Guided Data Generation for Understanding Model Behavior

E. Mehmet Kıral
RIKEN AIP, Japan
eren.kiral@riken.jp

Nurşen Aydın
Warwick Business School, UK
nursen.aydin@wbs.ac.uk

Ş. İlker Birbil
University of Amsterdam
s.i.birbil@uva.nl

Abstract

We propose a method for generating distributions over the input space as an inspection tool for understanding trained models. Our framework poses questions of the form "which inputs would make a trained model exhibit a specified behavior?" and encodes each question through a guidance function. The guidance function induces a Gibbs distribution over the input space, from which we generate samples. The generated data provide insights into how the models behave. To showcase our framework, we pose queries such as generating distributions of data where a specified label would be predicted by the model, where two distinct models would disagree, where the output is sensitive to parameter perturbations, and where the model is uncertain. Our method can be applied to a variety of classification and regression tasks and across a range of model types.

1 Introduction

Machine learning models are widely used in today's data-driven world, powering critical decision-making processes in sectors ranging from healthcare to human resources. Their widespread adoption in high-stakes scenarios raises important questions about how trained models behave and whether their decisions align with human values. Understanding how these models operate has become a critical concern. Our quest along this line starts with the following inquiry: *What kind of data can we generate to understand the behavior of trained models?*

To respond to this inquiry, we study the implicit data distribution of trained models under various scenarios. In other words, our approach to understanding a model is based on creating samples in the data domain that the trained model considers favorable for a specific task. Unlike conventional ML pipelines that focus on static datasets and predictive accuracy, our approach enables dynamic interrogation of model behavior via investigating the answers of the model to specific questions. We show that these questions can be customized to each situation and expressed mathematically through a loss function that evaluates data based on a combination of data characteristics and model parameters. We consider the problem of understanding a model to be a more nuanced endeavor that requires exploration across multiple dimensions. Rather than focusing on a single form of explanation, our framework supports a range of guided data generation objectives that reveal different aspects of model behavior. For instance, insights into model behavior can be gained by generating parameter-sensitive data samples. When two models showing similar performances give different predictions, also known as predictive multiplicity [28], our approach can be used to generate data to systematically compare their behaviors in diverse scenarios. These questions, and other custom guidance functions which can be designed by the users, provide a qualitative understanding of the model. The users can then form their hypotheses on how the models function, and quantitatively test them.

We begin by asking a high-level question about a trained model. This question defines a scenario of interest, which we encode through a guidance function on the data space. The guidance function induces a distribution that places greater probability on inputs exhibiting the specified behaviour,

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

and we use a sampling method such as MALA to generate samples from this distribution. We then inspect the generated data to understand how the trained model behaves under the specified scenario. In this work we focus on developing this methodology, how it can be deployed, and the generality of its application.

To highlight the potential downstream uses of the methodology, consider a model for electricity-demand prediction or capacity planning. Our method can generate input scenarios under which predicted demand reaches a specified, potentially undesirable, level. The guidance function provides a targeted mechanism for exploring such critical scenarios without an exhaustive grid search. As another example, using contrastive data generation, we can identify data points where two otherwise well-performing models disagree. This can reveal regions of the input space that are poorly represented in the available data. If these regions are practically relevant, they could motivate targeted data collection or active-learning strategies to improve model reliability there. The generated distributions may also reveal potential biases reflected in the models' behavior, for example when disagreement is concentrated on a protected class or another policy-relevant subgroup. Such complete downstream application pipelines are beyond the scope of the present paper. In this initial work, we focus on introducing the methodology for generating targeted data distributions under fairly generic scenarios and across multiple model types.

We nevertheless demonstrate one complete diagnostic pipeline in our numerical experiments. Given a pretrained ResNet50 model, we generate images that the model classifies as goldfish (Figure 6). The generated images consistently contain the color orange, suggesting the hypothesis that color is an important cue for the model's goldfish predictions. We then test this hypothesis independently of the generation process by evaluating the model on genuine goldfish validation images after grayscale and color-channel transformations. This illustrates how guided generation can lead from model probing to hypothesis formulation and independent validation.

Related Literature. Our work complements extensive research in synthetic data generation that has been pivotal in addressing fairness, bias detection [19] and reduction [35, 32] as well as dataset augmentation [34, 8]. Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have been widely used to approximate data distributions [13, 35, 18, 4], focusing on privacy, diversity, and fidelity as primary goals.

Energy-based models (EBMs) have also emerged as a promising framework, combining generative and discriminative modeling tasks. By treating classifier logits as an energy function, EBMs can model joint distributions over data and labels [20, 7]. Applications of EBMs include adversarial robustness, out-of-distribution detection, and data augmentation [37, 24, 2, 26]. For instance, [7] demonstrated improved out-of-distribution detection using a joint energy-based model, while [25] extended EBMs to tabular data for synthetic data generation. Using data samples to draw out and understand the implicit preferences of humans [30, 14] and of machines [38] has been tried out before.

The proposed framework draws inspiration from these works while introducing a distinct perspective. Our guidance function can be viewed as an energy function and leads to a Gibbs distribution. However, rather than learning the energy function to capture the data distribution, possibly conditioned on labels, we construct a guidance function using trained models. This design allows the resulting distribution to generate samples that address a specific behavioral question. Related works, such as [7] and [25], adopt a similar approach by utilizing a trained classifier to obtain an energy function and using Langevin dynamics for sampling from the Gibbs distribution. However, their main objective is to mimic the true data distribution. In fact, the former paper combines training of the energy function and classifier. In contrast, we propose a flexible framework for guided data generation that allows users to direct diverse questions to trained models through guidance functions reflecting different objectives, such as identifying prediction-risky, parameter-sensitive, or model-contrastive data samples.

Contributions. We introduce a new inductive approach that generates data samples through a flexible guidance function designed to analyze and reveal the behavior of a trained model. Our method can be tailored to suit various classification and regression tasks, demonstrating its versatility in producing data that meet specific queries. This work serves as a foundational step in establishing the effectiveness and potential of for understanding model behavior.

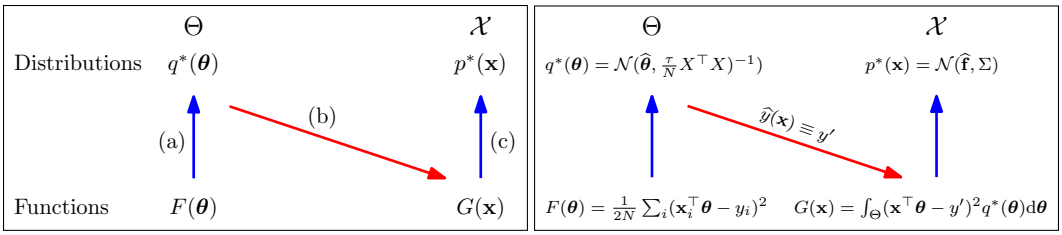


Figure 1: (Left) Overview of guided data generation for understanding model behavior. Samples from $p^*(\mathbf{x})$ answer the question posed by G. The vertical arrows (a) and (c) start with functions and lead to distributions on the same space by solving equation 2 and equation 3. The diagonal arrow (b) starts with a distribution on the parameter space and obtains a loss function on the data space by integrating out $\boldsymbol{\theta}$ dependence of a function on $\Theta \times \mathcal{X}$ against a distribution $q(\boldsymbol{\theta})$. (Right) The special case of the Linear Regression (LR) model with mean square error admits an analytic solution. The G function is designed to find data points $\mathbf{x}$ whose solutions are close to a chosen prediction y' under the trained distribution of linear models. The distribution $p^*(\mathbf{x})$ is calculated to be a Gaussian.

2 The Mathematical Framework

First, we present our notation. The labeled data lie in $\mathcal{X} \times \mathcal{Y}$, and the model defines a predictor function $f(\boldsymbol{\theta}, \cdot) : \mathcal{X} \to \mathcal{Y}'$ for any given set of model parameters $\boldsymbol{\theta} \in \Theta$. For a given sample $\mathbf{x} \in \mathcal{X}$, the predicted label $y_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathcal{Y}$ is obtained from the predictor function. The cost function $\ell_F : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ measures the distance between the predicted and true labels.

The standard construction of the parameter loss function is

$$F(\boldsymbol{\theta}) = \int_{\mathcal{X}\times\mathcal{Y}} (\ell_F(y_{\boldsymbol{\theta}}(\mathbf{x}), y) + R_F(\boldsymbol{\theta})) \mathrm{d}\nu(\mathbf{x}, y) = \frac{1}{N} \sum_{i=1}^{N} \ell_F(y_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + R_F(\boldsymbol{\theta}), \tag{1}$$

which can be seen as an integral of $\ell_F + R_F$ against the empirical distribution given by the training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N \subseteq \mathcal{X} \times \mathcal{Y}$. Here, $R_F(\boldsymbol{\theta})$ is a regularizer term that depends only on $\boldsymbol{\theta}$.

Figure 1 provides an overview of our framework. Just as the training process, which uses F to find the right parameters, our framework uses a guidance function G defined on the data space to generate samples that reveal model behavior. In the variational setting, the symmetry is clear, where we get a distribution over the parameters (or data) instead of a single $\boldsymbol{\theta}^*$ (or $\mathbf{x}^*$). The loss function F in equation 1 is an average over observed data, and, similarly, we construct the data loss G by integrating out $\boldsymbol{\theta}$ from a curated function (described later) depending on both data features and model parameters. We design this function of data and parameters to attain low values when the desiderata of our guided generation objective are met. Different choices of G functions pose different questions to the model with answers given in terms of samples from the data space. Observing the generated data, we gain valuable insights, both qualitatively and through population-level statistics.

The blue arrows (a) and (c) in Figure 1 map (loss) functions on the respective spaces Θ (parameter space) and $\mathcal{X}$ (data space) to distributions over the same spaces. This corresponds to solving the Bayesian Learning Problem (BLP), which—in the case of (a)—is

$$\arg\min_{q \in \mathcal{Q}} \mathbb{E}_q[F] - \tau \mathcal{H}(q), \tag{2}$$

where $\mathcal{Q}$ is a choice of candidate distributions on Θ, and $\mathcal{H}(q) = -\int_\Theta q \log q \, \mathrm{d}\mu$ is the entropy with respect to a base measure μ. The problem can be interpreted as an implementation of the exploration-exploitation trade-off in the parameter space. The constant $\tau > 0$ is called the temperature and balances these two objectives. If $\mathcal{Q}$ is the set of all density functions, then the Gibbs-Boltzmann distribution $q^*(\boldsymbol{\theta}) \propto e^{-\frac{1}{\tau}F(\boldsymbol{\theta})}$ is the unique solution to equation 2.

Symmetrically on the data space $\mathcal{X}$, the blue arrow labeled (c) in Figure 1 represents solving

$$\arg\min_{p \in \mathcal{P}} \mathbb{E}_p[G] - \tau \mathcal{H}(p). \tag{3}$$

The distribution $p^*(\mathbf{x}) \propto e^{-\frac{1}{\tau}G(\mathbf{x})}$ is its global solution, balancing the expectation term's effect of mass concentration at low G-values, with the entropy term's effect of exploring the data space.

We can analytically solve for q^* and p^* for the case of the Least Squares regression problem (Figure 1, right), which results in a Gaussian distribution.

Theorem 1. *Given a linear model $y_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$ and a loss criterion $\ell_F(y, y') = \ell_G(y, y') = \frac{1}{2}(y - y')^2$, the full Bayesian posterior, i.e., the solution q^* to equation 2 is a Gaussian distribution with mean equal to the Ordinary Least Squares (OLS) solution $\hat{\boldsymbol{\theta}}$ and covariance $(D^\top D)/(N\tau)$ where D is the design matrix of the OLS. Furthermore using the G function obtained from equation 4 using the full Bayesian posterior $q = q^*$ leads to a Gibbs distribution that is Gaussian on the data space with mean $\hat{\mathbf{f}}$ and covariance Σ given by equation 9 and equation 10, respectively.*

See Appendix A for the proof and explicit forms of $\hat{\mathbf{f}}$ and Σ.

