference pairs

Given a candidate pool $\{c_1, \ldots, c_N\}$ with GP posterior means μ_i, variances σ_i^2, and acquisition scores α_i, we construct preference pairs by (i) ranking candidates by α, (ii) splitting the ranked list at the median into winners $\mathcal{W}$ (top half) and losers $\mathcal{L}$ (bottom half), and (iii) forming the full

cross-product $\{(w, l) : w \in \mathcal{W}, l \in \mathcal{L},\ \mu_w > \mu_l\}$. Each pair is weighted by the posterior z-score of the mean-gap:

$$z_{wl} \;=\; \frac{\mu_w - \mu_l}{\sqrt{\sigma_w^2 + \sigma_l^2}}\,, \qquad \text{pair weight} = \min(z_{wl},\, 1). \tag{2}$$

The z-score has a direct interpretation: under independent Gaussian posteriors, $\Phi(z_{wl}) = P(y_w > y_l)$, so z_{wl} is monotonic in the posterior probability that y_w beats y_l. Pairs the GP cannot confidently separate contribute little to the IPO loss, while pairs with clear mean separation relative to combined uncertainty contribute most. Early in the BO run, when the GP is uncertain over much of the design space, *all* z-scores are small and the effective number of informative pairs is low; as data accumulates and the GP sharpens, the mean pair weight grows as we show in Figure 2.

5. Weighted Identity Preference Optimization

To prevent the mode collapse characteristic of standard DPO, we use the Identity Preference Optimization (IPO) variant (Azar et al., 2024), which replaces the saturating log-sigmoid loss with a bounded squared loss:

$$\mathcal{L}_{\text{IPO}}(w, l \mid x) = \left(\log \tfrac{\pi_\theta(w|x)}{\pi_{\text{ref}}(w|x)} - \log \tfrac{\pi_\theta(l|x)}{\pi_{\text{ref}}(l|x)} - \tfrac{1}{2\beta}\right)^2. \tag{3}$$

Our effective loss is the z-score-weighted mean over the pair set:

$$\mathcal{L} \;=\; \frac{1}{|\mathcal{P}|} \sum_{(w,l)\in\mathcal{P}} \min(z_{wl}, 1) \cdot \mathcal{L}_{\text{IPO}}(w, l \mid x), \tag{4}$$

with $\beta = 0.02$, learning rate 5×10^{-4}, and 5 optimizer epochs per BO iteration on minibatches of 32 pairs. The reference policy π_{ref} is always the frozen base LLM with LoRA disabled, for all BO iterations; only the LoRA adapter on top of π_{ref} is updated, and the adapter accumulates across iterations rather than being reset.

6. Iterative IPO and the sampler chain

A key property of GGOLLuM is that IPO updates accumulate over BO iterations (Xu et al., 2023; Xiong et al., 2023). Concretely, we let π_t denote the actor at the start of BO iteration t, with $\pi_0 = \pi_{\text{base}}$ (the frozen base LLM). At iteration t:

1. **Sample:** π_t generates a pool $\mathcal{C}_t = \{c_1, \ldots, c_N\}$ from its current distribution, turning the actor into the next iteration's data source.
2. **Score:** the critic (a deep-kernel GP refit on all observations so far) assigns acquisition scores $\alpha_t(c_i)$.
3. **Pair:** the candidates are split into winners/losers and converted into preference pairs.
4. **Optimize:** IPO updates the LoRA weights starting from π_t, with $\pi_{\text{ref}} = \pi_{\text{base}}$ for all iterations. The result is π_{t+1}.

Algorithm 1 summarizes one BO iteration.

Algorithm 1 One GGOLLuM BO iteration at step t.

input: history $\mathcal{D}_t$, base LLM π_{base}, current actor π_t
refit critic on $\mathcal{D}_t$: reset critic LoRA, retrain (LLM + GP) jointly
  $g_t \leftarrow \arg\max\ \log p(\mathcal{D}_t)$ {GP marginal likelihood}
sample candidate pool from the actor
  $\mathcal{C}_t \leftarrow \pi_t(\cdot), \quad |\mathcal{C}_t| = N$
score the pool with the refit critic
  $\alpha_t(c) = \mu_{g_t}(c) + \sqrt{\beta_{\text{UCB}}}\,\sigma_{g_t}(c)$ for $c \in \mathcal{C}_t$
build confidence-weighted preference pairs
  $\mathcal{P}_t \leftarrow$ pairs$(\mathcal{C}_t)$ with weights $\min(z_{wl}, 1)$
update actor via IPO
  $\pi_{t+1} \leftarrow$ IPO$(\pi_t,\ \pi_{\text{ref}}{=}\pi_{\text{base}},\ \mathcal{P}_t)$
pick first-b valid novel candidates and evaluate
  $\{c_1^\star, \ldots, c_b^\star\} \subset \mathcal{C}_t, \quad y_j^\star \leftarrow$ Evaluate$(c_j^\star)$
  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(c_j^\star, y_j^\star)\}_{j=1}^b$ {update dataset}
return $\mathcal{D}_{t+1},\ \pi_{t+1},\ g_t$

7. Experimental setup

Benchmarks. We evaluate on three scientific optimization tasks that span the typical inputs a practitioner might describe to an LLM: structured reaction parameters, molecular SMILES (Anderson et al., 1987; Weininger, 1988), and peptide sequences (Table 1). Each task is paired with a base LLM that provides a reasonable prior over that input type: a general-purpose base model for reaction optimization, and domain-specialized generators for molecules and peptides.

The base LLM used for the molecular design benchmark is pretrained to produce drug-like molecules. However, drug-likeness does not imply DRD2 activity (Olivecrona et al., 2017), and DRD2 actives constitute only a small portion of the chemical space the base model covers. The base model is in the vicinity of the target distribution but not aligned to it. The PeptideGPT variant we use (Shah et al., 2024) is pretrained on UniProt sequences and selected for non-hemolytic output. It emits valid peptide-like sequences but places no mass on the antimicrobial-likeness composite that defines our objective (Appendix A). Qwen2.5-7B (Bai et al., 2023) is general-purpose with no chemistry bias and serves to test whether the GP signal alone can carry a non-specialist base. In all three cases the base LLM is fluent in the domain but not aligned with the task, and GGOLLuM's job is to close that gap using only the critic's signal.

Methods compared. We compare seven methods along a ladder of increasing posterior awareness (Table 2). This

Table 1. Benchmarks and base LLMs used in the main evaluation.

