LARGE LANGUAGE MODELS AS GENERATIVE BAYESIAN POLICIES

Anonymous authors
Paper under double-blind review

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 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 ask whether Bayesian optimization can supply the training signal for preference-based LLM alignment. We train a Gaussian process critic through deep kernel learning on the LLM's own embeddings. 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 five tasks spanning reaction optimization, molecular design and peptide design under a budget of 100 experiments. 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 be the training signal that aligns language models with the decision-making demands of experimental discovery.

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). Combining the two, prior work keeps the LLM in a supporting role by proposing candidates, supplying priors, or providing features, while the BO machinery often performs the actual uncertainty-aware decision making (Section 2).

Here we ask: *can we teach an LLM to think like a Bayesian optimizer?* Preference optimization is agnostic to *where* its preference signal originates (Figure 1); we supply it from a Bayesian critic. We introduce GGOLLuM (Generative Gaussian Process Optimized LLM), a framework that uses

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

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. (2023); 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.

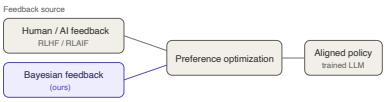


Figure 1: **Bayesian feedback.** Preference optimization is agnostic to where its labels come from; GGOLLuM takes them from a GP critic's posterior rather than from human or AI judgments.

We evaluate GGOLLuM on five tasks spanning three representation regimes: mixed-variable reaction optimization (catechol rearrangement reactions) Boyne et al. (2025), molecular design (dopamine receptor D2 affinity, with and without a synthesizability constraint), and sequence-based biological design (antimicrobial and lytic peptides). Against a ladder of baselines 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. Our contributions are:

- **Bayesian feedback:** GP-posterior preference pairs over a generator's own samples, confidence-weighted and anchored by evaluated points, that train stably from 100 labels (Sec. 4).
- **A generator that is the optimizer:** It requires no candidate set to search, and can sample a batch of candidates across reactions, molecules and peptides (Sec. 6).
- **What makes it work:** the critic's labels on sampled candidates, a bounded loss, and anchors against policy collapse (Sec. 6.4).

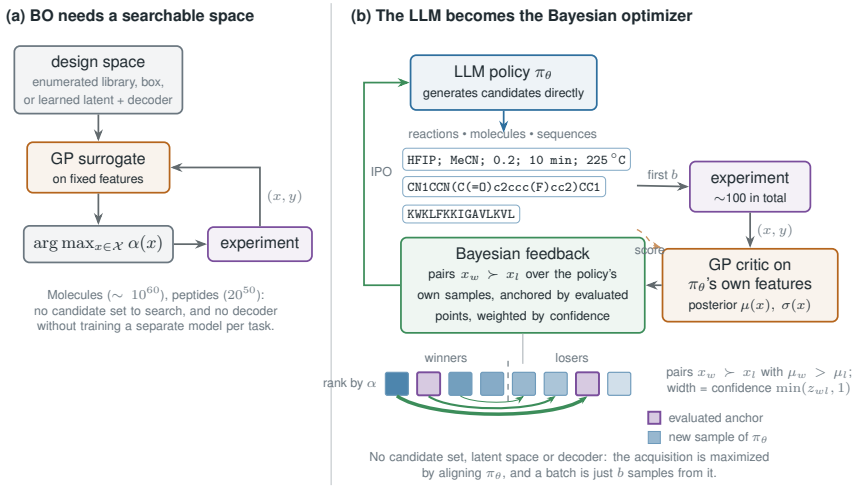


Figure 2: **From Bayesian optimization to a generative Bayesian policy. (a)** Standard BO maximizes an acquisition function over a searchable space – an enumerated library, a box, or a learned latent space with a decoder – none of which exists for molecules or peptides. **(b)** GGOLLuM removes the search: a GP critic on the LLM's own features scores the policy's samples, those scores become preference pairs over the samples, anchored by evaluated points and weighted by the critic's confidence, and IPO moves the policy towards high-acquisition regions. The next experiments are the policy's first b samples.

2 Related work

LLMs in Bayesian optimization. Prior work gives the LLM one of four roles: a *proposer* whose suggestions are filtered or prompted with the history (Liu et al., 2024; MacKnight et al., 2025; Mahammadli & Ertekin, 2024; Cissé et al., 2025; 2026; Agarwal et al.; Wang et al., 2025; Nguyen & Grover, 2025); a *feature map* for a GP surrogate (Ranković & Schwaller, 2023; Kristiadi et al., 2024; Ranković & Schwaller, 2025); a *prior* that reshapes the acquisition (Hvarfner et al., 2022; Xu et al., 2024; Chang et al., 2025); or a *surrogate* (Requeima et al., 2024). GGOLLuM instead trains the generator on the surrogate's posterior, so the LLM becomes Bayesian policy. The closest related work to ours is ToSFiT (Menet et al., 2025), which finetunes an LLM with an on-policy policy gradient on a GP-based advantage; GGOLLuM instead trains on off-policy preference pairs, and we compare the two with the same deep-kernel critic (Section 6).

Generative and latent-space BO. Optimizing over structured spaces usually means a learned latent space with a decoder (Gómez-Bombarelli et al., 2018; Tripp et al., 2020; Maus et al., 2023; Lee et al., 2025) or generators trained with RL (Olivecrona et al., 2017) or guided diffusion (Stanton et al., 2022; Gruver et al., 2023), which need far more than 100 evaluations (Gao et al., 2022b). GGOLLuM instead searches directly in the distribution of a pretrained language model, adapting it with $\sim$100 labels.

Where preference signals come from. Preference optimization (Rafailov et al., 2023; Azar et al., 2024) is agnostic to the source of its labels: human judgments (Christiano et al., 2023; Ouyang et al., 2022) or an AI judge (Bai et al., 2022; Lee et al., 2024). Learned rewards are prone to over-optimization (Gao et al., 2022a; Coste et al., 2024) and noisy labels (Chowdhury et al., 2024). A GP posterior addresses both: it states how confident each preference is, and it is refit on real data after every experiment.

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 & Schwaller, 2025).

Acquisition functions. Given a fitted GP with posterior mean $\mu(\mathbf{x})$ and standard deviation $\sigma(\mathbf{x})$, we use the upper confidence bound (Auer, 2002; Srinivas et al., 2009) $\alpha_{\text{UCB}}(\mathbf{x}) = \mu(\mathbf{x}) + \sqrt{\beta_{\text{UCB}}}\,\sigma(\mathbf{x})$ with $\beta_{\text{UCB}} = 4$ throughout. For batches (b>1) the GP baselines use sequential-greedy qUCB (Balandat et al., 2020) while GGOLLuM simply evaluates the first b samples of its policy in their generation order.

Direct preference optimization. DPO (Rafailov et al., 2023) trains a policy π_θ to prefer a "chosen" completion y_w over a "rejected" y_l without an explicit reward model, by pushing up the log-ratio margin $h_\theta(y_w, y_l) = \log \frac{\pi_\theta(y_w)}{\pi_{\text{ref}}(y_w)} - \log \frac{\pi_\theta(y_l)}{\pi_{\text{ref}}(y_l)}$ through a log-sigmoid loss $-\mathbb{E} \log \sigma(\beta h_\theta)$, where π_{ref} is a fixed reference policy and β controls the KL penalty.

4 Method

GGOLLuM runs in rounds of Bayesian optimization (Figure 2). In each, the policy samples N=16 candidates; a GP critic refit on all evaluations so far scores them; the candidates, together with k=10 evaluated points (*anchors*), are turned into confidence-weighted preference pairs; the policy is updated on those pairs; and the first b samples are evaluated. Policy and critic are two LoRA adapters (Hu et al., 2022) on one frozen base model, which also serves as π_{ref}: the policy adapter accumulates across rounds, while the critic adapter is reset and refit with the GP by marginal likelihood each round.

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:

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

Pairs the GP cannot confidently separate contribute little to the loss, while pairs with a clear mean gap relative to their combined uncertainty contribute most (Appendix F).

