arXiv:2406.01813v1 [stat.ML] 3 Jun 2024

Diffusion Boosted Trees

Xizewen Han and Mingyuan Zhou
xizewen.han@utexas.edu, mingyuan.zhou@mccombs.utexas.edu
The University of Texas at Austin
Austin, TX 78712

Abstract

Combining the merits of both denoising diffusion probabilistic models and gradient boosting, the diffusion boosting paradigm is introduced for tackling supervised learning problems. We develop Diffusion Boosted Trees (DBT), which can be viewed as both a new denoising diffusion generative model parameterized by decision trees (one single tree for each diffusion timestep), and a new boosting algorithm that combines the weak learners into a strong learner of conditional distributions without making explicit parametric assumptions on their density forms. We demonstrate through experiments the advantages of DBT over deep neural network-based diffusion models as well as the competence of DBT on real-world regression tasks, and present a business application (fraud detection) of DBT for classification on tabular data with the ability of learning to defer.

1 Introduction

A series of pivotal works in recent years (Song and Ermon, 2019; Ho et al., 2020; Song et al., 2021; Dhariwal and Nichol, 2021; Rombach et al., 2022; Karras et al., 2022) has propelled diffusion-based generative models (Sohl-Dickstein et al., 2015) to the forefront of generative AI, capturing a significant amount of academic and industrial interest by the success of this class of models in content generation. Meanwhile, another line of work, Classification and Regression Diffusion Models (CARD) (Han et al., 2022), has been proposed to tackle supervised learning problems with a denoising diffusion probabilistic modeling framework, shedding new lights on both the foundational machine learning paradigm and the new elite in the generative AI family.

More specifically, CARD learns the target conditional distribution of the response variable $\boldsymbol{y}$ given the covariates $\boldsymbol{x}$, $p(\boldsymbol{y} \mid \boldsymbol{x})$, without imposing explicit parametric assumptions on its probability density function, and makes predictions by utilizing the stochastic nature of its output to directly generate samples that resemble $\boldsymbol{y}$ from this target distribution. This framework has demonstrated outstanding results on both regression and image classification tasks: in regression, it shows the capability of modeling conditional distributions with flexible statistical attributes, and achieves state-of-the-art metrics on real-world datasets; for image classification, it introduces a novel paradigm to evaluate instance-level prediction confidence besides improving the prediction accuracy by a deterministic classifier.

However, CARD models are parameterized by deep neural networks. The work of Grinsztajn et al. (2022) has illustrated that tree-based models remain the state-of-the-art function choice for modeling tabular data, and could outperform neural networks by a wide margin. *Tabular data* is a crucial type of dataset for many supervised learning tasks, characterized by its table-format structure similar to a spreadsheet or a relational database, where each row represents an individual record or observation, and each column represents a feature or attribute of that record. Importantly, the features of tabular datasets are heterogeneous, including various types such as numerical (discrete or continuous) and categorical (nominal or ordinal), enabling the

representation of diverse information about each record. This contrasts with image data, where the raw information is solely represented as pixel values. CARD has not addressed classification tasks on tabular data, which represents an essential class of supervised learning tasks with wide applications in many areas. Therefore, significant potential remains to enhance the CARD framework to establish it as a universally applicable method in the realm of supervised learning.

In this work, we aim to improve the CARD framework by incorporating trees as its function choice: trees are another vital class of universal approximators besides neural networks (Watt et al., 2020; Nisan and Szegedy, 1994; Hornik et al., 1989), and offer several advantages, including the automatic handling of missing values without the need for imputation, no requirement for data normalization during preprocessing, effective performance with less data, better interpretability, and robustness to outliers and irrelevant features. Additionally, we fill an important gap by applying the framework to classification on tabular data, which was not explored in the experiments presented by Han et al. (2022). We start this quest by studying one of the most powerful supervised learning paradigms parameterized by trees: gradient boosting (Friedman, 2001).

Our main contributions are summarized as follows:

- We establish the connections between diffusion-based generative models and gradient boosting, a classic ensemble method for function estimation.
- We develop the **Diffusion Boosting** paradigm for supervised learning, which is simultaneously 1) a new denoising diffusion generative model that can be parameterized by decision trees — a single tree for each diffusion timestep — with a novel sequential training paradigm; and 2) a new boosting algorithm that combines the weak learners into a strong learner of conditional distributions without any assumptions on their parametric forms.
- Through experiments, we demonstrate that **D**iffusion **B**oosting **T**rees (DBT), the tree-based parameterization of our proposed paradigm, outperforms CARD on piecewise-defined functions and datasets with a large number of categorical features, while achieving competitive results in real-world regression tasks. DBT also excels in several other key areas: it offers interpretability at each diffusion timestep, maintains robust performance in the presence of missing data, and acts as an effective binary classifier on tabular data, featuring the ability to defer decisions with adjustable confidence levels.

2 Background

We contextualize gradient boosting as a method for tackling *supervised learning* tasks. Given a set of covariates $\boldsymbol{x} = \{x_1, \ldots, x_p\}$ and a response variable $\boldsymbol{y}$, we seek to learn a mapping F that takes $\boldsymbol{x}$ as input and predicts $\boldsymbol{y}$ as its output. It is common practice to impose a parametric form $\boldsymbol{\theta}$ on F, casting supervised learning as a *parameter optimization* problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\big[L\big(\boldsymbol{y}, F(\boldsymbol{x};\boldsymbol{\theta})\big)\big], \tag{1}$$

where $\boldsymbol{\theta}^*$ is achieved by minimizing the expected value of some loss function $L(\boldsymbol{y}, F)$. When gradient descent (Cauchy, 1847) is used to find the descent direction during the numerical optimization procedure, the optimal parameter is:

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 + \sum_{m=1}^{M} \rho_m \cdot \big(- \nabla_{\boldsymbol{\theta}_{m-1}} \Phi(\boldsymbol{\theta}_{m-1})\big), \tag{2}$$

where M is the total number of update steps, $\boldsymbol{\theta}_0$ is the initialization, and ρ_m is the step size.

