Calibrating Generative Models to Feature Distributions with MMD Finetuning

Nathaniel L. Diamant
Stanford University
diamant@stanford.edu

Brian L. Trippe
Stanford University
btrippe@stanford.edu

Abstract

Generative models can produce individually plausible samples while deviating substantially from a target set in the distribution of key features. For example, a model pretrained on broad drug-like chemical space may generate molecules whose molecular features differ from those of a therapeutic class of interest, such as known antibiotics. Correcting such distributional *miscalibration* is challenging: direct finetuning on the target set can overfit and does not control which features are matched. To fill this gap, we introduce kernel Calibrating Generative Models (kCGM). kCGM minimizes a maximum mean discrepancy (MMD) between generated and target feature distributions using an unbiased score-function estimator, with KL regularization to remain close to the pretrained model. On a target set of 174 antibiotics, direct finetuning sacrifices chemical validity for feature-distribution matching, whereas kCGM improves target feature matching while increasing validity. We further demonstrate kCGM in protein and DNA generation tasks, showing it can adapt autoregressive, continuous-space diffusion, and discrete diffusion models using only feature-level supervision. Code is available at https://github.com/smithhenryd/cgm.

1 Introduction

Generative models of molecules frequently suffer misalignment of the distribution of features of interest. Consider antibiotic drug discovery: models trained on large, diverse small-molecule datasets can generate plausible molecules whose chemical-group distributions differ substantially from known antibiotics. Finetuning directly on known antibiotics risks overfitting and cannot specify which features of the target set should be matched. More broadly, aligning a generative model to a target feature distribution is a recurring goal in biomolecular generative modeling: protein generators fail to match natural secondary-structure distributions [Lu et al., 2025], and regulatory DNA generators fail to reproduce realistic activity-profile distributions [Sarkar et al., 2024]. While aligning, or *calibrating*, these models to match the distribution of such features to a reference set could enable more faithful modeling and design of biomolecules, satisfactory finetuning algorithms do not exist.

An ideal finetuning method for calibrating feature distributions should satisfy two properties. First, it should work from feature-level supervision, matching selected aspects of a target distribution without requiring full target samples needed for direct finetuning. Second, it should allow discrete-valued and black-box features such as the bit-vector Morgan fingerprints commonly used in small-molecule drug discovery.

To satisfy these goals, we introduce kernel Calibrating Generative Models (kCGM), a method for finetuning generative models to match target feature distributions from feature-space samples alone. kCGM extends Calibrating Generative Models (CGM) [Smith et al., 2026], which finetunes a pretrained model so that feature *means* match target values. Like CGM, kCGM applies when one can sample from the model and compute score-function gradients of the sampling trajectory, while allowing the feature

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

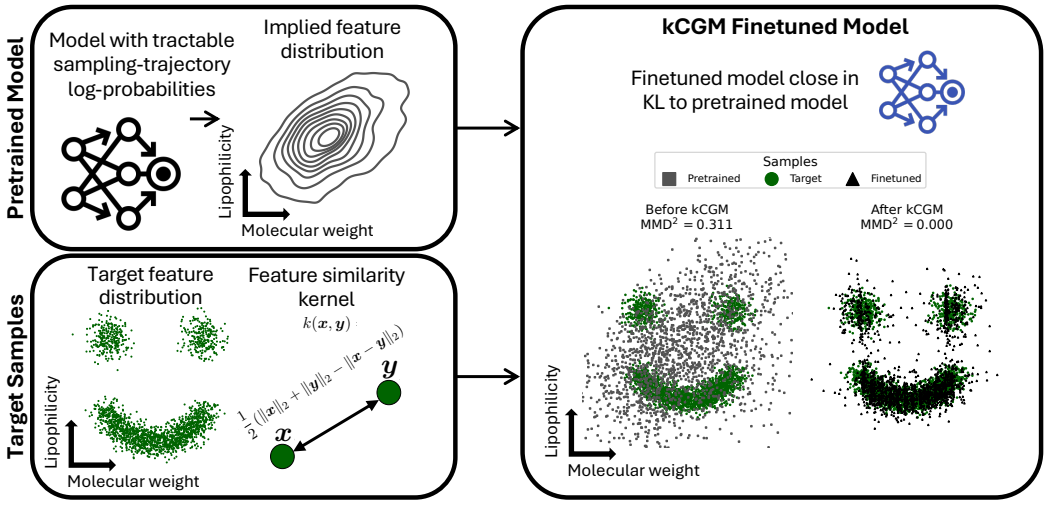

Figure 1: kCGM overview. The user specifies a pretrained model with tractable sampling-trajectory log-probabilities (top left) and a target feature distribution through samples in feature space (bottom left). kCGM compares these distributions using a kernel similarity between feature samples and finetunes the model, with KL regularization toward the pretrained model. The result is a finetuned model whose generated samples more closely match the target feature distribution (right).

map to be black-box or non-differentiable. We show an overview of kCGM in Figure 1. Rather than discretizing the target distribution as one would with CGM, kCGM minimizes an unbiased finite-sample estimator of the kernel Maximum Mean Discrepancy (MMD), enabling more favorable scaling to higher-dimensional features and tighter agreement with the target distribution.

This paper presents three contributions. First, we formulate feature-distribution matching as KL-regularized MMD finetuning from target feature samples alone. Second, we derive an unbiased score-function gradient estimator for the resulting objective, enabling non-differentiable feature maps and non-reparameterizable samplers. Finally, across protein, small-molecule, and regulatory DNA generation tasks, kCGM outperforms CGM over the tested KL range and outperforms direct finetuning on the antibiotic task while improving chemical validity.

1.1 Related work

Several lines of work study how to adapt a generative model to a target population or task. We discuss the most relevant: direct finetuning on samples, reward finetuning, CGM, and prior uses of MMD in generative modeling. No prior work has directly addressed the task of finetuning a generative model to match a target distribution over user-specified features.

Direct/supervised finetuning. A standard way to adapt a generative model is to continue likelihood training on a smaller, more relevant target dataset [Yosinski et al., 2014]. This two-stage procedure, often called supervised finetuning, is widely used to adapt pretrained models to task-specific data [Chen et al., 2021, Ouyang et al., 2022]. In drug design, models are pretrained on large, broad molecular datasets and then finetuned on molecules with a desired activity [Segler et al., 2018]. While direct finetuning can make generations more target-like, it requires full target samples, can overfit small target sets, and cannot specify which aspects of the target distribution should be preserved.

Reward finetuning. Reward finetuning methods adapt a generative model by assigning a scalar reward to each generated sample and optimizing the model to increase expected reward, e.g. Christiano et al. [2017], Rafailov et al. [2023], Uehara et al. [2024], Domingo-Enrich et al. [2025], Zheng et al. [2025]. In contrast, kCGM is designed for settings where the goal is to match a target feature distribution rather than maximize a reward assigned to individual samples.

Calibrating Generative Models (CGM). kCGM generalizes CGM [Smith et al., 2026], the prior method closest to the present work. Whereas CGM matches target feature means, kCGM matches feature distributions through an MMD. The special case of kCGM with the dot-product kernel recovers

CGM-relax, a variant of CGM; Appendix Section A.1 formalizes this observation and provides a proof.

Smith et al. [2026] show CGM can be extended toward distribution matching using a discretization of the empirical multivariate cumulative distribution function (CDF) of target set features. However, this introduces discretization hyperparameters and, because the CDF representation grows exponentially with dimension, is impractical for features of dimension larger than three. Moreover, in application this approach is shown to leave significant miscalibration error.

MMD in generative modeling. MMD has many uses in training and finetuning generative models, including DISCO Nets [Bouchacourt et al., 2016], MMD-GAN [Li et al., 2017], and MMD-based losses for diffusion models [De Bortoli et al., 2025, De Santi et al., 2025]. These works rely on reparameterization gradients, which are not directly applicable when sampling or the features of interest are non-differentiable.

Score-function estimators have also been used to optimize MMD objectives in non-reparameterizable settings. Masood and Doshi-Velez [2019] use policy-gradient estimators for an MMD diversity term between policy trajectory distributions, and Manenti et al. [2025] use a score-function estimator for an MMD loss between stochastic predictive output distributions in graph structure learning. Miao et al. [2024] use an MMD-based signal to finetune image diffusion models, but target diversity maximization rather than exact feature-distribution matching, and rely on a heuristic decomposition of set-level rewards into per-sample rewards. In contrast, kCGM finetunes pretrained generative models to match externally specified target feature distributions, incorporates KL regularization to the pretrained model, and derives the finite-sample MMD coefficients and leave-one-out baselines needed for practical score-function optimization.

2 Feature distribution matching with MMD

This section presents kCGM. We first make precise our notation and assumptions and then present our MMD objective, regularization, and finetuning algorithm.

Notation and assumptions. Let $p_{\theta_{\mathrm{pre}}}(\boldsymbol{x})$ be a pretrained generative model with parameters θ_{pre}, and let $\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})$ denote features of a generated sample in a set $\mathcal{Y}$. In addition, we have examples of features $\boldsymbol{y}_{1:N}^{\mathrm{targ}} := (\boldsymbol{y}_1^{\mathrm{targ}}, \ldots, \boldsymbol{y}_N^{\mathrm{targ}})$, from a target distribution.

Our goal is to choose θ so that if $\boldsymbol{x} \sim p_\theta$, then the induced feature distribution of $\boldsymbol{h}(\boldsymbol{x})$ matches a target distribution. The features may be extracted by a black-box, non-differentiable procedure.

We assume samples $\boldsymbol{x} \sim p_\theta$ and their log-probabilities may be readily drawn and computed, respectively; this assumption enables score-function gradients [Williams, 1992]. Here, $\boldsymbol{x}$ may denote the full sampling trajectory rather than only the terminal generated object. For an autoregressive model, $\boldsymbol{x}$ is the generated token sequence and $\log p_\theta(\boldsymbol{x})$ is the usual autoregressive log-probability. For a diffusion sampler, $\boldsymbol{x}$ includes the intermediate reverse-process states, so $\log p_\theta(\boldsymbol{x})$ decomposes into transition log-probabilities along the sampled denoising path. The feature map $\boldsymbol{h}$ acts on the terminal generated sample of the trajectory, such as the fully denoised output of a diffusion sampler. Thus the score-function estimator does not require differentiating through $\boldsymbol{h}$ or the sampling procedure.

MMD objective and the kernel choice. To compare feature distributions from samples, we use a maximum mean discrepancy (MMD). For distributions P and Q and a positive definite kernel $k(\cdot, \cdot)$, the population squared MMD is

$$\mathrm{MMD}_k^2(P, Q) = \mathbb{E}_{\boldsymbol{y}, \boldsymbol{y}' \sim P}[k(\boldsymbol{y}, \boldsymbol{y}')] - 2\mathbb{E}_{\boldsymbol{y} \sim P, \boldsymbol{y}' \sim Q}[k(\boldsymbol{y}, \boldsymbol{y}')] + \mathbb{E}_{\boldsymbol{y}, \boldsymbol{y}' \sim Q}[k(\boldsymbol{y}, \boldsymbol{y}')]. \tag{1}$$

In our setting, P is the model-induced feature distribution of $\boldsymbol{h}(\boldsymbol{x})$ for $\boldsymbol{x} \sim p_\theta$, and Q is the empirical distribution of target sample features $\boldsymbol{y}_{1:N}^{\mathrm{targ}}$.

The kernel k determines how differences between feature distributions contribute to the MMD. For example, the MMD defined by the dot-product kernel $k(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v}$ compares only means. In fact, with the dot-product kernel, $\mathrm{MMD}_k^2(P, Q)$ exactly recovers the CGM-relax objective and so does not distinguish distributions with identical means, even if their higher-order structure differs. Appendix Section A.1 provides details.

By contrast, we consider kernels for which the MMD defines a metric on probability distributions that is equal to zero only for identical distributions. For continuous Euclidean features, we use the energy-distance kernel,

$$k(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left(\|\boldsymbol{u}\|_2 + \|\boldsymbol{v}\|_2 - \|\boldsymbol{u} - \boldsymbol{v}\|_2\right). \tag{2}$$

In our small-molecule experiments with bit-vector fingerprints as features (Section 3.1), we use the Tanimoto / Jaccard kernel [Bouchard et al., 2013],

$$k(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^\top \boldsymbol{v}}{\|\boldsymbol{u}\|_2^2 + \|\boldsymbol{v}\|_2^2 - \boldsymbol{u}^\top \boldsymbol{v}}. \tag{3}$$

Both kernels avoid the bandwidth hyperparameter required by RBF kernels. For continuous molecular features, the energy-distance kernel produces results similar to a multiscale RBF mixture (Appendix Section D.1.2). See Duvenaud [2014, Chapter 2] for an overview of common kernel choices that might be preferable in other settings.

Training with the MMD objective. An attractive feature of MMD for finetuning feature distributions is that the expectations in the population estimand in Equation (1) may be approximated without bias by sample averages. The squared MMD has the unbiased estimator

$$\widehat{\mathrm{MMD}}_k^2[\boldsymbol{y}_{1:M}, \boldsymbol{y}^{\mathrm{targ}}_{1:N}] = \underbrace{\frac{1}{M(M-1)} \sum_{m' \neq m} k(\boldsymbol{y}_m, \boldsymbol{y}_{m'})}_{\text{Discourage mode collapse}} - \underbrace{\frac{2}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} k(\boldsymbol{y}_m, \boldsymbol{y}^{\mathrm{targ}}_n)}_{\text{Encourage similarity to target features}} + C_{\mathrm{targ}}, \tag{4}$$

where $C_{\mathrm{targ}} = \frac{1}{N(N-1)} \sum_{n' \neq n} k(\boldsymbol{y}^{\mathrm{targ}}_n, \boldsymbol{y}^{\mathrm{targ}}_{n'})$ depends only on target features, and so may be ignored in the optimization.

Score gradients of the MMD loss with leave-one-out baseline to reduce variance. Relating the sample-based discrepancy estimate in Equation (4) to practical gradient estimates requires care. Reparameterization gradients can provide low-variance gradient estimates when both $\boldsymbol{h}$ and the sampling procedure are differentiable. However, in the present setting where we assume only that we can tractably sample and compute $\log p_\theta(\boldsymbol{x})$, score-function gradients are a natural choice.

Algorithm 1 and Section A present and derive the score-gradient updates. A key component is the inclusion of a novel *leave-one-out baseline*, a Monte Carlo control variate used to reduce variance in score-function gradient estimates [Sutton and Barto, 2018, Ch. 13]. Standard leave-one-out baselines use the average coefficient from the other samples in the batch [Kool et al., 2019]. However, this approach is not directly applicable to the MMD loss because the coefficients are coupled across samples, so directly averaging them would introduce bias. Intuitively, the baseline should approximate the average coefficient that would have been obtained had $\boldsymbol{x}_m$ not appeared in the minibatch. Algorithm 1 uses a baseline constructed along this intuition, denoted `MMD-LOO`, and Section A.4 provides the mathematical expression and derivation. Empirically, we find including the LOO baseline can more than double the gradient signal-to-noise ratio at a given batch size (see Appendix Section D.1.3).

Algorithm 1: `kCGM` finetuning

Input: Pretrained model $p_{\theta_{\mathrm{pre}}}$, feature map $\boldsymbol{h}(\cdot)$, target features $\boldsymbol{y}^{\mathrm{targ}}_{1:N}$, kernel $k(\cdot, \cdot)$, batch size M, KL weight λ.