There are various methods of sampling from equation 3. In this work, we used Metropolis Adjusted Langevin Algorithm (MALA) to sample directly from the Gibbs-Boltzmann distribution $p^*(\mathbf{x}) \propto e^{-\frac{1}{\tau}G(\mathbf{x})}$. This method is a kind of noisy gradient descent, with an acceptance/rejection step ensuring that the limiting distribution is $e^{-\frac{1}{\tau}G(\mathbf{x})}$. Details of this method are given in Appendix C. There are alternative sampling methods such as Hamiltonian Monte Carlo variant No U-Turn Sampler (NUTS) [15]. In another direction, Variational Inference restricts the search to a statistical manifold $\mathcal{P}$, and effectively reaches a distribution $p^* \in \mathcal{P}$; see [10, 11]. For example, if $\mathcal{P}$ were the Gaussians, then only keeping track of the mean and the covariance would be sufficient to learn $p^* \in \mathcal{P}$.

The red arrow (b) constructs the function G as an integral of a function over both the data and parameter spaces, obtained by integrating out $\boldsymbol{\theta}$ with respect a measure $q(\boldsymbol{\theta})$, in direct analogy with the construction of F in equation 1. This construction of F is an integral against the empirical data distribution and, therefore, the learned parameters are compatible with the training data. Analogously, constructing G ensures that the search over $\mathcal{X}$ remains compatible with parameters sampled from $q(\boldsymbol{\theta})$. In particular, if $q(\boldsymbol{\theta})$ is a is a Dirac-delta distribution, this reduces to a single parameter vector $\boldsymbol{\theta}^* \in \Theta$. The specific choice of the integrand for G determines which samples from $p^*(\mathbf{x})$ are the data points that answer a question posed about the trained model.

The quality of the generated data points can be assessed in two ways. First, one can examine whether the generated points satisfy the property for which the function G was designed. This can be verified directly by evaluating the generated points under the trained model.

Second, one can assess how well the MCMC method samples from the target Gibbs–Boltzmann distribution induced by a given G function. This is a well-studied problem, and several diagnostic statistics are available for detecting discrepancies. One such approach is the Kernelized Stein Discrepancy (KSD), which tests whether the generated samples are consistent with the target Gibbs distribution. The idea was introduced by [23], who proposed using the Stein discrepancy measure $\mathbb{S}(p, q)$, associated with a reproducing kernel Hilbert space, as a goodness-of-fit test for a given empirical distribution and a possibly unnormalised density. Additional MCMC diagnostics can also be used. Effective Sample Size (ESS) adjusts the nominal length of an MCMC chain by accounting for autocorrelation across lags. When multiple chains are available, the Gelman–Rubin statistic, also known as $\hat{R}$, compares between-chain and within-chain variability. If the chains have mixed well, these should agree giving $\hat{R} \approx 1$ [33]. Implementations of these diagnostics are available in open-source packages such as `divergence` [29] and `arviz` [27]. These diagnostics also guide our choice of hyperparameters, including the step size and the length of the sampling chain.

Lastly, note that we can change our search space by replacing G with $G \circ \varphi$ for some $\varphi : \mathcal{Z} \to \mathcal{X}$. In high-dimensional data spaces $\mathcal{X}$, we will use this setup with $\mathcal{Z}$ as the latent space and φ as the decoder function of a pre-trained Variational AutoEncoder (VAE). In this case, $p^*(\mathbf{z})$ becomes a distribution on $\mathcal{Z}$, and mapping its samples to $\mathcal{X}$ by φ, gives points on the data manifold. Using a pre-trained VAE for image models reduces the data space dimension, and therefore, the efficiency of the MALA sampling process, but there are also conceptual benefits. There are vast regions of the high-dimensional input space that do not correspond to plausible images and which the model did not encounter during the training process. Some of these regions may also have low G values. Therefore, using a pre-trained VAE decoder reflects a plausible image requirement, if imposed.

3 Guided Data Generation for Trained Models

We start with a general structure of the loss function curated for questioning trained models:

$$G(\mathbf{x}) = \int_{\Theta} \ell_G(y_{\boldsymbol{\theta}}(\mathbf{x}), \widehat{y}(\mathbf{x}))q(\boldsymbol{\theta})\mathrm{d}\,\boldsymbol{\theta} + R_G(\mathbf{x}), \tag{4}$$

where $\widehat{y}$ stands for a predictor and R_G is a regularizer function that can be chosen to put additional soft constraints on the generated samples, and q is a distribution on the parameter space (could be a posterior distribution $q = q^*$ coming from the Bayesian problem equation 2, or a Dirac-delta distribution coming from classical, i.e., non-variational training). This general guidance function enables us to express a wide range of model-inspection tasks. Theoretically, a choice of G-function is equivalent to a probabilistic model $p(\mathbf{x}|q)$ on the data, where $G(\mathbf{x})$ is the negative $\log$ likelihood of sampling $\mathbf{x}$ given the model distribution q (which can be a Dirac delta at $\boldsymbol{\theta}^*$). But our probabilistic model can be more general than $y_{\boldsymbol{\theta}}(\mathbf{x})$ predicting the true label with a given noise model, as we see in the examples below. The best conceptualization of our method is that of an energy based model, where instead of learning the energy function from a data distribution, we start with an energy function G constructed using a model that has been trained (on the data distribution). With various choices of G as a guidance function, we aim at revealing the behavior of the model.

In the remainder of this section, we walk through several representative cases, each capturing a specific type of question one might ask about a trained model's behavior. These are not exhaustive, but illustrative scenarios that demonstrate the flexibility of the proposed framework. Figure 2 showcases what sort of data points would be produced in a synthetic dataset.

In many cases, we choose the regularizer term as $R_G(\mathbf{x}) = \lambda\|\mathbf{x} - \mathbf{x}_a\|_r^r$ for $r \geq 1$, so that the generated synthetic data is localized to an anchor $\mathbf{x}_a$. The parameter $\lambda > 0$ explicitly adjusts the trade-off between the guidance objective and localization strength. In fact, different weights can be applied to different columns to enforce this constraint more or less stringently for different features.

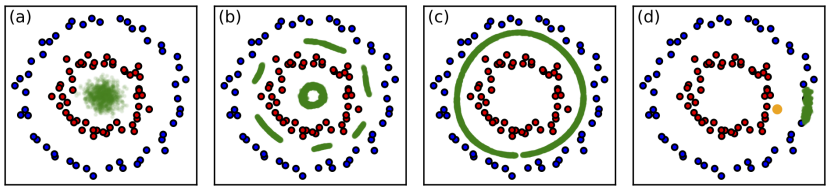


Figure 2: Given a dataset of two concentric circles labeled red and blue, two Support Vector Machine (SVM) models are trained on the binary classification task with kernels chosen as Radial Basis Function (RBF) and cubic polynomial, respectively. Generated data points are in green. In (a), we contrast the two SVM models, generating samples where the predictions differ, and discover that this is the case in a region near the origin lacking any training points. In (b) and (c), we inquire about data points that would be considered risky by the two models using RBF and cubic kernels, respectively. In (d) we design G so that it generates data points which are classified with the opposite label of the orange point by the RBF-SVM without straying too far from it.

Model-contrasting samples. Given two models, finding data points where their predictions disagree is illuminating either to interpret model-specific biases or to audit consistency between two different models. This is particularly useful when comparing models with different inductive biases (MLP vs. CNN, linear vs. nonlinear, and so on). Given $\widehat{y}$ as the predictor function of the model being compared against, which can be non-parametric, as in boosted trees, we solve equation 3 with the function

$$G(\mathbf{x}) = \ell_G(y_{\boldsymbol{\theta}^*}(\mathbf{x}), 1 - \widehat{y}(\mathbf{x})) + R_G(\mathbf{x}). \tag{5}$$

In Figure 2(a), we contrast two SVM models with different kernels and discover a region near the origin, not containing any samples from the dataset but would give conflicting predictions if a new sample were to come from it.

Prediction-risky samples. To identify the indecisive regions in a model's decision surface, we either put $G(\mathbf{x}) = \|f(\mathbf{x}, \boldsymbol{\theta}^*) - \alpha\|_r^r$ for $r \geq 1$, where f may be the decision function of a binary classification model such as in SVM and logistic regression and α denotes the cutoff point, or given a multi-class classification where f gives the prediction probabilities, we set $G(\mathbf{x}) = -H(f(\mathbf{x}))$,

the negative entropy. Solving equation 3 thus corresponds to generating "risky data points" near the decision boundary. These samples provide insight into whether the model identifies important aspects of the data for decision-making and whether this behaviour aligns with the user's expectations. As an illustrative example in Figure 2(b), we search for the decision boundary of a RBF-SVM trained on a dataset of two concentric circles, and discover a central ring as an unexpected decision boundary.

Parameter-sensitive samples. Given a set of parameters $\boldsymbol{\theta}^*$ and a distribution $q^*(\boldsymbol{\theta})$ of parameter values, we ask the model for data samples whose classifications would flip if the model parameters were to be (perhaps slightly) perturbed. This can be achieved by solving equation 3 using

$$G(\mathbf{x}) = \int_{\Theta} \ell_G(y_{\boldsymbol{\theta}}(\mathbf{x}), 1 - y_{\boldsymbol{\theta}^*}(\mathbf{x}))q(\boldsymbol{\theta})\mathrm{d}\,\boldsymbol{\theta} + R_G(\mathbf{x}). \tag{6}$$

This integral is approximated by samples from q. When q is chosen from a restricted family of distributions, e.g. fixed variance Gaussians, sampling from $q = \mathcal{N}(\boldsymbol{\theta}^*, \sigma)$ means perturbing $\boldsymbol{\theta}^*$. This guidance function is particularly useful for examining model prediction's consistency under small parameter shifts. By identifying inputs whose predictions vary significantly with minor parameter changes, we highlight sensitive regions in the input space that might indicate over-dependence on specific parameter configurations. This can overlap with, but is distinct from, prediction-risky samples, as we demonstrate in our computational study section. Parameter-sensitive samples could be generated far from the decision boundary, especially in non-linear models.

In equation 5, we formulated the guidance function for the binary case. For regression models, choosing $G(\mathbf{x}) = \int_{\Theta} \exp\left(-\|y_{\boldsymbol{\theta}}(\mathbf{x}) - y_{\boldsymbol{\theta}^*}(\mathbf{x})\|^2/\sigma^2\right) q^*(\boldsymbol{\theta})\mathrm{d}\,\boldsymbol{\theta} + R_G(\mathbf{x})$ yields lower values when prediction differences are large (with large measured using a yardstick of size σ of our choosing). A similar construction applies to equation 5. As for multi-class prediction, we can also use this G, with $y_{\boldsymbol{\theta}}(\mathbf{x})$ representing the (post-softmax) class probabilities.

Fixed-label samples. Finally, we probe the model for what it thinks are good data samples that fit the bill for the prediction y', either for a single parameter $\boldsymbol{\theta}^*$,

$$G(\mathbf{x}) = \ell_G(y_{\boldsymbol{\theta}^*}(\mathbf{x}), y') + R_G(\mathbf{x}) \tag{7}$$

or for a distribution $q^*(\boldsymbol{\theta})$ of parameters

$$G(\mathbf{x}) = \int_{\Theta} \ell_G(y_{\boldsymbol{\theta}}(\mathbf{x}), y')q^*(\boldsymbol{\theta})\mathrm{d}\,\boldsymbol{\theta} + R_G(\mathbf{x}). \tag{8}$$

Here, $R_G(\mathbf{x})$ is a localizer at an anchor point. We can take a data point $(\mathbf{x}_0.y_0)$ to be this anchor.

Figure 1 demonstrates the steps when $y_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^\top\boldsymbol{\theta}$ corresponds to linear regression, and both ℓ_F and ℓ_G are the mean squared errors. For this special case, we obtain analytical solutions for all steps of our framework. The details of this observation are given in Appendix A.