2.1 Gradient Boosting

While gradient descent can be described as a numerical optimization method *in the parameter space*, gradient boosting (Friedman, 2001) is essentially gradient descent *in the function space*. With the objective function at the instance level,

$$\Phi\big(F(\boldsymbol{x})\big) = \mathbb{E}_{p(\boldsymbol{y}\,|\,\boldsymbol{x})}\big[L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)\big], \tag{3}$$

by considering $F(\boldsymbol{x})$ evaluated at each $\boldsymbol{x}$ as a parameter, its gradient can be computed as

$$\nabla_{F(\boldsymbol{x})}\Phi\big(F(\boldsymbol{x})\big) = \frac{\partial\Phi\big(F(\boldsymbol{x})\big)}{\partial F(\boldsymbol{x})} = \mathbb{E}_{p(\boldsymbol{y}\,|\,\boldsymbol{x})}\left[\frac{\partial L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)}{\partial F(\boldsymbol{x})}\right], \tag{4}$$

assuming sufficient regularity to interchange differentiation and integration. Following the gradient-based numerical optimization paradigm as in Eq. (2), we obtain the optimal solution in the function space:

$$F^*(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \sum_{m=1}^{M} \rho_m \cdot \big(-g_m(\boldsymbol{x})\big), \tag{5}$$

where $f_0(\boldsymbol{x})$ is the initial guess, and $g_m(\boldsymbol{x}) = \nabla_{F_{m-1}(\boldsymbol{x})}\Phi\big(F_{m-1}(\boldsymbol{x})\big)$ is the gradient at optimization step m.

Given a finite set of samples $\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_1^N$ from $p(\boldsymbol{x}, \boldsymbol{y})$, we have the data-based analogue of $g_m(\boldsymbol{x})$ defined only at these training instances: $g_m(\boldsymbol{x}_i) = \frac{\partial L\left(\boldsymbol{y}_i, \hat{F}_{m-1}(\boldsymbol{x}_i)\right)}{\partial \hat{F}_{m-1}(\boldsymbol{x}_i)}$. Since the goal of supervised learning is to generalize the predictive function F to unseen data, Friedman (2001) proposes to use a parameterized class of functions $h(\boldsymbol{x}; \boldsymbol{\alpha})$ to estimate the negative gradient term for any $\boldsymbol{x}$ at every gradient descent step. Specifically, $h(\boldsymbol{x}; \boldsymbol{\alpha})$ is trained with the squared-error loss at step m to produce $\{h(\boldsymbol{x}_i; \boldsymbol{\alpha}_m)\}_1^N$ most parallel to $\{-g_m(\boldsymbol{x}_i)\}_1^N$, and the solution $h(\boldsymbol{x}; \boldsymbol{\alpha}_m)$ can be applied to approximate $-g_m(\boldsymbol{x})$ for any $\boldsymbol{x}$:

$$\boldsymbol{\alpha}_m = \arg\min_{\boldsymbol{\xi}, \omega} \sum_{i=1}^{N} \big(-g_m(\boldsymbol{x}_i) - \omega \cdot h(\boldsymbol{x}_i; \boldsymbol{\xi})\big)^2. \tag{6}$$

Therefore, with finite data, the gradient descent update in the function space at step m is

$$\hat{F}_m(\boldsymbol{x}) = \hat{F}_{m-1}(\boldsymbol{x}) + \rho_m \cdot h(\boldsymbol{x}; \boldsymbol{\alpha}_m), \tag{7}$$

and the prediction of $\boldsymbol{y}$ given any $\boldsymbol{x}$ can be obtained through

$$\hat{\boldsymbol{y}} = \hat{F}^*(\boldsymbol{x}) = \hat{F}_0(\boldsymbol{x}) + \sum_{m=1}^{M} \rho_m \cdot h(\boldsymbol{x}; \boldsymbol{\alpha}_m). \tag{8}$$

The function $h(\boldsymbol{x}; \boldsymbol{\alpha})$ is termed a *weak learner* or *base learner*, and is often parameterized by a simple Classification And Regression Tree (CART) (Breiman et al., 1984). Eq. (8) has the form of an ensemble of weak learners, trained sequentially and combined via weighted sum. It is worth noting that when the loss function $L(\boldsymbol{y}, F)$ is chosen to be the squared-error loss, its negative gradient is the residual: $-\frac{\partial L}{\partial F(\boldsymbol{x})} = \boldsymbol{y} - F(\boldsymbol{x})$, and the optimal solution for minimizing this loss is the conditional mean, $\mathbb{E}[\boldsymbol{y}\,|\,\boldsymbol{x}]$.

2.2 Classification and Regression Diffusion Models (CARD)

With the same goal as gradient boosting of tackling supervised learning problems, CARD (Han et al., 2022) approaches them from a different angle: by adopting a generative modeling framework, a CARD model directly outputs samples from $p(\boldsymbol{y}\,|\,\boldsymbol{x})$, instead of summary statistics such as $\mathbb{E}[\boldsymbol{y}\,|\,\boldsymbol{x}]$. This finer level of granularity in model output helps to paint a more complete picture of $p(\boldsymbol{y}\,|\,\boldsymbol{x})$. A unique advantage of CARD is that it does not require $p(\boldsymbol{y}\,|\,\boldsymbol{x})$ to adhere to a parametric form.

At its core, CARD is a generative model that aims to learn a function parameterized by $\boldsymbol{\theta}$ that maps a sample from a simple known distribution (*i.e.*, the *noise distribution*) to a sample from the target distribution

$p(\boldsymbol{y} \mid \boldsymbol{x})$. As a generative model, its objective function is rooted in distribution matching: re-denoting the ground truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ as $q(\boldsymbol{y}_0 \mid \boldsymbol{x})$, we wish to learn $\boldsymbol{\theta}$ so that $p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \mid \boldsymbol{x})$ approximates $q(\boldsymbol{y}_0 \mid \boldsymbol{x})$ well, *i.e.*,

$$D_{\mathrm{KL}}\big(q(\boldsymbol{y}_0 \mid \boldsymbol{x}) \,\big\|\, p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \mid \boldsymbol{x})\big) \approx 0. \tag{9}$$

As a class of diffusion models, CARD produces a *less noisy version* of $\boldsymbol{y}$ after each function evaluation, which is then fed into the *same* function to produce the next one. The final output $\boldsymbol{y}_0$ can be viewed as a noiseless sample of $\boldsymbol{y}$ from $p(\boldsymbol{y} \mid \boldsymbol{x})$. This autoregressive fashion of computing can be described as *iterative refinement* or *progressive denoising*.

The noisy samples of $\boldsymbol{y}$ from the intermediate steps are treated as latent variables, linked together by a Markov chain with $T+1$ timesteps constructed in the direction *opposite* to the data *generation* process: with the stepwise transition distribution $q(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{x})$, the *forward diffusion process* is defined as $q(\boldsymbol{y}_{1:T} \mid \boldsymbol{y}_0, \boldsymbol{x}) = \prod_{t=1}^{T} q(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{x})$. Meanwhile, the *reverse diffusion process* is defined as $p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \mid \boldsymbol{x}) = p(\boldsymbol{y}_T \mid \boldsymbol{x}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{x})$, in which $p(\boldsymbol{y}_T \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{I})$ is *the* noise distribution, also referred to as the *prior distribution*.