▷ Initialize and optimize

$p_\theta \leftarrow p_{\theta_{\mathrm{pre}}}$

while not converged **do**

  ▷ Sample and compute KL coefficients

  $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M \overset{\text{i.i.d.}}{\sim} p_{\texttt{stop-grad}(\theta)}$

  $l_m \leftarrow \big(\log p_{\texttt{stop-grad}(\theta)}(\boldsymbol{x}_m) - \log p_{\theta_{\mathrm{pre}}}(\boldsymbol{x}_m)\big)$

  $c^{\mathrm{KL}}_m \leftarrow \frac{1}{M}\texttt{KL-LOO}_m(l_{1:M})$

  ▷ Kernel discrepancy coefficients

  $\boldsymbol{y}_m \leftarrow \boldsymbol{h}(\boldsymbol{x}_m)$

  $s_m \leftarrow \frac{1}{M-1}\sum_{m' \neq m} k(\boldsymbol{y}_m, \boldsymbol{y}_{m'})$

  $r_m \leftarrow \frac{1}{N}\sum_{n=1}^{N} k(\boldsymbol{y}_m, \boldsymbol{y}^{\mathrm{targ}}_n)$

  $c^{\mathrm{k}}_m \leftarrow \frac{2}{M}\big(s_m - r_m\big)$

  $c^{\mathrm{k,LOO}}_m \leftarrow \texttt{MMD-LOO}_m(c^{\mathrm{k}}_{1:M})$

  ▷ Total coefficients and update

  $c_m \leftarrow \lambda c^{\mathrm{KL}}_m + c^{\mathrm{k,LOO}}_m$

  $\widehat{\mathcal{L}}^{\texttt{kCGM-surrogate}} \leftarrow \sum_{m=1}^{M} c_m \log p_\theta(\boldsymbol{x}_m)$

  $\theta \leftarrow \text{gradient-step}(\theta, \nabla_\theta \widehat{\mathcal{L}}^{\texttt{kCGM-surrogate}})$

end while

KL regularization and the complete algorithm. In addition to the squared MMD loss, we regularize KL divergence to the pretrained model $D_{\mathrm{KL}}(p_\theta || p_{\theta_{\mathrm{pre}}})$ with weight hyperparameter λ to ensure the finetuned model does not deviate too far from the pretrained model. The combined objective is

$$\mathcal{L}_\theta[k, \boldsymbol{y}_{1:N}^{\mathrm{targ}}] = \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{i.i.d.}{\sim} p_\theta} \left[\widehat{\mathrm{MMD}}_k^2[\boldsymbol{y}_{1:M}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}] \right] + \lambda \, \mathrm{D}_{\mathrm{KL}}(p_\theta \,\|\, p_{\theta_{\mathrm{pre}}}). \tag{5}$$

KL regularization is a standard approach in reward finetuning [Fan et al., 2023, Uehara et al., 2024] and is used by `CGM`. As the KL estimator decomposes into independent per-sample terms, we apply the standard leave-one-out centering baseline to the KL coefficients, denoted `KL-LOO` in Algorithm 1. We find varying λ enables exploring the tradeoff between matching the target feature distribution and preserving the pretrained model's distribution.

The final algorithm is shown in Algorithm 1. It consists of an optimization loop, wherein **(1)** samples are taken from the current model weights without tracking gradients, **(2)** the MMD and KL objectives are decomposed into per-sample coefficients with leave-one-out baselines, and **(3)** the coefficients are used to update the model weights with score-function gradients. While the MMD computation is $O(M^2 + MN)$, it is negligible compared to model gradient calculations.

3 Experiments

We evaluate how well `kCGM` matches the target feature distribution in three applications with diverse feature maps. Across these cases, `kCGM` substantially increases agreement with the distribution of target features and outperforms `CGM` and direct finetuning baselines. Before describing each experiment, we outline a shared protocol with additional details in Appendix Section C.

Evaluation details shared across experiments. Because both `kCGM` and `CGM` include a KL regularizer to the pretrained model weighted by λ, no single λ value determines a method's performance. Instead, each method gives a tradeoff curve between distance from the pretrained model (measured as KL-to-pretrained) and distribution-matching error. In practice, λ must be chosen by considering this tradeoff.

The distribution-matching metric differs between experiments and is specified in each subsection. Error bars are reported as the standard error of the mean from three replicates with different random seeds.

Baselines. Our primary baseline is `CGM`, shorthand for the `CGM`-relax variant of Smith et al. [2026], the closest prior method in our setting. In the protein experiment (Section 3.2), the 2D feature space makes the CDF-binning variant of `CGM` tractable, and we use it as described by Smith et al. [2026]. In the small-molecule and DNA experiments (Sections 3.1 and 3.3), binning is impractical in the higher-dimensional feature spaces, so we compare against using `CGM` to target the mean of the feature distribution. The small-molecule experiment additionally includes direct finetuning on the target set as a baseline.

Compute. All experiments run on a single GPU (H100 or L40S) and range from one to 30 hours in duration per finetuning run. Further details are provided in Appendix C.

3.1 Matching antibiotic feature distributions

We first apply `kCGM` to calibrate a generative model of small molecules to match properties of a set of 174 small-molecule antibiotics. `kCGM` delivers stronger agreement with chosen features than `CGM` while maintaining sample quality. By contrast, direct finetuning degrades model quality and leads to a high proportion of invalid molecules.

Background. Small-molecule generative models are increasingly important in early-stage drug discovery [Du et al., 2024], where they can help explore the vast space of synthesizable, drug-like molecules. While the space of drug-like molecules may contain 10^{60} molecules [Polishchuk et al., 2013], molecule sets with a desired function are often small.

The number of known antibiotics, and especially the number of unique antibiotic scaffolds, is limited [Farha et al., 2025]. This makes antibiotics a natural target for `kCGM`. Rather than finetuning a generator to reproduce the target set molecule-by-molecule, we ask whether it can be adapted to match selected *aspects* of the antibiotic distribution.

G2PT model. We use G2PT, an autoregressive graph generative model in which molecular graphs are flattened into sequences of node and edge tokens [Chen et al., 2025]. Specifically, we use the 11-million-parameter G2PT-small model pretrained on GuacaMol [Brown et al., 2019], a benchmark of 1.3M drug-like molecules filtered from ChEMBL. Thus, pretraining provides a broad prior over drug-like chemistry, while the antibiotic set represents a smaller and specialized target distribution.

Antibiotic features and kernels. We target a set of 174 small-molecule antibiotics curated by Scalia et al. [2025]. An advantage of `kCGM` is that the user can choose which aspects of the target distribution to match. We consider three complementary feature spaces, capturing local chemical substructures, broader scaffold-level organization, and coarse physicochemical properties.

For local substructure, we use Morgan fingerprints [Rogers and Hahn, 2010], sparse binary vectors indicating the presence of hashed local subgraphs, paired with the Tanimoto kernel.

For scaffold-level organization, we use generic Murcko scaffolds, which ignore atom type and bond order to capture only the topology of the molecular scaffold [Bemis and Murcko, 1996]. This captures a different notion of similarity from Morgan fingerprints, as two molecules can share scaffolds while differing in local substructure, or share local substructures while differing in scaffold organization. We embed generic scaffolds using the representation-learning strategy of Fréchet ChemNet Distance, taking the penultimate-layer activations of a molecular property predictor [Preuer et al., 2018]. Appendix Figure 8 shows that scaffold embedding space separates molecules with different scaffolds, whereas Morgan fingerprints do so very weakly.

For coarse properties, we use a vector of eleven standard molecular descriptors (full list in Appendix Section C.2) summarizing physicochemical, structural, and drug-likeness properties. We use the energy-distance kernel for both scaffold embeddings and descriptors.

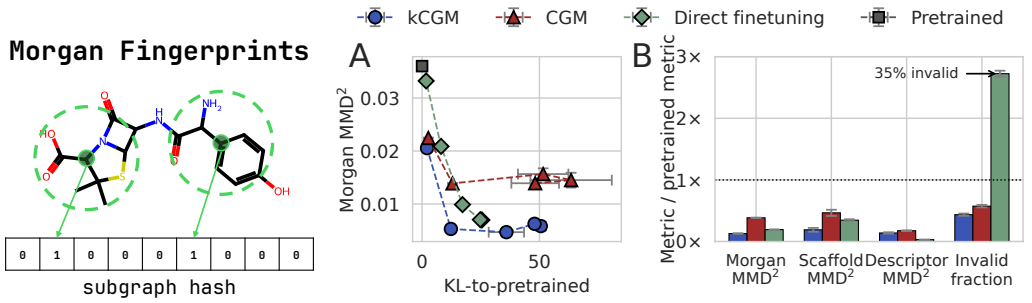

Figure 2: Comparison of direct finetuning, `kCGM`, and `CGM` for adapting G2PT to the antibiotics set. Panel A evaluates `kCGM` and `CGM` models trained to match the target Morgan fingerprint distribution compared to direct finetuning according to MMD^2 over five values of the regularization strength, λ. Panel B selects the best λ for each method according to Morgan MMD^2. It then evaluates the selected models using scaffold MMD^2, descriptor MMD^2, and the fraction of invalid generated molecules.

Distribution matching results. We first focus on Morgan fingerprints, a common representation of local chemical substructure. This setting is a natural test for `kCGM` because Morgan fingerprints are discrete bit vectors, making the feature map non-differentiable, and because the Tanimoto kernel provides a chemically meaningful similarity measure on the resulting feature space.

As baselines, we compare against `CGM`, which matches only the target Morgan fingerprint mean, and direct finetuning on the 174 antibiotics. Direct finetuning uses the standard autoregressive log-likelihood objective on full target molecules in the original sample space, with the same KL-to-pretrained regularization used for `kCGM` and `CGM`; details are given in Appendix Section C.2. For each method, we sweep five values of λ and evaluate the tradeoff between Morgan fingerprint distribution matching, measured only on chemically valid generations, and distance from the pretrained model.

kCGM achieves a better Morgan MMD^2–KL tradeoff than both CGM and direct finetuning (Figure 2A). This shows that matching only the average fingerprint, as in CGM, is insufficient for matching the antibiotic fingerprint distribution. Direct finetuning can also reduce Morgan MMD^2, but does so by moving farther from the pretrained model and, as discussed below, at a cost to sample validity.

We next ask whether Morgan fingerprint finetuning improves other molecular feature distributions. For each method, we select the best λ according to Morgan MMD^2 and evaluate the resulting models on scaffold, descriptor, and validity metrics (Figure 2B). Although kCGM is trained only on Morgan fingerprints, it also improves scaffold and descriptor MMD relative to the pretrained model, suggesting that local substructure matching transfers partially to broader molecular properties. In contrast, direct finetuning produces a high fraction of chemically invalid graphs. This occurs because direct finetuning imitates the small antibiotic set molecule-by-molecule and does not explicitly discourage invalid generations, whereas invalid molecular graphs are mapped to a default feature value and are therefore penalized by the CGM and kCGM objectives. In addition, all kCGM runs maintain greater than 98% uniqueness among generated molecules, indicating that the improved feature matching does not arise from memorization.

Finally, we repeat the same comparison when kCGM and CGM are trained directly on scaffold embeddings or molecular descriptors. These additional feature-targeted experiments are deferred to Appendix Figure 11 and show that kCGM remains effective across multiple user-chosen notions of molecular similarity. A KL-matched summary of best interpolated metric values and cross-feature transfer is provided in Appendix Table 1.

Ablation Results. We next ablate two components of kCGM on the Morgan fingerprint experiment. First, we replace the Tanimoto kernel with the dot-product kernel, which reduces kCGM to mean matching rather than full distribution matching. Second, we remove the leave-one-out baseline used for variance reduction. Both changes degrade performance (Figure 3). Using the dot-product kernel substantially increases MMD, confirming that distribution rather than mean matching is important. Removing the leave-one-out baseline also substantially worsens kCGM, showing that the baseline is important for effective optimization. A direct analysis of gradient signal-to-noise ratio further supports the effectiveness of the baseline (Appendix Section D.1.3). kCGM with the dot-product kernel can be viewed as CGM with our leave-one-out baseline, and substantially outperforms CGM in this experiment. This suggests the leave-one-out baseline is a drop-in improvement for CGM.

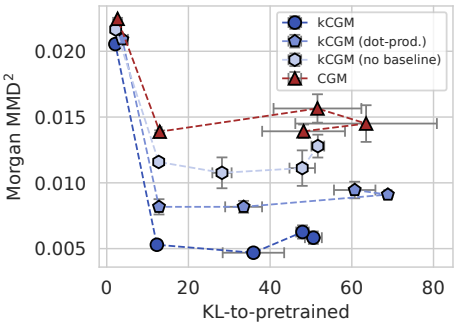

Figure 3: Ablations of kCGM with Morgan fingerprint features. We compare standard kCGM, kCGM with the dot-product kernel, kCGM without the leave-one-out baseline, and CGM.

3.2 Increasing protein structure diversity

We adopt an experiment from Smith et al. [2026] in which Genie 2 [Lin et al., 2024], a protein structure diffusion model, is finetuned so that its secondary structure distribution matches a diverse target distribution of secondary structural proportions taken from protein domains in the CATH dataset [Sillitoe et al., 2021]. While Smith et al. [2026] demonstrated improved agreement with CATH, substantial disagreement remained. With kCGM, the disagreement is roughly $2\times$ smaller than that of CGM. In these experiments, we additionally explore a *self-repulsion weight* hyperparameter that we find may be tuned to improve performance.

Background and setup. Due to a sampling heuristic necessary for high-quality generations, Genie 2 produces samples that consist almost entirely of alpha-helices. In contrast, CATH protein domains exhibit high structural diversity and thus present a useful target distribution.

Following Smith et al. [2026], we generate protein domains with 100 residues and define the feature function as the proportion of the residues in alpha-helices and beta-strands:

$$\boldsymbol{h}(\boldsymbol{x}) = (\#\ \alpha\text{-helix residues}/100, \#\ \beta\text{-strand residues}/100).$$

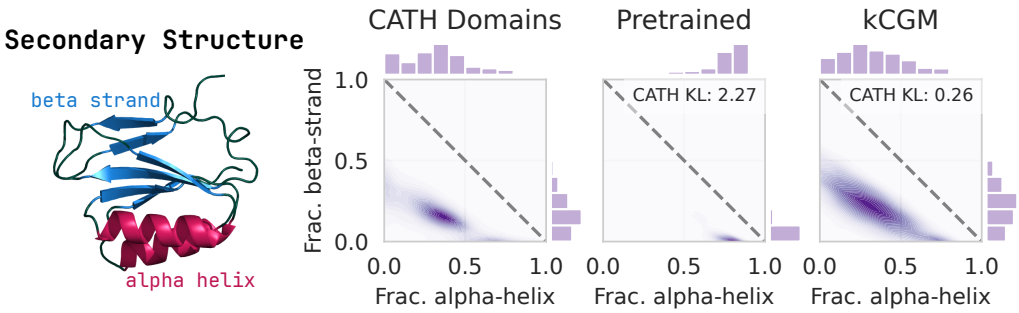

Figure 4: Joint KDEs and marginal histograms of secondary-structure features for the target CATH domains, the pretrained Genie 2 model, and the best kCGM model. The pretrained model is concentrated near all-α structures, whereas kCGM shifts the distribution toward the diverse target distribution.

kCGM matches this 2D feature distribution directly, using the energy-distance kernel (Equation (2)) which has no additional hyperparameters. CGM, by contrast, cannot use this 2D feature directly. Its mean-matching objective would only constrain the average helix and strand proportions, not the shape of the joint distribution. To recover distributional information, Smith et al. [2026] discretize the CATH distribution into 99 bivariate CDF bins and constrain CGM to match the proportion of samples falling in each bin. In effect, kCGM matches the full 2D distribution through sample-based kernel comparisons, whereas CGM matches 99 scalar bin-proportion means. Because the feature space is only 2D, the CDF-binning strategy is tractable, making this a favorable setting for CGM. We reuse the CGM hyperparameters (batch size, learning rate, etc.) without hyperoptimizing them for kCGM.

Self-repulsion hyperparameter. We additionally explore a *self-repulsion* weight hyperparameter α, which we include as a multiplicative factor on the first term in the MMD estimate in Equation (4), introducing a mode-seeking bias to the objective. This results in a change to the MMD coefficients in Algorithm 1, $c_m^{\mathrm{k}} \leftarrow \frac{2}{M}\left(\alpha s_m - r_m\right)$. We search three α values $(0.25, 0.5, 1.0)$, though this is not necessary to outperform the baseline. Section A.5 provides greater detail, as well as examples to provide intuition for the impact of α.

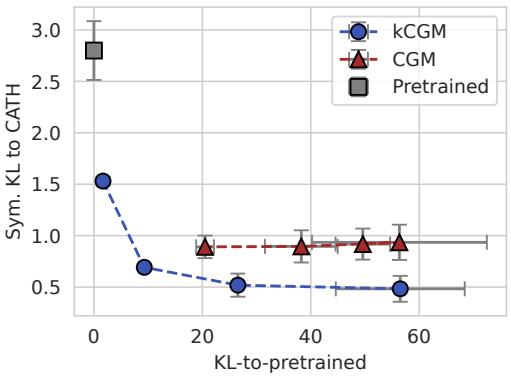

Figure 5: Distribution-matching error versus KL-to-pretrained for the protein secondary-structure experiment for four values of the KL-regularization weight λ. kCGM with $\alpha = 0.5$ achieves lower symmetrized KL to the target CATH secondary-structure distribution than CGM across the tested tradeoff range.

Results. Figure 5 shows the tradeoff curve between KL-to-pretrained and matching error, measured as the symmetric KL between Gaussian KDEs of the 2D features (Figure 4). kCGM achieves lower matching error than CGM at every KL-to-pretrained budget we tested. Moreover, kCGM's matching error varies smoothly with λ, whereas CGM's matching error is relatively insensitive to λ, making it difficult for a practitioner to optimize the tradeoff.