Anchors. The pool is not only the policy's new samples: we add k=10 points drawn uniformly from the evaluated set. Their values are measured, so their posterior variance is near zero and the pairs they join are ones the critic is confident about. They act as a replay buffer that keeps every update tied to real data as otherwise, the policy can concentrate on a few of its own samples, run out of distinct candidates to compare, and collapse (Section 6.4).

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:

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

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

$$\mathcal{L} \;=\; \frac{1}{|\mathcal{P}|} \sum_{(w,l)\in\mathcal{P}} \min(z_{wl}, 1) \cdot \mathcal{L}_{\text{IPO}}(w, l \mid x), \tag{3}$$

with $\beta = 0.033$ (target margin $1/(2\beta) \approx 15$), 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.

Updates accumulate. IPO updates accumulate over BO iterations (**??**). 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 in Appendix E summarizes one BO iteration.

5 Experimental setup

Benchmarks. We evaluate on five 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.

In every case the base model is fluent in the domain but not aligned with the task: the molecular model writes drug-like SMILES, yet DRD2 actives are a small part of what it covers (Olivecrona et al., 2017); PeptideGPT writes valid peptides but places no mass on the antimicrobial composite; and Qwen2.5-7B (Bai et al., 2023) has no chemistry bias at all (Appendix A).

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
Molecular	SMILES-DRD2-Synth	SMILES	gpt2-zinc-87M
Peptide	Peptide-AMP	sequence	PeptideGPT
Peptide	Peptide-Lytic	sequence	PeptideGPT

Baselines. We compare methods along a ladder of increasing posterior awareness (Appendix G). Non-LLM generative baselines built for thousands of evaluations are discussed in Appendix B; a graph genetic algorithm from the same cold start stays far below every LLM-based method (0.19 on SMILES-DRD2). This 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 & Hahn, 2010); for peptides, dipeptide ($k = 2$) count vectors (400 dims). These are the task-standard classical descriptors in each domain. 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.

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. For BO, the candidate pool depends on the design space. On Catechol, where the space is bounded and can be sampled directly, the GP chooses among 512 conditions drawn uniformly from the full parameter space. For molecules and peptides, which are unbounded and cannot be sampled uniformly, the pool is instead the 16 candidates proposed by the base LLM each iteration, featurized with Morgan fingerprints (molecules) or dipeptide counts (peptides) and the GP selects the next evaluation among them by acquisition.

Metrics. We report the best objective found (**BSF**) against the number of evaluations, and, in Table 2, the mean objective of the candidates a method actually evaluates (**Mean**) and the mean of each run's ten best evaluations (**Top-10**); the last two are insensitive to a single lucky draw. We also track **pool mean**, the mean objective of the $N{=}16$ candidates a method samples per iteration, which measures the quality of its suggestions rather than of its selection. Because IPO updates accumulate, GGOLLuM's sampler at iteration t has already absorbed the updates of iterations $1, \ldots, t-1$, so pool mean tracks how its generation distribution shifts over the run under the same metric used for every other method.

6 Results and discussion

Figure 3 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 five tasks at batch size one. Table 2 reports final BSF, trajectory-mean selected objective, and top-10 selected-candidate objective over 20 seeds.

6.1 GGOLLuM outperforms other baselines on every task

GGOLLuM achieves the highest BSF on all five tasks: 2.77 (Catechol), 0.90 and 0.84 (SMILES-DRD2 and -Synth), 0.87 and 0.53 (Peptide-AMP and -Lytic). 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 (1.10) is more than double that of GOLLuM (0.55) 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 task (Top-10 of

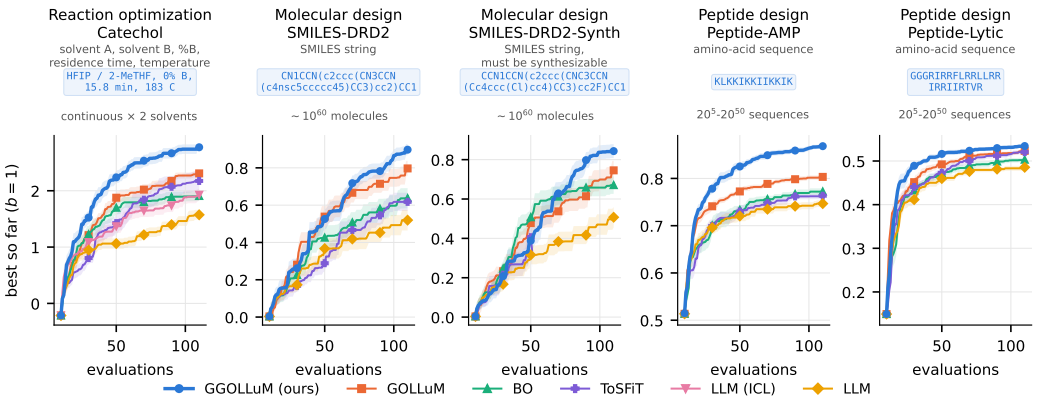


Figure 3: **Optimization in three kinds of open-ended design space. Top:** what the model writes in each space, with an example design GGOLLuM found and the size of the space. **Bottom:** best objective found against the number of evaluations, b=1, mean ± one standard error over 20 seeds from a cold start of 10 points below the library median. BO is a Gaussian process over 512 uniformly random conditions on Catechol, where the space can be sampled directly, and over the untrained model's samples featurized with Morgan fingerprints (molecules) or dipeptide counts (peptides) on the open-ended tasks, where no candidate set exists. GOLLuM is the same deep-kernel critic selecting among the untrained model's samples. LLM samples the untrained model; LLM (ICL) additionally places the ten best evaluations so far in the prompt, and exists only where the base model has a prompt interface. ToSFiT is on-policy RL with the same critic, tuned and sample-matched to GGOLLuM (Appendix I.7).

Table 2: Best objective found (BSF), mean objective of the evaluated candidates (Mean) and mean of each run's ten best evaluations (Top-10), b=1, mean ± std over 20 seeds. Methods as in Figure 3; on Catechol the LLM cell also reports the model prompted with the ten best evaluations with in-context learning (ICL). Bold: best per row.

		LLM	BO	ToSFiT	GOLLuM	GGOLLuM (ours)
Catechol	BSF	1.57±0.39 ICL: 1.93±0.47	1.91±0.39	2.17±0.76	2.31±0.19	**2.77**±0.25
	Mean	-0.133±0.041 ICL: 0.265±0.159	0.705±0.201	0.912±0.571	0.548±0.105	**1.10**±0.34
	Top-10	0.814±0.121 ICL: 1.46±0.49	1.67±0.36	2.09±0.80	1.75±0.22	**2.48**±0.43
SMILES-DRD2	BSF	0.520±0.248	0.637±0.242	0.618±0.218	0.797±0.179	**0.897**±0.133
	Mean	0.022±0.007	0.043±0.020	0.033±0.012	0.046±0.013	**0.150**±0.087
	Top-10	0.162±0.068	0.282±0.139	0.215±0.088	0.327±0.097	**0.678**±0.222
SMILES-DRD2-Synth	BSF	0.507±0.232	0.670±0.198	—	0.746±0.215	**0.843**±0.158
	Mean	0.020±0.006	0.047±0.017	—	0.038±0.012	**0.134**±0.072
	Top-10	0.148±0.051	0.311±0.121	—	0.279±0.095	**0.644**±0.206
Peptide-AMP	BSF	0.747±0.031	0.772±0.032	0.763±0.027	0.803±0.021	**0.868**±0.019
	Mean	0.388±0.018	0.450±0.021	0.504±0.048	0.608±0.012	**0.717**±0.020
	Top-10	0.677±0.017	0.712±0.021	0.719±0.026	0.762±0.013	**0.849**±0.020
Peptide-Lytic	BSF	0.486±0.023	0.503±0.021	0.520±0.023	0.523±0.017	**0.534**±0.019
	Mean	0.116±0.013	0.181±0.024	0.271±0.056	0.317±0.015	**0.415**±0.019
	Top-10	0.400±0.021	0.448±0.019	0.483±0.030	0.487±0.012	**0.523**±0.021

2.48, 0.68, 0.64, 0.85 and 0.52), with the nearest baseline trailing by 0.04 (Peptide-Lytic) to 0.39 (Catechol). 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.