Utilizing the decomposition of cross entropy and Jensen's inequality, the variational bound (*i.e.*, the negative ELBO) (Blei et al., 2017) can be derived from Eq. (9) as a new objective function, which can be further decomposed into terms at different timesteps (Sohl-Dickstein et al., 2015; Ho et al., 2020):

$$L =: \mathbb{E}_{q(\boldsymbol{y}_{0:T} \mid \boldsymbol{x})}\left[\log \frac{q(\boldsymbol{y}_{1:T} \mid \boldsymbol{y}_0, \boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \mid \boldsymbol{x})}\right] = \mathbb{E}_{q(\boldsymbol{y}_{0:T} \mid \boldsymbol{x})}\left[L_T + \sum_{t=2}^{T} L_{t-1} + L_0\right]. \tag{10}$$

It can be shown that the main focus for optimizing $\boldsymbol{\theta}$ is on the L_{t-1} terms for $t = 2, \ldots, T$, where

$$L_{t-1} \coloneqq D_{\mathrm{KL}}\big(q(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) \,\big\|\, p_{\boldsymbol{\theta}}(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{x})\big). \tag{11}$$

An in-depth walkthrough of the objective function construction can be found in Appendix A.1.4.

The distribution $q(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x})$ in each L_{t-1} is called the *forward process posterior distribution*, which is tractable and can be derived by applying Bayes' rule:

$$q(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) \propto q\big(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{x}\big) \cdot q\big(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_0, \boldsymbol{x}\big). \tag{12}$$

Both $q\big(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{x}\big)$ and $q\big(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_0, \boldsymbol{x}\big)$ in Eq. (12) are Gaussian: the former is the stepwise transition distribution in the forward process, defined as $q(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{y}_t; \sqrt{\alpha_t}\boldsymbol{y}_{t-1} + (1-\sqrt{\alpha_t})\boldsymbol{\mu}_T, \beta_t \boldsymbol{I}\big)$, where β_t is the t-th term of a predefined noise schedule $\beta_1, \ldots, \beta_T$, and $\alpha_t \coloneqq 1 - \beta_t$. This design gives rise to a closed-form distribution to sample $\boldsymbol{y}_t$ at any arbitrary timestep t:

$$q(\boldsymbol{y}_t \mid \boldsymbol{y}_0, \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{y}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{y}_0 + (1-\sqrt{\bar{\alpha}_t})\boldsymbol{\mu}_T, (1-\bar{\alpha}_t)\boldsymbol{I}\big), \tag{13}$$

in which $\bar{\alpha}_t \coloneqq \prod_{j=1}^{t} \alpha_j$. Each of the forward process posteriors thus has the form of

$$q(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) = \mathcal{N}\Big(\boldsymbol{y}_{t-1}; \tilde{\boldsymbol{\mu}}(\boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{\mu}_T), \tilde{\beta}_t \boldsymbol{I}\Big), \tag{14}$$

where the variance $\tilde{\beta}_t \coloneqq \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, and the mean

$$\tilde{\boldsymbol{\mu}}(\boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{\mu}_T) \coloneqq \gamma_0 \cdot \boldsymbol{y}_0 + \gamma_1 \cdot \boldsymbol{y}_t + \gamma_2 \cdot \boldsymbol{\mu}_T, \tag{15}$$

in which the value of coefficients can be found in Appendix A.1.4.

Now to minimize each L_{t-1}, $p_{\boldsymbol{\theta}}(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{x})$ needs to approximate the Gaussian distribution $q(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x})$, whose variance $\tilde{\beta}_t$ is already known. Therefore, the learning task is reduced to optimizing $\boldsymbol{\theta}$ for the estimation of the forward process posterior mean $\tilde{\boldsymbol{\mu}}(\boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{\mu}_T)$. In other words, the data *generation* process can now be modeled analytically, in the sense that an explicit distributional form (*i.e.*, Gaussian) can be imposed upon adjacent latent variables. CARD adopts the noise-prediction loss introduced in Ho et al. (2020), a simplification of L_{t-1}:

$$\mathcal{L}_{\mathrm{CARD}} = \mathbb{E}_{p(t, \boldsymbol{y}_0 \mid \boldsymbol{x}, \boldsymbol{\epsilon})}\left[\big|\big|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\big(\boldsymbol{x}, \boldsymbol{y}_t, f_\phi(\boldsymbol{x}), t\big)\big|\big|^2\right], \tag{16}$$

in which $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is sampled as the *forward* process noise term, $\boldsymbol{y}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{y}_0 + (1 - \sqrt{\bar{\alpha}_t})\boldsymbol{\mu}_T + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$ is the sample from the *forward* process distribution (13), and $f_\phi(\boldsymbol{x})$ is the point estimate of $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$.

3 The Diffusion Boosting Framework

Having established the objective functions of gradient boosting and CARD in Section 2, we now proceed to discuss the connections between these two methods.

3.1 Connections between Gradient Boosting and CARD

To begin with, we note that the functions in both methods can be viewed as ***gradient estimators***. For gradient boosting, each weak learner approximates the negative gradient of the objective at a particular optimization step (6). Meanwhile, Song and Ermon (2019) approach the training of diffusion models from the perspective of denoising score matching (Vincent, 2011).

Specifically, in our supervised learning context, the conditional distribution of the noisy response variable given the covariates $\boldsymbol{x}$ can be modeled as a semi-implicit distribution (Yin and Zhou, 2018; Yu et al., 2023):

$$q(\boldsymbol{y}_t \,|\, \boldsymbol{x}) = \int q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, \boldsymbol{x}) q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})\, d\boldsymbol{y}_0, \tag{17}$$

which generally lacks an analytic form since $q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$ is unknown and is the target of our estimation from the observed $(\boldsymbol{x}, \boldsymbol{y}_0)$ pairs. This semi-implicit form allows for the estimation of its *score*, the gradient of its log-likelihood with respect to the noisy sample that can be expressed as $\nabla_{\boldsymbol{y}_t} \log q(\boldsymbol{y}_t \,|\, \boldsymbol{x})$, using a score matching network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}_t)$ (Zhou et al., 2024), as discussed below.