While sweeping over α further improves distribution matching (Appendix Figure 10), every α value we test outperforms the baseline and maintains a controlled regularization curve. This shows that kCGM is robust to both the regularization weight and the self-repulsion weight and reliably improves distribution matching compared to the baseline.

3.3 Generating DNA with realistic cell-type-specific activity profiles

Regulatory DNA sequence generation provides a challenging test of feature-distribution matching. A useful enhancer generator should not only produce sequences that resemble the training set, but should also reproduce the distribution of regulatory activity associated with each biological condition

[Lal et al., 2025a]. This is difficult to enforce with standard likelihood training and even large DNA generative models often fail to learn regulatory syntax [Patel et al., 2024].

Background and setup. We study this setting using the human melanoma enhancer DeepMEL2 dataset [Atak et al., 2021]. The dataset consists of 500-base-pair regulatory sequences annotated with 47 binary cis-regulatory topics derived from melanoma ATAC-seq data. These topics summarize melanoma cell states and provide multi-label conditioning variables. We train a convolutional masked discrete diffusion model [Sahoo et al., 2024] to generate 500-base-pair DNA sequences conditional on these topic vectors. For finetuning, we construct a separate target feature distribution for each topic using real training sequences positive for that topic. Thus, when the model is sampled conditionally on a given topic, `kCGM` compares the generated AlphaGenome feature distribution to the target AlphaGenome feature distribution for that same topic. Appendix Section C.3 describes the model architecture, target construction, and training procedure. Whereas sequences with high levels of predicted activity are rare in pretrained model samples, after `kCGM` finetuning DNA samples exhibit similar activity profiles to real enhancers with the corresponding topic annotations.

Regulatory activity features and kernel. To measure regulatory activity, we use AlphaGenome [Avsec et al., 2026] as an independent sequence-to-function predictor. For each generated or target sequence, we summarize AlphaGenome-predicted ATAC-seq profiles by their peak heights, yielding a vector of predicted cell-type-specific activity features as shown in Figure 6. These feature distributions are matched separately for each cis-regulatory topic. We apply the preprocessing described in Appendix Section C.3 and use the energy-distance kernel for both training and evaluation.

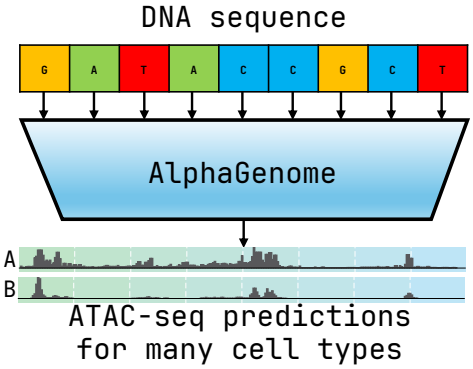

Figure 6: Diagram of AlphaGenome activity features for `kCGM` finetuning. The maximum height of each ATAC-seq track is used to create a vector of maximum accessibility for each AlphaGenome cell type.

Results. We sweep four regularization strengths λ for `kCGM` and `CGM`, using only training-set AlphaGenome features during finetuning to avoid leakage. Each finetuning update samples a topic condition and compares generated sequences for that condition to the corresponding topic-specific target feature distribution. In this experiment, `CGM` targets only the mean activity profile for each condition, whereas `kCGM` matches the full activity-feature distribution. `kCGM` achieves a better KL–MMD tradeoff than `CGM`, as measured against held-out target feature distributions for the same topics (Figure 7A). At the per-condition level, `kCGM` improves activity-profile matching for most conditioning topics (Figure 7B).

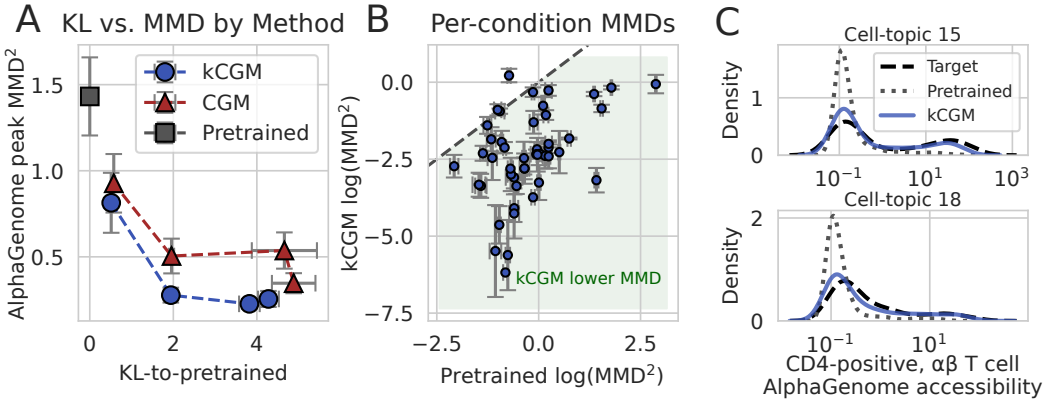

Figure 7: `kCGM` for conditional regulatory DNA generation. Panel A shows the tradeoff curve between KL to the pretrained model vs. matching the target AlphaGenome-predicted cell-type activity profiles for four values of the KL-regularization weight λ. Panel B shows MMDs for `kCGM` with $\lambda = 10^{-2}$ vs. the pretrained model for each of the 47 cis-regulatory topics. Panel C shows example AlphaGenome peak distributions for CD4-positive $\alpha\beta$ T cells before and after `kCGM`.

Representative AlphaGenome output tracks show that the pretrained model generates under-active sequences, while kCGM shifts the predicted activity distributions toward the topic-specific target (Figure 7C). This shows that kCGM can calibrate a conditional discrete diffusion model using a black-box sequence-to-function predictor.

As an additional validation, we evaluate the finetuned models using an independent accessibility predictor (CATlas) and sequence-level 3-mer statistics (Appendix Section D.3). Both CGM and kCGM improve agreement under CATlas, with kCGM achieving the lowest CATlas MMD^2 across evaluated runs.

4 Discussion and conclusion

kCGM effectively aligns diffusion and autoregressive models to sample-defined feature distributions. kCGM works without requiring differentiable features or target samples in the original sample space and maintains or even improves the sample quality of the pretrained model. Across experiments, the same objective supports hand-designed, learned-embedding, and predictor-derived features, suggesting that the feature map can be chosen to match the scientific question. Our results indicate an alternative to direct finetuning when the target dataset is small and only certain aspects of the target distribution are important to preserve.

We provide some practical guidance for picking kCGM hyperparameters in Appendix Section B. Our empirical recommendations should be interpreted alongside some technical limitations. For instance, MMD may be ineffective for very high-dimensional features. This can be seen in a hypothesis-testing context, where the two-sample statistic we rely on for our objective (Equation (4)) has power that drops as a polynomial function of the feature dimension [Ramdas et al., 2015]. In practice, however, kernel and feature choice, and pretrained-model support may matter more than ambient dimension. In one G2PT antibiotic experiment, we find that kCGM performance is relatively insensitive to using 32–128 retained principal components or the original 512-dimensional scaffold embeddings (Appendix Section D.1.1). Additionally, kCGM does not immediately apply to generative models without tractable sampling-trajectory log-probabilities, such as flow matching models. Both limitations may be directions for future work.

In our experiments, the target features are constructed by applying user-chosen feature functions to full target samples. A natural next step is to use the same framework in settings where experiments provide distributional constraints without full samples in the original generative space. Such constraints arise in many scientific measurements, including cryo-EM for protein structure [Shoemaker and Ando, 2018], correlated chemical probing for RNA structure [Mustoe et al., 2023], and proteomics-informed calibration of immunofluorescence image generators. This would extend feature-distribution matching into a general interface between generative models and partial experimental measurements.

Acknowledgments and disclosure of funding

We thank Henry Smith for helpful discussions, help setting up Genie 2 experiments, and providing manuscript feedback; Anshul Kundaje for inspiration for the enhancer DNA generation experiment; Arthur Deng for manuscript feedback; and Shiye Su for discussions about applications of kCGM.

NLD is supported by the Stanford Graduate Fellowship (SGF).

References

Zeynep Kalender Atak, Ibrahim Ihsan Taskiran, Jonas Demeulemeester, Christopher Flerin, David Mauduit, Liesbeth Minnoye, Gert Hulselmans, Valerie Christiaens, Ghanem-Elias Ghanem, Jasper Wouters, et al. Interpretation of allele-specific chromatin accessibility using cell state–aware deep learning. *Genome Research*, 2021.

Žiga Avsec, Natasha Latysheva, Jun Cheng, Guido Novati, Kyle R Taylor, Tom Ward, Clare Bycroft, Lauren Nicolaisen, Eirini Arvaniti, Joshua Pan, et al. Advancing regulatory variant effect prediction with AlphaGenome. *Nature*, 2026.

Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. Molecular frameworks. *Journal of Medicinal Chemistry*, 1996.

Diane Bouchacourt, Pawan K Mudigonda, and Sebastian Nowozin. DISCO Nets : DISsimilarity COefficients Networks. In *Advances in Neural Information Processing Systems*, 2016.

Mathieu Bouchard, Anne-Laure Jousselme, and Pierre-Emmanuel Doré. A proof for the positive definiteness of the Jaccard index matrix. *International Journal of Approximate Reasoning*, 2013.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. GuacaMol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 2019.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. arXiv:2107.03374.

Xiaohui Chen, Yinkai Wang, Jiaxing He, Yuanqi Du, Soha Hassoun, Xiaolin Xu, and Liping Liu. Graph generative pre-trained transformer. In *International Conference on Machine Learning*, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

Valentin De Bortoli, Alexandre Galashov, J Swaroop Guntupalli, Guangyao Zhou, Kevin Murphy, Arthur Gretton, and Arnaud Doucet. Distributional diffusion models with scoring rules, 2025. arXiv:2502.02483.

Riccardo De Santi, Marin Vlastelica, Ya-Ping Hsieh, Zebang Shen, Niao He, and Andreas Krause. Flow density control: Generative optimization beyond entropy-regularized fine-tuning. *Advances in Neural Information Processing Systems*, 2025.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *International Conference on Learning Representations*, 2025.

Yuanqi Du, Arian R Jamasb, Jeff Guo, Tianfan Fu, Charles Harris, Yingheng Wang, Chenru Duan, Pietro Liò, Philippe Schwaller, and Tom L Blundell. Machine learning-aided generative molecular design. *Nature Machine Intelligence*, 2024.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the $\ell 1$–ball for learning in high dimensions. In *International Conference on Machine Learning*, 2008.

David Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Maya A Farha, Megan M Tu, and Eric D Brown. Important challenges to finding new leads for new antibiotics. *Current Opinion in Microbiology*, 2025.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in Neural Information Processing Systems*, 2021.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free! In *Deep Reinforcement Learning Meets Structured Prediction Workshop, International Conference on Learning Representations*, 2019. Workshop paper.

Avantika Lal, Laura Gunsalus, Anay Gupta, Tommaso Biancalani, and Gokcen Eraslan. Polygraph: A software framework for the systematic assessment of synthetic regulatory DNA elements. *Genome Biology*, 2025a.

Avantika Lal, Laura Gunsalus, Surag Nair, Tommaso Biancalani, and Gokcen Eraslan. gReLU: a comprehensive framework for DNA sequence modeling and design. *Nature Methods*, 2025b.

Greg Landrum, Paolo Tosco, Brian Kelley, Ricardo Rodriguez, David Cosgrove, Riccardo Vianello, et al. RDKit: 2024_03_1 (q1 2024) release, 2024.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD-GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, 2017.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, 2015.

Yeqing Lin, Minji Lee, Zhao Zhang, and Mohammed AlQuraishi. Out of many, one: Designing and scaffolding proteins at the scale of the structural universe with Genie 2, 2024. arXiv:2405.15489.

Tianyu Lu, Melissa Liu, Yilin Chen, Jinho Kim, and Po-Ssu Huang. Assessing generative model coverage of protein structures with SHAPES. *Cell Systems*, 2025.

Alessandro Manenti, Daniele Zambon, and Cesare Alippi. Learning latent graph structures and their uncertainty. In *International Conference on Machine Learning*, 2025.

Muhammad A Masood and Finale Doshi-Velez. Diversity-inducing policy gradient: Using maximum mean discrepancy to find a set of diverse policies, 2019. arXiv:1906.00088.

Zichen Miao, Jiang Wang, Ze Wang, Zhengyuan Yang, Lijuan Wang, Qiang Qiu, and Zicheng Liu. Training diffusion models towards diverse image generation with reinforcement learning. In *Conference on Computer Vision and Pattern Recognition*, 2024.

Anthony M Mustoe, Chase A Weidmann, and Kevin M Weeks. Single-molecule correlated chemical probing: A revolution in RNA structure analysis. *Accounts of Chemical Research*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022.

Anusri Pampari, Anna Shcherbina, Evgeny Z Kvon, Michael Kosicki, Surag Nair, Soumya Kundu, Arwa S Kathiria, Viviana I Risca, Kristiina Kuningas, Kaur Alasoo, et al. ChromBP-Net: Bias factorized, base-resolution deep learning models of chromatin accessibility reveal cis-regulatory sequence syntax, transcription factor footprints and regulatory variants, 2025. BioRxiv:10.1101/2024.12.25.630221v2.

Aman Patel, Arpita Singhal, Austin Wang, Anusri Pampari, Maya Kasowski, and Anshul Kundaje. DART-eval: A comprehensive DNA language model evaluation benchmark on regulatory DNA. *Advances in Neural Information Processing Systems*, 2024.

Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of Computer-Aided Molecular Design*, 2013.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Fréchet ChemNet distance: A metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 2018.

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. In *Advances in Neural Information Processing Systems*, 2023.

Aaditya Ramdas, Sashank Jakkam Reddi, Barnabás Póczos, Aarti Singh, and Larry Wasserman. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. In *AAAI Conference on Artificial Intelligence*, 2015.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 2010.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 2024.

Anirban Sarkar, Alejandra Duran, Yiyang Yu, Da-Wei Lin, Yijie Kang, Nirali Somia, Pablo Mantilla, Jessica Zhou, Masayuki Nagai, Ziqi Tang, et al. Designing DNA with tunable regulatory activity using discrete diffusion, 2024. bioRxiv:10.1101/2024.05.23.595630v3.

Gabriele Scalia, Steven T Rutherford, Ziqing Lu, Kerry R Buchholz, Nicholas Skelton, Kangway Chuang, Nathaniel Diamant, Jan-Christian Hütter, Jerome-Maxim Luescher, Anh Miu, et al. Deep-learning-based virtual screening of antibacterial compounds. *Nature Biotechnology*, 2025.

Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 2018.

Susannah C Shoemaker and Nozomi Ando. X-rays in the cryo-electron microscopy era: Structural biology's dynamic future. *Biochemistry*, 2018.

Ian Sillitoe, Nicola Bordin, Natalie Dawson, Vaishali P. Waman, Paul Ashford, Harry M. Scholes, Camilla S. M. Pang, Laurel Woodridge, Clemens Rauer, Neeladri Sen, Mahnaz Abbasian, Sean Le Cornu, Su Datt Lam, Karel Berka, Ivana Hutařová Vareková, Radka Svobodova, Jon Lees, and Christine A. Orengo. CATH: Increased structural coverage of functional space. *Nucleic Acids Research*, 2021.

Henry D Smith, Nathaniel L Diamant, and Brian L Trippe. Calibrating generative models to distributional constraints. In *International Conference on Machine Learning*, 2026.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2nd edition, 2018.

Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel L Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control, 2024. arXiv:2402.15194.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, 2014.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process, 2025. arXiv:2509.16117.

Appendix

Appendix contents

A Derivations **15**

A.1 `CGM`-relax as a special case of `kCGM` . . . 15

A.2 Kernel discrepancy coefficients for score-function gradients . . . 16

A.3 Unbiased surrogate gradients . . . 17

A.4 Leave-one-out baselines . . . 18

A.5 Behavior of self-repulsion weight less than one for categorical target features . . . 19

B Practical usage recommendations **20**

C Additional experimental details **20**

C.1 Protein secondary structure distribution matching . . . 20

C.2 Antibiotics feature distribution matching . . . 21

C.3 Generating DNA with realistic cell-type-specific activity profiles . . . 23

C.4 Overview figure: a smile made of SMILES . . . 25

C.5 One-dimensional self-repulsion demonstration . . . 26

D Additional results **27**

D.1 G2PT . . . 27

D.1.1 Sensitivity to number of principal components used for target features . . . 28

D.1.2 Effect of kernel choice on performance metrics . . . 29

D.1.3 Effect of leave-one-out baseline and kernel choice on gradient signal-to-noise ratio . . . 29

D.2 Genie 2 . . . 30

D.3 DNA discrete diffusion . . . 30

E Example generations **31**

E.1 Genie 2 . . . 31

E.2 G2PT . . . 31

E.3 Asset licenses . . . 35

A Derivations

A.1 CGM-relax as a special case of kCGM

We show the connection between kCGM and CGM-relax using the dot-product kernel. With this kernel, kCGM does not compare full feature distributions and reduces exactly to matching mean feature vectors. Thus, CGM-relax can be viewed as the dot-product kernel special case of kCGM, while other kernels extend the objective to capture richer aspects of the target feature distribution.