Feature-restricted samples. By restricting $\mathcal{P}$ to be supported on data with certain features fixed, such as those features corresponding to age, race, and so on, we can ask the model for all of the above questions but conditioning on certain fixed characteristics. This falls into the class of optimizations, where instead of G we consider $G(\varphi(\boldsymbol{z}))$ on some other (latent) space $\boldsymbol{z} \in Z$. In case of image data, for example, to have our samples conform to the data manifold, φ can be taken as the trained decoder module from a VAE. Pushforwards $\varphi_*\tilde{p}$ of measures $\tilde{p} \in \mathcal{P}(Z)$ on the latent space then lie on the data manifold, i.e., sampling $\boldsymbol{z} \sim \tilde{p}$ and computing $\varphi(\boldsymbol{z})$ gives a data sample. See Figure 4 for this method in a concrete application.

4 Computational Study

In this section, we conduct a series of experiments to evaluate the cases presented in Section 3. Our experiments aim to evaluate the proposed framework by demonstrating its ability to generate data samples across various scenarios. We use well-established datasets that have been recently adopted in related literature (e.g., [12, 22, 31]), whose specifics are outlined in Appendix B. Implementation details and code for reproducing these experiments are available on our GitHub repository.[1]

Model-contrasting samples. This experiment investigates differences between two predictive models by generating data samples for features that drive contrasting predictions for the same data. Through our framework, we pose the following question:

[1] https://github.com/sibirbil/GuidedDataGeneration

Which features or input changes lead to disagreement between the predictions of two models?

To explore this, we apply the framework to datasets of different modalities. For tabular data, we use the FICO dataset [9]; for image dataset, we use MNIST [21].

We begin by investigating model divergence in scenarios where the comparison model is non-differentiable. To this end, we train XGBoost [5]–a non-parametric model–alongside logistic regression on the FICO dataset, which consists of credit applications with features related to financial history and risk performance. This setup highlights the flexibility of our framework, as it enables the analysis of differences between models with fundamentally distinct modeling approaches. The two models have comparable predictive performance on the test set (71.8% accuracy for XGBoost and 70.5% for logistic regression) and agree on 94.5% of the test observations. Thus, under standard test-set evaluation, disagreement is observed in only 5.5% of the samples. In contrast, our framework generates a distribution in which the models disagree on 90.1% of the samples, i.e., XGBoost predicts one class while logistic regression predicts the opposite. This contrast demonstrates how guided generation can systematically identify regions of model disagreement that are rarely encountered when sampling from the original test distribution. The details of the experiment are in Appendix D, together with diagnostic statistics such as KSD. Figure 3 presents the distributions of three representative features for the generated model-contrasting samples and the test data. The generated samples differ noticeably from the test distribution across all three features. For `NumTotalTrades` (total number of trades), the generated samples are concentrated at lower values. The largest difference is observed for `MSinceMostRecentTradeOpen` (months since the most recent trade was opened), where the generated samples extend substantially toward larger values while the test data are concentrated near the lower end of the range. Similarly, `NumTradesOpeninLast12M` (number of trades opened in the last 12 months) is shifted toward higher values and exhibits a longer right tail in the generated samples. This indicates that our method can surface disagreement patterns that would remain undetected through standard evaluation alone. We also include in Appendix D a comparison between a linear model and a Support Vector Regression (SVR) model using a different tabular dataset.

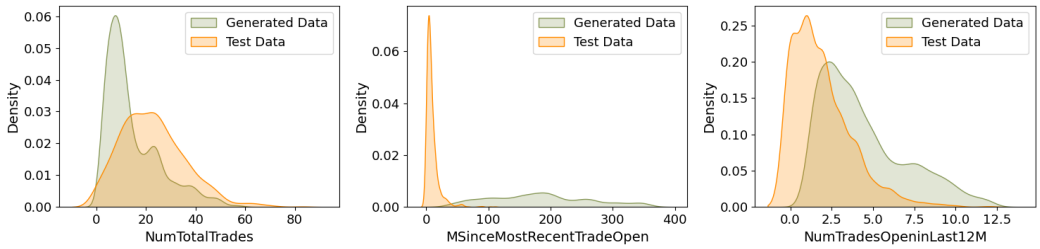


Figure 3: The distributions of three representative features in the generated samples. Here, XGBoost predicts "Bad" for `RiskPerformance`, while logistic regression predicts "Good".

Our framework can also be used to compare and contrast two models trained on image data. To demonstrate this, we consider a Convolutional Neural Network (CNN) and an MLP, both trained on MNIST. The architectures of these networks are provided in Appendix B.2. To better capture the data manifold, we also train a VAE with a latent dimension of 10. The trained encoder module of the VAE is denoted by $z \mapsto \varphi(z)$. Further details on the VAE training process are provided in Appendix E. In Figure 4, we present an example computation illustrating how this setup works. Starting with a latent vector encoding an image with label '3', we sample from a distribution that prefers the label '8' jointly for both a trained CNN (LeNet5) and an MLP.

We use this setup to systematically compare the CNN and MLP models. In Figure 4, we showcase some samples generated by forcing functions G that pull the data toward incompatible directions, for example, resulting in amorphous data points that exhibit characteristics of both '1' and '0'. The third column highlights cases where the label '8' is preferred (top: MLP, bottom: CNN) while remaining close to an actual MNIST image labeled '3', which is enforced through two-norm regularization.

Prediction-risky samples. As our first example, we train an MLP on the FICO dataset to classify credit risks as "Good" or "Bad." Prediction-risky samples are those for which the model outputs softmax probabilities close to 0.5, reflecting high uncertainty. A detailed analysis of this experiment, along with the generated samples, is provided in Appendix D. For example, we see in Figure 11

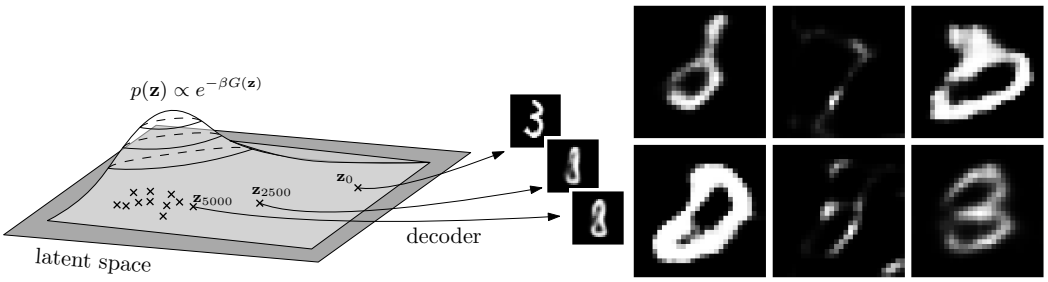


Figure 4: (Left) Using Langevin dynamics in the latent space, we obtain a sequence of latent vectors that, when passed through the decoder φ, correspond to a walk on the data manifold. In this image, the function G is the sum of cross-entropy predictions of trained MLP and LeNet5 networks for the label '8' and for the data $\varphi(z)$. (Right) Images in the first and second columns are generated to prefer a given label on an MLP model and another one on a CNN model. upper-left: CNN-'0' MLP-'1', upper-middle: CNN-'1' MLP-'7', lower-left: CNN-'0' MLP-'8', lower-middle: CNN-'2' MLP-'5'. On the third column, the upper image prefers the label '8' for the MLP model whilst being close to a data sample with label '3', and the same for the lower image for the CNN model.

that the model assesses between three-to-five delinquencies (`MaxDelqEver`) as being on the edge between being a bad credit risk versus a good credit risk. A domain expert can then assess whether this aligns with expectations about the model's behaviour.

As another example, to demonstrate the versatility of our approach, we apply it to tree-based models. Using these models poses a challenge due to the locally constant nature of their prediction functions, and hence, one cannot directly use the gradient-based methods. However, as described in Appendix C.2, approximate gradient information can still be leveraged effectively to overcome this limitation.

Using the wine dataset from `scikit-learn`, we train a Random Forest (RF) classifier. This dataset is designed for classification tasks and consists of numerical features that describe various characteristics of wine, such as hue and alcohol content. The target variable represents the wine's region of origin, which falls into one of three distinct classes. We ask the following question:

Which input samples drive the RF classifier to produce nearly uniform class probabilities?

To highlight the flexibility of our method, we also impose a regularizer that encourages that a Decision Tree (DT), trained on the same dataset, to predict a given region with certainty. This can be achieved by letting $G(\mathbf{x})$ to simultaneously maximize the entropy of the RF's prediction probabilities (encouraging uncertainty) and minimize the distance between the DT's prediction probabilities and a fixed one-hot vector, thereby enforcing certainty on a chosen class. Both models have high accuracy on the validation set (RF: 94.4%, DT: 88.8%). Therefore, the generated data's features necessarily lie outside the empirical data distribution. We generated 50 data points such that the DT predicts `class_1` with full certainity, and the RF's prediction probabillites are $(0.31_{\pm 0.03}, 0.4_{\pm 0.06}, 0.29_{\pm 0.06})$. See Table 1 in Appendix D to compare the feature values of this generated wine feature dataset versus those from each of the three regions. Following the decision path of the DT, we observe that the generated samples are identified as belonging to `class_1` solely based on their color intensity. In contrast, the Langevin process resulted in a set of wine features for which the Random Forest remains uncertain across the three classes.

Parameter-sensitive samples. This experiment investigates data samples that are sensitive to small perturbations in the model parameters. Unlike prediction-risky samples, parameter-sensitive samples may exist anywhere in the input space, as their classification changes with slight shifts in the model's parameters. To guide this analysis, we pose the following question:

What kind of data samples vary in classification due to small changes in model parameters?

We train an MLP on the FICO dataset and generate parameter-sensitive samples by perturbing the model parameters using a Gaussian distribution centered at the original weights with fixed variance. Using the guidance function in (6) with ℓ_G given by the cross entropy loss, we generate and analyze such samples to identify instances most susceptible to model variation, and compare them with

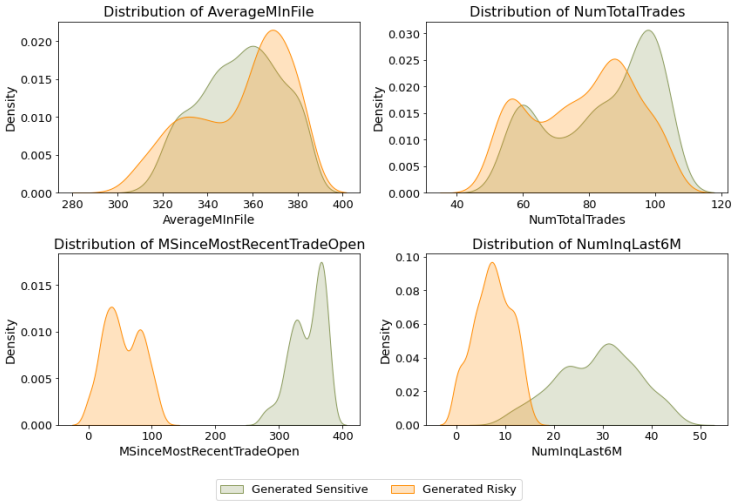


Figure 5: Feature distributions in generated parameter-sensitive and prediction-risky samples.

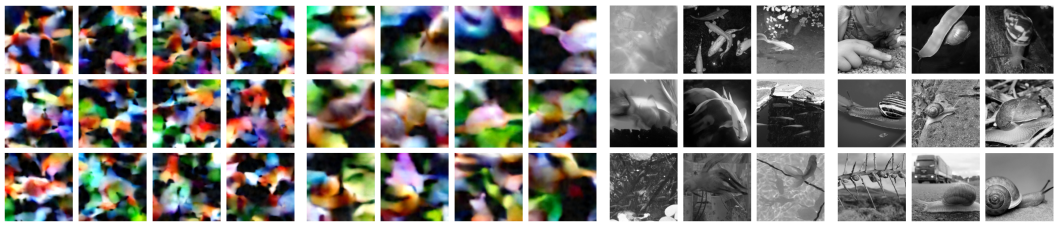
prediction-risky samples. We run four MCMC chains with 10k data points each. More details about the data generation and the statistics of the generated samples are given in Appendix D.