Realizing score matching for supervised learning involves estimating the score by minimizing the explicit score matching (ESM) loss:

$$\mathcal{L}_{\text{ESM}} = \mathbb{E}_{t, \boldsymbol{y}_0, \boldsymbol{y}_t} \left[\lambda(t) || \nabla_{\boldsymbol{y}_t} \log q(\boldsymbol{y}_t \,|\, \boldsymbol{x}) - \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}_t) ||^2 \right], \tag{18}$$

where $\lambda(t)$ is a positive weighting function. However, this objective function is intractable in practice since $\nabla_{\boldsymbol{y}_t} \log q(\boldsymbol{y}_t \,|\, \boldsymbol{x})$ is generally unknown. To address this issue, following the idea of denoising score matching (DSM) (Vincent, 2011; Song and Ermon, 2019), this intractable objective can be rewritten into an equivalent form:

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{t, \boldsymbol{y}_0, \boldsymbol{y}_t} \left[\lambda(t) || \nabla_{\boldsymbol{y}_t} \log q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, \boldsymbol{x}) - \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}_t) ||^2 \right], \tag{19}$$

where $q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, \boldsymbol{x})$ is the forward sampling distribution whose gradient is analytic. With the Gaussian formulation of the forward process sampling distributions, Eqs. (16) and (19) are connected via $\nabla_{\boldsymbol{y}_t} \log q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, \boldsymbol{x}) = -\frac{\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}_t}}$, thus denoting $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)} \coloneqq \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\big(\boldsymbol{x}, \boldsymbol{y}_t, f_\phi(\boldsymbol{x}), t\big)$, we have $||\nabla_{\boldsymbol{y}_t} \log q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, \boldsymbol{x}) - \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}_t)||^2 = \frac{1}{1-\bar{\alpha}_t} ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}||^2$. In other words, $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)} \equiv -\sqrt{1 - \bar{\alpha}_t} \cdot \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}_t)$ and CARD estimates the (scaled) gradient of $\log q(\boldsymbol{y}_t \,|\, \boldsymbol{x})$ at each diffusion timestep.

Additionally, we highlight that the core mechanism of both methods is ***iterative refinement***: gradient boosting essentially performs gradient descent in the function space (Section 2.1), and CARD generates each sample by making small and incremental changes to the initial noise sample over multiple steps to progressively refine it into a sample that resembles one from the target distribution (Section 2.2). The final form of gradient boosting is a strong function estimator (of a summary statistic), while CARD constructs a strong *implicit* conditional distribution estimator.

Moreover, we point out that the iterative refinement mechanism implies ***the adequacy of a weak learner*** at each refining step. This is already evident for gradient boosting, as each base learner is usually a single tree (Friedman, 2001). For CARD, we revisit the crucial term L_{t-1} in the learning objective (10): in Section 2.2, we showed that the task of learning the diffusion model parameter $\boldsymbol{\theta}$ is reframed as approximating the

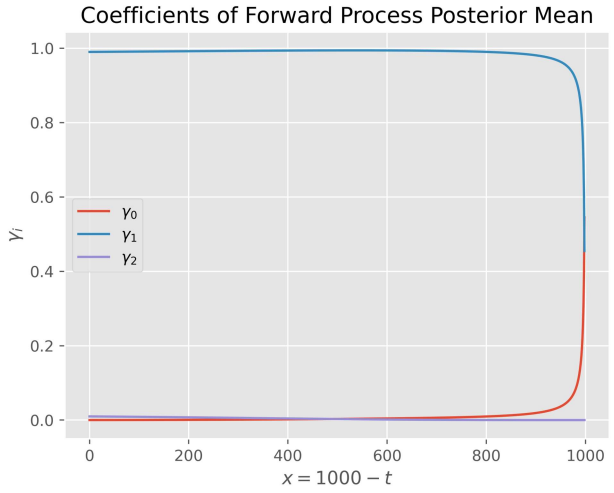

Figure 1: CARD posterior mean coefficients in Eq. (15) across all timesteps during sampling.

forward process posteriors $q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x})$ with $p_{\boldsymbol{\theta}}(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{x})$ — more specifically, estimating the mean term $\tilde{\boldsymbol{\mu}}(\boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{\mu}_T)$ with the diffusion model, since the variance can already be computed analytically. Note that Eq. (15) formulates $\tilde{\boldsymbol{\mu}}$ as a linear combination of the target variable $\boldsymbol{y}_0$, the noisy sample $\boldsymbol{y}_t$ from the previous timestep t, and the prior mean $\boldsymbol{\mu}_T$, with their corresponding coefficients γ_0, γ_1, and γ_2. During the data generation process, $\boldsymbol{y}_0$ is unknown (and is the target that we seek to generate), thus at each timestep of this reverse process, CARD approximates this term via the reparameterization of the *forward* process sampling distribution (13), in which the noise term is estimated by the noise-predicting network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$:

$$\hat{\boldsymbol{y}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\Big(\boldsymbol{y}_t - (1-\sqrt{\bar{\alpha}_t})\boldsymbol{\mu}_T - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}^{(t)}\Big). \tag{20}$$

In other words, CARD generates new samples of $\boldsymbol{y}$ by exploiting the Bayesian formulation of the reverse process stepwise transition distribution, *i.e.*, the forward process posterior $q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x})$: more specifically, CARD computes the *surrogate* of the true $\boldsymbol{y}_0$, providing the only missing piece in the analytical form of the posterior mean $\tilde{\boldsymbol{\mu}}$ to kickstart the sampling process.

We take a closer look at the role of each term in the linear combination that forms the forward process posterior mean $\tilde{\boldsymbol{\mu}}$ (15), by plotting the coefficient values across all timesteps during the reverse process in Figure 1, where we set the total number of timesteps $T = 1000$, and apply a linear noise schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. The process starts at timestep T (with label 0 at the x-axis). Notice that γ_2 stays consistently close to 0 for all timesteps, which makes intuitive sense, since the information of the prior mean $\boldsymbol{\mu}_T$ has been largely absorbed by the noise sample $\boldsymbol{y}_T$. The more interesting part is the arcs of γ_0 and γ_1: across the vast majority of the timesteps (*i.e.*, from $t = 1000$ to around $t = 100$), γ_0 stays very close to 0, while γ_1 stays very close to 1 — this shows that prior to about the last 10% of timesteps of the reverse process, the value of the posterior mean $\tilde{\boldsymbol{\mu}}$ is predominantly determined by $\boldsymbol{y}_t$. In other words, the *mean* of the next $\boldsymbol{y}$ sample is almost the same as the *value* of the current $\boldsymbol{y}$ sample; at the same time, the contribution of $\hat{\boldsymbol{y}}_0$ — the surrogate of the true $\boldsymbol{y}_0$ predicted by the diffusion model (20) — to the computation of $\tilde{\boldsymbol{\mu}}$ is basically negligible. The value of γ_0 begins to surge around the very end of the reverse process, by which time $\hat{\boldsymbol{y}}_0$ shall be close enough to $\boldsymbol{y}_0$ when the model is well-trained.