We first state the population-level relationship.

Proposition A.1. *Let Q be a target distribution on feature space, and let $\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})$ for $\boldsymbol{x} \sim p_\theta$, with distribution $P_{\boldsymbol{y}}$. If $k(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v}$ is the dot-product kernel, then*

$$\mathrm{MMD}_k^2(P_{\boldsymbol{y}}, Q) = \left\| \mathbb{E}[\boldsymbol{y}] - \mathbb{E}_{\boldsymbol{y}^{\mathrm{targ}} \sim Q}[\boldsymbol{y}^{\mathrm{targ}}] \right\|_2^2 .$$

Equivalently, kCGM with the dot-product kernel matches only the mean feature vector of the target distribution.

Proof. By the population definition of squared MMD,

$$\mathrm{MMD}_k^2(P_{\boldsymbol{y}}, Q) = \mathbb{E}[\boldsymbol{y}^\top \boldsymbol{y}'] - 2\mathbb{E}[\boldsymbol{y}^\top \boldsymbol{y}^{\mathrm{targ}}] + \mathbb{E}[(\boldsymbol{y}^{\mathrm{targ}})^\top (\boldsymbol{y}^{\mathrm{targ}})'],$$

where $\boldsymbol{y}, \boldsymbol{y}' \overset{\text{i.i.d.}}{\sim} P_{\boldsymbol{y}}$ and $\boldsymbol{y}^{\mathrm{targ}}, (\boldsymbol{y}^{\mathrm{targ}})' \overset{\text{i.i.d.}}{\sim} Q$ are independent. Using independence,

$$\mathbb{E}[\boldsymbol{y}^\top \boldsymbol{y}'] = \mathbb{E}[\boldsymbol{y}]^\top \mathbb{E}[\boldsymbol{y}], \quad \mathbb{E}[\boldsymbol{y}^\top \boldsymbol{y}^{\mathrm{targ}}] = \mathbb{E}[\boldsymbol{y}]^\top \mathbb{E}[\boldsymbol{y}^{\mathrm{targ}}], \quad \mathbb{E}[(\boldsymbol{y}^{\mathrm{targ}})^\top (\boldsymbol{y}^{\mathrm{targ}})'] = \mathbb{E}[\boldsymbol{y}^{\mathrm{targ}}]^\top \mathbb{E}[\boldsymbol{y}^{\mathrm{targ}}].$$

Substituting gives

$$\mathrm{MMD}_k^2(P_{\boldsymbol{y}}, Q) = \mathbb{E}[\boldsymbol{y}]^\top \mathbb{E}[\boldsymbol{y}] - 2\mathbb{E}[\boldsymbol{y}]^\top \mathbb{E}[\boldsymbol{y}^{\mathrm{targ}}] + \mathbb{E}[\boldsymbol{y}^{\mathrm{targ}}]^\top \mathbb{E}[\boldsymbol{y}^{\mathrm{targ}}] = \left\| \mathbb{E}[\boldsymbol{y}] - \mathbb{E}[\boldsymbol{y}^{\mathrm{targ}}] \right\|_2^2 .$$

□

As an immediate corollary, if the target distribution Q is concentrated at a single value $\boldsymbol{h}^*$, then dot-product kCGM reduces to matching $\mathbb{E}[\boldsymbol{h}(\boldsymbol{x})]$ to $\boldsymbol{h}^*$, which is exactly the population objective used by CGM-relax:

$$\mathcal{L}_\theta[\boldsymbol{h}^*] = \|\mathbb{E}_{\boldsymbol{x} \sim p_\theta}[\boldsymbol{h}(\boldsymbol{x})] - \boldsymbol{h}^*\|_2^2 + \lambda \, \mathrm{D}_{\mathrm{KL}}(p_\theta || p_{\theta_{\mathrm{pre}}}).$$

At the stochastic-optimization level, the connection is even tighter. Let $\hat{\mathcal{L}}^{\mathrm{viol}}$ denote the finite-sample CGM-relax violation objective used to match the empirical target mean. When the target feature distribution is represented by empirical samples, dot-product kCGM depends on those samples only through their empirical mean, up to a target-only constant. As a result, dot-product kCGM recovers the same score-function optimization objective as CGM-relax.

Proposition A.2. *Let $\boldsymbol{y}_{1:N}^{\mathrm{targ}}$ be target feature samples with empirical mean*

$$\bar{\boldsymbol{y}}^{\mathrm{targ}} := \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{y}_n^{\mathrm{targ}}.$$

Let $\boldsymbol{y}_{1:M}$ be model feature samples, where $\boldsymbol{y}_m = \boldsymbol{h}(\boldsymbol{x}_m)$ for $\boldsymbol{x}_m \sim p_\theta$, and take the dot-product kernel

$$k(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v}.$$

Then the unbiased finite-sample estimator of squared MMD satisfies

$$\widehat{\mathrm{MMD}}_k^2[\boldsymbol{y}_{1:M}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}] = \frac{1}{M(M-1)} \sum_{m \neq m'} \boldsymbol{y}_m^\top \boldsymbol{y}_{m'} - \frac{2}{M} \sum_{m=1}^{M} \boldsymbol{y}_m^\top \bar{\boldsymbol{y}}^{\mathrm{targ}} + C_{\mathrm{targ}},$$

where

$$C_{\mathrm{targ}} := \frac{1}{N(N-1)} \sum_{n \neq n'} (\boldsymbol{y}_n^{\mathrm{targ}})^\top \boldsymbol{y}_{n'}^{\mathrm{targ}}$$

depends only on the target samples.

Equivalently,

$$\widehat{\mathrm{MMD}}_k^2[\boldsymbol{y}_{1:M}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}] = \hat{\mathcal{L}}^{\mathrm{viol}}(\boldsymbol{y}_{1:M}; \bar{\boldsymbol{y}}^{\mathrm{targ}}) + C'_{\mathrm{targ}},$$

where

$$\hat{\mathcal{L}}^{\mathrm{viol}}(\boldsymbol{y}_{1:M}; \bar{\boldsymbol{y}}^{\mathrm{targ}}) := \frac{1}{M(M-1)} \sum_{m \neq m'} (\boldsymbol{y}_m - \bar{\boldsymbol{y}}^{\mathrm{targ}})^\top (\boldsymbol{y}_{m'} - \bar{\boldsymbol{y}}^{\mathrm{targ}}),$$

and C'_{targ} is constant with respect to θ.

Proof. For the dot-product kernel,

$$\begin{aligned}\widehat{\mathrm{MMD}}_k^2[\boldsymbol{y}_{1:M}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}] =& \frac{1}{M(M-1)} \sum_{m \neq m'} \boldsymbol{y}_m^\top \boldsymbol{y}_{m'} - \frac{2}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \boldsymbol{y}_m^\top \boldsymbol{y}_n^{\mathrm{targ}} \\ &+ \frac{1}{N(N-1)} \sum_{n \neq n'} (\boldsymbol{y}_n^{\mathrm{targ}})^\top \boldsymbol{y}_{n'}^{\mathrm{targ}}.\end{aligned}$$

Rearranging the middle term gives

$$\frac{2}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \boldsymbol{y}_m^\top \boldsymbol{y}_n^{\mathrm{targ}} = \frac{2}{M} \sum_{m=1}^{M} \boldsymbol{y}_m^\top \bar{\boldsymbol{y}}^{\mathrm{targ}},$$

so

$$\widehat{\mathrm{MMD}}_k^2[\boldsymbol{y}_{1:M}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}] = \frac{1}{M(M-1)} \sum_{m \neq m'} \boldsymbol{y}_m^\top \boldsymbol{y}_{m'} - \frac{2}{M} \sum_{m=1}^{M} \boldsymbol{y}_m^\top \bar{\boldsymbol{y}}^{\mathrm{targ}} + C_{\mathrm{targ}}.$$

On the other hand,

$$\hat{\mathcal{L}}^{\mathrm{viol}}(\boldsymbol{y}_{1:M}; \bar{\boldsymbol{y}}^{\mathrm{targ}}) = \frac{1}{M(M-1)} \sum_{m \neq m'} (\boldsymbol{y}_m - \bar{\boldsymbol{y}}^{\mathrm{targ}})^\top (\boldsymbol{y}_{m'} - \bar{\boldsymbol{y}}^{\mathrm{targ}}),$$

which expands to

$$\frac{1}{M(M-1)} \sum_{m \neq m'} \boldsymbol{y}_m^\top \boldsymbol{y}_{m'} - \frac{2}{M} \sum_{m=1}^{M} \boldsymbol{y}_m^\top \bar{\boldsymbol{y}}^{\mathrm{targ}} + \|\bar{\boldsymbol{y}}^{\mathrm{targ}}\|_2^2.$$

Therefore

$$\widehat{\mathrm{MMD}}_k^2[\boldsymbol{y}_{1:M}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}] = \hat{\mathcal{L}}^{\mathrm{viol}}(\boldsymbol{y}_{1:M}; \bar{\boldsymbol{y}}^{\mathrm{targ}}) + \left(C_{\mathrm{targ}} - \|\bar{\boldsymbol{y}}^{\mathrm{targ}}\|_2^2\right),$$

and the difference depends only on the target samples. □

As a consequence, the expected dot-product `kCGM` objective differs from the `CGM`-relax mean-matching objective with target mean $\bar{\boldsymbol{y}}^{\mathrm{targ}}$ only by a constant independent of θ. Therefore the score-function gradient of dot-product `kCGM` is identical to that of `CGM`-relax with target mean $\bar{\boldsymbol{y}}^{\mathrm{targ}}$.

A.2 Kernel discrepancy coefficients for score-function gradients

As a reminder, $\boldsymbol{x} \sim p_\theta$ denotes a full sampling trajectory. The feature map $\boldsymbol{h}(\boldsymbol{x})$ is understood to act on the terminal generated sample of that trajectory. Thus $\log p_\theta(\boldsymbol{x})$ denotes the sampling-trajectory log-probability.

We now derive the per-sample kernel discrepancy coefficients used in Algorithm 1. Since `kCGM` is designed to handle non-differentiable feature functions $\boldsymbol{h}(\cdot)$ and non-differentiable sampling procedures, we derive these coefficients directly from the score-function gradient of the expected squared MMD objective

$$\mathcal{L}_\theta^{\mathrm{k}}[\boldsymbol{y}_{1:N}^{\mathrm{targ}}] := \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\widehat{\mathrm{MMD}}_k^2[\boldsymbol{y}_{1:M}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}]\right], \qquad \boldsymbol{y}_m := \boldsymbol{h}(\boldsymbol{x}_m). \tag{6}$$

The target–target term in Equation (4) is constant with respect to θ, so it can be ignored when taking gradients. Thus

$$\nabla_\theta \mathcal{L}^{\mathrm{k}}_\theta[\boldsymbol{y}^{\mathrm{targ}}_{1:N}] = \nabla_\theta \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \underbrace{\left[\frac{1}{M(M-1)} \sum_{m \neq m'}^{M} k(\boldsymbol{y}_m, \boldsymbol{y}_{m'}) - \frac{2}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} k(\boldsymbol{y}_m, \boldsymbol{y}^{\mathrm{targ}}_n) \right]}_{\text{[kernel terms]}}$$

$$= \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\text{[kernel terms]} \sum_{m=1}^{M} \nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right]. \tag{7}$$

We first derive the contribution of the model–model term. Define

$$\phi(\boldsymbol{x}, \boldsymbol{x}') := k(\boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}')).$$

Then

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{\boldsymbol{x}, \boldsymbol{x}' \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\phi(\boldsymbol{x}, \boldsymbol{x}') \right] \\
&= \nabla_\theta \int \phi(\boldsymbol{x}, \boldsymbol{x}')\, p_\theta(\boldsymbol{x})\, p_\theta(\boldsymbol{x}')\, d\boldsymbol{x}\, d\boldsymbol{x}' \\
&= \int \phi(\boldsymbol{x}, \boldsymbol{x}') \Big(\nabla_\theta p_\theta(\boldsymbol{x})\, p_\theta(\boldsymbol{x}') + p_\theta(\boldsymbol{x})\, \nabla_\theta p_\theta(\boldsymbol{x}') \Big) d\boldsymbol{x}\, d\boldsymbol{x}' \\
&= \mathbb{E}_{\boldsymbol{x}, \boldsymbol{x}' \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\phi(\boldsymbol{x}, \boldsymbol{x}') \big(\nabla_\theta \log p_\theta(\boldsymbol{x}) + \nabla_\theta \log p_\theta(\boldsymbol{x}') \big) \right].
\end{aligned} \tag{8}$$

Since $\phi(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x}', \boldsymbol{x})$ by symmetry of the kernel and $(\boldsymbol{x}, \boldsymbol{x}')$ are exchangeable, the two expectation terms are equal, so

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x}, \boldsymbol{x}' \overset{\text{i.i.d.}}{\sim} p_\theta} \left[k(\boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}')) \right] = 2\, \mathbb{E}_{\boldsymbol{x}, \boldsymbol{x}' \overset{\text{i.i.d.}}{\sim} p_\theta} \left[k(\boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}')) \nabla_\theta \log p_\theta(\boldsymbol{x}) \right]. \tag{9}$$

Applying Equation (9) to the model–model term in Equation (7), and the ordinary score-function trick to the model–target term, gives

$$\nabla_\theta \mathcal{L}^{\mathrm{k}}_\theta[\boldsymbol{y}^{\mathrm{targ}}_{1:N}] = \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\sum_{m=1}^{M} \left(\frac{2}{M(M-1)} \sum_{m' \neq m} k(\boldsymbol{y}_m, \boldsymbol{y}_{m'}) - \frac{2}{MN} \sum_{n=1}^{N} k(\boldsymbol{y}_m, \boldsymbol{y}^{\mathrm{targ}}_n) \right) \nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right]. \tag{10}$$

Define

$$s_m := \frac{1}{M-1} \sum_{m' \neq m} k(\boldsymbol{y}_m, \boldsymbol{y}_{m'}), \tag{11}$$

$$r_m := \frac{1}{N} \sum_{n=1}^{N} k(\boldsymbol{y}_m, \boldsymbol{y}^{\mathrm{targ}}_n), \tag{12}$$

$$g_m := 2\,(s_m - r_m)\,, \tag{13}$$

$$c^{\mathrm{k}}_m := \frac{1}{M} g_m. \tag{14}$$

Then

$$\nabla_\theta \mathcal{L}^{\mathrm{k}}_\theta[\boldsymbol{y}^{\mathrm{targ}}_{1:N}] = \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\sum_{m=1}^{M} c^{\mathrm{k}}_m \nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right]. \tag{15}$$

Thus c^{k}_m are exactly the kernel discrepancy coefficients used in Algorithm 1.

A.3 Unbiased surrogate gradients

Under standard regularity conditions, we may exchange gradient and expectation in the score-function expressions above. We now show that the stop-gradient surrogate used in Algorithm 1 gives an unbiased estimator of the gradient of the regularized objective.