Figure 5 shows density plots of four representative features (see Appendix D for additional ones), comparing parameter-sensitive and prediction-risky samples. By comparing these two distributions, we gain insights into how the model perceives uncertainty from different perspectives. While prediction-risky samples are associated with uncertainty near the decision boundary, parameter-sensitive samples highlight regions in the feature space where small parameter changes can flip predictions. The features `AverageMinFile` (average observation period) and `NumTotalTrades` (total number of trades) exhibit similar distributions across both sets. In contrast, the features `MSinceMostRecentTradeOpen` (months since most recent trade) and `NumInqLast6M` (inquiries in the past six months) diverge. For example, `NumInqLast6M`, which signals recent credit-seeking activity, is lower among prediction-risky samples, indicating that individuals with fewer recent inquiries are more likely to fall near the decision boundary. In contrast, parameter-sensitive samples exhibit a broader distribution, indicating that parameter shifts affect individuals across a wider range of credit inquiry patterns. This may be because frequent inquiries reflect diverse financial behaviors, making these samples more vulnerable to prediction instability. These findings suggest that some features contribute more to robustness under parameter variation, while others primarily influence boundary-sensitive classifications.

Fixed-label samples. We apply the guidance function G in equation 7 to generate samples that a trained model assigns to a specified target label. In this setting, the target label is fixed in advance, and the guidance function directs the generation process toward samples that the model assigns to that label with high confidence. The resulting samples reveal the input patterns that the model has learned to associate with the target output, providing a direct way to inspect its class-specific behavior.

To demonstrate this setting, we use a ResNet50 model pre-trained on ImageNet-1k (available in `torchvision.models`) and generate image samples guided toward selected target classes. Specifically, we consider the classes "goldfish" or "snail." The guidance function encourages generated samples to be classified by ResNet50 as the chosen target label. Since the input space is high-dimensional, we perform generation in the latent space of a pre-trained TAESD autoencoder [3] and decode the generated latent vectors back into image space. Figure 6(a) shows generated samples classified as "goldfish," while Figure 6(b) shows generated samples classified as "snail." The generated goldfish samples predominantly contain orange regions, often accompanied by a small dark region resembling an eye. In contrast, the generated snail samples tend to contain curved, antenna-like structures. These visual patterns suggest that the model relies on different class-specific cues when assigning these labels. In particular, the generated goldfish samples indicate that color may play an important role in the model's recognition of this class. We test this hypothesis by evaluating ResNet50 on 50 color-modified validation samples. Results for goldfish, snail, zebra, and other classes are reported in Table 2 in Appendix F. Notably, swapping green and blue channels does not

substantially impair goldfish detection (suggesting that the model relies primarily on the red channel), while grayscale conversion affects goldfish but not zebra classification.



(a) (b) (c) (d)

Figure 6: (a) generated samples classified as "goldfish", (b) generated samples classified as "snail", (c) "goldfish" validation images misclassified by ResNet50 when converted to grayscale, (d) "snail" validation images converted to grayscale, only the first is misclassified by ResNet50.

This experiment illustrates how fixed-label guided generation can be used to study the input patterns that a trained model associates with a specified output class. The generated samples act as diagnostic artifacts, revealing class-specific visual cues learned by the model rather than explanations tied to a single observed instance.

5 Conclusion

We introduce a mathematical framework for guided data generation to support the analysis of trained machine learning models, going beyond traditional interpretability methods. By formulating guidance functions, we demonstrate how to generate samples for scenarios involving prediction uncertainty, parameter sensitivity, model disagreement, and class-specific behavior. Our computational study demonstrates the flexibility of the framework across classification and regression tasks on diverse datasets, revealing insights into decision boundaries, input sensitivities, model disagreement, and class-specific behavior. Our goal is to support a deeper understanding of trained machine learning models, with guided data generation providing a flexible tool for model analysis.

Our framework also presents several opportunities for improvement and future research. Incorporating implicit constraints among features (e.g., monotonic relationships) could enable the generation of samples that more accurately reflect the structure of the data and enhance their interpretability and reliability. The choice of regularization and sampling hyperparameters can affect the generated distributions. Extending the framework to more complex models such as LLMs is not straightforward and introduces both technical and conceptual challenges that warrant further investigation. In addition, domain experts may identify application-specific downstream uses beyond those considered here.

As noted above, for high-dimensional inputs, we use a data generation procedure such as a VAE, which may itself introduce biases. Therefore, any hypotheses developed using our method should be independently tested, consistent with the recommendation in the closing remarks of [14]: "... should treat the results as exploratory, and ideally validate the results with well-controlled experiments that do not rely on the generative model." Furthermore, approaches such as [16] could be incorporated in place of standard VAEs to steer the data generation process using interpretable concepts. Addressing these considerations will help refine and build upon the foundational study presented here.

Acknowledgments and Disclosure of Funding

The first author is supported by JSPS Kakenhi Grant number JSPS, KAKENHI Grant Number 26K14955.

References

[1] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5PC7J.

[2] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. In *International Conference on Learning Representations (ICLR)*, 2021.

[3] Ollin Boer Bohan. Taesd, 2023.

[4] Boris van Breugel, Trent Kyono, Jeroen Berrevoets, and Mihaela van der Schaar. Decaf: generating fair synthetic data using causally-aware generative networks. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2024. Curran Associates Inc.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[7] David Duvenaud, Jackson Wang, Jorn Jacobsen, Kevin Swersky, Mohammad Norouzi, and Will Grathwohl. Your classifier is secretly an energy-based model and you should treat it like one. In *International Conference on Learning Representations (ICLR)*, 2020.

[8] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Data augmentation using synthetic data for time series classification with deep residual networks. *arXiv preprint arXiv:1808.02455*, 2018.

[9] FICO. Home Equity Line of Credit (HELOC) Dataset. FICO Community, 2018. https://community.fico.com/s/explainable-machine-learning-challenge.

[10] Ankush Ganguly and Samuel WF Earp. An introduction to variational inference. *arXiv preprint arXiv:2108.13083*, 2021.

[11] Jürgen Geiser. Numerical Picard iteration methods for simulation of non-Lipschitz stochastic differential equations. *Symmetry*, 12(3), 2020.

[12] Jack Good, Torin Kovach, Kyle Miller, and Artur Dubrawski. Feature learning for interpretable, performant decision trees. In *Advances in Neural Information Processing Systems*, volume 36, pages 66571–66582. Curran Associates, Inc., 2023.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, 2014.

[14] Peter Harrison, Raja Marjieh, Federico Adolfi, Pol van Rijn, Manuel Anglada-Tort, Ofer Tchernichovski, Pauline Larrouy-Maestri, and Nori Jacoby. Gibbs Sampling with People. *Advances in neural information processing systems*, 33:10659–10671, 2020.

[15] Matthew D Hoffman, Andrew Gelman, et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[16] Aya Abdelsalam Ismail, Julius Adebayo, Hector Corrada Bravo, Stephen Ra, and Kyunghyun Cho. Concept bottleneck generative models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 34114–34132, 2024.

[17] Kaggle. Housing Prices Dataset, 2021. https://www.kaggle.com/datasets/yasserh/housing-prices-dataset.

[18] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[19] Matt Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4069–4079, 2017.

[20] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, , and Fu-Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

[21] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. MNIST Handwritten Digit Database. AT & T Labs, 2010. http://yann.lecun.com/exdb/mnist.

[22] Dan Ley, Saumitra Mishra, and Daniele Magazzeni. GLOBE-CE: a translation based approach for global counterfactual explanations. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[23] Qiang Liu, Jason Lee, and Michael Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR, 2016.

[24] Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2020. Curran Associates Inc.

[25] Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. TabPFGen – Tabular data generation with TabPFN. *arXiv preprint arXiv:2406.05216*, 2024.

[26] Andrei Margeloiu, Xiangjian Jiang, Nikola Simidjievski, and Mateja Jamnik. TabEBM: A tabular data augmentation method with distinct class-specific energy-based models. *arXiv preprint arXiv:2409.16118*, 2024.

[27] Osvaldo A. Martin, Oriol Abril-Pla, Jordan Deklerk, Seth D. Axen, Colin Carroll, Ari Hartikainen, and Aki Vehtari. Arviz: a modular and flexible library for exploratory analysis of bayesian models. *Journal of Open Source Software*, 11(119):9889, 2026.

[28] Charles T. Marx, Flavio du Pin Calmon, and Berk Ustun. Predictive multiplicity in classification. In *International Conference on Machine Learning*, pages 6765–6774. PMLR, 2020.

[29] Michael Christoph Nowotny. *divergence: A Python library for Shannon measures and f-divergences*, 2026.

[30] Adam Sanborn and Thomas Griffiths. Markov Chain Monte Carlo with People. *Advances in neural information processing systems*, 20, 2007.

[31] Jacob Si, Wendy Yusi Cheng, Michael Cooper, and Rahul G. Krishnan. InterpreTabNet: distilling predictive signals from tabular data by salient feature interpretation. ICML'24. JMLR.org, 2024.

[32] Boris van Breugel, Trent Kyono, Jeroen Berrevoets, and Mihaela van der Schaar. DECAF: Generating fair synthetic data using causally-aware generative networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021.

[33] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved r-hat for assessing convergence of mcmc (with discussion). *Bayesian analysis*, 16(2):667–718, 2021.

[34] Sebastien C. Wong, Adam Gatt, Victor Stamatescu, and Mark D. McDonnell. Understanding data augmentation for classification: When to warp? In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6, 2016.

[35] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. Fairgan: Fairness-aware generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 570–575, 2018.

[36] Arnold Zellner. Optimal information processing and bayes's theorem. *The American Statistician*, 42(4):278–280, 1988.

[37] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[38] Jian-Qiao Zhu and Thomas L Griffiths. Eliciting the Priors of Large Language Models using Iterated in-context Learning. *arXiv preprint arXiv:2406.01860*, 2024.

A Linear Regression with Gaussian Data

Proof of Theorem 1. Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, we construct the loss function $F(\theta)$ as the integral of $\ell_F(y_{\boldsymbol{\theta}}(\mathbf{x}), y)$, over the data distribution, which is approximated by the Dirac delta comb $\nu = \frac{1}{N}\sum_{i=1}^N \delta_{(\mathbf{x}_i, y_i)}$:

$$F(\theta) = \int_{\mathcal{X}\times\mathcal{Y}} \ell_F(y_{\boldsymbol{\theta}}(\mathbf{x}), y) \mathrm{d}\,\nu(\mathbf{x}, y) = \frac{1}{2N}\sum_{i=1}^N \left(\mathbf{x}_i^\top \theta - y_i\right)^2.$$

Assume, for convenience, that a constant feature of 1 is included as the last coordinate of $\mathbf{x}$, allowing us to explicitly represent the intercept. Using this notation, we define

$$\mathbf{x} = \begin{bmatrix}\mathbf{f}\\ 1\end{bmatrix}, \qquad \theta = [\boldsymbol{\xi} \quad b], \text{ so that } \mathbf{x}^\top\theta = \mathbf{f}^\top\boldsymbol{\xi} + b.$$

We write the design matrix as

$$D = \begin{bmatrix} \cdots & \mathbf{x}_1^\top & \cdots & 1\\ \cdots & \mathbf{x}_2^\top & \cdots & 1\\ & \vdots & & \vdots\\ \cdots & \mathbf{x}_N^\top & \cdots & 1 \end{bmatrix} = [\ X \quad \mathbf{1}\ ].$$

The quadratic loss function can then be expressed as

$$F(\theta) = \frac{1}{2N}\|D\theta - \mathbf{y}\|^2,$$

where $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_N]^\top$ is the label vector. We can reorder the terms so that

$$\begin{aligned} F(\theta) &= \frac{1}{2N}(D\theta - y)^\top(D\theta - y) = \frac{1}{2N}\left(\theta^\top D^\top D\theta - 2\theta^\top X^\top \mathbf{y}\right) + \text{const.}\\ &= \frac{1}{2}(\theta - \widehat{\theta})^\top \frac{D^\top D}{N}(\theta - \widehat{\theta}) + \text{const.} \end{aligned}$$

where $\widehat{\theta} = (D^\top D)^{-1}D^\top\mathbf{y}$. Note that this is precisely the ordinary least squares solution.