Task	Benchmark	Input	Base LLM
Reaction	Catechol	tabular	Qwen2.5-7B
Molecular	SMILES-DRD2	SMILES	gpt2-zinc-87M
Peptide	Peptide-AMP	sequence	PeptideGPT

structure isolates three independent design choices: (i) *does the method use a GP?*, (ii) *is the candidate pool drawn by an LLM or from the raw search space?*, and (iii) *does the LLM's generation distribution adapt to the objective?*

For the reaction task, classical features are one-hot encoding of categoricals plus standard-scaled continuous parameters; for molecules, 2048-bit Morgan fingerprints (radius 2) (Rogers and Hahn, 2010); for peptides, dipeptide ($k = 2$) count vectors (400 dims). These are the task-standard classical descriptors in each domain. GOLLuM (LLM pool) and GGOLLuM both use the same deep kernel (Wilson et al., 2016) with LoRA-finetuned top 25% of transformer layers. All GP-based baselines use UCB as the single-point acquisition function with $\beta_{\text{UCB}} = 4$ and sequential-greedy qUCB for batched selection. GGOLLuM's critic uses the same UCB for ranking candidates during IPO pair construction; its batch acquisition takes the first-b valid novel candidates in the actor's generation order. For GGOLLuM we use IPO with confidence-weighted pair construction, $\beta = 0.02$, learning rate 5×10^{-4}, 5 IPO epochs per iteration with minibatches of 32 pairs. LoRA weights accumulate across iterations, and the reference policy is always the frozen base LLM.

We report 20 random seeds per method, with 10 initial points sampled via cold-start initialization (excluding above-median performers), 100 iterations, and 16 LLM-sampled candidates per iteration. The random pool in BO (512 pool) is drawn from the full unconstrained parameter space.

Metrics. **Best-so-far (BSF)**: maximum objective value found, plotted against iteration. **Pool mean**: mean objective of the $N = 16$ candidates generated per iteration, measuring the overall quality of the LLM's suggestions. Because IPO updates accumulate across iterations, the sampler at iteration t for GGOLLuM is the actor that has already absorbed all IPO updates from iterations $1, \ldots, t-1$; pool mean therefore tracks how the actor's generation distribution shifts over the run under the same metric used for all other methods. **Mean selected-candidate objective (Mean)**: mean per-iteration objective of the candidate the method actually picks for evaluation, averaged across all 100 optimization iterations. Unlike BSF, Mean is not dominated by a lucky outlier; unlike pool mean, it conditions on what the method selects rather than what it samples. **Top-10 mean**: mean of the 10 highest selected-candidate objectives per run—a depth-of-discovery proxy that rewards methods which find a *set* of strong candidates rather than a single high-scoring outlier, and in conjunction with pool mean distinguishes generation-distribution shift from mode collapse.

8. Results and discussion

Figure 1 summarizes the main results. Row 1 shows best-so-far (BSF) trajectories and Row 2 shows the pool-mean trajectory of the LLM generator across the three benchmarks at batch size one. Table 3 reports final BSF, trajectory-mean selected objective, and top-10 selected-candidate objective over 20 seeds.

8.1. GGOLLuM outperforms other baselines on every benchmark

GGOLLuM achieves the highest BSF on all three tasks: 2.48 (reaction), 0.84 (molecular), 0.85 (peptide). The advantage is not a lucky last-iteration outlier: the same ordering holds for the mean objective of the selected candidate over the whole trajectory (Mean column) and for the mean of the top-10 selected candidates (Top-10 column). On reaction optimization in particular, GGOLLuM's Mean (1.26) is more than double that of GOLLuM (LLM pool) (0.53) and larger than every other method's Mean—the actor is selecting better-than-baseline candidates not just occasionally but on average. The Top-10 column makes the same point in a different language: GGOLLuM discovers a *set* of strong candidates on every benchmark (2.26, 0.59, 0.83), with the nearest baseline trailing by at least 0.08 Top-10 units. This rules out the mode-collapse interpretation as the actor's generation distribution is broadening into the high-utility region, not contracting onto a single lucky point.

The margin over the strongest non-IPO baseline varies with how well the other components of the stack are already matched to the domain: $+0.17$ BSF over GOLLuM (LLM pool) on reaction, $+0.12$ over BO (LLM pool+FPs) on molecular, $+0.05$ over GOLLuM (LLM pool) on peptide. The trajectory-quality gap (Mean column) is much larger—$+0.73$ on reaction, $+0.06$ on molecular, $+0.18$ on peptide—showing that the BSF margin understates how much more useful an entire GGOLLuM run is compared to surrogate-only stacks.

8.2. The generator matters

The method table stratifies the contribution of each component in the stack. Three observations follow from Table 3.

LLM-generated pools outperform random pools. On reaction optimization, LLM-GO (1.57) already outperforms random search (0.90) and LLM-GO (ICL) (1.93) further beats the non-LLM BO floor (1.91), showing that the base LLM's textual prior over reaction conditions is itself a non-trivial source of optimization signal.

Table 2. Methods compared. A method is characterized by its candidate pool (random vs LLM), its surrogate (none, classical GP on hand-crafted features, or deep-kernel GP on LLM features), and whether the LLM itself is adapted during optimization.

Method	Pool	Surrogate	LLM adapted?	Role
Random	random	–	–	trivial floor
BO (512 pool)	random	Matérn-5/2 on classical feats.	–	non-LLM BO floor
LLM-GO	LLM	–	base only	LLM generator floor
LLM-GO (ICL)	LLM	–	in-context	ICL ablation (MacKnight et al., 2025)
TOSFIT	LLM	deep-kernel GP	on-policy PG	LLM-BO baseline (Menet et al., 2025)
BO (LLM pool+FPs)	LLM	Matérn-5/2 on FPs	–	classical kernel on LLM pool
GOLLuM (LLM pool)	LLM	deep-kernel GP	marginal lik.	GOLLuM critic (Ranković and Schwaller, 2025)
GGOLLuM	LLM	deep-kernel GP	MLL + IPO	**ours**

Table 3. Final best-so-far (BSF), mean selected-candidate objective across the trajectory (Mean), and mean of the top-10 selected candidates (Top-10). Values are mean ± std across 20 seeds (peptide GOLLuM (LLM pool): 19). BSF is the best ever observed; Mean measures typical selection quality along the optimization trajectory and is insensitive to lucky outliers; Top-10 is a depth-of-discovery proxy that asks whether a method finds a *set* of strong candidates rather than a single one. Bold indicates best per row; "—" indicates the method is not applicable to that benchmark.