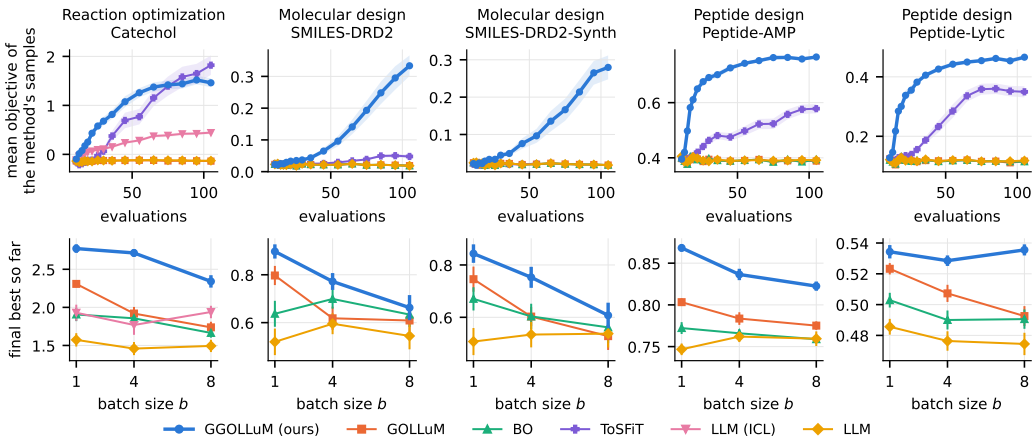


Figure 4: **The generator becomes the optimizer. Top:** mean objective of all valid candidates a method samples in an iteration, scored for this figure only and never counted against the budget; for GGOLLuM these are the policy's own samples, for the baselines the untrained model's. BO is omitted because it draws from a uniform pool rather than a generator. **Bottom:** final best-so-far against batch size $b \in \{1, 4, 8\}$. GGOLLuM evaluates the first b samples of its policy with no batch acquisition function, while the GP baselines select batches with sequential-greedy qUCB. Bands and bars: one standard error over 20 seeds.

The margin over the strongest non-IPO baseline varies with how well the other components of the stack are already matched to the domain: $+0.46$ BSF over GOLLuM on Catechol, $+0.10$ on both molecular tasks, and $+0.07$ and $+0.01$ on the two peptide tasks. The trajectory-quality gap (Mean column) is larger, $+0.56$ on Catechol and about $+0.10$ on each of the other four tasks, showing that the BSF margin understates how much more useful an entire GGOLLuM run is compared to surrogate-only stacks.

6.2 The generator matters

The base LLM's prior is itself a signal. On reaction optimization, prompting the base model with the ten best evaluations (LLM (ICL), 1.93) already matches a GP choosing among 512 random conditions (BO, 1.91), showing that its textual prior over reaction conditions carries real optimization signal.

On-policy RL with the same critic helps, but less than the critic alone. ToSFiT finetunes the LLM with a policy gradient whose advantage combines the GP posterior mean with a σ-scaled exploration bonus. We tuned it over learning rate, optimizer and the number of updates per iteration, and report it sample-matched to GGOLLuM (16 samples per iteration; Appendix I.7). It improves on the untrained model on Catechol and SMILES-DRD2 (2.17 vs 1.57, 0.62 vs 0.52) but stays at or below GOLLuM, which never trains the generator (2.31, 0.80). Giving it four times the samples per iteration does not close the gap, and replacing its advantage with our UCB acquisition changes nothing, so the difference lies in how the signal updates the policy: on-policy gradients on individual samples versus anchored pairwise preferences.

Preference optimization is the decisive step. GGOLLuM beats every stack that differs from it only in the absence of IPO, and the top row of Figure 4 shows why: its pool mean – the average quality of what the policy samples – climbs on every task, from ≈ 0.1 to ≈ 1.5 on Catechol, ≈ 0.02 to ≈ 0.3 on SMILES-DRD2 and ≈ 0.4 to ≈ 0.8 on Peptide-AMP. Generators that never see the preference signal (LLM, BO, GOLLuM) stay flat by construction. Two methods also move. LLM (ICL) on Catechol climbs early as the prompt fills with good examples but saturates, since in-context conditioning does not reshape the next-token distribution. ToSFiT's samples climb too, and on Catechol end above GGOLLuM's, yet its best-so-far is lower: it concentrates on the neighbourhood of what already works. A better average sample is necessary but not sufficient; the optimizer also has to keep proposing candidates worth the risk.

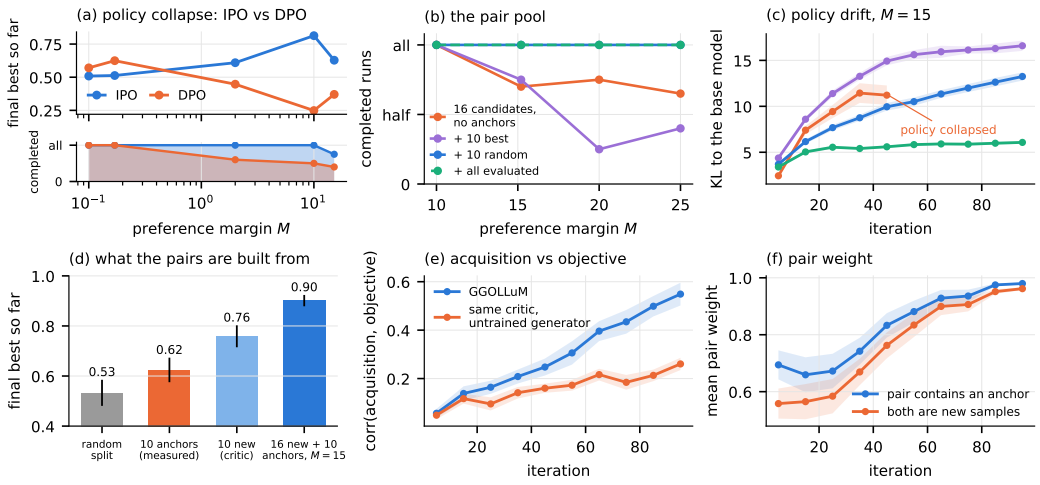


Figure 5: **What makes Bayesian feedback work** (SMILES-DRD2, 20 seeds per cell). *Top: what breaks, and what prevents it.* **(a)** Final best-so-far against the margin for both losses, with the fraction of runs reaching 110 evaluations shaded below. **(b)** With IPO, which evaluated points join the pair pool decides whether a large margin survives. **(c)** KL to the base model at $M{=}15$, colours as in (b); without anchors the policy collapses to about two unique samples per iteration and stops updating. *Bottom: how the loop learns.* **(d)** Pairs from a random split, from the measured values of 10 evaluated points, or from the critic's labels on 10 new samples ($M{=}10$); hatched: final configuration. **(e)** Correlation of the critic's acquisition with the objective on each method's own candidates. **(f)** Mean pair weight for pairs with and without an anchor. Bands and bars: one standard error.

6.3 Batch acquisition

In practice, experiments are often run in parallel. The bottom row of Figure 4 repeats every comparison at batch sizes $b \in \{1, 4, 8\}$ under the same budget of 110 evaluations. GGOLLuM leads at every batch size on every task, and on Catechol the gap to GOLLuM widens as the batch grows ($2.77/2.71/2.34$ against $2.31/1.92/1.74$). The GP baselines select batches jointly with qUCB, but no batching rule can extract a strong batch from an unadapted generator's weak pool. GGOLLuM has no batch acquisition function at all, it evaluates the first b samples of its policy but because IPO has already moved the policy towards high-acquisition regions, those samples are individually strong. The advantage narrows at $b{=}8$ on the molecular tasks (SMILES-DRD2 0.66 against 0.63 for BO), where fewer iterations leave fewer policy updates.