Since the loss function is quadratic, we can explicitly write the Gibbs distribution (which is the unrestricted solution to the Bayesian Learning Problem with F) as the Gaussian distribution

$$q^*(\theta) \propto e^{-\beta F(\theta)} \propto e^{-\frac{1}{2}(\theta-\widehat{\theta})^\top \frac{D^\top D}{N/\beta}(\theta-\widehat{\theta})} \quad \text{thus } q^*(\theta) = \mathcal{N}\left(\widehat{\theta}, \left(\frac{D^\top D}{N/\beta}\right)^{-1}\right).$$

Here, the variable β is the inverse temperature defined as $\beta = 1/\tau$.

Next, we construct G, a loss function on $\mathcal{X}\times\mathcal{Y}$. By fixing the label, we may also consider G as a loss function only on $\mathcal{X}$, from which we derive a distribution over $\mathcal{X}$. To avoid overusing $\mathbf{x}$ and y, we denote elements of the labeled dataset as $(\mathbf{z}, w) \in \mathcal{X}\times\mathcal{Y}$ with $\mathbf{z} = [\begin{smallmatrix}\mathbf{f}\\1\end{smallmatrix}]$. Using the first and second moments of Gaussians, we calculate

$$\begin{aligned} G(\mathbf{z}, w) &= \int_\Theta \left(\mathbf{z}^\top\theta - w\right)^2 q^*(\theta)\mathrm{d}\,\theta\\ &= \mathbf{z}^\top \mathbb{E}_{q^*}[\theta\theta^\top]\mathbf{z} - 2w\mathbf{z}^\top\mathbb{E}_{q^*}[\theta] + \text{ const}\\ &= \mathbf{z}^\top\left(\widehat{\theta}\widehat{\theta}^\top + \left(\frac{D^\top D}{N\tau}\right)^{-1}\right)\mathbf{z} - 2w\mathbf{z}^\top\widehat{\theta} + \text{ const.} \end{aligned}$$

which is again a quadratic function in $\mathbf{z}$. Let us now write this quadratic in terms of $\mathbf{f}$. We write $\widehat{\theta} = \left[\begin{smallmatrix}\widehat{\boldsymbol{\xi}}\\ \widehat{b}\end{smallmatrix}\right]$.

First, a quick calculation gives the block diagonal form

$$\left(\frac{D^\top D}{N\tau}\right)^{-1} = \tau \begin{bmatrix} \frac{X^\top X}{N} & \overline{\mathbf{x}} \\ \overline{\mathbf{x}}^\top & 1 \end{bmatrix}^{-1} = \tau \begin{bmatrix} A^{-1} & -A^{-1}\overline{\mathbf{x}} \\ -\overline{\mathbf{x}}^\top A^{-1} & * \end{bmatrix},$$

where $A = \frac{X^\top X}{N} - \overline{\mathbf{x}}\overline{\mathbf{x}}^\top$ is the Schur complement and $\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i$ is the mean data vector.
We can write G as a quadratic function of $\mathbf{f}$ (fixing w) as

$$\begin{aligned} G_w(\mathbf{f}) &= \mathbf{f}^\top \left(\tau A^{-1} + \widehat{\boldsymbol{\xi}}\widehat{\boldsymbol{\xi}}^\top\right)\mathbf{f} - 2\mathbf{f}^\top\left(\tau A^{-1}\overline{\mathbf{x}} - \widehat{\boldsymbol{\xi}}\hat{b} + w\hat{b}\right) + \text{const.} \\ &= (\mathbf{f} - \widehat{\mathbf{f}})\left(\tau A^{-1} + \widehat{\boldsymbol{\xi}}\widehat{\boldsymbol{\xi}}^\top\right)(\mathbf{f} - \widehat{\mathbf{f}}) + \text{const.} \end{aligned}$$

Here, $\widehat{\mathbf{f}}$ is calculated as

$$\begin{aligned} \widehat{\mathbf{f}} &= \left(\tau A^{-1} + \widehat{\boldsymbol{\xi}}\widehat{\boldsymbol{\xi}}^\top\right)^{-1}\left(\tau A^{-1}\overline{\mathbf{x}} + \widehat{\boldsymbol{\xi}}(w - \hat{b})\right) \\ &= \left(A_\tau - \frac{A_\tau \widehat{\boldsymbol{\xi}}\widehat{\boldsymbol{\xi}}^\top A_\tau}{1 + \widehat{\boldsymbol{\xi}}^\top A_\tau \widehat{\boldsymbol{\xi}}}\right)\left(A_\tau^{-1}\overline{\mathbf{x}} + \widehat{\boldsymbol{\xi}}^\top(w - \hat{b})\right), \end{aligned}$$

where $A_\tau = \frac{1}{\tau}A$ and the Sherman-Morrison formula is used for inverting the matrix.
Now expanding the product, we obtain

$$\widehat{\mathbf{f}} = \overline{\mathbf{x}} + A_\tau\widehat{\boldsymbol{\xi}}(w - \hat{b}) - \frac{A_\tau\widehat{\boldsymbol{\xi}}\widehat{\boldsymbol{\xi}}^\top\overline{\mathbf{x}}}{1 + \widehat{\boldsymbol{\xi}}^\top A_\tau\widehat{\boldsymbol{\xi}}} - A_\tau\widehat{\boldsymbol{\xi}}\frac{\widehat{\boldsymbol{\xi}}^\top A_\tau\widehat{\boldsymbol{\xi}}}{1 + \widehat{\boldsymbol{\xi}}^\top A_\tau\widehat{\boldsymbol{\xi}}}(w - \hat{b}).$$

Note that if we denote the predictions of the linear model as $\mathbf{x}_i^\top\widehat{\boldsymbol{\xi}} + \widehat{b} = \hat{y}_i$, we can rewrite the above formula as follows:

$$\begin{aligned} \widehat{\mathbf{f}} &= \bar{\mathbf{x}} + A_\tau\widehat{\boldsymbol{\xi}}(w - \widehat{b})\frac{1}{1 + \hat{\boldsymbol{\xi}}^\top A_\tau\hat{\boldsymbol{\xi}}} - A_\tau\hat{\boldsymbol{\xi}}\frac{\hat{\boldsymbol{\xi}}^\top\bar{\mathbf{x}}}{1 + \hat{\boldsymbol{\xi}}^\top A_\tau\hat{\boldsymbol{\xi}}} \\ &= \bar{\mathbf{x}} + A_\tau\hat{\boldsymbol{\xi}}\frac{(w - \hat{b})}{1 + \hat{\boldsymbol{\xi}}^\top A_\tau\hat{\boldsymbol{\xi}}} - A_\tau\hat{\boldsymbol{\xi}}\frac{(\widehat{\bar{\mathbf{y}}} - \hat{b})}{1 + \hat{\boldsymbol{\xi}}^\top A_\tau\hat{\boldsymbol{\xi}}} \\ &= \bar{\mathbf{x}} + A_\tau\hat{\boldsymbol{\xi}}\frac{w - \widehat{\bar{\mathbf{y}}}}{1 + \hat{\boldsymbol{\xi}}^\top A_\tau\hat{\boldsymbol{\xi}}}. \end{aligned}$$

Here, we denoted the prediction of the average data by $\widehat{\bar{\mathbf{y}}} = \hat{\xi}^\top\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^N \hat{y}_i$.

Finally, let's rewrite $A_\tau\widehat{\boldsymbol{\xi}}$ and $\widehat{\boldsymbol{\xi}}^\top A_\tau\widehat{\boldsymbol{\xi}}$ in terms of interpretable statistical quantities. Recall that $A_\tau = \frac{1}{\tau}\left(\frac{X^\top X}{N} - \overline{\mathbf{x}}\overline{\mathbf{x}}^\top\right)$. Using this, we compute

$$\begin{aligned} A_\tau\widehat{\boldsymbol{\xi}} &= \frac{1}{\tau}\left(\frac{1}{N}\sum_{i=1}^N \mathbf{x}_i(\underbrace{\mathbf{x}_i^\top\widehat{\boldsymbol{\xi}}}_{=\hat{y}_i - \hat{b}}) - \overline{\mathbf{x}}\underbrace{\overline{\mathbf{x}}^\top\hat{\boldsymbol{\xi}}}_{=\frac{1}{N}\sum_{i=1}^N \hat{y}_i - b}\right) \\ &= \frac{1}{N\tau}\sum_{i=1}^N(\mathbf{x}_i - \overline{\mathbf{x}})(\hat{y}_i - \hat{b}) \\ &= \frac{1}{N\tau}\sum_{i=1}^N(\mathbf{x}_i - \overline{\mathbf{x}})(\hat{y}_i - \hat{\bar{\mathbf{y}}}) \\ &= \frac{1}{\tau}\operatorname{Cov}(X, \widehat{\mathbf{y}}). \end{aligned}$$

In the final expression, the term $\mathrm{Cov}(X, \widehat{\mathbf{y}})$ corresponds directly to the previous line. This covariance is a vector that averages data deviations, weighted by prediction deviations. In the line before last, we replaced $\widehat{b}$ with any constant since it is independent of i, and the first factor sums to the zero vector. Additionally, we leveraged a key property of linear models: the average of the predictions is the same as the prediction of the average.

A similar calculation yields,

$$\hat{\boldsymbol{\xi}}^\top A_\tau \hat{\boldsymbol{\xi}} = \frac{1}{\tau}\left(\frac{1}{N}\sum_{i=1}^{N}\hat{y}_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N}\hat{y}_i\right)^2\right) = \frac{1}{\tau}\,\mathrm{Var}(\widehat{\mathbf{y}}).$$

Therefore, we obtain an explicit quadratic formulation of the data loss function G in terms of $\mathbf{f}$ at a fixed w. This means that the data distribution $p^*(\mathbf{x})$, which solves the unrestricted Bayesian Learning Problem, follows a Gaussian distribution given as

$$p^*(\mathbf{f}) \propto e^{-G_w(\mathbf{f})} \propto \mathcal{N}(\widehat{\mathbf{f}}, \Sigma),$$

where

$$\widehat{\mathbf{f}} = \overline{\mathbf{x}} + \frac{\mathrm{Cov}(X, \widehat{\mathbf{y}})}{\tau + \mathrm{Var}(\widehat{\mathbf{y}})}\left(w - \frac{1}{N}\sum_{i=1}^{N}\widehat{y}_i\right), \tag{9}$$

and

$$\Sigma^{-1} = \left(\tau\left(\frac{X^\top X}{N} - \bar{\mathbf{x}}\bar{\mathbf{x}}^\top\right)^{-1} + \widehat{\boldsymbol{\xi}}\widehat{\boldsymbol{\xi}}^\top\right). \tag{10}$$

□

The interpretation of the mean $\widehat{\mathbf{f}}$ is as follows: if you want to sample from a data distribution that will produce a given ω, then you should not sample around $\bar{\mathbf{x}}$ (which would be the case without output restrictions). Instead, you shift $\bar{\mathbf{x}}$ in proportion to the difference between ω and the mean of the training label predictions, following the direction of the covariance between the training data and predicted labels.

B Computational Setup

In this section, we supplement our computational study by presenting the datasets and detailing the neural network architectures.

B.1 Datasets Used in the Experiments

Our experiments are conducted using four numerical datasets and one visual dataset from the literature. The details of the datasets are provided below.