		Random	BO (512)	LLM-GO	LLM-GO (ICL)	TOSFIT	BO (LLM+FP)	GOLLuM (LLM pool)	**GGOLLuM**
Reaction	BSF	0.90±0.33	1.91±0.39	1.57±0.39	1.93±0.47	1.62±0.47	—	2.31±0.18	**2.48±0.60**
	Mean	−0.31±0.04	0.71±0.20	−0.13±0.04	0.27±0.16	−0.13±0.04	—	0.53±0.11	**1.26±0.67**
	Top-10	0.41±0.11	1.67±0.36	0.81±0.12	1.46±0.49	0.83±0.14	—	1.69±0.19	**2.26±0.65**
Molecular	BSF	—	—	0.52±0.25	—	0.51±0.25	0.72±0.20	0.63±0.23	**0.84±0.18**
	Mean	—	—	0.02±0.01	—	0.02±0.01	0.05±0.02	0.04±0.01	**0.11±0.06**
	Top-10	—	—	0.16±0.07	—	0.16±0.07	0.32±0.16	0.27±0.09	**0.59±0.21**
Peptide	BSF	—	—	0.75±0.03	—	0.75±0.03	0.78±0.04	0.80±0.03	**0.85±0.05**
	Mean	—	—	0.39±0.02	—	0.39±0.02	0.45±0.02	0.56±0.02	**0.74±0.06**
	Top-10	—	—	0.68±0.02	—	0.68±0.02	0.70±0.02	0.75±0.01	**0.83±0.06**

Classical kernels on LLM pools are already competitive. BO (LLM pool+FPs), which reranks an LLM-generated pool with a Matérn GP on standard domain fingerprints, is the strongest non-IPO baseline on molecular design (0.72 vs 0.63 for GOLLuM (LLM pool)). This indicates that on a narrow, LLM-filtered pool the choice of surrogate kernel is secondary—most of the work is done by the LLM generator. The classical kernel falters when the fingerprints themselves are weak: on peptide design, BO (LLM pool+FPs) (0.78) drops to LLM-GO-level performance while the deep kernel retains a $+0.02$ BSF advantage.

TOSFIT matches LLM-GO at our horizon. TOSFIT, which finetunes the LLM via an on-policy policy gradient whose advantage combines the GP posterior mean with a σ-scaled exploration bonus, sits within noise of LLM-GO on every benchmark (1.62 vs 1.57 on reaction, 0.51 vs 0.52 on molecular, 0.75 vs 0.75 on peptide). The original ToSFiT work (Menet et al., 2025) reports stronger gains over longer optimization horizons; at the 100-iteration budget we study here, the benefit of its update is not yet separated from the LLM-generator floor.

Preference optimization is the decisive step. GGOLLuM beats every stack that differs from it only in the absence of IPO. Row 2 of Figure 1 makes this visible: across all three benchmarks, only GGOLLuM's pool mean climbs substantially across iterations. The non-IPO generators (LLM-GO, TOSFIT, BO (LLM pool+FPs), GOLLuM (LLM pool)) keep roughly the same pool-mean throughout, because their generation distribution never sees the GP's pairwise preference feedback. LLM-GO with in-context learning (ICL) (Brown et al., 2020) on catechol is the one partial exception: its pool mean also climbs in early iterations as the in-context examples seed the prompt with high-utility points, but it saturates below GGOLLuM because in-context conditioning does not reshape the underlying next-token distribution explicitly. GGOLLuM's pool mean on reaction rises from ≈ 0.2 to ≈ 1.7, on molecular from ≈ 0 to ≈ 0.2, and on peptide from ≈ 0.4 to ≈ 0.8. These results suggest a behavioural signature of the actor internalizing acquisition-guided preferences.

8.3. What drives the pool-mean climb: critic signal, pair confidence, and actor drift

Figure 2 unpacks the mechanism behind GGOLLuM's rising pool mean into three per-iteration diagnostics, measured on the SMILES-DRD2 runs where the critic genuinely starts weak and all three curves have headroom to