6.4 What makes Bayesian feedback work

Figure 5 examines what makes this work on SMILES-DRD2, where the critic starts weak and every quantity has room to move.

Stability. Preference optimization from about a hundred noisy labels is fragile. DPO's unbounded objective collapses the policy as the preference margin grows: its best-so-far falls to 0.25 at $M{=}10$ with half its runs failing, while IPO peaks there at 0.81 with every run completing (Figure 5a). A bounded loss is not enough on its own. With IPO fixed, pairs built only from the policy's new samples still collapse at larger margins, and the pair pool decides survival: at $M{=}25$, 65% of runs without anchors complete, 40% with the ten best evaluated points, and all runs with ten random ones (Figure 5b). Without anchors the policy concentrates to a few unique candidates, not enough to form pairs, and stops updating at a median of iteration 42 (Figure 5c).

Signal. With 10 points per update and no collapse ($M{=}10$), pairs labelled by the critic on the policy's own unevaluated samples reach 0.76, against 0.62 for pairs ranked by the measured values of 10 evaluated points and 0.53 for a random split (Figure 5d): the critic's labels carry information the measured data alone do not. The loop also improves its own judge. The critic's acquisition correlates 0.55 with the true objective on GGOLLuM's candidates over the last ten iterations, against 0.26 when

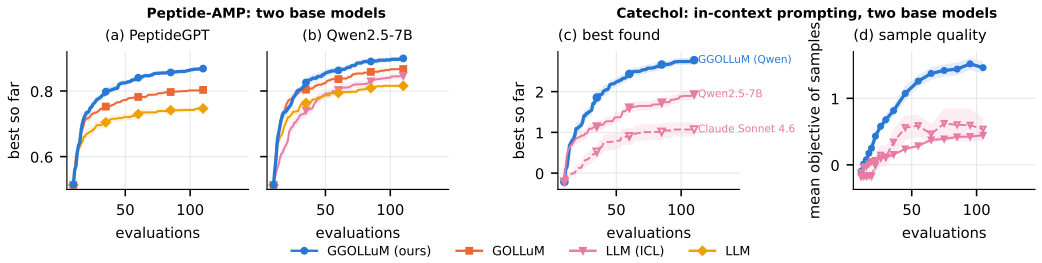


Figure 6: **Base models. Left:** Peptide-AMP with a 124M peptide-specialist **(a)** and a general-purpose 7B model **(b)**; colours denote methods as in Figure 3. **Right:** Catechol with in-context prompting on the open 7B base (solid) and on a frontier hosted model (dashed, open markers), alongside GGOLLuM on the open base; **(c)** best found, **(d)** mean objective of the samples. The frontier model is not fine-tuned, as its weights are not accessible; its runs are 10 seeds, of which 5 stopped early on API failures, and this comparison exists only on Catechol because the peptide prompts trigger provider safety filters. PeptideGPT has no prompt interface, so LLM (ICL) appears only for the general model.

the same critic ranks an untrained generator's samples (Figure 5e), and pairs containing an anchor are the ones it is most confident about early on (Figure 5f).

6.5 Does a stronger base model explain it?

The headline peptide numbers use PeptideGPT, a domain-specialized generator. Running every LLM-based method with the general-purpose Qwen2.5-7B instead lifts the whole ladder – the untrained model alone reaches 0.82 against 0.75 – but leaves the ordering unchanged, and GGOLLuM on the small specialist (0.87) still beats the untrained 7B model and the same model prompted with the history (0.85). On Catechol, a frontier hosted model given the history in context proposes better candidates on average than the open 7B base yet finds worse designs (1.07 against 1.93), because it commits early to the incumbent instead of exploring. A stronger proposer is therefore not a better optimizer (Appendix J).

7 Limitations and conclusion

The loop is only as good as what the policy proposes. Without anchors the policy collapses to about two unique samples per iteration and the update starves (Figure 5c). Anchors are an effective but blunt remedy; samplers that trade LLM-guided relevance against coverage of the design space are an open direction.

Our objectives are computable proxies, and proxies can be gamed. The best Peptide-AMP designs are poly-lysine-like repeats that score highly on the antimicrobial composite but that no peptide chemist would call a discovery. The optimizer is doing what it was asked; our numbers measure optimization ability, not design quality, and wet-lab feedback is the real test.

Sample efficiency is bought with compute. A GGOLLuM run costs roughly 40 GPU-minutes on the molecular tasks against about one for sampling the untrained model, which pays off only when experiments are expensive. Repeated runs of one configuration differ by about 0.12 in final best-so-far, so smaller gaps are within noise (Appendix H).

Conclusion. GGOLLuM turns a pretrained language model into the optimizer: preference pairs from a GP posterior move its generation distribution towards high-acquisition regions, so Bayesian uncertainty becomes policy alignment.

AI USE STATEMENT

We used an LLM-based assistant to help design and give feedback on experiments. All code was run and all results were checked by the authors, who take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

REFERENCES

Masaki Adachi, Brady Planden, David A. Howey, Michael A. Osborne, Sebastian Orbell, Natalia Ares, Krikamol Muandet, and Siu Lun Chau. Looping in the human collaborative and explainable bayesian optimization, 2024. URL https://arxiv.org/abs/2310.17273.

Dhruv Agarwal, Manoj Ghuhan Arivazhagan, Rajarshi Das, Sandesh Swamy, Sopan Khosla, and Rashmi Gangadharaiah. Searching for optimal solutions with llms via bayesian optimization. In *The Thirteenth International Conference on Learning Representations*.

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.

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

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*, pp. 4447–4455. PMLR, 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.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

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

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.

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.

Chih-Yu Chang, Milad Azvar, Chinedum Okwudire, and Raed Al Kontar. Llinbo: Trustworthy llm-in-the-loop bayesian optimization, 2025. URL https://arxiv.org/abs/2505.14756.

Sayak Ray Chowdhury, Anush Kini, and Nagarajan Natarajan. Provably robust dpo: Aligning language models with noisy feedback, 2024. URL https://arxiv.org/abs/2403.00409.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

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.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization, 2024. URL https://arxiv.org/abs/2310.02743.

Nathan C. Frey. gpt2_zinc_87m: a GPT-2 model pretrained on ZINC SMILES, 2022. URL https://huggingface.co/entropy/gpt2_zinc_87m. Hugging Face model.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022a. URL https://arxiv.org/abs/2210.10760.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W. Coley. Sample efficiency matters: A benchmark for practical molecular optimization, 2022b. URL https://arxiv.org/abs/2206.12411.

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

Samuel Genheden, Amol Thakkar, Veronika Chadimová, Jean-Louis Reymond, Ola Engkvist, and Esben Bjerrum. Aizynthfinder: a fast, robust and flexible open-source software for retrosynthetic planning. *Journal of Cheminformatics*, 12(1):1–9, 2020.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Nate Gruver, Samuel Stanton, Nathan C. Frey, Tim G. J. Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion, 2023. URL https://arxiv.org/abs/2305.20009.

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.

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, 2021. URL https://arxiv.org/abs/2102.09548.

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.

Carl Hvarfner, Danny Stoll, Artur Souza, Marius Lindauer, Frank Hutter, and Luigi Nardi. πbo: Augmenting acquisition functions with user beliefs for bayesian optimization, 2022. URL https://arxiv.org/abs/2204.11051.

Jan H. Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical Science*, 10(12):3567–3572, 2019. ISSN 2041-6539. doi: 10.1039/c8sc05372c. URL http://dx.doi.org/10.1039/C8SC05372C.

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

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.

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.

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, AlexanderSavelyev, 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.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback, 2024. URL https://arxiv.org/abs/2309.00267.

Seunghun Lee, Jinyoung Park, Jaewon Chu, Minseo Yoon, and Hyunwoo J. Kim. Latent bayesian optimization via autoregressive normalizing flows, 2025. URL https://arxiv.org/abs/2504.14889.

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

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.