FICO. The FICO (HELOC) dataset (CC0: Public Domain) consists of home equity line of credit applications submitted by homeowners [9]. It includes 10,459 records with 23 features, comprising both numerical and ordinal variables. The primary objective is to classify applications based on their risk performance, identifying whether an applicant is likely to meet payment obligations or become delinquent. Data preprocessing steps are applied to address missing values.

Housing. The Housing dataset (CC0: Public Domain), sourced from Kaggle, includes information on various house attributes such as lot size, number of rooms, and number of stories [17]. The dataset contains 535 records and 12 features, comprising both numerical and ordinal variables. The primary objective is to predict housing prices based on these features.

MNIST. The MNIST dataset (Data files © Original Authors) is a widely used benchmark in computer vision, consisting of 70,000 grayscale images of handwritten digits (0–9), each represented as a 28×28 pixel matrix [21]. The dataset is divided into 60,000 training samples and 10,000 test samples. The primary objective is to classify images based on the digit they represent. We normalized each of the images to be arrays of shape $(28, 28, 1)$ with FP32 values in the interval $[0, 1]$.

Wine Recognition Dataset. The Wine Recognition dataset contains the results of chemical and physical analyses of wines produced in three regions of Italy [1]. Each record is described by 13 continuous features obtained through chemical analysis, including alcohol content, malic acid, magnesium, color intensity, and hue. The primary objective is to classify wines according to their region of origin.

B.2 Experimental Setup

We ran all the experiments on a personal Apple M1 Pro with 32 GB RAM. In our experiments compute time was not a significant factor, all experiments were completed within minutes.

For the parameter-sensitive and prediction-risky experiments on the FICO dataset, we trained an MLP with ReLU activation functions and layer widths of $128 - 32 - 8 - 2$. Dropout with a rate of 0.2 was applied after each activation layer to prevent overfitting. The model was trained using a batch size of 128 for $10,000$ steps.

For the image experiments, we used an MLP with layer widths of 1024-128-10, where each layer included a ReLU activation, followed by a dropout layer with a rate of 0.2. The CNN architecture consisted of two convolutional blocks with feature sizes $32 - 64$. Each block followed the structure: Conv $\rightarrow$ ReLU $\rightarrow$ Conv $\rightarrow$ ReLU $\rightarrow$ max_pool $\rightarrow$ Dropout, where the convolutional kernels had a size of 3×3, the max pooling window was 2×2, and the dropout rate was 0.2.

Both the CNN and MLP models were trained for 10,000 update steps using a batch size of 128 and the Adam optimizer. The learning rate followed an exponential decay schedule, starting with a maximum learning rate of 0.1, decaying by a rate of 0.9 every 100 steps.

C Langevin Dynamics solving the Bayesian Learning Problem

In solving equation 3, we sample data points by solving a Bayesian optimization problem over distributions on the data space:

$$p^* = \arg\min_{p \in \mathcal{Q}} \mathbb{E}[G] - \tau \mathcal{H}(p),$$

where $\mathcal{H}(p)$ denotes the entropy and $\tau > 0$ is a temperature parameter.

The intuition behind this objective is that instead of minimizing a function G we find a distribution over the space X which balances having its distributional mass focused on points where G is low and also a spread widely. This objective is also called the Evidence Lower BOund (ELBO) and interpreting the loss as negative log-likelihood the solution can be interpreted as Bayes's formula updating beliefs upon observing new data, as shown by Zellner [36].

C.1 Metropolis Hastings Langevin Algorithm (MALA)

On unconstrained data space, i.e., when $\mathcal{Q} = \mathcal{P}(\mathbb{R}^d)$, one approach of sampling from p^* is to simulate the Langevin dynamics, which is a Stochastic Differential Equation (SDE) given by

$$\frac{\mathrm{d}\, X}{\mathrm{d}\, t} = -\frac{1}{\tau} \nabla G(X) + \sqrt{2} \frac{\mathrm{d}\, W}{\mathrm{d}\, t},$$

where W is a Wiener process (i.e., standard Brownian motion), the G is called a drift term. The limiting distribution of this SDE converges to $p_\infty(X) \propto e^{-\frac{1}{\tau} G(x)}$.

For concrete computation, we work with the standard Euler-Maruyama discretization

$$\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t - \eta \nabla G(\mathbf{x}_t) + \sqrt{2\eta\tau}\varepsilon_t,$$

where $\eta > 0$ is the step size, and $\varepsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ is sampled independently from the unit normal distribution at each time step.

To improve convergence and ensure the correctness of the sampling distribution, we apply a Metropolis–Hastings acceptance step. Specifically, the proposed update $\mathbf{x}_{t+1}$ is accepted as $\mathbf{x}_{t+1} = \tilde{\mathbf{x}}_{t+1}$ with probability

$$\alpha := \min\left\{1, \frac{\pi(\tilde{\mathbf{x}}_{t+1}) q(\mathbf{x}_t | \tilde{\mathbf{x}}_{t+1})}{\pi(\mathbf{x}_t) q(\tilde{\mathbf{x}}_{t+1} | \mathbf{x}_t)}\right\}$$

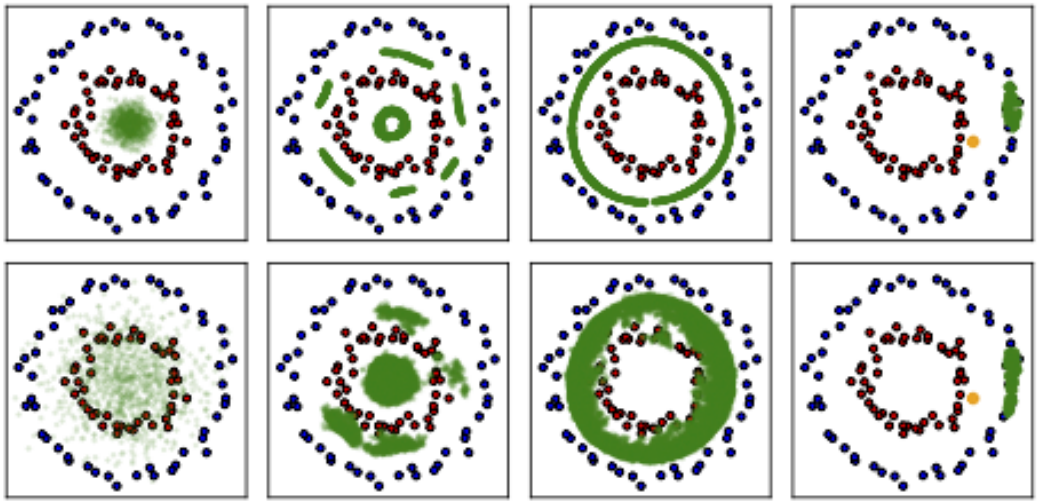


Figure 7: The guidance functions in Figure 2, where in the first there columns the data generation in the second line trained with different temperatures. The last column shows the effect of a ℓ_2 regularizing term (with anchor point shown in orange) with a smaller strength.

or otherwise rejected, in which case $\mathbf{x}_{t+1} = \mathbf{x}_t$. Here, $\pi \propto e^{-\frac{1}{\tau}G}$ and $q(\mathbf{x}'|\mathbf{x})$ is the transition probability of stepping from $\mathbf{x}$ to $\mathbf{x}'$ given by $q(\mathbf{x}'|\mathbf{x}) = \exp(-\frac{1}{4\tau}\|\mathbf{x}' - \mathbf{x} + \tau\nabla G(\mathbf{x})\|^2)$. An implementation of this sampling procedure is provided in the submitted code repository, specifically in the file `langevin.py` using JAX/PyTorch and Numpy.

The temperature hyperparameter τ determines the balance between exploration of the data space versus minimizing G. In the figure below we show the effect of changing this parameter (as well as the effect of changing the tunable parameter of the regularizing term).

C.2 Smoothing

The acceptance/rejection is the critical step to ensure convergence of the trajectory to the limiting Gibbs-Boltzmann distribution $\propto e^{-\frac{1}{\tau}G}$. The drift term $\nabla G(\mathbf{x})$ speeds up the convergence by ensuring that more of the proposals $\widetilde{\mathbf{x}}$ will be accepted since this becomes (noisy) gradient descent.

In case the function G is locally flat, such as those functions created from tree-based models (Decision Tree, Random Forest, XGBoost) then the gradient term is always $\mathbf{0}$, reducing the proposals to simple random walk. This is called the Metropolis Hastings (MH) algorithm and it also has the same limiting distribution, and the transition probabilities in the acceptance ratio cancel since $q(\mathbf{x}|\mathbf{x}') = q(\mathbf{x}'|\mathbf{x})$.

However in this case at any step the proposals are random and the point $\mathbf{x}$ does not see if it is near a decision boundary or not. For this purpose we propose using the gradients of the smoothed function $G_s(\mathbf{m}) = \int G(\mathbf{x})\mathcal{N}(\mathbf{m}, \sigma)(\mathbf{x})\mathrm{d}\mathbf{x}$ for the proposals. With a larger σ, the point $\mathbf{x}$ "sees" a wider horizon for its proposals. Indeed the gradient of G_s is

$$\nabla G_s(\mathbf{x}) = \int G(\mathbf{x} + \sigma\varepsilon)\varepsilon\mathcal{N}(\mathbf{0}, I)(\varepsilon)\mathrm{d}\varepsilon \approx \frac{1}{J}\sum_{j=1}^{J} G(\mathbf{x} + \sigma\varepsilon_j)\varepsilon_j, \qquad \text{with } \varepsilon_j \sim \mathcal{N}(\mathbf{0}, I).$$

In our implementation we applied proposals with ∇G_s and acceptance/rejection with G In the code, we also assumed $\nabla G_s(\mathbf{x}) = \nabla G_s(\mathbf{x}')$ for computational simplicity in calculating the transition probabilities $q(\mathbf{x}'|\mathbf{x})$. This is approximately true when G is a step function and the step size η is small. Regardless of whether one uses this approximation, the limiting distribution is still the Gibbs-Boltzmann distribution since that is ensured by the Metropolis-Hastings acceptance step, regardless of the specific proposal.

C.3 Other Methods and Variants

Beyond the Langevin dynamics approach, several alternative methods exist for sampling data points. For instance, one may use the Picard iteration method or various splitting schemes (see [11]) to discretize and solve the above SDE. Another approach is to restrict the distribution family $\mathcal{Q}$ to be a tractable statistical manifold and apply Variational Inference techniques [10].

It is also worth noting that when sampling over a data manifold, if there exists a mapping $\varphi : \mathbb{R}^d \to \mathcal{X}$ and $G : \mathcal{X} \to \mathbb{R}$, then the corresponding Gibbs distribution on $\mathbb{R}^d$ (with respect to the Lebesgue measure) is given by $e^{-\frac{1}{\tau}H}$, where $H = G \circ \varphi : \mathbb{R}^d \to \mathbb{R}$. Samples drawn from this distribution can be pushed forward via φ to obtain samples in $\mathcal{X}$. These pushed-forward samples follow a distribution that can be interpreted as a Gibbs distribution over $\mathcal{X}$ with respect to the base measure ν, which is the pushforward of the Lebesgue measure under φ.

D Experimental Details and Additional Numerical Results

This section presents additional results that complement the findings discussed in Section 4. These results provide further insights into the generated data distributions, feature variations, and model behavior under different guided data generation settings.

Model-contrasting samples. Given the FICO dataset, we apply min–max scaling to the features and fit a logistic regression model and an XGBoost classifier. We take G as the model-contrast function equation 5, where ℓ_G is the binary cross-entropy loss, $\hat{y}$ is the predictor of the XGBoost classifier, $\tau = 0.1$, and $\varphi(z)$ is the sigmoid function to map the variables to the normalized range $[0, 1]$. We generate four chains of length $N = 20,000$ each. After omitting the burn-in period and thinning by retaining every 10th sample to reduce autocorrelation, we obtain 6000 data points with an Effective Sample Size (ESS) ≈ 3780. The KSD value is 0.5719 for the Inverse MultiQuadratic (IMQ) kernel.