Proposition A.3. *The stop-gradient surrogate gradient used by* `kCGM` *is an unbiased estimator of*

$$\nabla_\theta \mathcal{L}_\theta[\boldsymbol{y}_{1:N}^{\text{targ}}] = \nabla_\theta \mathcal{L}_\theta^{\text{k}}[\boldsymbol{y}_{1:N}^{\text{targ}}] + \lambda \nabla_\theta \, \mathrm{D}_{\mathrm{KL}}(p_\theta || p_{\theta_{\text{pre}}}).$$

Proof. From Section A.2, we have

$$\nabla_\theta \mathcal{L}_\theta^{\text{k}}[\boldsymbol{y}_{1:N}^{\text{targ}}] = \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\sum_{m=1}^{M} c_m^{\text{k}} \nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right].$$

For the KL term, Smith et al. [2026] derive the analogous score-function form

$$\nabla_\theta \, \mathrm{D}_{\mathrm{KL}}(p_\theta || p_{\theta_{\text{pre}}}) = \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\sum_{m=1}^{M} c_m^{\mathrm{KL}} \nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right],$$

where

$$l_m := \log p_\theta(\boldsymbol{x}_m) - \log p_{\theta_{\text{pre}}}(\boldsymbol{x}_m), \qquad c_m^{\mathrm{KL}} := \frac{1}{M} l_m.$$

Combining the two terms gives

$$\nabla_\theta \mathcal{L}_\theta[\boldsymbol{y}_{1:N}^{\text{targ}}] = \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\sum_{m=1}^{M} \left(\lambda c_m^{\mathrm{KL}} + c_m^{\text{k}} \right) \nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right].$$

Now define

$$c_m := \lambda c_m^{\mathrm{KL}} + c_m^{\text{k}}, \qquad \widehat{\mathcal{L}}^{\texttt{kCGM}\text{-surrogate}} := \sum_{m=1}^{M} c_m \log p_\theta(\boldsymbol{x}_m),$$

where the coefficients c_m are computed from samples drawn from $p_{\texttt{stop-grad}(\theta)}$ and are therefore treated as constants with respect to the gradient step. Hence

$$\nabla_\theta \widehat{\mathcal{L}}^{\texttt{kCGM}\text{-surrogate}} = \sum_{m=1}^{M} c_m \nabla_\theta \log p_\theta(\boldsymbol{x}_m).$$

Taking expectations over $\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta$ yields

$$\mathbb{E}_{\boldsymbol{x}_{1:M} \overset{\text{i.i.d.}}{\sim} p_\theta} \left[\nabla_\theta \widehat{\mathcal{L}}^{\texttt{kCGM}\text{-surrogate}} \right] = \nabla_\theta \mathcal{L}_\theta[\boldsymbol{y}_{1:N}^{\text{targ}}].$$

Thus the surrogate gradient is unbiased. □

A.4 Leave-one-out baselines

To reduce variance, Algorithm 1 applies leave-one-out centering to the kernel coefficients. Intuitively, the baseline for sample m should approximate the average coefficient that would have been obtained had $\boldsymbol{x}_m$ not appeared in the batch. This does not change the expectation provided the baseline subtracted from the coefficient for sample m is a function only of $\boldsymbol{x}_{-m}$, since for any $b_m(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\text{targ}})$ that does not depend on $\boldsymbol{x}_m$,

$$\mathbb{E}_{\boldsymbol{x}_{1:M} \sim p_\theta} \left[b_m(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\text{targ}}) \nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right] = 0.$$

Indeed,

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{x}_{1:M} \sim p_\theta} \left[b_m(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\text{targ}}) \nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right] \\ &\quad = \mathbb{E}_{\boldsymbol{x}_{-m} \sim p_\theta} \left[b_m(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\text{targ}}) \mathbb{E}_{\boldsymbol{x}_m \sim p_\theta} \left[\nabla_\theta \log p_\theta(\boldsymbol{x}_m) \right] \right] = 0. \end{aligned}$$

For the kernel term, we therefore define the leave-one-out baseline for sample m by recomputing the coefficients of the remaining samples on the reduced batch $\boldsymbol{x}_{-m}$. Concretely, for $j \neq m$, define

$$s_j^{(-m)} := \frac{1}{M-2} \sum_{\ell \neq j,m} k(\boldsymbol{y}_j, \boldsymbol{y}_\ell), \qquad r_j^{(-m)} := r_j,$$

and

$$g_j^{(-m)} := 2\big(s_j^{(-m)} - r_j\big), \qquad c_j^{\text{k},(-m)} := \frac{1}{M} g_j^{(-m)}.$$

We then set

$$b_m^{\mathrm{k}}(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}) := \frac{1}{M-1} \sum_{j \neq m} c_j^{\mathrm{k},(-m)},$$

and use the centered coefficient

$$c_m^{\mathrm{k,LOO}} := c_m^{\mathrm{k}} - b_m^{\mathrm{k}}(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}).$$

By construction, $b_m^{\mathrm{k}}(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\mathrm{targ}})$ depends only on $\boldsymbol{x}_{-m}$ and $\boldsymbol{y}_{1:N}^{\mathrm{targ}}$, so replacing c_m^{k} with $c_m^{\mathrm{k,LOO}}$ preserves unbiasedness.

Naively computing $b_m^{\mathrm{k}}(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\mathrm{targ}})$ for every m would require recomputing the model–model coefficients separately for each leave-one-out batch. However, all leave-one-out coefficients can be obtained efficiently from the same model feature kernel matrix $K_{\boldsymbol{yy}}$, where $(K_{\boldsymbol{yy}})_{ij} = k(\boldsymbol{y}_i, \boldsymbol{y}_j)$. In particular, for $j \neq m$,

$$s_j^{(-m)} := \frac{1}{M-2} \sum_{\ell \neq j,m} k(\boldsymbol{y}_j, \boldsymbol{y}_\ell) \tag{16}$$

$$= \frac{\sum_{\ell \neq j} k(\boldsymbol{y}_j, \boldsymbol{y}_\ell) - k(\boldsymbol{y}_j, \boldsymbol{y}_m)}{M-2} \tag{17}$$

$$= \frac{(M-1)s_j - k(\boldsymbol{y}_j, \boldsymbol{y}_m)}{M-2}, \tag{18}$$

$$g_j^{(-m)} := 2\big(s_j^{(-m)} - r_j\big), \qquad c_j^{\mathrm{k},(-m)} := \frac{1}{M} g_j^{(-m)}, \tag{19}$$

$$b_m^{\mathrm{k}}(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}) := \frac{1}{M-1} \sum_{j \neq m} c_j^{\mathrm{k},(-m)}, \qquad c_m^{\mathrm{k,LOO}} := c_m^{\mathrm{k}} - b_m^{\mathrm{k}}(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\mathrm{targ}}). \tag{20}$$

Thus all $b_m^{\mathrm{k}}(\boldsymbol{x}_{-m}, \boldsymbol{y}_{1:N}^{\mathrm{targ}})$ can be computed in $O(M^2)$ time from the row sums of $K_{\boldsymbol{yy}}$, without forming additional kernel matrices. Algorithm 1 uses the centered coefficients $c_m^{\mathrm{k,LOO}}$ from Equation (20).

Algorithm 1 also applies the ordinary leave-one-out baseline to the KL coefficients, exactly as in Smith et al. [2026]. The final coefficients used by `kCGM` therefore remain unbiased while typically having lower variance.

A.5 Behavior of self-repulsion weight less than one for categorical target features

We now show explicitly how the self-repulsion weight $\alpha < 1$ changes the target of the population objective in a simple categorical setting. Suppose the feature space is $\{1, \ldots, C\}$, the target feature distribution is

$$Q = (Q_1, \ldots, Q_C) \in \Delta_C,$$

and the model-induced feature distribution is

$$P = (P_1, \ldots, P_C) \in \Delta_C.$$

Equivalently, Q_i is the probability that a target feature equals category i, and P_i is the probability that a model sample has feature category i. We represent category i by the one-hot vector $\boldsymbol{e}_i \in \mathbb{R}^C$ and use the dot-product kernel

$$k(\boldsymbol{e}_i, \boldsymbol{e}_j) = \boldsymbol{e}_i^\top \boldsymbol{e}_j = \mathbb{1}\{i = j\}.$$

The population version of the α-weighted MMD objective is then

$$\mathcal{L}_\alpha(P; Q) = \alpha \mathbb{E}_{\boldsymbol{z}, \boldsymbol{z}' \overset{\text{iid}}{\sim} P}\left[k(\boldsymbol{z}, \boldsymbol{z}')\right] - 2\mathbb{E}_{\boldsymbol{z} \sim P, \boldsymbol{y} \sim Q}\left[k(\boldsymbol{z}, \boldsymbol{y})\right] + \mathbb{E}_{\boldsymbol{y}, \boldsymbol{y}' \overset{\text{iid}}{\sim} Q}\left[k(\boldsymbol{y}, \boldsymbol{y}')\right] \tag{21}$$

$$= \alpha \sum_{i=1}^{C} P_i^2 - 2 \sum_{i=1}^{C} P_i Q_i + \sum_{i=1}^{C} Q_i^2. \tag{22}$$

Completing the square gives

$$\mathcal{L}_\alpha(P; Q) = \alpha \sum_{i=1}^{C} \left(P_i - \frac{Q_i}{\alpha}\right)^2 + \left(1 - \frac{1}{\alpha}\right) \sum_{i=1}^{C} Q_i^2. \tag{23}$$

The second term is constant with respect to P, and $\alpha > 0$, so the optimizer over categorical distributions is

$$P^\star = \arg\min_{P \in \Delta_C} \left\| P - \frac{Q}{\alpha} \right\|_2^2$$

This problem is known as simplex projection and has the closed form solution [Duchi et al., 2008]

$$P_i^\star = \left(\frac{Q_i}{\alpha} - \nu \right)_+,$$

where $(u)_+ := \max\{u, 0\}$ and ν is chosen so that $\sum_i P_i^\star = 1$. Equivalently, defining $\tau_\alpha := \alpha\nu$, we can write

$$P_i^\star = \frac{(Q_i - \tau_\alpha)_+}{\alpha}, \qquad \sum_{i=1}^{C} (Q_i - \tau_\alpha)_+ = \alpha.$$

When $\alpha = 1$, we may take $\tau_\alpha = 0$, giving $P^\star = Q$. When $\alpha < 1$, the optimizer subtracts a common threshold from the target probabilities, sets categories below that threshold to zero, and rescales the remaining mass by $1/\alpha$. Therefore, in this categorical setting, reducing α does not target the original distribution Q exactly. Instead, it produces a thresholded sharpening of Q, concentrating probability mass on higher-probability categories.

B Practical usage recommendations

Our experiments suggest some practical guidance. If the target features are continuous, then the energy-distance kernel will likely work well and has no extra hyperparameters. We recommend doing a small sweep over the regularization weight $\lambda = \{10^{-1}, 10^{-2}, 10^{-3}\}$ while keeping self-repulsion $\alpha = 1$. Lower λ can make optimization unstable. Trying $\alpha = \{0.25, 0.5\}$ can additionally improve distribution-matching performance as seen in our protein structure experiment. For discrete-valued or otherwise non-Euclidean kernels, use domain-specific kernels such as the Tanimoto kernel.

We also recommend two scaling tricks to help hyperparameters like learning rate transfer across λ and kernel choices. First, rescale the MMD and KL-regularization terms so that their ratio remains λ, but the overall scale of the objective stays roughly constant via Equation (24). Second, scale the kernel values so that the initial squared MMD between pretrained generations and target features is fixed.

C Additional experimental details

In this section, we provide additional information about each of our experiments, including how we compute log-probabilities and how we define the feature function $\boldsymbol{h}$ and the feature target samples $\boldsymbol{y}_{1:N}^{\text{targ}}$.

All experiments use Adam with momentum hyperparameters $\beta = (0.9, 0.999)$ and either a cosine or linear decay learning rate schedule. We train all models on either a single H100 or L40S GPU with at most 4 CPUs. Genie 2 experiments take 30 hours per finetuning run on L40S and G2PT runs take one hour.

C.1 Protein secondary structure distribution matching

Features and kernels. Following Smith et al. [2026], we use the Biotite package (`https://www.biotite-python.org/latest/index.html`) to annotate secondary structure.

For `kCGM`, we use the proportion of alpha-helix and beta-strand residues to compute a two-vector of features. These are compared using the energy-distance kernel (Equation (2)).

For `CGM`, we use the bivariate quantile features implemented by Smith et al. [2026]. The quantile bins are initialized using the empirical quantiles from the CATH target dataset.

Sampling and log-probabilities. All optimization hyperparameters and score-function gradient computations were taken from the code of Smith et al. [2026] (`https://github.com/`

smithhenryd/cgm/blob/main/genie2/calibrate_genie2.py). In particular, see their appendix section on diffusion models for a more detailed treatment than follows.

For completeness, we briefly outline the approach to sampling and computing score-function gradients. We use Euler-Maruyama sampling with 100 sampling steps. Each sampling step i consists of estimating a drift $\boldsymbol{b}_\theta(\boldsymbol{x}_i, i/100)$ using the model, then sampling the next time $\boldsymbol{x}_{i+1}$ from a Gaussian with mean and variance determined by the drift and noise schedule. Therefore the log-probability of the sampling trajectory can be decomposed into a sum of log probabilities of Gaussians with means determined by the model weights, enabling tractable gradient computations.

The decomposition over sampling steps also enables accumulating gradients over time steps, avoiding holding the entire computational graph in memory, making finetuning tractable on one GPU even with 100 sampling steps.

Additional training and evaluation details. Following Smith et al. [2026], we train for 100 epochs with batch size 64 and a learning rate that decays linearly from 10^{-5} to 10^{-7} and sampling noise level $\sigma = 0.5$ over 100 sampling steps. Each batch is split into 20 temporal sampling chunks by 32 batch dimension chunks.

For the symmetrized KL evaluation metric, we compare two empirical bivariate feature distributions using kernel density estimates. Each sample was represented by two features taking values in $[0, 1]^2$, such as secondary-structure proportions. Given samples from two distributions, we fit two Gaussian kernel density estimates using `scipy.stats.gaussian_kde` with default bandwidth selection rule, Scott's rule. We then approximated the KL divergence by deterministic numerical integration on a uniform 50×50 grid over $[0, 1]^2$.

C.2 Antibiotics feature distribution matching

Sampling and log-probabilities. Sampling and log-probabilities for the G2PT model proceed in the standard way for autoregressive models. That is, tokens are sequentially sampled according to $\boldsymbol{x}_i \sim p_\theta(\boldsymbol{x}_i \mid \boldsymbol{x}_{<i})$, where $\boldsymbol{x}_{<i}$ indicates the previous $i-1$ tokens in the sequence. Autoregressive log-probabilities are the same as used for training,

$$\log p_\theta(\boldsymbol{x}) = \sum_{i=1}^{T} \log p_\theta(\boldsymbol{x}_i \mid \boldsymbol{x}_{<i}),$$

where T is the sequence length including an end-of-sequence token. This can be computed with one forward pass since G2PT is an autoregressive transformer.

Kernel rescaling. In order to make it easier to share hyperparameters across all three target features, we rescale each kernel during training so that the MMD^2 is 10 for the pretrained model. We do this by sampling five batches from the pretrained model, computing average MMD^2 against the target dataset for each batch $\overline{\mathrm{MMD}^2}[\boldsymbol{y}_{1:M}, \boldsymbol{y}^{\mathrm{targ}}_{1:N}]$, then defining

$$k'(\boldsymbol{x}, \boldsymbol{y}) = \frac{10k(\boldsymbol{x}, \boldsymbol{y})}{\overline{\mathrm{MMD}^2}[\boldsymbol{y}_{1:M}, \boldsymbol{y}^{\mathrm{targ}}_{1:N}]}.$$

Features and kernels. For all features, chemically invalid generations are mapped to the zero vector.

For fingerprint embeddings, we use RDKit [Landrum et al., 2024] to compute Morgan fingerprints with 256 bits and radius two and use the Tanimoto kernel (Equation (3)).

For scaffold embeddings, we use RDKit to extract Murcko generic scaffolds, then convert those to SMILES strings. Those SMILES are then passed to `fcd_torch` (`https://github.com/insilicomedicine/fcd_torch`) in order to get ChemNet embeddings. Chemically invalid generations are mapped to the zero vector. Finally, we use PCA to reduce the embedding dimension to 64. The PCs are fit on the target antibiotics dataset embeddings.

For our molecular descriptor features, we use RDKit to compute molecular weight, logP, topological polar surface area, number of rotatable bonds, QED, synthetic accessibility score, fraction of sp^3 carbons, numbers of hydrogen-bond donors and acceptors, number of aromatic rings, and heavy atom

count. We divide each by its empirical standard deviation in the antibiotics target set so they are all roughly equally weighted, but do not zero-center them.

In Figure 8 we compare the distance between embeddings that share a generic Murcko scaffold against embeddings with different scaffolds. ChemNet embeddings of Murcko scaffolds (with and without PCA to 64 components) compared via Euclidean distance both have distance zero when scaffolds are shared by construction, but also show a wide range of distances for distinct scaffolds. Morgan fingerprints with Tanimoto distance (one minus Tanimoto similarity) do not separate by scaffold identity.