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

Natalie Maus, Haydn T. Jones, Juston S. Moore, Matt J. Kusner, John Bradshaw, and Jacob R. Gardner. Local latent space bayesian optimization over structured inputs, 2023. URL https://arxiv.org/abs/2201.11872.

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

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.

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.

Tung Nguyen and Aditya Grover. Lico: Large language models for in-context molecular optimization, 2025. URL https://arxiv.org/abs/2406.18851.

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.

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.

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.

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

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

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.

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

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

James Requeima, John Bronskill, Dami Choi, Richard E. Turner, and David Duvenaud. Llm processes: Numerical predictive distributions conditioned on natural language, 2024. URL https://arxiv.org/abs/2405.12856.

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

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*, pp. e1604, 2022.

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.

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.

Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating bayesian optimization for biological sequence design with denoising autoencoders. In *International conference on machine learning*, pp. 20459–20478. PMLR, 2022.

Teague Sterling and John J. Irwin. Zinc 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, November 2015. ISSN 1549-960X. doi: 10.1021/acs.jcim.5b00559. URL http://dx.doi.org/10.1021/acs.jcim.5b00559.

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

Austin Tripp and José Miguel Hernández-Lobato. Genetic algorithms are strong baselines for molecule generation, 2023. URL https://arxiv.org/abs/2310.09267.

Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining, 2020. URL https://arxiv.org/abs/2006.09191.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alán Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models, 2025. URL https://arxiv.org/abs/2406.16976.

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.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models, 2023. URL https://arxiv.org/abs/2307.09702.

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

Wenjie Xu, Masaki Adachi, Colin N. Jones, and Michael A. Osborne. Principled bayesian optimisation in collaboration with human experts, 2024. URL https://arxiv.org/abs/2410.10452.

A BENCHMARK DETAILS

All five tasks start from 10 cold-start points (Appendix E) and allow 100 further evaluations, so every method sees exactly 110 evaluations of the objective. Table 3 summarizes them.

Table 3: Benchmarks. "Base LLM" is the generator shared by every LLM-based method on that task.

Task	Candidate	Objective	Base LLM	Constrained
Catechol	2 solvents + 3 continuous	scalarized yield/selectivity	Qwen2.5-7B	yes
SMILES-DRD2	SMILES string	DRD2 activity	gpt2-zinc-87M	no
SMILES-DRD2-Synth	SMILES string	DRD2 $\times$ synthesizability	gpt2-zinc-87M	no
Peptide-AMP	sequence, length 5–50	composite AMP score	PeptideGPT	yes
Peptide-Lytic	sequence, length 5–50	charge $\times$ amphipathicity	PeptideGPT	yes

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 and two products P_2 and P_3. We adopt the scalarized objective of the original benchmark study, which rewards total product yield and selectivity toward P_2 while penalizing harsher operating conditions:

$$f(S_A, S_B, b, \tau, T) = \lambda_1(P_2 + P_3) + \lambda_2 \frac{P_2}{P_2 + P_3} - \lambda_3 \frac{T - 175}{50} - \lambda_4 \tau, \tag{4}$$

with $\lambda_1 = 5$, $\lambda_2 = 1$, $\lambda_3 = 3$, $\lambda_4 = \frac{1}{20}$; the best measured condition scores 2.989. Because the measured dataset covers only part of the categorical-continuous product, novel combinations are evaluated through a digital twin: a random forest (500 trees, one-hot categoricals, continuous parameters passed through) fit on the measured table, with cross-validated $R^2 \approx 0.94$. Candidates within 10^{-3} of a measured point on every continuous parameter are scored by exact lookup.

SMILES-DRD2 (molecular design). The objective is dopamine receptor D2 activity, scored by the Therapeutics Data Commons (Huang et al., 2021) `DRD2` oracle, which wraps the SVM classifier of Olivecrona et al. (2017) (range $[0, 1]$). Candidates are SMILES strings canonicalized with RDKit (Landrum et al., 2022); unparseable strings are invalid. The cold-start pool is a curated library of about 150 drug-like molecules spanning major therapeutic classes, restricted to molecules below the library's median DRD2 score so that no method starts from an active.

SMILES-DRD2-Synth (molecular design with a synthesis constraint). The objective multiplies DRD2 activity by a synthesizability score from an AiZynthFinder (Genheden et al., 2020) retrosynthetic route search to commercial building blocks (the score of the top route, in $[0, 1]$; 0 when no route is found). Because the terms multiply, a potent but unmakeable molecule scores near zero, as does a makeable inactive one. Molecules with DRD2 < 0.01 are not sent to the route search and score their DRD2 value. Base model and cold start are as for SMILES-DRD2. The route search dominates the cost of a run (about 11 h for 110 evaluations).

Peptide-AMP (peptide design). The objective is a closed-form antimicrobial-peptide score computed with the `peptides` library: a weighted sum of hydrophobic moment (35%, mapped through $h/(h + 0.5)$), net charge at pH 7.4 (30%, normalized over $[0, +5]$), stability ($\max(0,\, 1 - \mathrm{II}/80)$ for instability index II, 20%) and Boman index ($b/(b + 2)$ for $b > 0$, 15%). Candidates are sequences over the 20 standard amino acids with length in $[5, 50]$. The cold-start pool is a curated set of about 200 known antimicrobial peptides (LL-37, magainins, cecropins, melittin, defensins, dermaseptins, protegrins, temporins and others) augmented with diverse short sequences.

Peptide-Lytic (membrane-lytic peptide design). The objective is $\min\big(1, \max(0, q)/6\big) \cdot h/(h+1)$, where q is the net charge at pH 7.0 and $h \geq 0$ the hydrophobic moment. The product cannot be

maximized by one term alone: poly-lysine maximizes charge but has near-zero hydrophobic moment, and a hydrophobic run has neither. Only amphipathic cationic sequences score well. The base model, PeptideGPT, was trained toward non-hemolytic output, i.e. away from this region of sequence space, which leaves a real gap for the policy to close.

B Extended related work

The role of the LLM in Bayesian optimization. We group prior work by what the LLM does. (i) *Feature map*: the LLM embeds candidates for a GP surrogate, fixed (Ranković & Schwaller, 2023; Kristiadi et al., 2024) or fine-tuned by marginal likelihood (Ranković & Schwaller, 2025); our critic is of this kind, and the GOLLuM baseline is exactly this critic without a trained generator. (ii) *Proposer*: the LLM suggests candidates, optionally conditioned on the history, and a surrogate or the evaluator selects among them (Liu et al., 2024; MacKnight et al., 2025; Agarwal et al.); evolutionary variants use the LLM as a mutation operator (Wang et al., 2025), and in-context methods condition on past evaluations (Nguyen & Grover, 2025). Our LLM and LLM (ICL) baselines, and the frontier-model comparison (Appendix I.8), cover this role; the generator is never trained. (iii) *Prior or acquisition modifier*: human or LLM knowledge reshapes the acquisition (Hvarfner et al., 2022; Xu et al., 2024; Adachi et al., 2024; Chang et al., 2025); these need a hand-specified prior or an expert in the loop, which our setting does not assume. (iv) *Surrogate*: the LLM itself predicts the objective (Requeima et al., 2024). (v) *Trained policy*: ToSFiT (Menet et al., 2025) fine-tunes the LLM with an on-policy policy gradient on a GP-derived reward, and GGOLLuM fine-tunes it with preference pairs from the GP posterior. Role (v) is sparsely populated, and our comparison with ToSFiT holds the critic, adapter and budget fixed (Appendix I.7).

Generative design at small budgets. Latent-space BO trains a variational autoencoder and optimizes in its latent space (Gómez-Bombarelli et al., 2018; Tripp et al., 2020; Maus et al., 2023; Lee et al., 2025); reinforcement learning (Olivecrona et al., 2017), guided diffusion (Stanton et al., 2022; Gruver et al., 2023) and genetic algorithms (Jensen, 2019; Tripp & Hernández-Lobato, 2023) are strong molecular baselines when thousands of oracle calls are available (Gao et al., 2022b). Our setting is about a hundred evaluations from a cold start, where a graph genetic algorithm seeded with the ten initial molecules reaches a best-so-far of 0.19 on SMILES-DRD2 and 0.14 on SMILES-DRD2-Synth, far below every LLM-based method.