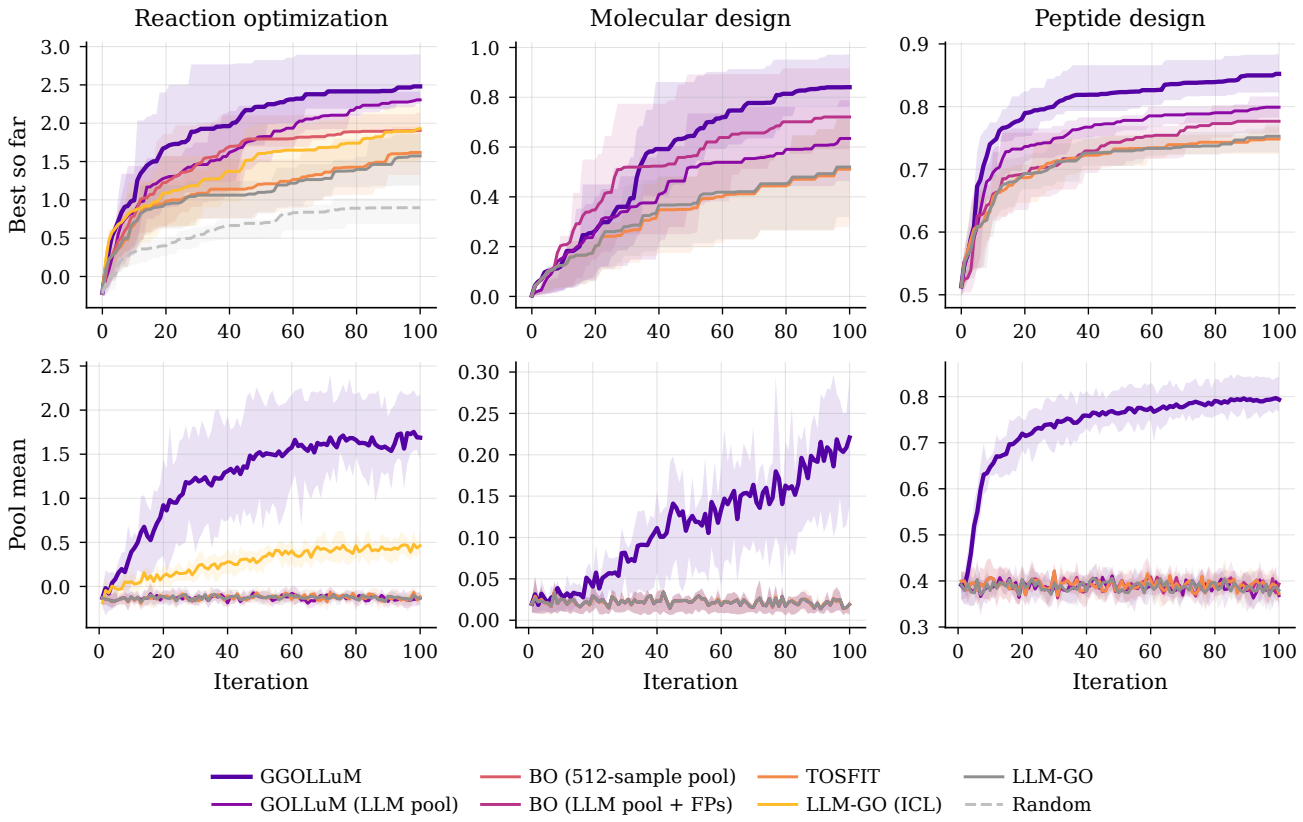

Figure 1. Main results across three benchmarks. **Top:** Best-so-far (BSF) as a function of optimization iteration. **Bottom:** Pool-mean quality of the 16 candidates generated by the LLM at each iteration. For GGOLLuM, the sampler at iteration t is the actor that has accumulated all IPO updates from iterations $1, \ldots, t-1$, so pool mean is the natural readout of how the actor's generation distribution evolves over the run. Lines are means over 20 seeds; shading is the inter-quartile range.

move. (Left) Running correlation between the critic's acquisition scores and the observed objective on each iteration's candidate pool. The critic starts near zero and becomes progressively better at ranking its own proposals as more data accumulates. (Middle) Mean confidence-weighted pair weight $z_{wl} = (\mu_w - \mu_l)/\sqrt{\sigma_w^2 + \sigma_l^2}$ clipped to $[0, 1]$. Early iterations start with most weight at the low end as winners and losers are barely distinguishable in GP-confidence. Progressively the weights rise as the GP becomes more decisive, which suppresses ambiguous pairs and sharpens the preference signal. (Right) Mean $|\log \pi_t(c_w) - \log \pi_{\text{base}}(c_w)|$ on the winner sequence, averaged over the pairs in iteration t's IPO step. The actor reliably absorbs each iteration's preferences (the drift proxy is always non-trivial) while staying within the neighbourhood enforced by the rolling-window threshold. Together, the three panels show the GGOLLuM advantage: a sharper critic produces higher-confidence pairs, which drive a larger-but-bounded policy shift, which yields a better pool mean at the next iteration.

8.4. Batch acquisition

In practice, experiments are often run in parallel. Figure 3 repeats the reaction-optimization comparison at batch sizes $b \in \{1, 4, 8\}$. GGOLLuM is the best method at every batch size (BSF $2.48/2.53/2.44$ at $b = 1, 4, 8$) while the classical deep-kernel baseline, GOLLuM (LLM pool), degrades substantially ($2.31/1.98/1.73$). GOLLuM (LLM pool) jointly selects b candidates from its LLM-generated pool via BoTorch's sequential-greedy qUCB, but joint selection cannot rescue a bad pool. The acquisition can only choose from the candidates it is handed, and when an unadapted generator produces uniformly low-utility candidates, even an informative batching strategy cannot extract a good batch from it. GGOLLuM instead takes the first-b valid novel candidates in the actor's generation order. Because IPO has already shifted actor mass toward the high-acquisition region, the leading candidates it emits are individually strong and collectively more diverse than a top-b GP reranking of a fixed generator's pool, so joint-selection quality holds up with b.

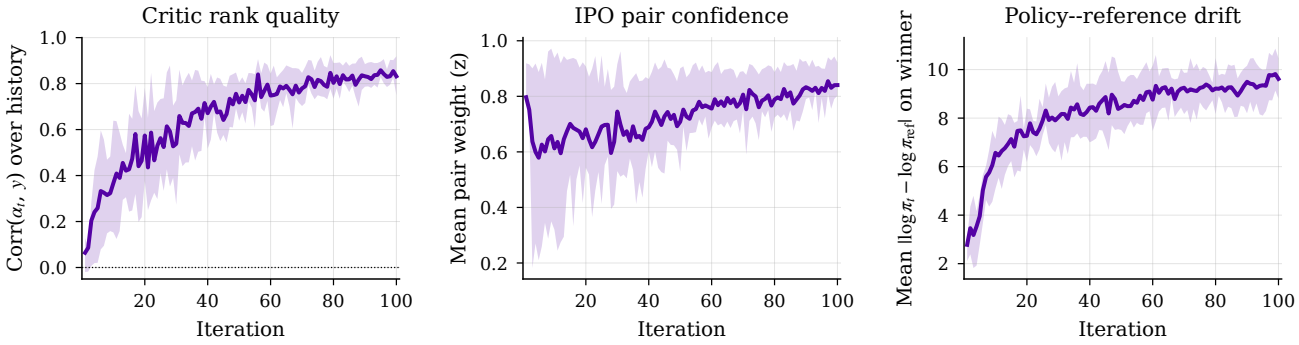

Figure 2. GGOLLuM mechanism diagnostics on SMILES-DRD2 (means with IQR shading, 20 seeds). **Left:** running correlation of the critic's acquisition score with the observed objective on each iteration's candidate pool. **Middle:** mean IPO pair weight (confidence-weighted z-score clipped to $[0, 1]$). **Right:** mean absolute log-probability gap on the winner between actor and base LLM, the proxy the trainer bounds.

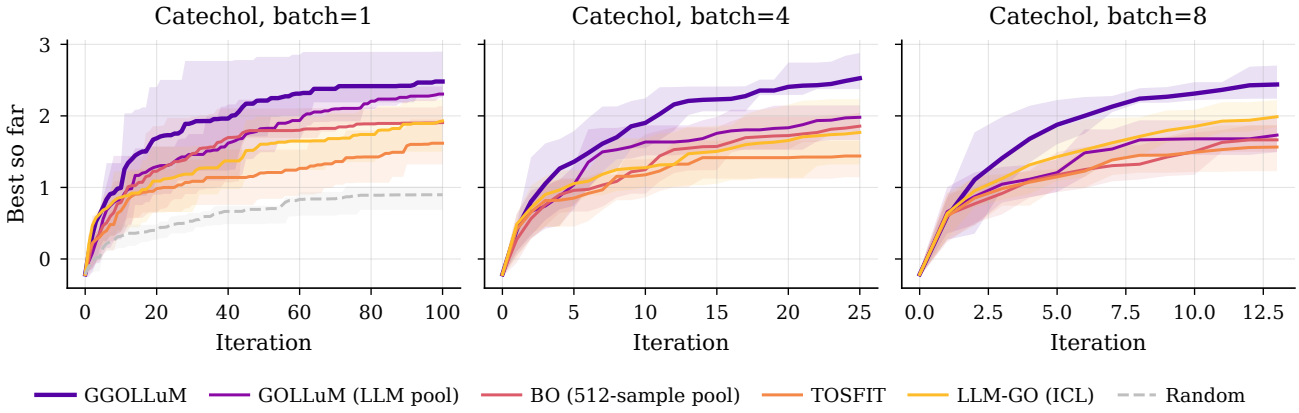

Figure 3. Batch acquisition on reaction optimization. GGOLLuM retains its advantage at batch sizes 4 and 8 while the classical deep-kernel GP baseline degrades markedly, suggesting that actor adaptation helps the joint-selection problem as much as the point-selection one.