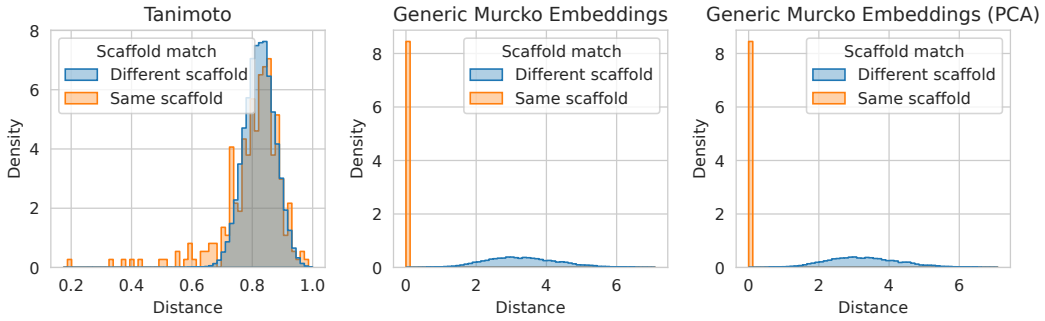

Figure 8: Comparison of molecular similarity measures for identifying shared generic Murcko scaffolds. Histograms show pairwise distances for molecule pairs with the same generic scaffold and with different scaffolds. Morgan fingerprint Tanimoto distance only weakly separates these groups, while distances in the generic Murcko embedding space separate them much more clearly. PCA preserves this separation, suggesting that dimensionality reduction has little effect on scaffold identification.

Additional training and evaluation details. For each finetuning method, including the `CGM` and direct finetuning baselines, we train for 500 epochs with batch size 192 and evaluate with 9,600 samples. We use a cosine-annealing-with-warm-up learning rate schedule with 25 warm-up epochs and final learning rate of 0.01 times the initial. We tried learning rates $5 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-5}$ for each method and took the learning rate for each with the lowest distribution-matching loss for the Morgan fingerprint objective averaged over runs with $\lambda = 10$ and $\lambda = 0.01$ to ensure the learning rate worked for high and low regularization weights. Towards this end, we found it helpful to normalize the regularization and MMD objectives' weightings such that their ratio was preserved while keeping the absolute scale of the loss roughly conserved across regularization weights. This resulted in the reweighted objective

$$\mathcal{L}_\theta[k, \boldsymbol{y}^{\text{targ}}_{1:N}] = \frac{1}{1+\lambda} \mathbb{E}_{\boldsymbol{x}_{1:M} \overset{i.i.d.}{\sim} p_\theta} [\widehat{\text{MMD}}^2_k[\boldsymbol{y}_{1:M}, \boldsymbol{y}^{\text{targ}}_{1:N}]] + \frac{\lambda}{1+\lambda} \text{D}_{\text{KL}}(p_\theta || p_{\theta_{\text{pre}}}). \tag{24}$$

This resulted in all methods having optimal performance at learning rate 10^{-5}.

Direct finetuning baseline. The direct finetuning baseline uses the full antibiotic molecules as target samples in the original sample space, rather than using only feature-space samples. Starting from the same pretrained G2PT model as `kCGM` and `CGM`, we finetune by minimizing the standard autoregressive negative log-likelihood of target antibiotic token sequences denoted $\mathcal{D}_{\text{abx}}$,

$$\mathcal{L}_{\text{sup}}(\theta) = -\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\text{abx}}} [\log p_\theta(\boldsymbol{x})] + \lambda D_{\text{KL}}(p_\theta \| p_{\theta_{\text{pre}}}),$$

where $\log p_\theta(\boldsymbol{x})$ is the G2PT sequence log-probability. We sweep the same KL-regularization weights used for `kCGM` and `CGM` so that all methods are compared at matched KL-to-pretrained budgets. Unlike `kCGM` and `CGM`, this baseline requires full target molecules and trains the model to imitate the small antibiotic set directly.

Antibiotics target dataset. We constructed the antibiotic target set used in the G2PT experiments from the set of known antibiotics from Scalia et al. [2025]. We process this set to produce a target

set of SMILES strings that are unique, chemically valid, and compatible with the exact molecular representation pipeline used during G2PT finetuning and calibration. To do this, we iterated through the input SMILES records and split any multi-component entry at the "." separator, so that mixtures or salts contributed one row per molecular fragment rather than being treated as a single composite molecule. Each fragment was then parsed with RDKit and canonicalized with isomeric information removed, so duplicates differing only by SMILES syntax or stereochemical annotation collapsed to a common canonical representation. Fragments that could not be parsed by RDKit were discarded.

After canonicalization, each candidate SMILES string was filtered by running it through the same preprocessing steps used by the downstream G2PT pipeline. In particular, the molecule was converted into the graph representation used by G2PT, linearized with the breadth-first-search serialization used by the model, and then tokenized with the pretrained GuacaMol G2PT tokenizer. A candidate was retained only if all of these steps succeeded, the number of atoms did not exceed the tokenizer's supported indexed-atom vocabulary, and the resulting token sequence fit within the tokenizer's maximum input length. This ensures that every molecule is processable by the tokenizer used in the experiments. Finally, exact duplicate canonical SMILES were removed.

C.3 Generating DNA with realistic cell-type-specific activity profiles

Masked discrete diffusion model pretraining. We pretrained a conditional masked discrete diffusion model on DeepMEL2 enhancer sequences. DNA sequences were represented as categorical tokens over the four bases, with an additional absorbing mask token used by the diffusion process. The model was trained for 1,000 epochs on the DeepMEL2 training split on a single L40S GPU with batch size 256, learning rate 5×10^{-4}, Adam optimization, and validation-loss checkpointing. The run used a one-dimensional convolutional sequence model with 512 hidden channels, dropout 0.1, and one dilation block.

The architecture follows a ChromBPNet-style dilated convolutional design [Pampari et al., 2025]. Input tokens are one-hot encoded over the augmented alphabet $\{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}, \boldsymbol{m}\}$ and passed through an initial convolution where $\boldsymbol{m}$ denotes the mask token. The model then applies a residual stack of one-dimensional convolutional blocks with dilation pattern $(1, 1, 4, 16, 64)$. Each block uses channel-wise layer normalization, a same-padded dilated convolution, ReLU nonlinearity, and dropout. A final 1D convolution maps the hidden representation to logits over the four unmasked DNA bases at each position.

The model is conditional on the DeepMEL2 cell-topic annotation vector. The conditioning vector is passed through a small multilayer perceptron consisting of layer normalization, a linear layer to the model hidden dimension, a ReLU, and a second linear layer. The resulting embedding is added to the sequence hidden representation at the beginning of each dilation block, so the same condition vector can globally modulate predictions at all sequence positions.

Training uses the masked diffusion denoising objective. For each minibatch, a diffusion time $t \in [0, 1]$ is sampled using a low-discrepancy grid with a random offset [Kingma et al., 2021]. Given a clean sequence $\boldsymbol{x}_0$, the forward process independently keeps each token with probability

$$\alpha(t) = 1 - \cos\left(\frac{\pi}{2}(1-t)\right)$$

and replaces it by the mask token $\boldsymbol{m}$ otherwise. The network receives the corrupted sequence $\boldsymbol{x}_t$ and the condition vector, and is trained to predict the original base at masked positions. The per-sequence loss is a weighted masked cross entropy,

$$\mathcal{L}(\theta) = \sum_{j : x_t[j] = \boldsymbol{m}} w(t)\, \mathrm{CE}\left(x_0[j], \pi_\theta^{(j)}(\cdot \mid \boldsymbol{x}_t, \boldsymbol{c})\right),$$

normalized by sequence length. We use the cosine diffusion loss weighting

$$w(t) = \frac{\pi}{2} \tan\left(\frac{\pi}{2}(1-t)\right),$$

clamped in the implementation to avoid extreme weights.

AlphaGenome target features and kernel. For distributional calibration, we used AlphaGenome predictions as target features. Because AlphaGenome requires a minimum input length of 2,048

base pairs, each DeepMEL2 enhancer sequence was inserted into the center of a fixed random DNA background sequence before calling AlphaGenome. The same insertion procedure was used for both real target sequences and generated sequences, so the calibrated features reflect changes in the enhancer region rather than changes in sequence context. The random background was generated once with base probabilities ($P_A = 0.2910, P_C = 0.2085, P_G = 0.2087, P_T = 0.2918$) corresponding roughly to frequencies seen in regulatory DNA and then held fixed across feature evaluations.

For each sequence, we used the AlphaGenome ATAC predictions as enhancer activity features. The feature vector was defined by taking the maximum predicted ATAC value over genomic position for each retained ATAC channel, i.e. the peak height of each channel. We then applied a $\log(1 + x)$ transform to the peak-height features, which made the marginal feature distributions closer to Gaussian and reduced the influence of very large peaks.

The target distribution for each cell topic was constructed from real DeepMEL2 sequences positive for that topic. We used up to 256 target sequences per topic when precomputing the AlphaGenome feature cache. During calibration, each update selects one topic and samples condition vectors from the training examples positive for that topic. Generated sequences are then compared to the corresponding topic-specific target feature distribution.

For the calibration objective, the log-transformed AlphaGenome features were projected using PCA fit on the training target features. We used 32 principal components with whitening; these components explained more than 99% of the variance in the log-transformed target features. The PCA-whitened features were used for the MMD calibration objective, while final reported feature metrics were computed on the non-PCA log-transformed AlphaGenome features.

For `kCGM` and `CGM`, calibration was run for 20 epochs with batch size 128, 50 diffusion sampling steps, learning rate 10^{-4}, cosine learning-rate decay to 10% of the initial learning rate, and three random seeds. The final validation evaluation generated 512 samples per condition.

Sampling method. For a fixed conditioning vector $\boldsymbol{c}$, the sampler starts from a fully masked sequence $\boldsymbol{x}(0)$ of length L and progressively unmasks tokens. Let $\boldsymbol{m}$ denote the mask token and let

$$M(i-1) = \{j : \boldsymbol{x}(i-1)[j] = \boldsymbol{m}\}$$

be the masked positions before step i.

The implementation uses a cosine unmasking schedule. For a nominal number of sampling steps T, define

$$r(i) = \text{round}\left(L \cos\left(\frac{\pi}{2}\frac{i}{T}\right)\right), \qquad i = 0, \ldots, T.$$

The number of positions unmasked at step i is

$$k_i = r(i-1) - r(i),$$

with zero-sized steps omitted in the implementation. At each nonzero step:

1. Select a subset $U(i) \subset M(i-1)$ with $|U(i)| = k_i$ uniformly at random without replacement.
2. Evaluate the conditional model on the current partially masked sequence and condition vector $\boldsymbol{c}$. This gives logits $a_\theta^{(j)}(\cdot \mid \boldsymbol{x}(i-1), \boldsymbol{c})$ over the four DNA bases at each position j.
3. Convert logits to categorical probabilities,

$$\pi_\theta^{(j)}(\cdot \mid \boldsymbol{x}(i-1), \boldsymbol{c}) = \text{softmax}\left(a_\theta^{(j)}(\cdot \mid \boldsymbol{x}(i-1), \boldsymbol{c})\right),$$

   and sample each newly unmasked base independently:

$$\boldsymbol{x}(i)[j] \sim \pi_\theta^{(j)}(\cdot \mid \boldsymbol{x}(i-1), \boldsymbol{c}), \qquad j \in U(i).$$

All other positions are left unchanged. Thus the chain begins fully masked and ends at a fully specified DNA sequence.

Transition probabilities. Conditioned on $\boldsymbol{c}$, the initial distribution is deterministic:

$$\pi_0(\boldsymbol{x}(0)) = \mathbf{1}\{\boldsymbol{x}(0) \text{ is fully masked}\}.$$

For step i, the transition probability factors into a position-selection term and token-sampling terms:

$$p_\theta(\boldsymbol{x}(i) \mid \boldsymbol{x}(i-1), \boldsymbol{c}) = C_i \prod_{j \in U(i)} \pi_\theta^{(j)}\left(\boldsymbol{x}(i)[j] \mid \boldsymbol{x}(i-1), \boldsymbol{c}\right),$$

where

$$C_i = \binom{|M(i-1)|}{k_i}^{-1}$$

if the within-step unmasking set is treated as unordered. This constant accounts for choosing which masked positions to unmask. It does not depend on θ. Since the number of masked positions is fixed by the schedule, C_i is also the same across trajectories of the same length.

Trajectory log-probability. As in the Genie 2 setting, the score-function gradient is computed using the full sampling trajectory

$$\boldsymbol{x} := \big(\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(S)\big),$$

where $\boldsymbol{x}(S)$ is the final generated DNA sequence and S is the number of unmasking steps. The conditional trajectory log-probability is

$$\begin{aligned}\log p_\theta(\boldsymbol{x} \mid \boldsymbol{c}) &= \log \pi_0(\boldsymbol{x}(0)) + \sum_{i=1}^{S} \log p_\theta(\boldsymbol{x}(i) \mid \boldsymbol{x}(i-1), \boldsymbol{c}) \\ &= \sum_{i=1}^{S} \log C_i + \sum_{i=1}^{S} \sum_{j \in U(i)} \log \pi_\theta^{(j)}\left(\boldsymbol{x}(i)[j] \mid \boldsymbol{x}(i-1), \boldsymbol{c}\right).\end{aligned}$$

We use only the second term, since the $\log C_i$ terms are constant with respect to model parameters.

Parameter gradients. The gradient of the trajectory log-probability is therefore

$$\nabla_\theta \log p_\theta(\boldsymbol{x} \mid \boldsymbol{c}) = \sum_{i=1}^{S} \sum_{j \in U(i)} \nabla_\theta \log \pi_\theta^{(j)}\left(\boldsymbol{x}(i)[j] \mid \boldsymbol{x}(i-1), \boldsymbol{c}\right).$$

During finetuning, trajectories are sampled without gradients, then each sampling step is replayed under the current policy to compute the corresponding log-probability contribution. This lets the REINFORCE-style calibration gradient be accumulated one diffusion step at a time rather than attempting to backpropagate through the full sampling procedure.

C.4 Overview figure: a smile made of SMILES

We briefly outline the G2PT finetuning setup we use to produce Figure 1.

Features, kernel, and target features. For the smiley-face calibration experiment, we define the target distribution in a two-dimensional standardized descriptor space,

$$z = \big(z_{\mathrm{MolWt}}, z_{\mathrm{MolLogP}}\big).$$

The standardization is computed from samples drawn from the pretrained base model. We decode a presampled set of molecules, retain the valid ones, and compute the coordinate-wise mean μ and standard deviation σ of $(\mathrm{MolWt}, \mathrm{MolLogP})$. A valid molecule with raw descriptor vector $\boldsymbol{x}$ is then mapped to

$$z = \frac{x - \mu}{\sigma}.$$

Invalid molecules are assigned the placeholder value $(0, 0)$. The synthetic smiley target is constructed directly in this standardized space.

Given a target sample size n, we allocate $n_{\text{left}} = \lfloor 0.12n \rfloor$ points to the left eye, $n_{\text{right}} = \lfloor 0.12n \rfloor$ points to the right eye, and assign the remaining points to the mouth. The two eyes are sampled as isotropic Gaussian clusters with centers

$$c_{\text{left}} = (-0.8,\, 0.75), \qquad c_{\text{right}} = (0.8,\, 0.75),$$

and standard deviation 0.16 in each coordinate. Equivalently,

$$X_{\text{left}} \sim \mathcal{N}\big(c_{\text{left}},\, 0.16^2 I_2\big)\,, \qquad X_{\text{right}} \sim \mathcal{N}\big(c_{\text{right}},\, 0.16^2 I_2\big)\,.$$

The mouth is sampled from a noisy lower arc. We first draw

$$\theta \sim \text{Unif}(210^\circ,\, 330^\circ),$$

which traces the lower portion of a circle. We then form the corresponding point on a radius-1 arc centered at $(0, 0.15)$,

$$m(\theta) = (0,\, 0.15) + (\cos\theta,\, \sin\theta),$$

and add isotropic Gaussian perturbations with standard deviation 0.12:

$$X_{\text{mouth}} = m(\theta) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0,\, 0.12^2 I_2).$$

After concatenating the two eye clouds and the mouth samples, we remove any point whose Euclidean norm is smaller than 0.35, i.e.,

$$\|z\|_2 < 0.35.$$

Rejected points are replaced by newly drawn samples until exactly n target points are obtained. This rejection step creates a small hole around the origin, which is useful because invalid generated molecules are also mapped to $(0, 0)$ under the descriptor representation. The resulting empirical target distribution therefore consists of two Gaussian eye clusters, a noisy lower arc for the smile, and an empty region near the origin.

We use the energy-distance kernel for the MMD objective (Equation (2)).

Training settings. We train for 500 epochs with $\lambda = 10^{-4}$, $\alpha = 1$, learning rate 2×10^{-5}, and batch size 2,048 split into four chunks to reduce the memory footprint.