Alignment signals. RLHF (Christiano et al., 2023; Ouyang et al., 2022) and RLAIF (Bai et al., 2022; Lee et al., 2024) supply preferences from people or a model; DPO and IPO (Rafailov et al., 2023; Azar et al., 2024) optimize them without an explicit reward model. Point-estimate rewards are over-optimized (Gao et al., 2022a), which ensembles mitigate (Coste et al., 2024), and noisy labels call for robust objectives (Chowdhury et al., 2024). Bayesian feedback is a posterior rather than a point estimate, so each preference comes with a confidence (the pair weight) and the reward model is refit on real data after every experiment; anchors play a role similar to mixing pretraining data into the update to prevent the policy from degenerating (Ouyang et al., 2022).

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, ToSFiT, BO, GOLLuM, 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 & 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 and GGOLLuM pool-mean trajectories diverge in Figure 3 (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 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 & 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 (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 (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 (ICL)** baseline (Figure 3, Catechol), 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 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 IMPLEMENTATION DETAILS

Table 4 lists every setting of GGOLLuM; one configuration is used on all five tasks.

Table 4: GGOLLuM configuration, identical on all tasks.

Setting	Value
Candidates generated per iteration	$N = 16$, temperature 1
Evaluation budget	10 cold-start + 100 evaluations
Critic	deep-kernel GP, Matérn-5/2 on pooled LLM features, `ScaleToBounds`$(-1, 1)$
Critic training	one AdamW optimizer: critic LoRA lr 2×10^{-3}, GP hyperparameters lr 0.2
Acquisition	UCB, $\beta_{\mathrm{UCB}} = 4$
LoRA (actor and critic)	rank 4, $\alpha = 16$, dropout 0.2, RSLoRA, top 25% of layers
Pair pool	N new candidates + $k = 10$ evaluated points drawn uniformly at random
Pair construction	median split on α; keep pairs with $\mu_w > \mu_l$; weight $\min(z_{wl}, 1)$
Pairs per iteration	at most 96, drawn uniformly from the retained pairs; no per-winner cap
Preference loss	weighted IPO, $\beta_{\mathrm{IPO}} = 0.033$ (target margin $M = 15.15$)
Actor optimizer	AdamW, lr 5×10^{-4} halved after every epoch, 5 epochs, minibatch 32
Reference policy	frozen base model
Candidate selection	first b valid, novel candidates among the N samples
Run termination	110 evaluations, or 20 consecutive iterations without a valid novel candidate

Base models. Catechol uses **Qwen2.5-7B** (Bai et al., 2023), the *base* (not instruction-tuned) variant, so the preference signal reshapes a general next-token distribution rather than one already aligned to follow instructions. Both SMILES tasks use **gpt2-zinc-87M** (Frey, 2022), a GPT-2 model pretrained on ZINC that emits valid molecules but places almost no mass on DRD2-active chemistry. Both peptide tasks use **PeptideGPT**, a GPT-2 model pretrained on UniProt sequences and selected for non-hemolytic output. Every pairing is plausible for the task but not aligned with its objective.

Actor and critic on one base model. Two LoRA adapters sit on the same frozen base. The *actor* adapter generates candidates, is trained by weighted IPO, and accumulates across iterations. The *critic* adapter provides the GP's feature map; it is re-initialized every iteration and trained jointly with the GP hyperparameters by maximizing the exact marginal likelihood, with gradients flowing through the kernel and the pooling into the LoRA weights. Only the active adapter receives gradients, so neither overwrites the other. Features are the final hidden layer, pooled by the last token for Qwen2.5-7B and by the mean for the GPT-2 models.

From posterior to preference pairs. Each iteration scores the pool (the N new candidates plus k evaluated points) with the refit critic, giving the acquisition value α_i, posterior mean μ_i and variance σ_i^2 of every candidate. Candidates are sorted by α; the top half are winners and the bottom half losers. Every winner is paired with every loser for which $\mu_w > \mu_l$, and the pair is weighted by $\min(z_{wl}, 1)$ with $z_{wl} = (\mu_w - \mu_l)/\sqrt{\sigma_w^2 + \sigma_l^2}$. At most 96 of the retained pairs, drawn uniformly, enter the loss. The evaluated points have near-zero posterior variance, so they supply pairs the critic is confident about; Appendix I.2 shows why they are needed.

Hyperparameter selection. The configuration was selected once on SMILES-DRD2, from a single sweep over target margin $\times$ pair pool $\times$ weighting with 20 seeds per cell (Appendix I.2), and transferred unchanged to the other four tasks. We chose the setting whose performance varies least across margins rather than the single best cell, since it is the one expected to transfer.

F PREFERENCE PAIRS: THE CONFIDENCE WEIGHT

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 5.

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
$\mathcal{C}_t \leftarrow \pi_t(\cdot), \quad |\mathcal{C}_t| = N$
add anchors: k evaluated points drawn uniformly from $\mathcal{D}_t$
$\mathcal{C}_t \leftarrow \mathcal{C}_t \cup \mathcal{A}_t, \quad \mathcal{A}_t \subset \mathcal{D}_t, \ |\mathcal{A}_t| = k$
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
$\mathcal{P}_t \leftarrow \text{pairs}(\mathcal{C}_t)$ with weights $\min(z_{wl}, 1)$
update actor via IPO
$\pi_{t+1} \leftarrow \text{IPO}(\pi_t, \ \pi_{\text{ref}}{=}\pi_{\text{base}}, \ \mathcal{P}_t)$
pick first-b valid novel candidates and evaluate
$\{c_1^\star, \ldots, c_b^\star\} \subset \mathcal{C}_t, \quad y_j^\star \leftarrow \text{Evaluate}(c_j^\star)$
$\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(c_j^\star, y_j^\star)\}_{j=1}^b$ {update dataset}
return $\mathcal{D}_{t+1}, \ \pi_{t+1}, \ g_t$

G BASELINES

Table 5: 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
BO (512 random)	random	Matérn-5/2 on classical feats.	–	non-LLM BO floor
LLM	LLM	–	base only	LLM generator floor
LLM (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)
Graph GA	mutation/crossover	–	–	non-LLM molecular baseline (Jensen, 2019; Trip
BO (LLM samples)	LLM	Matérn-5/2 on FPs	–	classical kernel on LLM pool
GOLLuM	LLM	deep-kernel GP	marginal lik.	GOLLuM critic (Ranković & Schwaller, 2025)
GGOLLuM	LLM	deep-kernel GP	MLL + IPO	**ours**

Every baseline shares the task's base model, candidate budget ($N = 16$ per iteration), cold start, evaluation budget and constrained decoding.

- **LLM**: the base model samples N candidates and the first valid novel one is evaluated. No critic, no training. The name follows MacKnight et al. (2025).
- **LLM (ICL)**: as LLM, with the 10 best evaluations so far in the prompt. Catechol only, since the SMILES and peptide base models have no prompt interface.
- **BO**: a standard GP (Matérn-5/2 on Morgan fingerprints for molecules, dipeptide counts for peptides) selects from the same LLM-generated pool with UCB (sequential-greedy qUCB for $b > 1$).
- **BO**: a standard GP on one-hot and scaled tabular features selects from 512 candidates drawn from the design space. Catechol only: on the molecular and peptide tasks a uniformly random pool would be almost entirely invalid or meaningless, which is why BO is used there.
- **GOLLuM**: the GGOLLuM critic (deep-kernel GP with its LoRA feature map) selects from the pool of the *untrained* base model. This is GGOLLuM without the actor update.
- **ToSFiT**: ToSFiT (Menet et al., 2025) implemented inside our framework: same base model, same two adapters, same critic and prompts; the actor is trained on-policy with the ToSFiT (VBOS) pseudo-reward and GRPO-style advantage standardization instead of IPO on preference pairs. Appendix I.7 studies its update budget and learning rate.