8.5. Domain-specialist vs general-purpose base on peptide design

The headline peptide numbers in Table 3 use PeptideGPT, a domain-specialized generator. We additionally ran every LLM-based method with Qwen2.5-7B as the base (Figure 4) to isolate the contribution of domain-specialization from the rest of the stack.

Two findings emerge. First, GGOLLuM reaches the highest scores with either base: PeptideGPT: BSF 0.85 / Mean 0.74 / Top-10 0.83; Qwen2.5-7B: BSF 0.88 / Mean 0.74 / Top-10 0.86. The actor-adaptation advantage is not an artefact of pairing IPO with a weak domain-specialist base. Second, the non-IPO floor is higher for Qwen2.5-7B: LLM-GO reaches 0.82 BSF / 0.45 Mean with Qwen against only 0.75/0.39 with PeptideGPT, and adding ICL on top (LLM-GO ICL, Qwen-only because PeptideGPT is a BOS-conditioned sequence sampler with no prompt interface) pushes the floor to 0.85 BSF / 0.61 Mean. A larger general-purpose model is already a stronger peptide sampler than the domain-specialist, closing most but not all of the gap to GGOLLuM even without any GP in the loop. This result signalizes that GGOLLuM's uplift over the baselines is mediated by the *generation distribution*: whichever base we start from, IPO reshapes it into the high-utility region; a stronger base produces a higher non-IPO baseline but no stronger ceiling.

9. Limitations and future work

Sampler–trainer coupling. A central challenge is the tight coupling between the base LLM sampler and the preference training step. The quality and diversity of the candidates generated in Phase 1 directly determine what preference pairs are available for training. When the base model produces many duplicates or repeatedly proposes previously evaluated conditions, the preference optimization step receives a weak or degenerate training signal. We explored using random sampling from the design space to decouple the sampler from the trainer, but this proved less effective, suggesting that the LLM's pretrained knowledge provides a useful inductive bias for generating relevant training candidates. Developing better sampling strategies that balance

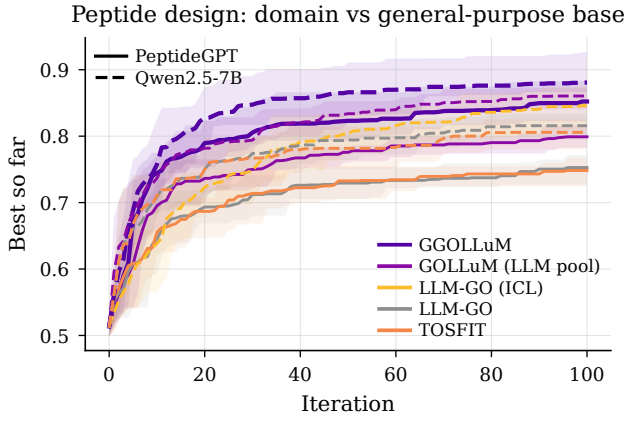

Figure 4. Peptide design with a domain-specialist base (PeptideGPT, solid) versus a general-purpose base (Qwen2.5-7B, dashed). Only LLM-based methods are shown. GGOLLuM is the best method with either base and the gap over non-IPO baselines is comparable, despite the two bases differing in parameter count and domain pretraining.

LLM-guided relevance with design-space diversity remains an important direction.

Actor–surrogate coupling. The quality of the preference training signal depends directly on the GP surrogate's acquisition scores, which in turn depend on the quality of the LLM's embeddings. This creates a feedback loop: a better surrogate produces more informative preference pairs, which trains a better actor, which generates higher-quality candidates for evaluation, which provides better training data for the surrogate in the next iteration. When this cycle is stable the optimization progressively improves; if the surrogate is poor the cycle can stall or misguide optimization. Even with joint finetuning of the embeddings, the preference signal can remain noisy in early iterations when the GP has seen few data points. The confidence-weight trajectories in Figure 2 (column 2) illustrate the cost: the critic initially cannot separate winners from losers, but once the GP has accumulated enough data for its acquisition scores to carry informative preferences, both the ranking quality and the generated pool mean improve.

10. Conclusion

GGOLLuM extends the GOLLuM critic with a preference-optimized actor, unifying candidate generation and evaluation inside a single language model. By using a GP's acquisition as the preference signal for IPO, we bridge Bayesian optimization and LLM alignment, transforming a pretrained LM into a generative Bayesian policy. Across three benchmarks spanning reaction, molecular, and peptide design, GGOLLuM outperforms a ladder of baselines isolating the contribution of LLM-generated pools, classical versus deep-kernel GPs, and policy-gradient versus preference-based adaptation. The advantage persists under batch acquisition, and GGOLLuM's progressive pool improvements over iterations suggest a behaviour of an actor internalizing Bayesian feedback.

References

Lewis H Mervin, Simon Johansson, Elizaveta Semenova, Kathryn A Giblin, and Ola Engkvist. Uncertainty quantification in drug design. *Drug discovery today*, 26(2): 474–489, 2021.

Francesca Tavazza, Brian DeCost, and Kamal Choudhary. Uncertainty prediction for machine learning models of material properties. *ACS omega*, 6(48):32431–32440, 2021.

Philippe Schwaller, Alain C Vaucher, Ruben Laplaza, Charlotte Bunne, Andreas Krause, Clemence Corminboeuf, and Teodoro Laino. Machine intelligence for chemical reaction space. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, page e1604, 2022.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Andres M Bran, Sam Cox, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

Mayk Caldas Ramos, Shane S. Michtavy, Marc D. Porosoff, and Andrew D. White. Bayesian optimization of catalysts with in-context learning, 2023.

Robert MacKnight, Jose Emilio Regio, Jeffrey G Ethier, Luke A Baldwin, and Gabe Gomes. Pre-trained knowledge elevates large language models beyond traditional chemical reaction optimizers. *arXiv preprint arXiv:2509.00103*, 2025.