Based on the above observations regarding the computation of $\tilde{\boldsymbol{\mu}}$ — specifically, that for most of the timesteps during sampling, the coefficient of $\boldsymbol{y}_0$ is close to 0, and during the remaining timesteps when $\hat{\boldsymbol{y}}_0$ is close enough to $\boldsymbol{y}_0$, the model only needs to capture small changes — it is reasonable to argue that a weak learner at each timestep during the reverse diffusion process shall be sufficient in terms of computational power. In other words, we do not necessarily need a strong model to estimate $\boldsymbol{y}_0$ at each timestep.

Having examined the similarities between gradient boosting and CARD, we now turn our attention to an analysis of their differences. More specifically, we come up with the following question.

3.2 What can CARD learn from Gradient Boosting?

In Section 3.1, we established that for CARD, a weak learner should suffice to meet the computational requirements at each reverse timestep. This is because the noise-predicting network in CARD performs a similar task to each weak learner in gradient boosting, namely, approximating a gradient term. Moreover, the $\boldsymbol{y}_0$ term, as part of the posterior mean computation, may not require a *precise* estimation for most of the timesteps. However, these insights are not currently reflected in CARD's implementation: CARD uses a deep neural network to parameterize the noise predictor $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ due to the need for amortization, *i.e.*, the same function is applied across all timesteps, necessitating it to possess abundant modeling and computational capacity. Therefore, based on our findings in Section 3.1, it may be beneficial to follow the gradient boosting paradigm and improve the function choice of CARD by modeling the gradient term at each timestep with a *different* weak learner.

Additionally, since CART (Breiman et al., 1984) is the orthodox function choice for gradient boosting and tree-based models are widely acknowledged as superior to deep neural networks for tabular data (Grinsztajn et al., 2022; Qin et al., 2021; Shwartz-Ziv and Armon, 2022; Borisov et al., 2022; Yang et al., 2018), incorporating CART into the CARD framework could potentially enhance CARD's ability to model tabular data, which represents the very type of data from which many supervised learning demands arise. Importantly, this new function choice could more clearly validate the DDPM framework (Ho et al., 2020) as a method whose success is grounded in statistical principles and computational practices (for example, the Bayesian formulation with Gaussian conjugacy shown in Eq. (12), the variational lower bound, and reparameterization). This perspective emphasizes the foundational statistical and computational mechanisms over the reliance on the architectural complexities of deep neural networks or the engineering nuances typically employed during training and sampling.

Furthermore, each weak learner in gradient boosting is trained sequentially (7). Although the reverse process of CARD (and diffusion models in general) conducts sampling in a sequential fashion, its training treats the latent variables at different timesteps as independent random variables: during inference, $\boldsymbol{y}_t$ is sampled from the approximation of the forward process posterior distribution (14) (where the true $\boldsymbol{y}_0$ is replaced with its surrogate), whose mean $\tilde{\boldsymbol{\mu}}$ depends on $\boldsymbol{y}_{t+1}$, the sample from the previous timestep (15); however, during training, $\boldsymbol{y}_t$ is drawn from the forward process sampling distribution (13). This creates a *discrepancy* in the noisy response input $\boldsymbol{y}_t$ for the $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ network *between training and sampling*. This phenomenon of model input mismatch is commonly referred to as *exposure bias* (Williams and Zipser, 1989; Bengio et al., 2015; Ranzato et al., 2016; Schmidt, 2019; Fan et al., 2020; Ning et al., 2023a,b). To address this issue, one could refer to the sequential training mechanism of gradient boosting (6, 7) to devise a method for aligning the computational graphs during training and sampling.

3.3 Diffusion Boosted Trees

Following our discussion in Section 3.2, we now propose the **Diffusion Boosting** framework.

First, we replace the amortized single model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ in the CARD framework with a series of weak learners $\{f_{\boldsymbol{\theta}_t}\}_{t=1}^{T}$, *one for each diffusion timestep*. For the input to each $f_{\boldsymbol{\theta}_t}$, we use the same set of variables as CARD except the timestep t. Since we train a distinct model for each timestep, the representation of the temporal dynamic is no longer needed. We concatenate the remaining variables — the noisy sample of $\boldsymbol{y}$, the covariates $\boldsymbol{x}$, and the conditional mean estimation $f_\phi(\boldsymbol{x})$ — to form the model input. For simplicity, each $f_{\boldsymbol{\theta}_t}$ directly predicts $\boldsymbol{y}_0$ as its target, instead of the forward process noise sample $\boldsymbol{\epsilon}$ (16), thus sparing the step of

Algorithm 1 Diffusion Boosted Trees Training

Input: Training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_{0,i})\}_{i=1}^N$
Output: Trained mean estimator $f_\phi(\boldsymbol{x})$ and tree ensemble $\{f_{\boldsymbol{\theta}_t}\}_{t=1}^T$

Pre-train $f_\phi(\boldsymbol{x})$ to estimate $\mathbb{E}[\boldsymbol{y}_0 \,|\, \boldsymbol{x}]$
for $t = T$ to 1 **do**
  if $t = T$ **then**
    Sample $\hat{\boldsymbol{y}}_t \sim \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{I})$, the prior distribution
  else
    Sample $\boldsymbol{y}_{t+1} \sim q(\boldsymbol{y}_{t+1} \,|\, \boldsymbol{y}_0, \boldsymbol{x})$
    Predict $\boldsymbol{y}_0$ with the newly trained model $f_{\boldsymbol{\theta}_{t+1}}$:

$$\hat{\boldsymbol{y}}_{0,t+1} = f_{\boldsymbol{\theta}_{t+1}}\big(\boldsymbol{y}_{t+1}, \boldsymbol{x}, f_\phi(\boldsymbol{x})\big)$$

    Compute $\tilde{\boldsymbol{\mu}}(\boldsymbol{y}_{t+1}, \boldsymbol{y}_0, \boldsymbol{\mu}_T)$, the forward process posterior mean:

$$\hat{\tilde{\boldsymbol{\mu}}}_t = \gamma_0 \cdot \hat{\boldsymbol{y}}_{0,t+1} + \gamma_1 \cdot \boldsymbol{y}_{t+1} + \gamma_2 \cdot \boldsymbol{\mu}_T$$