C.5 One-dimensional self-repulsion demonstration

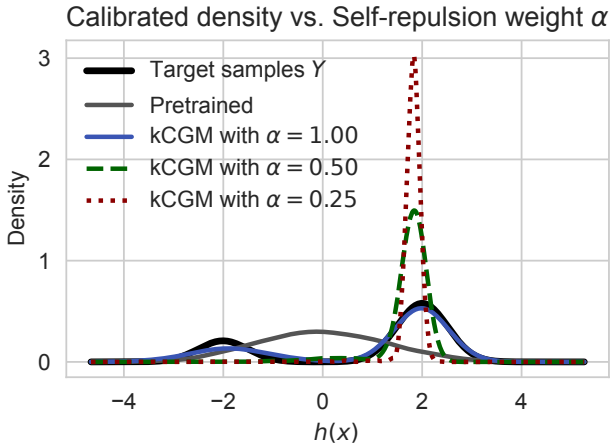

Figure 9: Effect of self-repulsion weight α in a synthetic bimodal distribution-matching task. Smaller α weakens self-repulsion and produces more mode-seeking behavior.

To illustrate the role of the self-repulsion term in `kCGM`, we constructed a one-dimensional synthetic example using a learnable two-component Gaussian mixture model. The target distribution is a symmetric bimodal mixture with modes at -2 and $+2$ and standard deviation 0.35, while the initial model has the same number of equally weighted mixture components but starts with means

$-0.75, 0.75$ and standard deviations 1.0. We use the identity feature map and the energy-distance kernel.

Target samples are obtained by drawing 4,000 points from the fixed target mixture, and the initial model is separately sampled before calibration to visualize its pre-finetuning distribution.

The demo then runs `kCGM` several times from the same initial model, varying only the self-repulsion weight over the values $\{0.25, 0.5, 0.75, 1.0\}$. In each run, the model is trained for 300 epochs with batch size 256, regularization parameter $\lambda = 10^{-2}$, and learning rate 3×10^{-2}. After calibration, the script draws fresh samples from each fitted model, reports the final logged loss values, and visualizes the resulting densities with one-dimensional kernel density estimates.

D Additional results

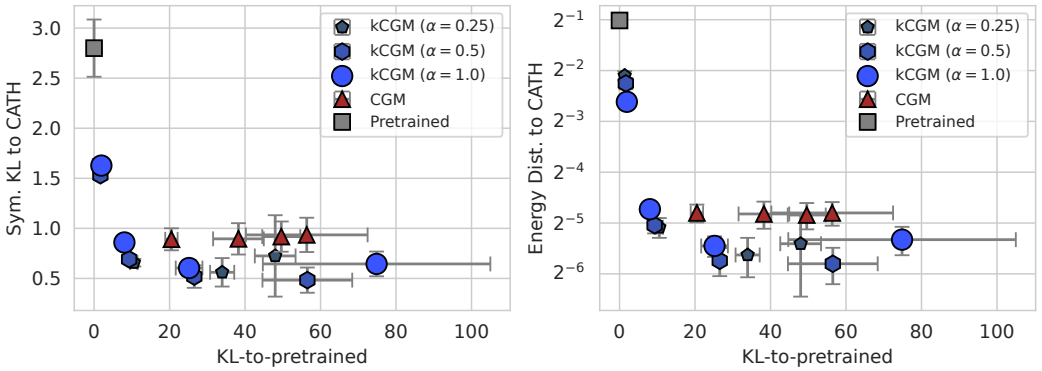

Figure 10: Extended Genie 2 protein secondary-structure finetuning results for `kCGM` across all self-repulsion weights α. The left panel reports symmetrized KL to the target CATH secondary-structure distribution, and the right panel reports energy distance to the same target distribution, both plotted against KL-to-pretrained. This figure extends Figure 5 by showing the full set of `kCGM` tradeoff curves obtained by varying α.

D.1 G2PT

Figure 11 repeats the experiment of Figure 2 for scaffold and chemical descriptor features, showing that `kCGM` works for matching distributions from diverse feature extractors.

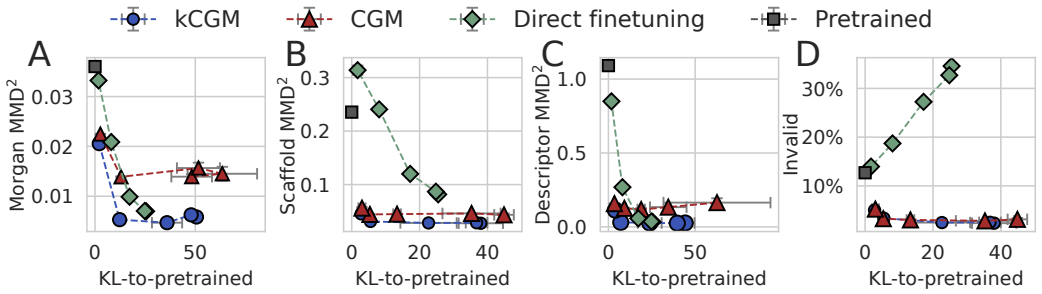

Figure 11: Comparison of direct finetuning, `kCGM`, and `CGM` for adapting G2PT to the antibiotics set for five values of the KL-regularization weight λ. Panels A–C evaluate separate `kCGM` and `CGM` models trained to match the target Morgan fingerprint, Murcko scaffold, and descriptor distributions, respectively, using the corresponding MMD^2 metric. Panel D shows the fraction of chemically invalid samples for scaffold finetuned `kCGM` and `CGM` and for direct finetuning. Direct finetuning does not depend on the feature target and is shared across all panels.

Table 1 compares the methods for each target feature space with metrics interpolated to have equal KL-from-pretrained to evaluate a KL-constrained comparison. Each `kCGM` model performs best on the features it was trained for, while transfer across feature spaces is asymmetric. For example, Morgan finetuning transfers better to descriptor similarity than scaffold finetuning transfers to Morgan similarity, showing that the target feature spaces differ in granularity and therefore preserve different aspects of the antibiotic distribution.

Table 1: Best metric values for runs with KL budget 25. Entries show linearly interpolated mean (SEM) at KL-to-pretrained equal to 25. SEMs are interpolated by treating the two neighboring estimates as independent and propagating variance under the linear interpolation weights. Bold indicates the best method for each metric; underline indicates the second-best method.

Method	Morgan MMD^2	Scaffold MMD^2	Descriptor MMD^2	Validity
kCGM Morgan	**0.005 (0.00017)**	0.039 (0.004)	0.15 (0.007)	95% (0.2%)
kCGM Scaffold	0.038 (0.00073)	**0.028 (0.0009)**	1.2 (0.04)	**98% (0.098%)**
kCGM Descriptor	0.024 (0.00028)	0.13 (0.009)	**0.027 (0.0042)**	96% (0.2%)
CGM Morgan	0.014 (0.00019)	0.15 (0.015)	0.26 (0.012)	92% (0.21%)
CGM Scaffold	0.037 (0.00083)	0.045 (0.00052)	1.2 (0.054)	97% (0.13%)
CGM Descriptor	0.024 (0.00028)	0.12 (0.0083)	0.12 (0.0059)	94% (0.29%)
Direct finetuning	0.007 (7e-05)	0.085 (0.0021)	0.036 (0.00049)	67% (0.093%)
Pretrained	0.036 (0.0002)	0.24 (0.003)	1.1 (0.00031)	87% (0.22%)

D.1.1 Sensitivity to number of principal components used for target features

In order to measure the effect of the number of principal components (PCs) used to define the feature function $\boldsymbol{h}$, we ran `kCGM` on the G2PT antibiotic task using scaffold embeddings reduced to 32, 64, or 128 principal components, as well as the original 512-dimensional embeddings without PCA. As seen in Figure 12, the number of PCs has little effect on the resulting model for this experiment for both the models' target scaffold features (panel B) and in transfer to other features and overall sample validity (panels A, C, D). Adding the number of retained PCs to a linear model controlling for regularization weight λ and training seed increased the explained variance in Morgan Tanimoto MMD^2 by only 3 percentage points ($\Delta R^2 = 0.03$). These results suggest that, for this feature representation, `kCGM` is relatively insensitive to the number of retained principal components across the tested range.

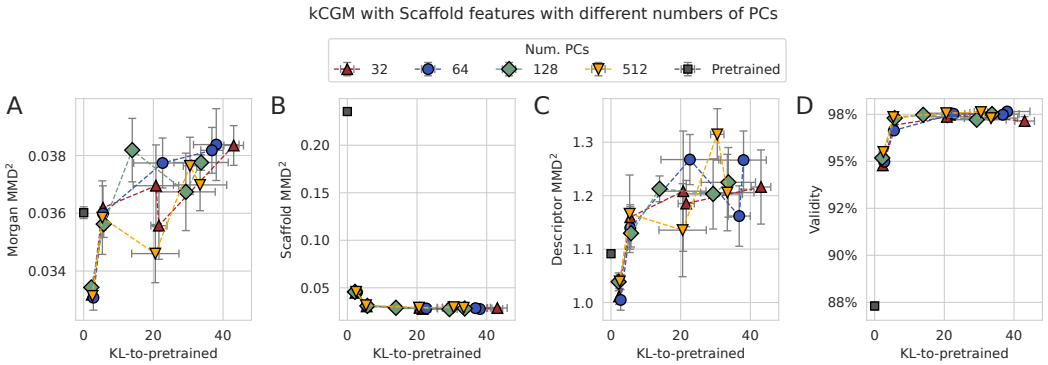

Figure 12: Comparison of `kCGM` for adapting G2PT to the antibiotics set with scaffold embedding features for four choices of embedding dimension and five values of the KL-regularization weight λ. Panels A–C evaluate distribution matching on Morgan fingerprint, Murcko scaffold, and descriptor features using the MMD^2 metric. Panel D shows the fraction of chemically valid samples.

D.1.2 Effect of kernel choice on performance metrics

To evaluate the effect of kernel choice on kCGM, we re-ran the G2PT finetuning experiments on the generic scaffold embedding and small-molecule descriptor feature sets with an alternative kernel. Specifically, we use a multiscale RBF mixture kernel defined as

$$k_{\mathrm{RBF}}(u, v) = \frac{1}{5} \sum_{s=1}^{5} \exp\left(-\frac{\|u - v\|_2^2}{2\sigma_s^2}\right), \qquad \sigma_s = a_s d_{\mathrm{med}}, \tag{25}$$

where $d_{\mathrm{med}} = \mathrm{median}_{i<j} \|y_i^{\mathrm{targ}} - y_j^{\mathrm{targ}}\|_2$ is the median pairwise Euclidean distance among target features and $(a_1, \ldots, a_5) = (0.25, 0.5, 1, 2, 4)$. Multiscale Gaussian-kernel mixtures have previously been used for MMD-based generative modeling [Li et al., 2015].

Replacing the energy-distance kernel with a multiscale RBF mixture yielded similar performance across most metrics (Figure 13). Across the 10 energy/RBF model pairs, the absolute relative change in Morgan MMD^2 averaged 2.1% (SEM 0.6%). However, when finetuning on molecular descriptors, the energy-distance kernel achieved better transfer to scaffold-distribution matching. These results support our recommendation of the energy-distance kernel as a default for continuous features, since it requires no bandwidth selection and performs comparably to a multiscale RBF mixture in our experiments. However, problem-specific kernels may still offer improvements.

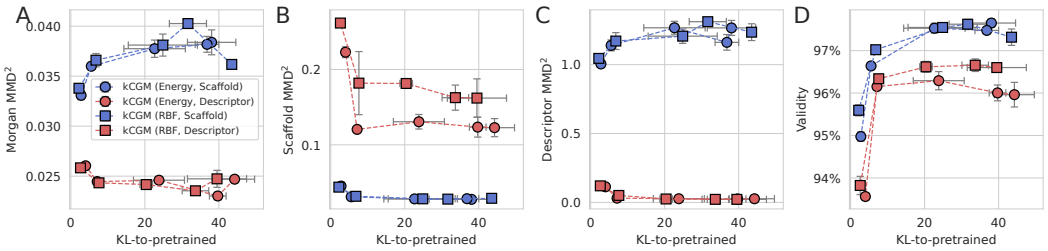

Figure 13: Comparison of `kCGM` for adapting G2PT to the antibiotics set with different kernel choices for scaffold embedding and descriptor features with five values of the KL-regularization weight λ. Panels A–C evaluate distribution matching on Morgan fingerprint, Murcko scaffold, and descriptor features using the MMD^2 metric. Panel D shows the fraction of chemically valid samples. Choice of target features has a much stronger effect than kernel choice.

D.1.3 Effect of leave-one-out baseline and kernel choice on gradient signal-to-noise ratio

Here we investigate how batch size, use of leave-one-out (LOO) baseline, and kernel choice affect gradient signal-to-noise ratio (SNR).

We estimate the gradient SNR at initialization of G2PT finetuning on molecular descriptor features. We define SNR as the squared norm of the expected gradient divided by the total gradient variance. For each batch size M, kernel, and LOO condition, we compute $B = 50$ independent batch-gradient estimates $G_{M,1}, \ldots, G_{M,B}$. We estimate the gradient noise power using

$$\widehat{N}_M = \frac{1}{B-1} \sum_{b=1}^{B} \|G_{M,b} - \bar{G}_M\|_2^2, \tag{26}$$

where $\bar{G}_M = B^{-1} \sum_{b=1}^{B} G_{M,b}$.

Because the gradient estimators are unbiased regardless of batch size and LOO centering, we estimate the signal power using the largest batch size $M_{\mathrm{ref}} = 192$ with LOO centering:

$$\widehat{S}_{\mathrm{ref}} = \|\bar{G}_{\mathrm{ref}}\|_2^2 - \frac{\widehat{N}_{\mathrm{ref}}}{B}. \tag{27}$$

We use this shared signal estimate separately for each kernel and compute

$$\widehat{\mathrm{SNR}}_M = \frac{\widehat{S}_{\mathrm{ref}}}{\widehat{N}_M}. \tag{28}$$

Although the signal and noise power estimators are individually unbiased, their ratio is generally biased.

Figure 14 shows the estimated SNR for batch sizes ranging from 8 to 192 with and without the LOO baseline. We also compare the energy-distance kernel with the multiscale RBF mixture kernel defined in Equation (25). We find that using the LOO baseline gives an increase in SNR equivalent to a 2.15 (0.05 SEM) multiplier on batch size on average and that there is a predictable linear increase in SNR with batch size (Pearson $r > 0.999$ for all four types of runs). The energy and RBF mixture kernels behaved similarly with a slightly higher SNR for the RBF kernel. These results indicate that, in this setting and at the beginning of training, the gradient SNR scales approximately linearly with batch size despite the within-batch kernel interactions.

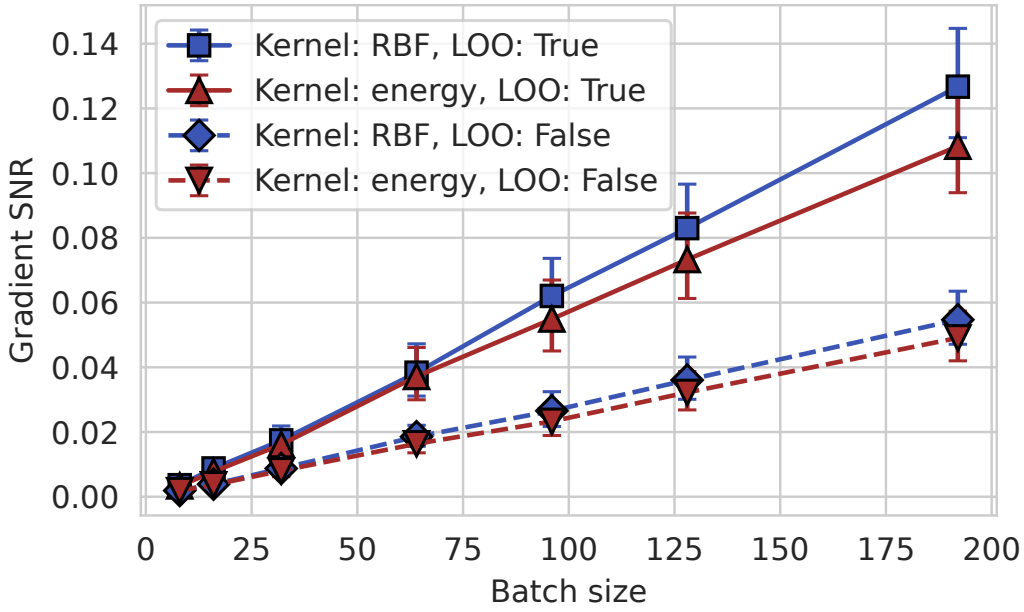

Figure 14: Comparison of `kCGM` gradient SNR with and without LOO baseline and two kernel choices. LOO increases the SNR by a factor of > 2 at a given batch size, while kernel choice has little effect at the measured batch sizes. Error bars indicate approximate 95% confidence intervals computed using a paired jackknife over 50 gradient replicates.