Adam Tauman Kalai, Ofir Nachum, Santosh S Vempala, and Edwin Zhang. Why language models hallucinate. *arXiv preprint arXiv:2509.04664*, 2025.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

Sebastian Musslick, Laura K Bartlett, Suyog H Chandramouli, Marina Dubova, Fernand Gobet, Thomas L Griffiths, Jessica Hullman, Ross D King, J Nathan Kutz, Christopher G Lucas, et al. Automating the practice of

science: Opportunities, challenges, and implications. *Proceedings of the National Academy of Sciences*, 122(5): e2401238121, 2025.

Mayk Caldas Ramos, Christopher J Collison, and Andrew D White. A review of large language models and autonomous agents in chemistry. *Chemical science*, 2025.

Harold J Kushner. A versatile stochastic model of a function of unknown and time varying form. *Journal of Mathematical Analysis and Applications*, 5(1):150–167, 1962.

Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. 1964.

Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Bojana Ranković and Philippe Schwaller. Bochemian: Large language model embeddings for bayesian optimization of chemical reactions. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921*, 2024.

Kanan Mahammadli and Seyda Ertekin. Sequential large language model-based hyper-parameter optimization. *arXiv preprint arXiv:2410.20302*, 2024.

Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. A sober look at LLMs for material discovery: Are they actually good for Bayesian optimization over molecules? In *ICML*, 2024.

Bojana Ranković and Philippe Schwaller. Gollum: Gaussian process optimized llms–reframing llm finetuning through bayesian optimization. *arXiv preprint arXiv:2504.06265*, 2025.

Abdoulatif Cissé, Xenophon Evangelopoulos, Vladimir V Gusev, and Andrew I Cooper. Language-based bayesian optimization research assistant (bora). *arXiv preprint arXiv:2501.16224*, 2025.

Abdoulatif Cissé, Max E Cooper, Mengjia Zhu, Xenophon Evangelopoulos, Andrew I Cooper, et al. Can we automate scientific reasoning in closed-loop experiments using large language models? 2026.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Nicolas Menet, Aleksandar Terzić, Michael Hersche, Andreas Krause, and Abbas Rahimi. Thompson sampling via fine-tuning of llms. *arXiv preprint arXiv:2510.13328*, 2025.

Toby Boyne, Juan S Campos, Becky D Langdon, Jixiang Qing, Yilin Xie, Shiqiang Zhang, Calvin Tsay, Ruth Misener, Daniel W Davies, Kim E Jelfs, et al. The catechol benchmark: Time-series solvent selection data for few-shot machine learning. *arXiv preprint arXiv:2506.07619*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 18, 2023.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. *arXiv preprint arXiv:2312.11456*, 2023.

Eric Anderson, Gilman D Veith, and David Weininger. *SMILES, a line notation and computerized interpreter for chemical structures*. US Environmental Protection Agency, Environmental Research Laboratory, 1987.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9 (1):1–14, 2017.

Aayush Shah, Chakradhar Guntuboina, and Amir Barati Farimani. Peptide-gpt: Generative design of peptides using generative pre-trained transformers and bio-informatic supervision. *arXiv preprint arXiv:2410.19222*, 2024.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*, 2021.

Greg Landrum, Paolo Tosco, Brian Kelley, Ric, sriniker, gedeck, Riccardo Vianello, NadineSchneider, David Cosgrove, Eisuke Kawashima, Andrew Dalke, Dan N, Gareth Jones, Brian Cole, Matt Swain, Samo Turk, Alexander-Savelyev, Alain Vaucher, Maciej Wójcikowski, Ichiru Take, Daniel Probst, Kazuya Ujihara, Vincent F. Scalfani, guillaume godin, Axel Pahl, Francois Berenger, JLVarjo, strets123, JP, and DoliathGavid. rdkit/rdkit: 2022_09_1b1 (q3 2022) release. October 2022. doi: 10.5281/zenodo.7179566. URL `https://doi.org/10.5281/zenodo.7179566`.

Entropy. gpt2_zinc_87m. `https://huggingface.co/entropy/gpt2_zinc_87m`, 2024. HuggingFace model card, accessed 2026-05-08.

Anna Szymańska, Marcin Król, and Krzysztof Baczyński. Comparative analysis of molecular embeddings for efficient compound similarity search using vector databases. 2025.

Brandon T Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

A. Benchmark details

Catechol (reaction optimization). We use the Catechol Benchmark of Boyne et al. (2025): 1227 experimentally measured process conditions for an allyl-substituted catechol rearrangement collected on a transient continuous-flow platform with online quantification of reaction outcomes. The controllable inputs are two categorical solvent identities (`SOLVENT A NAME`, `SOLVENT B NAME`) and three continuous parameters — volumetric solvent-B fraction $b \in [0.0, 1.0]$, residence time $\tau \in [2.0, 15.0]$ min, and temperature $T \in [175, 225]\,^\circ$C. The measured outcomes are the yields of the starting material (SM) and two products P_2 and P_3. We adopt the scalarized objective defined in the original benchmark study, which rewards total product yield and selectivity toward P_2 while penalizing harsher operating conditions:

$$f(S_A, S_B, b, \tau, T) = \lambda_1(P_2 + P_3) + \lambda_2 \frac{P_2}{P_2 + P_3} - \lambda_3 \frac{T - 175}{50} - \lambda_4 \tau, \quad (5)$$

with $\lambda_1 = 5$, $\lambda_2 = 1$, $\lambda_3 = 3$, $\lambda_4 = \frac{1}{20}$. Because the measured dataset only covers a fraction of the full categorical-continuous product, we evaluate novel combinations through a digital-twin surrogate: a Random Forest regressor (500 trees, one-hot encoding for categoricals, passthrough for continuous parameters) is fit on the measured $(S_A, S_B, b, \tau, T) \to f$ table and reaches cross-validated $R^2 \approx 0.94$. Candidates that match a measured point (within 10^{-3} tolerance on continuous parameters) are scored by exact lookup; novel candidates are scored by the surrogate.

SMILES-DRD2 (molecular design). The objective is dopamine receptor D2 (DRD2) activity, scored by the Therapeutics Data Commons (Huang et al., 2021) `DRD2` oracle, which wraps the standardized SVM classifier of Olivecrona et al. (2017) (range $[0, 1]$, higher is better). Candidates are SMILES strings emitted by the actor; we canonicalize with RDKit (Landrum et al., 2022) and assign `NaN` to unparseable strings. The reference library used to seed cold-start initialization is a curated set of ≈ 150 FDA-approved or well-known drug-like molecules spanning major therapeutic classes (analgesics, CNS, antibiotics, cardiovascular, antimalarials, natural products, simple fragments). Cold-start sampling restricts to molecules below the reference-library median DRD2 score so that no method begins with an active.