H EXPERIMENTAL PROTOCOL AND STATISTICS

Every cell in every table is 20 independent seeds. Running an identical configuration again can shift final best-so-far by about 0.11 on SMILES-DRD2, and the 20-seed noise floor is about 0.12, so we treat differences below 0.12 as directional. The 10 initial points of every run are drawn from the task's candidate library with the constraint that none exceeds the library's median objective (*cold start*), so no method begins from a good point. A run *completes* if it reaches 110 evaluations; it *collapses* if it terminates earlier because the actor produced no valid novel candidate for 20 consecutive iterations. Best-so-far is reported at 110 evaluations; for $b = 8$ the last batch overshoots to 114 and we read the value at 110.

I ABLATIONS

All ablations are on SMILES-DRD2 unless stated, with $M = 1/(2\beta_{\mathrm{IPO}})$. They are ordered so that each builds on the previous one: the preference loss, then the pool of candidates pairs are drawn from, then how the critic's posterior is turned into pairs, then what the critic gains from the actor, and finally alternatives to preference optimization.

I.1 PREFERENCE LOSS: IPO VERSUS DPO ACROSS MARGINS

Table 6: IPO vs DPO, pool of new candidates only, uniform pair weights, at most 8 pairs per winner, frozen-base reference. Mean final best-so-far over all 20 runs of a cell, including runs that stopped early; completed runs in parentheses. Rows marked † are one sweep containing both losses; the others pair two sweeps with identical grids that differ only in the loss.

M (β)	IPO	DPO
25 (0.02)	0.255 (0/20)	0.165 (0/20)
15.15 (0.033)†	0.628 (15/20)	0.370 (8/20)
10 (0.05)†	**0.813** (20/20)	0.249 (10/20)
5 (0.1)	0.649 (20/20)	0.284 (0/20)
2 (0.25)†	0.609 (20/20)	0.447 (12/20)
1 (0.5)	0.569 (20/20)	0.396 (0/20)
0.5 (1.0)	0.544 (20/20)	0.585 (12/20)
0.25 (2.0)	0.500 (20/20)	0.608 (20/20)
0.17 (3.0)†	0.513 (20/20)	**0.625** (20/20)
0.1 (5.0)†	0.509 (20/20)	0.571 (20/20)

The ordering reverses with the margin: IPO leads for $M \geq 1$ and DPO for $M \leq 0.5$, consistently across three sweeps on disjoint grids. IPO's best cell (0.813 at $M = 10$) is better than DPO's best (0.625 at $M = 0.17$), and at large margins DPO runs mostly stop early while IPO runs complete. This matches the motivation for IPO (Azar et al., 2024): DPO's implicit regularization weakens as preferences become near-deterministic, while IPO's squared loss keeps pulling toward the reference policy.

I.2 THE PAIR POOL AND ACTOR COLLAPSE

Table 7: Pair pool × margin, one sweep (360 runs), clip weights. BSF / fraction of runs completed.

M	new candidates only	+ all evaluated	+ 10 random evaluated
10	**0.826** / 1.00	0.682 / 1.00	0.813 / 1.00
15.15	0.513 / 0.70	0.824 / 1.00	**0.901** / 1.00
25	0.389 / 0.70	0.881 / 1.00	0.899 / 1.00

Without evaluated points in the pool, raising the margin makes the policy collapse (Table 8): with pairs built only from new samples, completion falls to about two thirds of runs by M=15 and best-so-far roughly halves. Adding evaluated points restores completion, but which points matters at

large margins: the ten best evaluated points slow the collapse without preventing it, while ten random ones or all evaluated points keep every run alive. At $M{=}10$, where nothing collapses, the pool does not help, so the effect of anchors is collapse prevention rather than "more data helps".

Table 8: Which evaluated points join the pair pool, across margins, SMILES-DRD2, clipped weights, 20 seeds per cell. Each entry is BSF / fraction of runs that reach 110 evaluations.

Pool	$M = 10$	$M = 15$	$M = 20$	$M = 25$
new samples only	0.83 / 1.00	0.51 / 0.70	0.49 / 0.75	0.39 / 0.65
+ 10 best evaluated	0.86 / 1.00	0.93 / 0.75	0.79 / 0.25	0.70 / 0.40
+ 10 random evaluated (ours)	0.81 / 1.00	0.90 / 1.00	0.90 / 1.00	0.90 / 1.00
+ all evaluated	0.68 / 1.00	0.82 / 1.00	0.81 / 1.00	0.88 / 1.00

The ten best evaluated points only partly prevent collapse; ten random ones prevent it almost entirely. What matters is the diversity of the anchors, not their quality.

I.3 From posterior to preferences: where the critic's signal enters

The critic enters pair construction in three places: which candidates become winners (the α split), which pairs are kept ($\mu_w > \mu_l$), and how much each counts ($\min(z_{wl}, 1)$). Table 9 switches them off one at a time.

Table 9: Pair-construction arms, SMILES-DRD2, final configuration otherwise, 20 seeds each. BSF over all runs; AUC is the mean best-so-far over evaluations 11–110.

Arm	Winners chosen by	μ filter	Weight	BSF	AUC
A (ours)	α (UCB)	yes	$\min(z, 1)$	0.866	**0.579**
B	α	yes	uniform	0.840	0.526
C	μ ($\beta_{\mathrm{UCB}} = 0$)	yes	$\min(z, 1)$	0.890	0.555
D	μ	yes	uniform	**0.902**	0.536
E	α	no	uniform	0.866	0.501
F	random	no	uniform	0.545	0.363

A vs B isolates the confidence weight, A vs C the role of σ in choosing winners, D removes σ entirely, E lets exploration decide the direction of a preference, and F removes the critic from pair construction altogether. All arms except F are within about 0.06 of each other, below the 0.12 that separates repeated runs of one configuration; removing the critic's ordering (F) costs 0.32. The reason is visible in the pairs themselves: in the final configuration only 4% of winners are winners because of σ (in the α top half but not the μ top half), 98% of pairs already satisfy the μ filter, and 58% of retained pairs contain an anchor, whose pairs carry a higher mean weight (0.83 against 0.77). The critic's posterior mean does most of the work.

Confidence weighting across pools and margins. Table 10 repeats the weighting comparison in all nine cells of the pool $\times$ margin sweep.

Table 10: Clip ($\min(z, 1)$) vs uniform weights, BSF, same sweep as Table 7.

	new only		+ all evaluated		+ 10 random	
M	clip	unif.	clip	unif.	clip	unif.
10	0.826	0.858	0.682	0.634	0.813	0.683
15.15	0.513	0.545	0.824	0.774	**0.901**	0.824
25	0.389	0.362	0.881	0.845	0.899	0.866

With evaluated points in the pool, confidence weighting improves best-so-far by 0.062 on average ($p = 0.0097$, 120 runs per arm); without them it has no effect (-0.012, $p = 0.83$, 60 runs per arm). The same holds for the exploration weight of the acquisition (Appendix I.4). Without anchors every pair compares two new candidates with similar posterior variance, so the weights are nearly uniform;

the anchors, which have near-zero variance, are what give the posterior uncertainty something to act on.

I.4 Exploration weight of the acquisition

Table 11: UCB exploration weight. Left: final configuration (one sweep). Right: pool of new candidates only, uniform weights, $M = 10$ (another sweep). BSF.

β_{UCB}	with evaluated points	new candidates only
0 (mean only)	0.855 ± 0.137	0.795
2	0.865 ± 0.148	0.755
4 (default)	0.886 ± 0.144	0.795
16	0.908 ± 0.099	0.792

With evaluated points, best-so-far rises monotonically with β_{UCB} and ranking by the mean alone is weakest; without them the setting has no effect (ANOVA $p = 0.90$). The spread with evaluated points (0.05) is below the noise floor, so we read it as directional.