D.2 Genie 2

Figure 10 shows symmetrized KL and energy distance to the target CATH distribution of secondary structure proportions for each `kCGM` (λ, α) combination. Both distribution-matching metrics result in the same ranking of finetuned models.

D.3 DNA discrete diffusion

We additionally evaluate generated DNA samples using a separate prediction model `human-atac-catlas` from the gReLU model family [Lal et al., 2025b], a cell-type-specific accessibility prediction model with a different training dataset and architecture than AlphaGenome. The CATlas model predicts an accessibility probability in 204 cell types from 200 bp of DNA. In order to apply it to the 500 bp sequences in our discrete diffusion experiments, we take the central 200 bp from real and generated sequences and binarize these predictions with a threshold of 0.5. We use these predictions to construct binary vectors with 204 entries for the generated DNA from each topic condition and compute MMDs to the corresponding topic condition in the test set DNA with the Tanimoto kernel.

The best model according to this metric is the `kCGM` AlphaGenome feature model with regularization weight $\lambda = 0.01$, which reduces the CATlas MMD^2 by more than fourfold compared to the pretrained

model (Figure 15). Both `CGM` and `kCGM` generally improve CATlas MMD^2 relative to the pretrained model while neither uniformly dominates across λ. This provides evidence that the improvements are not specific to the AlphaGenome predictor used during finetuning.

As an additional sanity check, for each topic condition and 3-mer, we compute the total variation (TV) distance between the Bernoulli distributions indicating whether that 3-mer occurs in real and generated sequences. We then average these distances across 3-mers and topic conditions. We find that the pretrained model and all `kCGM` models have <0.02 TV distance in 3-mer presence probability from the test set on average, indicating that `kCGM` does not substantially change the 3-mer occurrence probability.

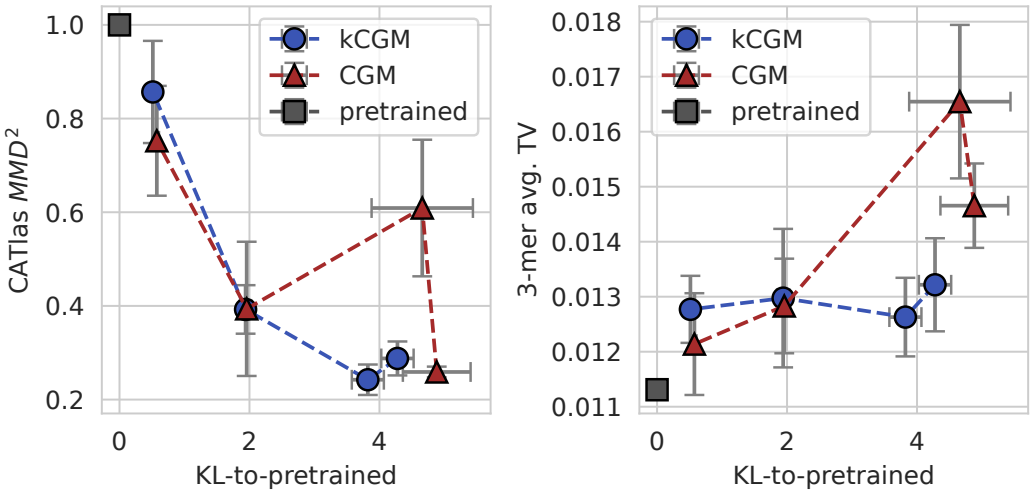

Figure 15: Comparison of `kCGM` and `CGM` finetuned with AlphaGenome features on external distribution matching metrics.

E Example generations

E.1 Genie 2

We include uniform randomly selected samples from the pretrained model in Figure 16, `CGM` with its best λ value in Figure 17, and `kCGM` with its best λ, α combination in Figure 18.

E.2 G2PT

We show example generations from the pretrained model (Figure 23), the target antibiotics set (Figure 19) and from all the other finetuned models at regularization strength $\lambda = 10^{-2}$.

- Figure 20: `kCGM` with Morgan fingerprint features.
- Figure 21: `kCGM` with generic Murcko scaffold features.
- Figure 22: `kCGM` with descriptor features.
- Figure 24: `CGM` with Morgan fingerprint features.
- Figure 25: `CGM` with generic Murcko scaffold features.
- Figure 26: `CGM` with descriptor features.
- Figure 27: Direct finetuning.

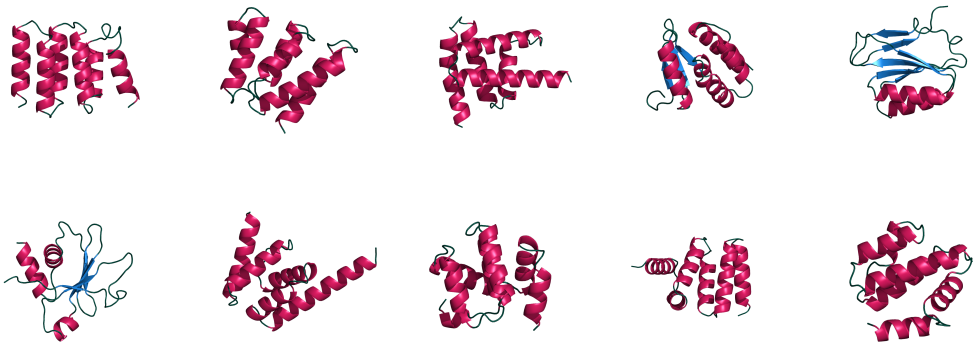

Figure 16: Genie 2 pretrained model samples.

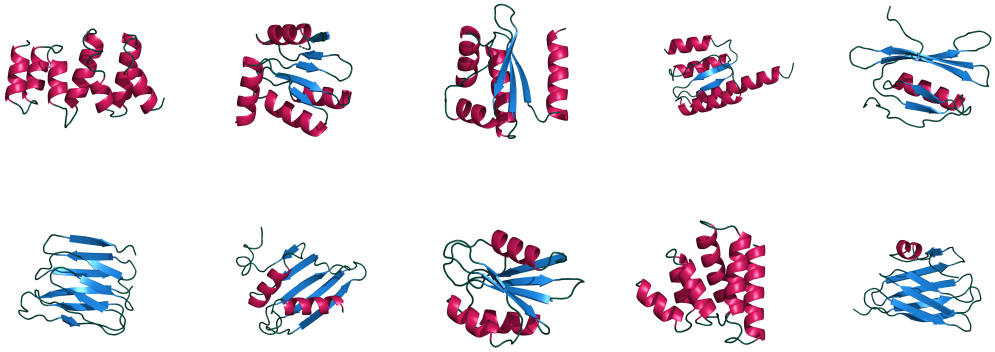

Figure 17: Genie 2 `CGM` $\lambda = 10^{-2}$.

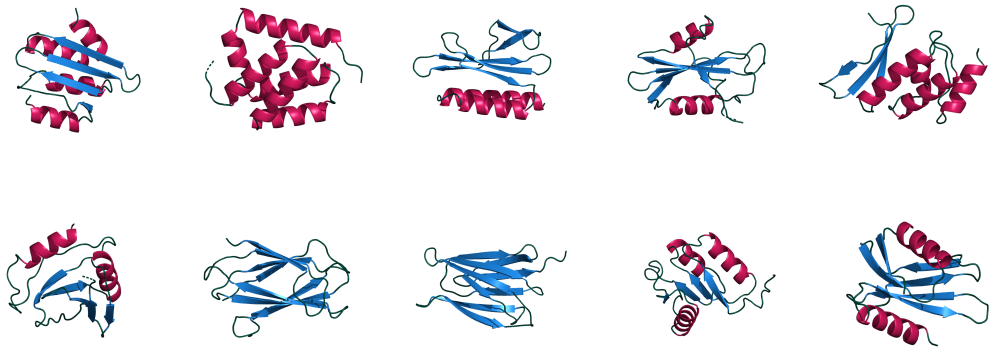

Figure 18: Genie 2 `kCGM` with $\lambda = 10^{-3}$, $\alpha = 0.25$.

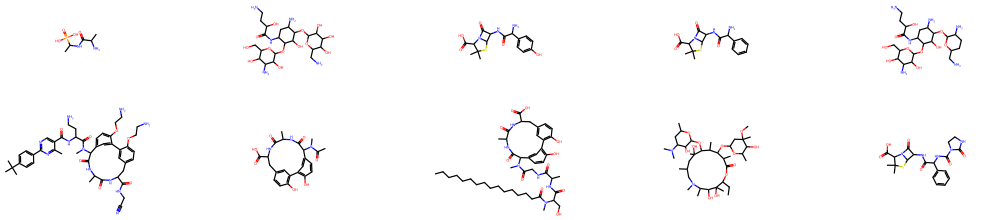

Figure 19: Antibiotics target set.

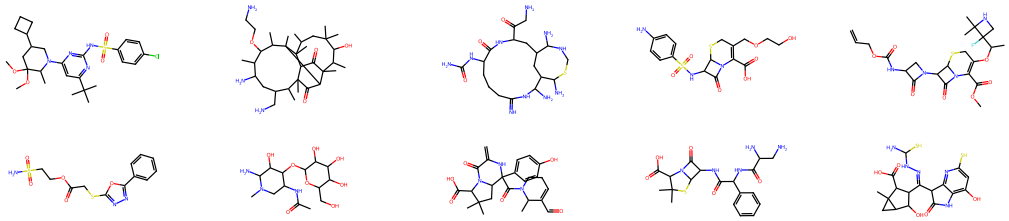

Figure 20: kCGM Morgan-FP tuned G2PT.

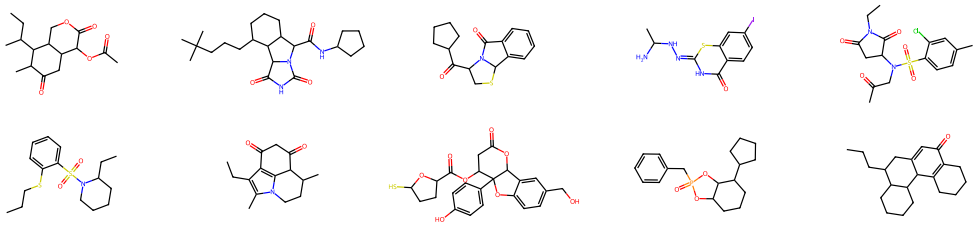

Figure 21: kCGM scaffold tuned G2PT.

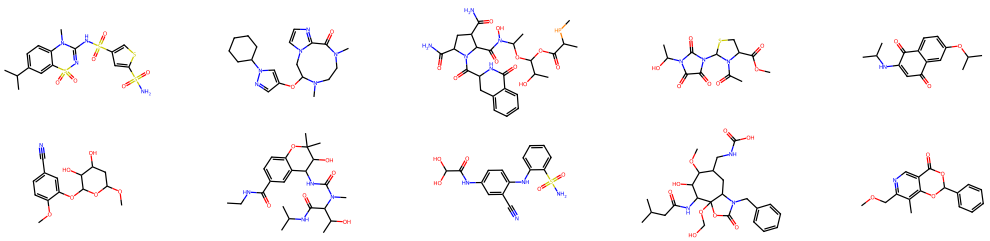

Figure 22: kCGM descriptor tuned G2PT.

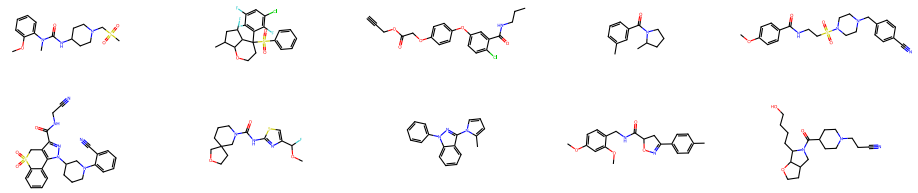

Figure 23: G2PT pretrained model samples.

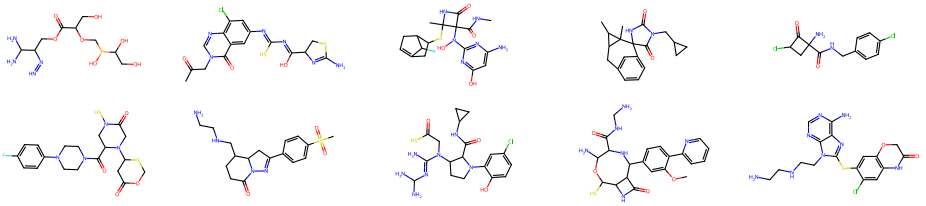

Figure 24: `CGM` Morgan-FP tuned G2PT.

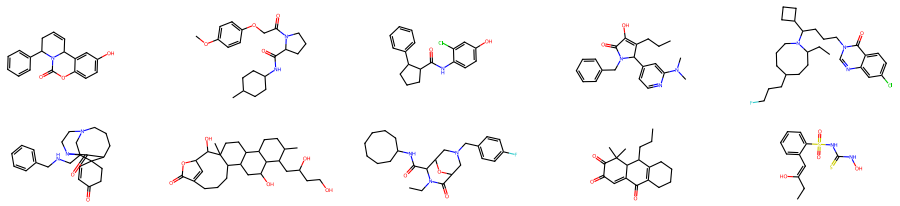

Figure 25: `CGM` scaffold tuned G2PT.

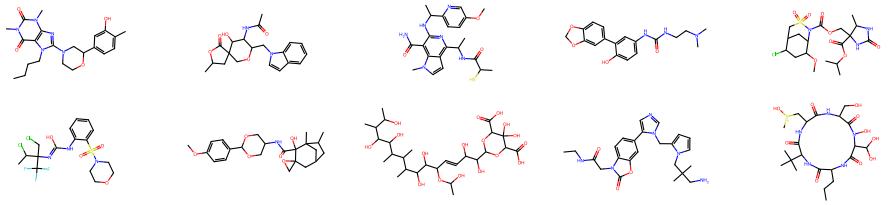

Figure 26: `CGM` descriptor tuned G2PT.

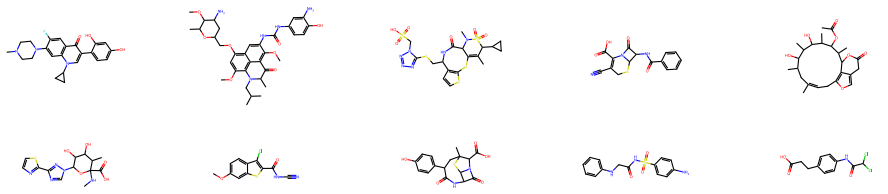

Figure 27: Direct finetuned G2PT.

E.3 Asset licenses

Table 2 lists the licenses for all assets used in our experiments.

Table 2: Licenses and usage terms for external models, datasets, and software used in this study.

Asset	License or usage terms	Source
G2PT framework	MIT	Repository
G2PT pretrained checkpoint	No separate license specified	Model card
Genie 2	Apache 2.0 (repository)	Repository
GuacaMol software	MIT	License
ChEMBL	CC BY-SA 3.0	Database
Antibiotic dataset (Scalia et al.)	No separate data license identified	Supplementary Table 11
CATH	CC BY 4.0 (CATH website)	CATH
Protein Data Bank	CC0 1.0	Usage policy
Biotite	BSD 3-Clause	Repository
RDKit	BSD 3-Clause	License
`fcd_torch`	MIT	Repository
DeepMEL2 enhancer data	GEO reuse policy; no separate data license verified	GEO policy
AlphaGenome PyTorch	Apache 2.0 (implementation); Google AlphaGenome Model Terms (weights and outputs)	Code, Model terms
gReLU	MIT	Repository
CATlas predictor	MIT	Model card
SciPy	BSD 3-Clause	Documentation
PyTorch	BSD 3-Clause (core project)	License

NeurIPS Paper Checklist

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Claims are backed by experiments, citations, and proofs.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the discussion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Derivations with assumptions are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Algorithms and training settings are provided and code is available at `https://github.com/smithhenryd/cgm`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and installation instructions are publicly available at `https://github.com/smithhenryd/cgm`. The paper identifies the pretrained models and datasets used in the experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Described in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars are provided and described in text.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Described in appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the code of ethics and there are no violations.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work develops a general-purpose distribution-calibration method without a direct deployment application. Potential societal impacts depend on the underlying generative models and their downstream uses.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not introduce a high-risk foundation model or release experimentally validated hazardous biological designs. The experiments primarily demonstrate a general-purpose finetuning algorithm.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Relevant papers for model and data are cited. Licenses listed in Appendix Section E.3.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This paper introduces a general finetuning method rather than a new pretrained model or dataset. The implementation is publicly available and documented at `https://github.com/smithhenryd/cgm`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: No non-standard usage according to the handbook.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.