    Sample $\hat{\boldsymbol{y}}_t \sim \mathcal{N}(\hat{\tilde{\boldsymbol{\mu}}}_t, \tilde{\beta}_{t+1}\boldsymbol{I})$
  end if
  Train $f_{\boldsymbol{\theta}_t}$ with MSE loss to predict the true response:

$$\mathcal{L}_{\boldsymbol{\theta}}^{(t)} = \mathbb{E}\left[\left|\left|\boldsymbol{y}_0 - f_{\boldsymbol{\theta}_t}\big(\hat{\boldsymbol{y}}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\big)\right|\right|^2\right]$$

end for

Algorithm 2 Diffusion Boosted Trees Sampling

Input: Test data $\{\boldsymbol{x}_j\}_{j=1}^M$, trained $f_\phi(\boldsymbol{x})$ and $\{f_{\boldsymbol{\theta}_t}\}_{t=1}^T$
Output: Response variable prediction $\hat{\boldsymbol{y}}_{0,1}$

Draw $\hat{\boldsymbol{y}}_T \sim \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{I})$
for $t = T$ to 1 **do**
  Predict the response $\hat{\boldsymbol{y}}_{0,t} = f_{\boldsymbol{\theta}_t}\big(\hat{\boldsymbol{y}}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\big)$
  if $t > 1$ **then**
    Draw the noisy sample $\hat{\boldsymbol{y}}_{t-1} \sim q\big(\boldsymbol{y}_{t-1} \,|\, \hat{\boldsymbol{y}}_t, \hat{\boldsymbol{y}}_{0,t}, f_\phi(\boldsymbol{x})\big)$
  end if
end for
return $\hat{\boldsymbol{y}}_{0,1}$

converting the estimated $\boldsymbol{\epsilon}$ to $\hat{\boldsymbol{y}}_0$ via Eq. (20).

We then choose CART (Breiman et al., 1984) as the default function to parameterize each weak learner. For the inaugural algorithm, we set the number of trees to 1 for each $f_{\boldsymbol{\theta}_t}$, which is the universal setting applied to all the experiments in Section 5. We argue that model performance could potentially be improved by using more trees when γ_0 (15) surges near the end of the generation process (Figure 1), *i.e.*, when the estimate of $\boldsymbol{y}_0$ has more impact on the computation of $\tilde{\boldsymbol{\mu}}$. We defer this attempt for future iterations of the algorithm.

Furthermore, to address the issue of exposure bias, we design a *sequential training* paradigm inspired by gradient boosting: we train the first weak learner at timestep T, then use its output to construct the input for training the next weak learner at timestep $T-1$, and so on. This approach creates a dependency for adjacent weak learners *during training*, emulating the computational graphs of consecutive timesteps during sampling.

Initially, we considered duplicating the sampling procedure during training, *i.e.*, sampling a set of noises from the prior $\mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{I})$ as the input to train $f_{\boldsymbol{\theta}_t}$, then using the trained model with the same set of noise samples

to predict $\hat{\boldsymbol{y}}_{0,T}$, which would be used for the training of $f_{\boldsymbol{\theta}_{T-1}}$, and so on. However, this would result in all $f_{\boldsymbol{\theta}_t}$'s being trained with the same set of noise samples, limiting the diversity of training data and introducing additional overhead for storing the noisy samples during training. Therefore, we directly sample $\boldsymbol{y}_{t+1}$ from $q(\boldsymbol{y}_{t+1} \,|\, \boldsymbol{y}_0, \boldsymbol{x})$ (13), to be used as the input to the trained $f_{\boldsymbol{\theta}_{t+1}}$ when training $f_{\boldsymbol{\theta}_t}$.

Incorporating the above-mentioned modifications into CARD, we propose **D**iffusion **B**oosted **T**rees (DBT) as a class of diffusion boosting models. The training and sampling procedures are presented in Algorithms 1 and 2, respectively, for both regression and classification tasks.

Notably, we made a slight adjustment to the CARD framework by writing the prior mean in the generic form $\boldsymbol{\mu}_T$ instead of $f_\phi(\boldsymbol{x})$. This introduces an extra degree of freedom by allowing the choice of the prior mean to differ from the conditional mean estimation $f_\phi(\boldsymbol{x})$, while still using $f_\phi(\boldsymbol{x})$ as an input to each $f_{\boldsymbol{\theta}_t}$ since it possesses the information about $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$.

The design choices and evaluation methods for diffusion boosting in both regression and classification tasks are presented as follows.

3.3.1 Diffusion Boosting Regressor

For regression, the conditional mean estimator $f_\phi(\boldsymbol{x})$ is pre-trained with the MSE loss. It can be parameterized by any type of model, including neural networks, tree-based models, linear models with the OLS solution, *etc.*

To evaluate a DBT model, we apply the conventional metrics **RMSE** and **NLL**, as well as the **QICE** metric proposed in Han et al. (2022), which is a quantile-based coverage metric that measures the level of distribution matching between the true and the learned distributions. For data whose $\boldsymbol{x}$ and $\boldsymbol{y}$ are both 1D, a scatter plot can also be made for visual inspection of true and generated samples.

3.3.2 Diffusion Boosting Classifier

For classification, we tailor the model to specifically tackle binary classification tasks on tabular data. This is a family of supervised learning tasks that CARD has not attempted, and it represents one of the most successful and common applications of tree-based models (Grinsztajn et al., 2022).

Unlike CARD, where the class representation of the response variable $\boldsymbol{y}$ has the same dimensionality as the number of classes, we adopt a 1D representation of $\boldsymbol{y}$. This choice is due to the fact that popular gradient boosting libraries do not natively support multi-dimensional outputs, and a scalar representation is sufficient for binary classification.

Binary classes are first encoded as scalar labels, 0 and 1, to pre-train $f_\phi(\boldsymbol{x})$ using the binary cross-entropy loss. This function outputs *the predicted probability of the class with label* 1 to guide the training of DBT. These class labels are then converted to the logit scale to serve as class representations, also known as class prototypes in Han et al. (2022), which have an unbounded range. This transformation aligns with the Gaussian assumption in the denoising diffusion framework, allowing us to use the same objective function to train DBT for both regression and classification.

We leverage the stochastic nature inherent in a generative model's output to evaluate DBT, following the paradigm proposed in Han et al. (2022), with modifications for 1D output. For *each test instance* $\boldsymbol{x}_j$, we generate S samples $\{\boldsymbol{y}_{j,s}\}_{s=1}^S$ as class predictions in logits. The evaluation process consists of two steps:

1. Class Prediction:

    (a) Apply the sigmoid function to convert each output to a probability, representing the predicted

probability of label 1: ${p_{j,s}}^{(1)} = \text{sigmoid}(\boldsymbol{y}_{j,s})$.

(b) Convert these probabilities to binary labels using a threshold: 0.5 for a balanced dataset, or the mean of the binary labels from the training set for an imbalanced one.

(c) Classify via the majority vote by the generated samples: selecting the more frequently predicted label as the class prediction.