Peptide-AMP (peptide design). The objective is a composite antimicrobial-peptide score computed in closed form via the `peptides` library: a weighted sum of (a) hydrophobic moment (35%, mapped through $h/(h + 0.5)$), (b) net positive charge at pH 7.4 (30%, normalized to $[0, 1]$ over $[0, +5]$), (c) stability ($\max(0,\, 1 - \mathrm{II}/80)$ for instability index II, 20%), and (d) Boman index ($b/(b + 2)$ for $b > 0$, 15%). All four components are normalized to roughly $[0, 1]$ and scored by the same library across methods, so the AMP objective is deterministic and benchmark-stable. Candidates are amino-acid sequences over the 20-letter alphabet `ACDEFGHIKLMNPQRSTVWY` with length $\in [5, 50]$. The reference library is a curated set of ≈ 200 known antimicrobial peptides (LL-37, magainins, cecropins, melittin, defensins, dermaseptins, protegrins, temporins, and others) augmented with diverse short sequences for exploration.

B. Implementation details

Base models. Each benchmark uses the base LLM that best matches the modality of its design space. Catechol solvent selection (tabular reaction inputs) uses **Qwen2.5-7B** (Bai et al., 2023), the *base* (not instruction-tuned) variant, so the IPO signal reshapes a general next-token distribution rather than fighting an already-aligned one. SMILES-DRD2 uses **gpt2-zinc-87M** (Entropy, 2024; Szymańska et al., 2025), a GPT-2 architecture pretrained on ZINC SMILES that natively emits grammatically valid molecules but places almost no mass on DRD2-active chemistry. Peptide-AMP uses **PeptideGPT**, a GPT-2 architecture pretrained on UniProt peptide sequences and selected for non-hemolytic output. It emits valid peptide-like sequences but is not aligned with the AMP objective. All three pairings are *plausible but not aligned*, which is precisely the regime where a generative policy shaped by a Bayesian critic is expected to have headroom.

LoRA configuration. Two LoRA adapters are registered on the same base: an *actor* adapter (accumulates across iterations, trained by IPO) and a *critic* adapter (reset and retrained each BO iteration, trained jointly with the GP). Both adapters use rank 4, $\alpha = 16$, dropout 0.2, and RSLoRA. The target ratio (fraction of top transformer layers wrapped with LoRA) is 25%.

GP surrogate. The GP surrogate is a Deep Kernel GP (Wilson et al., 2016) with a Matérn-5/2 kernel applied to mean-pooled LLM embeddings, passed through a `ScaleToBounds(-1, 1)` transform. The GP is re-initialized from scratch at the start of each BO iteration, and the critic LoRA adapter is reset alongside it to preserve the semantics of fresh marginal-likelihood optimization per iteration.

IPO training. We use Identity Preference Optimization (IPO) (Azar et al., 2024) with $\beta = 0.02$, learning rate 5×10^{-4}, and up to 5 epochs per iteration, with 32-pair mini-batches. Preference pairs are constructed via the confidence-weighted strategy described in the main text: winners are

drawn from the top 50% of the candidate set by acquisition score, losers from the bottom 50%, with a minimum score margin of $0.5 \times \text{std}(\alpha)$ and a cap on pairs per winner to prevent a single high-scoring candidate from dominating the batch.

Acquisition function. Unless otherwise specified, we use UCB with $\beta_{\text{UCB}} = 4$ for all GP-based methods and for the GGOLLuM critic. For batched BO we use BoTorch's sequential-greedy qUCB. The value $\beta_{\text{UCB}} = 4$ was selected by preliminary tuning on catechol: higher β_{UCB} yielded more stable best-so-far trajectories under strong regularization of the LLM kernel.

Initialization. All methods start from 10 initial points sampled via *cold-start* initialization: points are drawn uniformly from the curated design space with the constraint that none exceed the median objective value. This removes any advantage from lucky initial draws and tests each method's ability to recover from uninformative starting conditions.

Hyperparameter selection. Hyperparameters were fixed across all benchmarks after preliminary tuning on catechol; no benchmark-specific tuning was performed. The same $(\beta_{\text{IPO}}, \text{lr}, \beta_{\text{UCB}}, \text{epochs}, \text{LoRA rank})$ is used on tabular reaction inputs, SMILES strings, and peptide sequences.

C. Constrained decoding

For tabular reactions and peptides we apply domain-specific logits processors at every generation step to prevent the sampler from emitting strings that would never be evaluated. The same constraints are used identically for all LLM-based methods (LLM-GO, TOSFIT, BO with LLM pool, GOLLuM (LLM pool), and GGOLLuM); they guarantee *validity*, not quality. SMILES decoding is unconstrained; `gpt2-zinc-87M` is pretrained on ZINC and emits parseable strings with high frequency, and we simply assign `NaN` to the rare unparseable strings.

Tabular reactions (catechol). For mixed-categorical/continuous parameter spaces we compile the parameter description into a regular expression with one alternation group per parameter, separated by literal semicolons. Each categorical parameter contributes an alternation over its admissible values; each continuous parameter contributes an integer-range alternation followed by an optional two-decimal suffix `(int_lo|...|int_hi)(\.[0-9]{1,2})?`. The regex is compiled to a deterministic finite automaton via `outlines_core` (Willard and Louf, 2023), and the logits-processor masks any token whose addition would leave the Finite State Machine (FSM) in a dead state. Concretely, this means the actor cannot emit a categorical level that does not exist in the parameter space, cannot emit a continuous value outside the admissible range, and cannot emit a malformed parameter delimiter. Per-batch FSM state is tracked incrementally so the marginal cost per token is negligible.

Peptides. The peptide regex is the simpler character class `[ACDEFGHIKLMNPQRSTVWY]{5,50}\n?`. We use the same `outlines_core` FSM machinery: the processor allows only the 20 amino-acid one-letter codes, masks all non-amino-acid tokens, and allows EOS only once the current sequence length is at least 5. FASTA- trained protein LMs (PeptideGPT and similar GPT-2 variants) emit a literal newline as their natural sentence terminator (their EOS token is BOS, used only at document boundaries), so a trailing optional `\n` in the regex prevents the FSM from forcing every sample to the upper length bound; the newline is stripped before evaluation.

Why constrained decoding does not subsume IPO. The constraint guarantees *validity*, not *quality*: a constrained sampler still emits low-utility candidates with high probability if the base model is uninformed about the objective. The gap that IPO closes is precisely the gap between syntactically valid generation and high-utility generation, which is why the LLM-GO and GGOLLuM pool-mean trajectories diverge in Figure 1 (row 2) despite both running with identical constrained decoding.