In this subsection, we also our analysis of model-contrasting samples by applying the framework to a different tabular dataset. In this experiment, we examine prediction divergence between support vector regression (SVR) and linear regression (LR) models. We use the Housing dataset, where the primary objective is to predict house prices based on various structural and amenity-related features. We split the dataset into training-test sets and train both models on the same training data. To generate data samples where the two models diverge in their predictions, we formulate the cost function given in (5) as $\ell_G(y_1, 1 - y_2) = \exp(-(y_1 - y_2)^2)$. Using our framework, we generate data samples to identify the regions of the input space where the models exhibit significant disagreement, likely due to their differing assumptions about feature interactions and predictive mechanisms.

Figure 8 presents a scatter plot comparing the predictions of the SVR and LR models. The blue points represent the predictions of the models in the test data, demonstrating that the two models generally produce highly similar outputs, with minimal differences observed. The green points, on the other hand, represent generated samples, highlighting instances where the models exhibit contrasting predictions. The zoomed-in inset further emphasizes these discrepant predictions, demonstrating that our framework effectively identifies and generates data points that maximize the divergence between the two models.

Figure 9 compares the feature distributions between the synthetic dataset generated by our framework and the test data. The box plots represent the range of values for each feature, with blue corresponding to the test data and green representing the generated samples. The Housing dataset [17] used in this experiment contains real estate information such as lot size, number of bathrooms, number of stories, and heating/air conditioning types, aiming to predict house prices. This figure provides a clear visualization of how the generated data differs from the test data in terms of feature distributions. For instance, as the number of bathrooms and stories increases, the model predictions diverge. Additionally, hot water heating and air conditioning exhibit a distinct concentration in the synthetic data, with most generated samples clustering around higher values compared to the test data. This suggests that these features play a prominent role in distinguishing instances where the models behave differently. Overall, this figure offers insights into how the generated samples differ from the original dataset, highlighting key feature distributions that drive divergence in model predictions and providing a deeper understanding of how our framework probes model behavior.

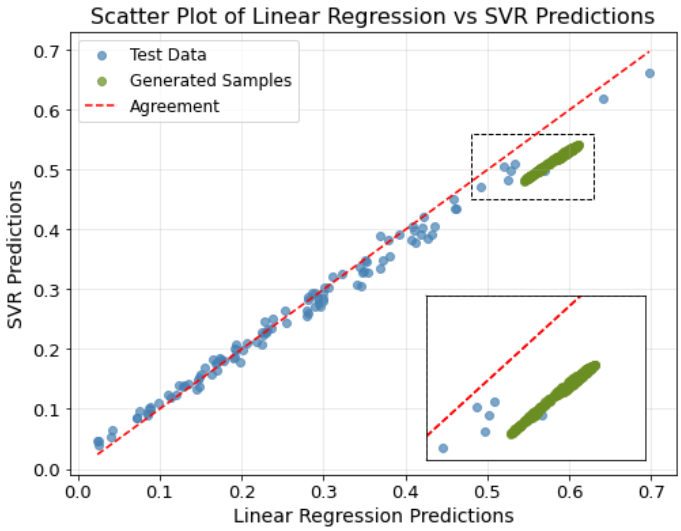


Figure 8: Comparison of SVR and LR predictions on test and generated data.

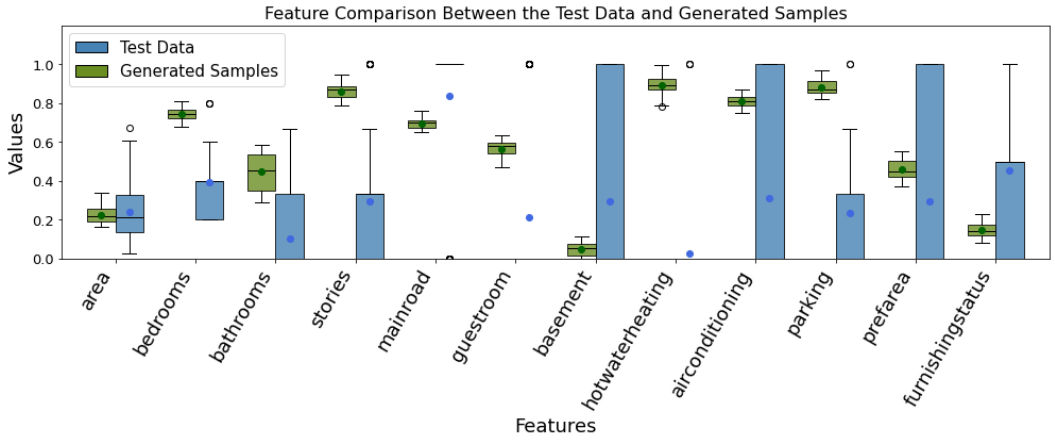


Figure 9: Features in test data and generated samples that produce different predictions for SVR and LR.

Prediction-risky samples.

This experiment explores data samples near the model's decision boundary, where predictions are inherently uncertain. To guide this analysis, we pose the question:

Which data samples are predicted to be risky due to being close to a specific anchor value?

In this experiment, we train a neural network (MLP) to classify customers in the FICO dataset as either "Good" or "Bad" credit risks. Prediction-risky samples are those with model outputs near the anchor value of 0.5. Using our framework, we generate 500 such samples to examine the characteristics of borderline classification cases. The average predicted probability of the "Bad" credit class among these samples is 0.525, with a standard deviation of 0.017.

The density plots in Figure 10 compare the distributions of two representative features in the original data and the generated samples. The feature `NumTrades60Ever2DerogPubRec` represents the number of past credit trades with payments delayed by at least 60 days, serving as a key indicator of delinquency. As shown in the figure, the distribution of risky samples follows the original data closely in the lower range but exhibits a stronger peak around zero. This suggests that the model considers individuals with few or no past delinquencies as borderline cases, likely due to the absence of strong negative or positive indicators, making classification more uncertain. The feature

`MSinceOldestTradeOpen` indicates the number of months since a customer's first credit line was opened, capturing the length of their credit history. As shown in Figure 10, the distribution of risky samples is sharply concentrated around 400 months ($\sim$33 years), whereas the original data spans a broader range. This suggests that the model associates long credit histories with greater uncertainty. The pronounced peak reflects how the model treats long-established credit profiles as ambiguous when making predictions.

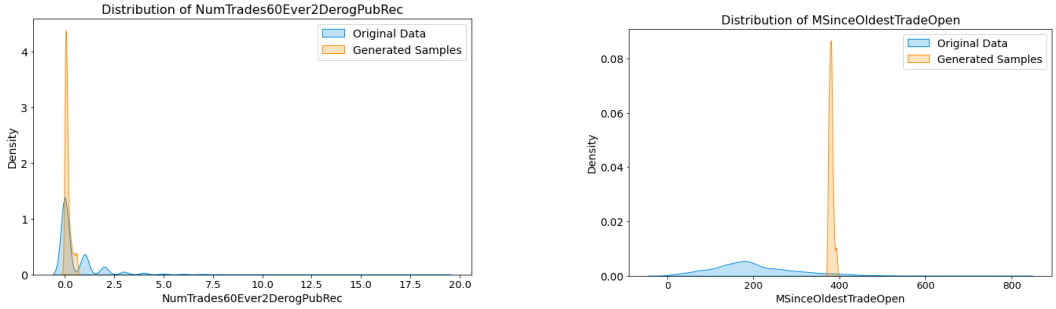


Figure 10: Feature distributions of the original data and generated prediction-risky samples.

To further investigate data samples near the decision boundary, we present the distributions of all features in the original dataset and the generated prediction-risky samples in Figure 11. These density plots provide a comprehensive view of the differences between the generated samples and the original data across multiple features. By analyzing these distributions, we can observe how the model identifies borderline cases based on different financial attributes. Across multiple features, the generated prediction-risky samples exhibit a much narrower distribution compared to the original data. This suggests that the model focuses on a specific subset of feature values when identifying borderline cases.

Table 1: Characteristics of the wine recognition dataset and the generated data

Characteristic	**Class 0**	**Class 1**	**Class 2**	**Generated**
Samples	59	71	48	50
Alcohol	$13.74_{\pm 0.63}$	$12.28_{\pm 0.53}$	$13.15_{\pm 0.52}$	$12.75_{\pm 0.61}$
Malic acid	$2.01_{\pm 0.68}$	$1.93_{\pm 1.00}$	$3.33_{\pm 1.08}$	$1.59_{\pm 0.46}$
Ash	$2.46_{\pm 0.22}$	$2.45_{\pm 0.31}$	$2.44_{\pm 0.18}$	$1.83_{\pm 0.25}$
Alcalinity of ash	$17.04_{\pm 2.52}$	$20.24_{\pm 3.33}$	$21.42_{\pm 2.23}$	$19.37_{\pm 4.95}$
Magnesium	$106.34_{\pm 10.41}$	$94.55_{\pm 16.63}$	$99.31_{\pm 10.78}$	$98.9_{\pm 28.91}$
Total phenols	$2.84_{\pm 0.34}$	$2.26_{\pm 0.54}$	$1.68_{\pm 0.35}$	$1.69_{\pm 0.43}$
Flavanoids	$2.98_{\pm 0.39}$	$2.08_{\pm 0.7}$	$0.78_{\pm 0.29}$	$2.11_{\pm 0.82}$
Nonflavanoid phenols	$0.29_{\pm 0.07}$	$0.36_{\pm 0.12}$	$0.45_{\pm 0.12}$	$0.3_{\pm 0.08}$
Proanthocyanins	$1.90_{\pm 0.4}$	$1.63_{\pm 0.6}$	$1.15_{\pm 0.4}$	$2.46_{\pm 0.41}$
Color intensity	$5.53_{\pm 1.23}$	$3.09_{\pm 0.92}$	$7.4_{\pm 2.3}$	$2.34_{\pm 0.6}$
Hue	$1.06_{\pm 0.16}$	$1.06_{\pm 0.2}$	$0.68_{\pm 0.11}$	$1.27_{\pm 0.14}$
OD280/OD315	$3.16_{\pm 0.35}$	$2.78_{\pm 0.49}$	$1.68_{\pm 0.27}$	$1.69_{\pm 0.39}$
Proline	$1115.71_{\pm 221.64}$	$519.5_{\pm 156.1}$	$629.9_{\pm 113.9}$	$1200.8_{\pm 233.3}$