2. Model Confidence Measurement:

(a) **Prediction Interval Width (PIW)**: Compute the PIW between two percentile levels (2.5^{th} and 97.5^{th} by default) of the S samples (in either logit or probability). A narrower PIW indicates higher model confidence for that particular test instance, as it suggests less variation among the S samples. Relative confidence can be assessed by comparing the PIWs of different test instances.

(b) **Paired Two-Sample t-Test**: Compute the corresponding class prediction of label 0 for each of the S samples: ${p_{j,s}}^{(0)} = 1 - {p_{j,s}}^{(1)}$. Perform a paired two-sample t-test to determine if $\{{p_{j,s}}^{(0)}\}_{s=1}^{S}$ and $\{{p_{j,s}}^{(1)}\}_{s=1}^{S}$ have significantly different sample means. Rejecting the t-test indicates the model is confident in its class prediction for $\boldsymbol{x}_j$. The significance level, set to $\alpha = 0.05$ by default, can be interpreted as an adjustable confidence level and can be modified based on the practical problem.

4 Related Work

Our work shares the same goal as CARD (Han et al., 2022) to model the conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ under a supervised learning setting from a generative modeling perspective, *i.e.*, directly generating samples from $p(\boldsymbol{y} \mid \boldsymbol{x})$, rather than providing a point estimate. This method allows for the direct calculation of various summary statistics from the generated samples, including the conditional expectation $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}]$, different quantiles such as the median, and measures of predictive uncertainty, thereby providing a more comprehensive representation of the target distribution. This generative method to capture $p(\boldsymbol{y} \mid \boldsymbol{x})$ has also been employed by Zhou et al. (2023) and Liu et al. (2021), both of which are based on GANs (Goodfellow et al., 2014) instead of diffusion models (Sohl-Dickstein et al., 2015) as the generative modeling framework. These works do not impose any parametric assumptions on the distributional form of $p(\boldsymbol{y} \mid \boldsymbol{x})$, allowing it to be learned completely from data.

The family of Bayesian neural networks (BNNs) (Blundell et al., 2015; Gal and Ghahramani, 2016; Hernández-Lobato and Adams, 2015; Kendall and Gal, 2017; Kingma et al., 2015; Gal et al., 2017) is another class of methods that is capable of capturing predictive uncertainty. Unlike our work and CARD, which focus exclusively on modeling aleatoric uncertainty, BNNs address both aleatoric and epistemic uncertainty (Hüllermeier and Waegeman, 2021) by treating network parameters as random variables. Furthermore, BNNs often explicitly assume $p(\boldsymbol{y} \mid \boldsymbol{x})$ to be Gaussian, facilitating the decomposition of these two types of predictive uncertainty (Depeweg et al., 2018). Another line of work related to ours applies a semi-implicit construction (Yin and Zhou, 2018), $\int p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) p(\boldsymbol{z} \mid \boldsymbol{x}) \mathrm{d}\boldsymbol{z}$, to model local uncertainties (Wang and Zhou, 2020). In this approach, local variables are typically infused with uncertainty through contextual dropout (Fan et al., 2021; Boluki et al., 2020), while auto-encoding variational inference (Kingma and Welling, 2014) is employed to obtain point estimates of the underlying neural networks.

Ensemble-based methods also assess predictive uncertainty by assuming a Gaussian form for $p(\boldsymbol{y} \mid \boldsymbol{x})$, attaining aleatoric uncertainty by learning both the mean and the variance parameter via the Gaussian negative log-likelihood objective, and quantifying epistemic uncertainty by training multiple base models. It can be parameterized by either deep neural networks (Lakshminarayanan et al., 2017) or trees (Duan et al., 2020; Malinin et al., 2021). Bayesian Additive Regression Trees (BART) (Chipman et al., 2010; Sparapani et al., 2021; He et al., 2019; Starling et al., 2020; Hill, 2011) is another class of ensemble methods, which approximates $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}]$ by a sum of regression trees. It assumes the conventional additive Gaussian noise regression model, and

obtains MCMC samples of the sum-of-trees model and the noise variance parameter from their corresponding posterior distributions. It can be viewed as a Bayesian form of gradient boosting.

Additionally, several studies feature components akin to our work. Forest-Diffusion (Jolicoeur-Martineau et al., 2024) is the first work to parameterize diffusion-based generative models with gradient boosted trees (GBTs), and is designed for unconditional tabular data generation and imputation. This approach involves training a distinct GBT model at each diffusion timestep, with each model comprising 100 trees. eDiff-I (Balaji et al., 2022) approaches text-to-image generation by training an ensemble of three expert denoisers, each specialized for a specific timestep interval, instead of using a single model across all timesteps. This method aims to capture the complex temporal dynamics observed at different stages of generation: throughout the process, the dependence of the denoising model gradually shifts from the input text prompt embedding towards the visual features. SGLB (Ustimenko and Prokhorenkova, 2021) introduces a gradient boosting algorithm that leverages the Langevin diffusion equation to facilitate convergence to the global optimum during numerical optimization, regardless of the loss function's convexity.

5 Experiments

Our current implementation of DBT is based on the LightGBM (Ke et al., 2017) library. For each $f_{\boldsymbol{\theta}_t}$, we fix the number of trees at 1, and set the default number of leaves to 101 and the learning rate to 1, leaving all other hyperparameters at their default settings. Since LightGBM requires loading the entire dataset for training, we need to construct the training data in its entirety, instead of iteratively updating the model via mini-batches. We set the number of noise samples for each instance $n_{noise} = 100$, and duplicate the entire dataset n_{noise} times to construct the training set. To address the inefficiency of duplicating the training set, we plan to incorporate the XGBoost (Chen and Guestrin, 2016) library in future iterations of our code. Recent versions of XGBoost offer the data iterator functionality for memory-efficient training with external memory, eliminating the need to duplicate the training set, as demonstrated by Jolicoeur-Martineau et al. (2024) in their updated Forest-Diffusion repository.

The input to each $f_{\boldsymbol{\theta}_t}$, $\big(\hat{\boldsymbol{y}}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\big)$, is formed via concatenation. The conditional mean estimator $f_\phi(\boldsymbol{x})$ is conceptually model-free. When training on complete data, we use the same parameterization as CARD: a feedforward neural network with two hidden layers, containing 100 and 50 hidden units, respectively. A Leaky ReLU activation function with a 0.01 negative slope is employed after each hidden layer. For datasets with missing covariates, we parameterize $f_\phi(\boldsymbol{x})$ using a gradient-boosted trees model. This model consists of 100 trees with 31 leaf nodes each, and it is trained with a learning rate of 0.05.