D. Prompt templates

We use the *direct* prompting style throughout: the model is asked for a single candidate per sample with no intermediate Analysis/Hypothesis scaffolding, and we use a *compact* output format (semicolon-separated values for tabular reactions, bare strings for SMILES and peptides). This is the format consumed by both LLM-GO and the GGOLLuM actor at inference time, and matches the regular expressions enforced by constrained decoding (Appendix C).

Tabular reactions (catechol). The system prompt lists the parameter space and asks for one semicolon-separated row in a fixed parameter order:

```
You are an expert optimizer. Your goal is to
find the optimal conditions to maximize the
objective.

Search space:
- SOLVENT A NAME: <admissible solvents>
- SOLVENT B NAME: <admissible solvents>
- SolventB%:      [0.0, 1.0]
- Residence Time: [2.0, 15.0]
- Temperature:    [175.0, 225.0]

Output format: one suggestion per line,
```

```
values separated by semicolons
in this exact order:
  SOLVENT A NAME; SOLVENT B NAME; SolventB%;
  Residence Time; Temperature
Example:
  1,1,1,3,3,3-Hexafluoropropan-2-ol;
  2-Methyltetrahydrofuran [2-MeTHF];
  0.50; 8.00; 200.00

Rules:
- Copy parameter values EXACTLY from the
lists above.
- Do NOT include parameter names, JSON,
quotes, or any
  other formatting.
- Output ONLY the semicolon-separated values,
nothing else.
```

The user message is the short trigger `Suggest a candidate.` Constrained decoding (Appendix C) guarantees that each emitted line conforms to the parameter-space regex, so the actor cannot emit invalid solvent names, out-of-range continuous values, or malformed delimiters.

Generative-only bases (SMILES-DRD2, Peptide-AMP). For `gpt2-zinc-87M` and PeptideGPT, the actor has no chat template and no tabular parameter space, so the prompt collapses to a bare BOS token and the model emits a single bare string per sample. Peptide samples are constrained to the 20-letter amino-acid regex (Appendix C); SMILES samples are emitted unconstrained and validated post hoc with RDKit (parseable rates are high since `gpt2-zinc-87M` is pretrained on ZINC SMILES (Sterling and Irwin, 2015)). Representative emissions are:

```
SMILES-DRD2:   CC(=O)Oc1ccccc1C(=O)O
Peptide-AMP:   KWKLFKKIEKVGQNIR
```

The IPO pair is then $(w, l) = (s_w, s_l)$ over the bare strings, with preference labels produced by the critic's acquisition score exactly as in the tabular case. This is also why the ICL ablation (LLM-GO ICL) is not reported for these two benchmarks: a BOS-conditioned sampler has no prompt interface into which in-context examples can be inserted.

History inclusion (LLM-GO ICL only). GGOLLuM does not include experiment history in the actor prompt: the IPO signal already shapes the next-token distribution toward high-utility candidates, and conditioning on a long prompt-suffix of past evaluations would bias the actor toward near-duplicates of recent best points and away from exploration.

The only configuration that uses history is the **LLM-GO ICL** baseline (Figure 5), which appends the top-10 previously-evaluated points (ranked by objective) to the system prompt as in-context examples:

```
Previous best experiments (use as guidance,
do NOT repeat these):

  HFIP; 2-MeTHF; 0.50; 8.00; 200.00 -> 0.9453
  Acetonitrile; 2-MeTHF; 0.30; 6.00; 195.00
  -> 0.8721
  ... (10 lines total) ...

Best so far: 0.9453
```

The 10-best window is small enough to fit comfortably in the Qwen2.5-7B context but large enough to expose the actor to the structure of high-utility regions. The non-ICL LLM-GO baseline uses the same direct prompt as GGOLLuM with no history, isolating the contribution of in-context examples.

IPO training prompt. For IPO, the pair (c_w, c_l) shares the same system/user prompt as inference (no history); the preferred and dispreferred completions are the bare compact-format candidates:

```
System: [parameter-space description and
output-format rules, as above]
User:   Suggest a candidate.

Preferred:    HFIP; 2-MeTHF; 0.50; 8.00;
200.00
Dispreferred: Toluene; THF;  0.10; 14.00;
180.00
```

For SMILES-DRD2 and Peptide-AMP the same structure holds with an empty system/user prompt and bare strings as winner/loser.

E. ICL ablation with a matched base model

Figure 5 reports the LLM-GO comparison with and without in-context learning, using Qwen2.5-7B as a matched base on both catechol and peptide. ICL helps most where the base model's textual prior is strong (tabular reaction inputs); on peptide sequences the gap is smaller. GGOLLuM (shown for reference) improves on top of the ICL baseline in both panels.

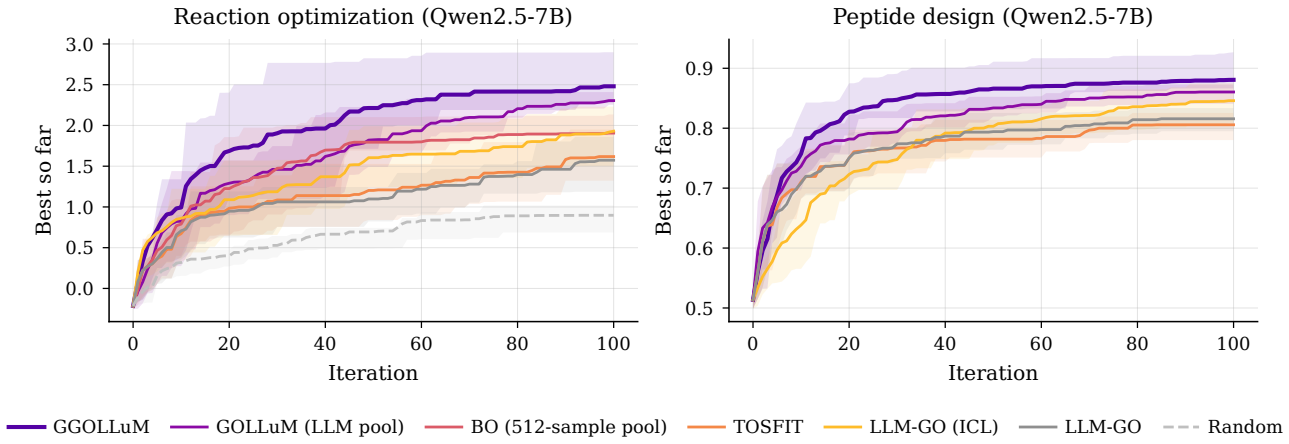

Figure 5. ICL ablation with Qwen2.5-7B as the common base. Left: catechol (reaction optimization). Right: peptide design. ICL helps most where the base model's textual prior is strong (tabular reaction inputs); on peptide sequences the gap is smaller. GGOLLuM improves on top of the ICL baseline in both panels.