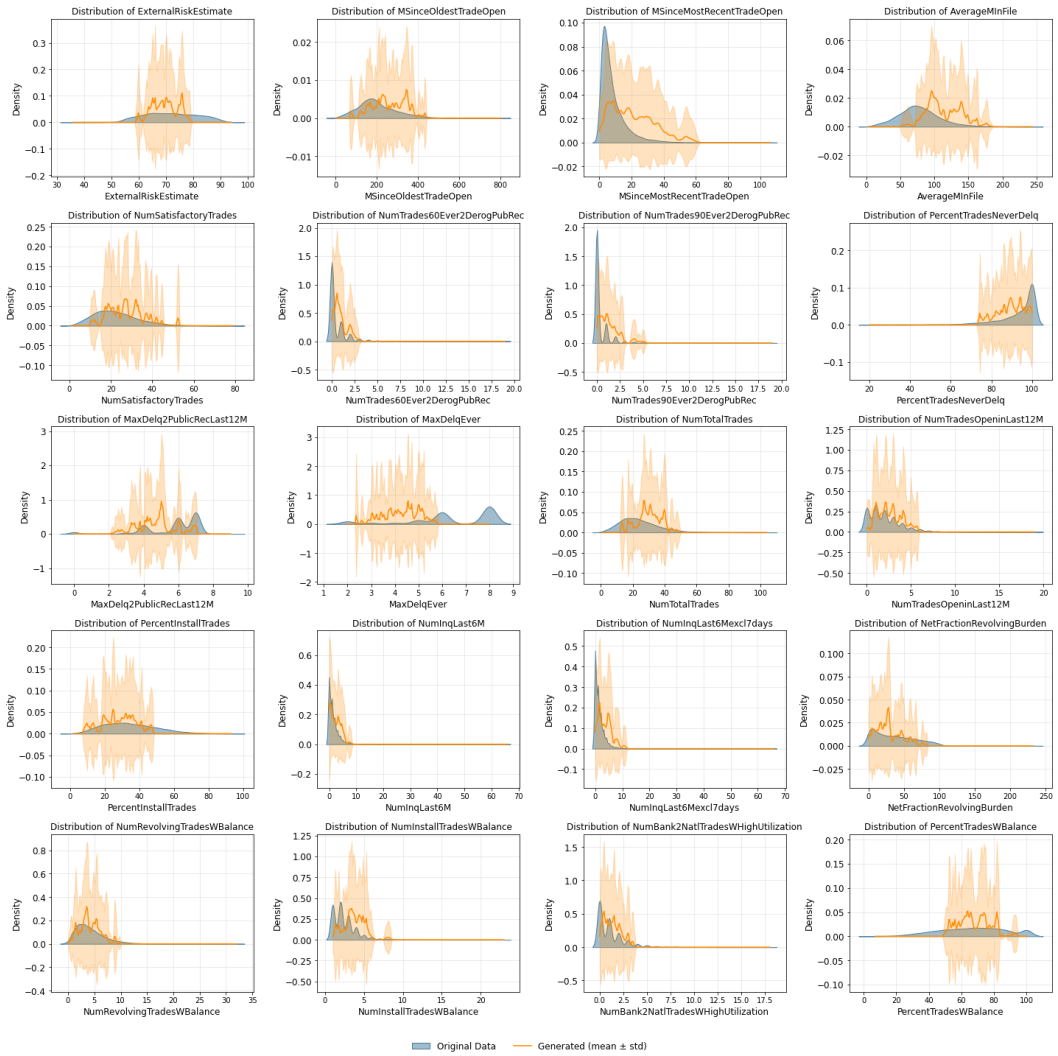


Figure 11: Feature distributions in the original data and generated prediction-risky samples.

Parameter-sensitive samples.

The details of the MALA are as follows. The temperature is set at $\tau = 0.1$ and four chains are calculated with length 10000 each. The step size is taken to be $\eta = 0.005/\tau$. The average R-hat value (among coordinates) of the chain is $1.0485_{\pm 0.008}$. We omitted the first 5000 samples as the burn-in period. We thinned out the chains by taking every tenth value to reduce autocorrelation. The KSD value of the pooled chains is calculated to be 1.1448 for the RBF kernel with bandwidth taken as 0.2.

To complement the findings presented in Section 4, we provide the full set of feature distributions comparing parameter-sensitive samples and prediction-risky samples in Figure 12. These density plots illustrate how the two types of generated samples differ. By analyzing these distributions, we observe that while some features exhibit similar trends across both sample types, others show notable divergences. Features with broader distributions in parameter-sensitive samples indicate that model perturbations impact a wider range of instances.

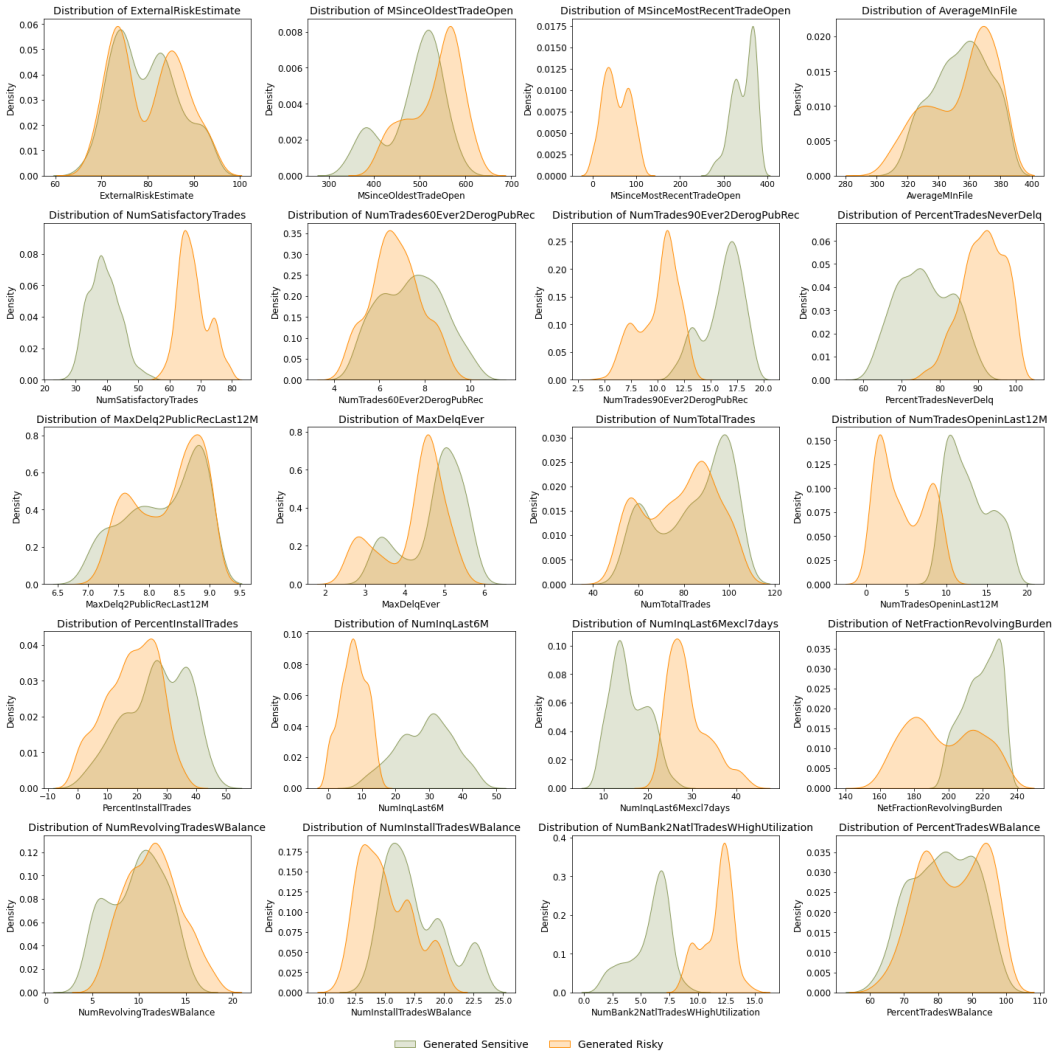


Figure 12: Feature distributions in generated parameter-sensitive and prediction-risky samples.

E The use of VAEs

A notable example of using pushforwards to obtain points on the data manifold comes from image datasets. We employ a VAE architecture with two convolutional layers each for the encoder and decoder submodules. Features in the convolutional layers are 32 and 64 with kernel sizes of (3,3) and a stride of (2,2). During training, the reconstruction loss is computed using bitwise entropy.

Figure 13 shows how this setup works for constructing loss functions G on the latent space. One may use a combination of models, each precomposed with the decoder of the trained VAE. The resulting distribution on the latent space, after pushforwarding (i.e., passing the samples through the decoder), corresponds to a distribution on the data that is closer to the original data distribution.

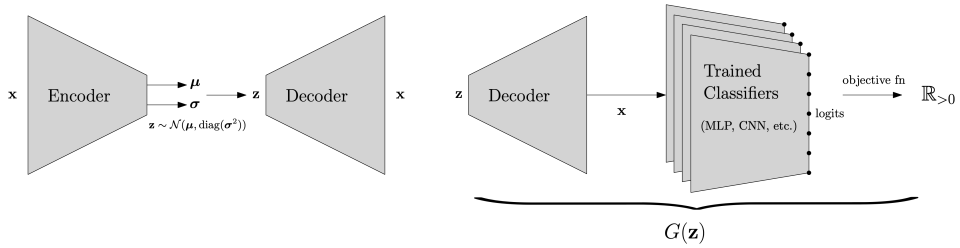


Figure 13: By precomposing with the decoder submodule of a trained neural network, we can define G functions on the lower-dimensional latent space, while still leveraging networks designed for higher-dimensional image inputs.

F High Dimensional Image Generation

To generate the images in Figure 6, using a VAE alone is insufficient for producing images of size 256×256. While a VAE ensures local consistency of colors in small patches, it does not guarantee global coherence. To address this, we successively increase the resolution of generated samples, making use of the TAESD autoencoder, which supports multiple input sizes.

The latent space of the TAESD has shape $16 \times \frac{n}{8} \times \frac{m}{8}$, where $n \times m$ is the image size. At each resolution (e.g., 64×64), we run MALA according to a probing function G. This function G simultaneously (1) reduces the cross entropy loss of the logits produced by decoder followed by the ResNet50 classifier, (2) minimizes the reconstruction loss of the high dimensional image corresponding to the latent vector, and (3) enforces closeness to the previously generated latent vector corresponding to the image at the lower resolution (e.g., 32×32). This procedure of successively increasing the resolution of the image, and starting from the resized version of the previous MALA run, maintains global consistency of the generated images whilst also satisfying the probing function requirements. Implementation details are provided in the `probe_resnet.py` script in our repository.

In Figure 6, we show samples generated with G function aiming to minimize the cross entropy loss between the ResNet50 logits and the label corresponding to goldfish:1, and snail:113. Each of the images in the figure has resolution 256×256.

By qualitatively observing these figures, we were able to form a hypothesis on how the pretrained ResNet50 (using the weights `ResNet50_Weights.IMAGENET1K_V2` from `torchvision.models`) detects the class label 1 corresponding to goldfish. The presence of the color orange in every image and also a black dot inside the orange corresponding to the eye of the fish was present in most of the images. In order to test the color hypothesis, we modify the 50 goldfish validation images from ImageNet-1k [6]. Using the luminance formula

$$\texttt{gray} = (0.299 \times R) + (0.587 \times G) + (0.114 \times B),$$

we convert RGB images to grayscale and evaluate model accuracy. Along with other color modifications such as swapping various color channels, and including classes other than goldfish and snail, we form the Table 2.

For goldfish, keeping the red channel intact while swapping green and blue does not substantially reduce accuracy, reinforcing the role of the orange color component.

Table 2: Correct ResNet50 predictions out of 50 validation images per class, with various color modifications.

	goldfish	snail	ox	broccoli	cucumber	zebra
original	48	45	32	46	36	48
grayscale	34	44	29	44	28	48
rg_swap	40	40	26	40	25	49
rb_swap	31	42	28	42	36	48
gb_swap	46	41	26	37	31	49

Also note that none of the modifications we use alter the black and white colors. The performance of the model on zebra images is completely unaffected by these changes, suggesting that it is these colors (and not, for example, the green grass on which the zebras might stand) that the model relies on to assign the class label zebra.

NeurIPS Paper Checklist

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We claim a framework for understanding trained models through guided data generation and throughout the paper show examples and uses of this framework where such data give insights into trained models.

   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: We mention our limitations in the conclusion section of our paper.

   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: Appendix A includes the proof of Theorem 1.

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: We provide the code for our work in a anonymized repository

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: We provide an anonymized repository.

   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: This is detailed in Appendix B

   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: In order to show that the samples obtained from the MCMC chains are true to the Gibbs distribution we report goodness of fit results using aggregate kernelized Stein discrepancy. We obtain these results using ksdagg package.

   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: The details are given in appendix B

   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: Our work complies with the code of ethics.

   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: [Yes]

    Justification: We discuss in the conclusion and the introduction how understanding models can have impact while using the framework.

    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: We do not release a pretrained model or datasets that have a high risk for misuse.

    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: We report the license name and cite the original owners of datasets and trained models.

    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: [N/A]

    Justification: We do not create a new asset. We develop a framework, and do not release a particular trained model.

    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: We do not use human subjects or experiments.

    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: As we do not use human subjects or crowdsourcing an IRB approval is not needed.

    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: We did not use LLMs in an important non-standard way.

    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.