I.5 Which candidate is evaluated

By default GGOLLuM evaluates the first valid novel sample from the actor. Letting the critic choose instead (the highest-α candidate) gives 0.893 ± 0.211 vs 0.855 ± 0.116 at 110 evaluations (Welch $p = 0.48$, one sweep). Critic selection converges faster (larger area under the best-so-far curve in 16 of 20 paired seeds, sign test $p = 0.012$) and the actor catches up by the end of the budget.

I.6 Does the critic depend on the actor?

GGOLLuM's critic is refit every iteration on data the actor chose, so it can improve because the actor feeds it better data or simply because data accumulate. We log the correlation between the critic's acquisition and the true objective on each iteration's candidates for GGOLLuM and for GOLLuM (same critic, untrained generator), 20 seeds each. By the end of the run the correlation is 0.55 for GGOLLuM and 0.26 for GOLLuM (mean over the last ten iterations, Welch $p = 8 \times 10^{-6}$), and the held-out negative log predictive density at the end of the run is 1.2 vs 7.5 (Welch $p = 0.009$); held-out R^2 is negative for both. The correlation is computed on each method's own candidates, so it reflects both the critic and the candidates it has to rank.

I.7 Preference optimization versus on-policy RL (ToSFiT)

ToSFiT (Menet et al., 2025) is the closest method: it also trains the generator from a GP, but on-policy, with a Thompson-sampling (VBOS) pseudo-reward and GRPO-style advantages. We implement it inside our framework, so the base model, adapters, critic and prompts are shared and only the actor objective differs. Its update budget is set by c, the number of on-policy gradient steps per BO iteration; each step samples N fresh candidates, scores them with the critic (no evaluation of the objective) and takes one step, and only the last sample is evaluated. The number of objective evaluations therefore stays matched, while compute grows with c (Table 12).

Table 12: Actor-update cost per BO iteration.

Method	LLM samples	critic scorings	gradient steps
GGOLLuM	16	26	$\approx$3
ToSFiT, $c = 1$	16	16	1
ToSFiT, $c = 4$	64	64	4
ToSFiT, $c = 8$	128	128	8

At the learning rate of the ToSFiT recipe (10^{-5}, SGD) the policy barely moves over 100 iterations, so we sweep the learning rate, the optimizer (SGD as in the original work, and AdamW as used by GGOLLuM) and the number of on-policy updates per iteration c, always with ToSFiT's own VBOS

pseudo-reward. The best cell uses c=4, i.e. four times the samples per iteration of GGOLLuM; at a matched sampling budget (c=1) the best is 0.618 (AdamW, 5×10^{-4}), which is the setting reported in the main text. More updates per iteration do not help, and large learning rates collapse the policy.

Table 13: ToSFiT with its VBOS pseudo-reward, SMILES-DRD2, BSF over all runs, 20 seeds per cell; c is the number of on-policy updates (and fresh batches of 16 samples) per iteration. Bold: best cell.

optimizer	lr	$c = 1$	$c = 4$	$c = 8$
SGD	10^{-5}	0.521	0.560	—
SGD	10^{-4}	—	0.602	0.520
SGD	10^{-3}	—	0.602	0.585
SGD	10^{-2}	—	0.360	—
AdamW	10^{-4}	0.534	**0.672**	0.522
AdamW	5×10^{-4}	0.618	0.398	—
AdamW	10^{-3}	0.437	0.285	0.342

I.8 A FRONTIER LLM AS THE PROPOSER

We ran Claude Sonnet 4.6 as the LLM (ICL) proposer on Catechol with 10 seeds; the API cost of the full protocol was the limiting factor, and five of the ten runs stopped early on API failures, keeping their last value. Table 14 reports every method at 56 evaluations, where all frontier runs are still active, and at the full budget of 110.

Table 14: Catechol, best-so-far (mean $\pm$ std over all runs; 20 seeds, 10 for the frontier model).

Method	Base model	56 evaluations	110 evaluations
GGOLLuM	Qwen2.5-7B	**2.33 $\pm$ 0.50**	**2.77 $\pm$ 0.25**
GOLLuM	Qwen2.5-7B	1.92 $\pm$ 0.47	2.31 $\pm$ 0.19
ToSFiT	Qwen2.5-7B	1.54 $\pm$ 0.82	2.17 $\pm$ 0.76
BO	–	1.79 $\pm$ 0.43	1.91 $\pm$ 0.39
LLM (ICL)	Qwen2.5-7B	1.52 $\pm$ 0.50	1.93 $\pm$ 0.47
LLM	Qwen2.5-7B	1.08 $\pm$ 0.41	1.57 $\pm$ 0.39
LLM (ICL)	Claude Sonnet 4.6	0.89 $\pm$ 0.64	1.07 $\pm$ 0.55

The frontier model is last at both budgets. It proposes better candidates on average than the same prompting with Qwen2.5-7B (Figure 6d) but the spread of what it evaluates is the narrowest of all methods: it commits early to the neighbourhood of the incumbent and repeats near-duplicates, which is the wrong strategy from a cold start. It generates JSON (the open models use a single-line format); all of its proposals parsed and validated.

I.9 BATCH ACQUISITION AND BATCH DIVERSITY

For $b > 1$, GGOLLuM evaluates the first b valid novel samples of the adapted actor, with no batch acquisition function; the GP baselines select their batch with sequential-greedy qUCB. All batch sizes use 110 evaluations, so $b = 8$ gets about 13 actor updates instead of 100. For every evaluated batch we log its diversity: the fraction of unique candidates and a mean pairwise distance in $[0, 1]$ (one minus Tanimoto similarity of Morgan fingerprints for molecules; edit distance divided by the longer length for peptides; distinct solvent pairs and the spread of the scaled continuous parameters for catechol). Diversity is read next to the batch's mean objective, since a random batch is also diverse.

J BASE MODELS: FULL RESULTS

The headline peptide numbers in Table 2 use PeptideGPT, a domain-specialized generator. We additionally ran every LLM-based method with Qwen2.5-7B as the base (Figure 6) to isolate the contribution of domain-specialization from the rest of the stack.

Table 15: Best-so-far within 110 evaluations for batch size $b = 1/4/8$, 20 seeds, mean over all runs. BO is a GP over 512 random conditions on Catechol and over fingerprint-featurized LLM samples elsewhere.

Task	GGOLLuM	GOLLuM	BO	LLM
Catechol	2.77/2.71/2.34	2.31/1.92/1.74	1.91/1.86/1.66	1.57/1.46/1.49
SMILES-DRD2	0.90/0.77/0.66	0.80/0.62/0.61	0.64/0.70/0.63	0.52/0.59/0.54
SMILES-DRD2-Synth	0.84/0.75/0.61	0.75/0.60/0.53	0.67/0.60/0.56	0.51/0.53/0.54
Peptide-AMP	0.87/0.84/0.82	0.80/0.78/0.78	0.77/0.77/0.76	0.75/0.76/0.76
Peptide-Lytic	0.53/0.53/0.54	0.52/0.51/0.49	0.50/0.49/0.49	0.49/0.48/0.47

Two findings emerge. First, GGOLLuM reaches the highest best-so-far with either base: 0.87 with PeptideGPT and 0.90 with Qwen2.5-7B, against 0.80 and 0.87 for GOLLuM. The advantage is not an artefact of pairing IPO with a weak domain-specialist base. Second, the floor is higher for the general model: the untrained Qwen2.5-7B reaches 0.82 against 0.75 for PeptideGPT, and prompting it with the ten best evaluations (LLM (ICL), Qwen only, since PeptideGPT is a BOS-conditioned sequence sampler with no prompt interface) reaches 0.85. A larger general-purpose model is already a stronger peptide sampler, closing part of the gap to GGOLLuM without any critic in the loop, yet GGOLLuM on the 124M specialist still exceeds both. On Catechol, the same in-context prompting with a frontier hosted model reaches 1.07 against 1.93 with Qwen2.5-7B and 2.77 for GGOLLuM, while proposing better candidates on average (Figure 6d): it concentrates on the neighbourhood of the incumbent. A stronger base raises the floor, not the ceiling.