For the diffusion model hyperparameters, we set the number of timesteps $T = 1000$, and use a linear noise schedule with $\beta_1 = 10^{-4}$ and $\beta_T = 0.02$. While we currently apply the same set of hyperparameters for $f_{\boldsymbol{\theta}_t}$ across all timesteps, each tree is free to be trained with different hyperparameter settings, achieving a new level of flexibility over an amortized deep neural network. We reserve the exploration in this direction for future work.

5.1 Regression

For regression, we incorporate experiments on both toy and real-world datasets.

5.1.1 Toy Examples

We designed several toy examples with diverse statistical attributes — including linear and non-linear piecewise-defined functions with additive Gaussian noise, multimodality, and heteroscedasticity — to demonstrate the following: 1) like CARD, DBT is versatile in modeling conditional distributions; 2) DBT is better suited for

functions where the response variable, $\boldsymbol{y}$, exhibits distinct, non-continuous values across subintervals of $\boldsymbol{x}$; and 3) DBT requires less data than CARD to reach effective performance levels.

We train both DBT and CARD on these datasets and create scatter plots with both true and generated samples, as illustrated in Figure 2. The tasks are denoted from left to right as a, b, c, d, and e. For tasks a, d, and e, where $\boldsymbol{y}$ is unimodal, we shade the region between the 2.5^{th} and 97.5^{th} percentiles of the generated $\boldsymbol{y}$ in grey. Tasks b and c are based on the same true data-generating function; however, Task c utilizes only $\frac{1}{5}$ of the training data compared to Task b.

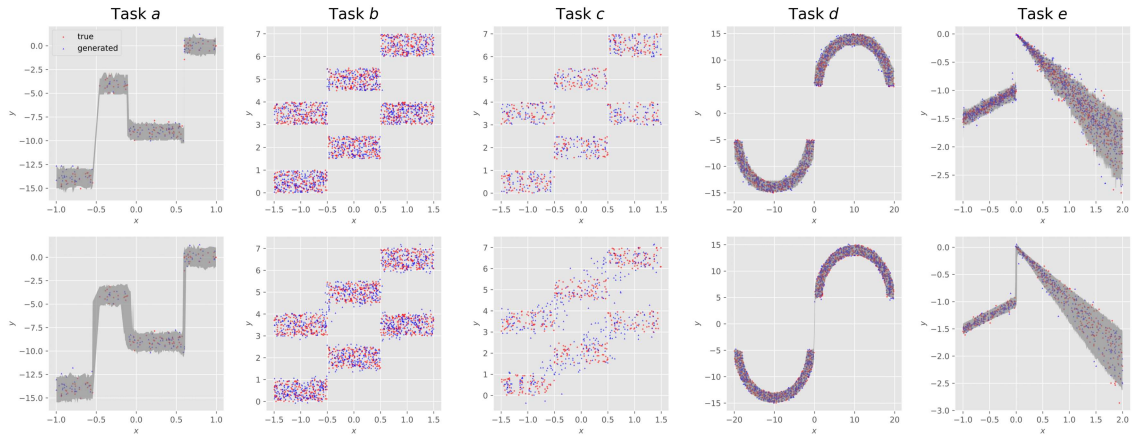

Figure 2: Comparison of DBT (**top row**) and CARD (**bottom row**) on toy regression examples.

We observe that DBT consistently generates samples that blend very well with the true data across all tasks, highlighting its capability to accurately capture the underlying data generation mechanisms.

Furthermore, for the uni-modal tasks a, d, and e, there is a notable distinction in the central 95% sample intervals between DBT and CARD. Specifically, in areas near the junctions of two adjacent $\boldsymbol{x}$ subintervals, CARD tends to create a visible "band" that bridges these subintervals. In contrast, DBT forms either a much narrower stripe or no stripe at all, more effectively capturing the "disjointness" of $\boldsymbol{y}$. This observation underscores a clear advantage of trees over deep neural networks as a function choice: trees make predictions by dividing the covariate space into discrete subregions, making them naturally suited for modeling piecewise-defined functions. Conversely, deep neural networks, which are smooth functions due to the use of activation functions, tend to interpolate between breakpoints in adjacent $\boldsymbol{x}$ subintervals by "borrowing" information from the different $\boldsymbol{y}$ values in these neighborhoods.

Lastly, for the multimodal tasks b and c, the consequences of the different function choices between DBT and CARD are amplified. While DBT generates samples with clear-cut boxes, CARD struggles to separate these regions in the generated samples. This is particularly evident in Task c: with only one-fifth the training data compared to Task b, DBT still produces samples that align closely with the true data, whereas the samples generated by CARD blend together, failing to separate $\boldsymbol{y}$ in each of the three $\boldsymbol{x}$ subintervals, revealing the challenges CARD faces in scenarios with limited data availability.

5.1.2 OpenML Examples

We now turn our attention to real-world datasets that exhibit discontinuities in the feature space rather than the response variable space. We conduct an ablation study to demonstrate the impact of the three key adjustments made to CARD for the construction of DBT (Section 3.2), namely: switching from one single amortized model to a series of weak learners, tree parameterization, and sequential training.

Initially, we planned to compare the performance of DBT and CARD along with two variants of CARD: one where the amortized neural network is replaced with an amortized GBT, and another in which the amortized neural network is replaced with distinct single tree models at different timesteps, trained independently rather than sequentially. The former model was eliminated from the list due to its consistently poor performance on UCI (Dua and Graff, 2017) benchmark datasets (Appendix A.9). The latter model demonstrated reasonable results and was retained. This model is named CARD-T — refer to Appendix A.5 for details on CARD-T's training and sampling algorithms.

We identified two benchmark datasets characterized by a large number of categorical features from Grinsztajn et al. (2022), available on OpenML (Vanschoren et al., 2013): *Mercedes_Benz_Greener_Manufacturing*, which contains $4,209$ instances with all 359 features being categorical, and *Allstate_Claims_Severity*, which comprises $188,318$ datapoints, with 110 out of 124 features being categorical.

For both datasets, we trained the models on 5 different train-test splits, each with a 90%/10% ratio, created using distinct random seeds. Table 1 presents the results evaluated with RMSE, NLL, and QICE (Section 3.3.1). Our observations indicate that both DBT and CARD-T significantly outperform CARD in terms of distribution matching, as reflected by lower NLL and QICE values, while also providing better mean estimates. Furthermore, DBT surpasses CARD-T in all but one instance, highlighting the effectiveness of the sequential training scheme in enhancing the tree-based model's performance.

5.1.3 SHAP Value Analysis on OpenML Dataset

One major strength of decision tree-based models over neural networks is their interpretability: they provide clear visualizations of decision paths and the influence of each feature on the outcome. By employing distinct models for each diffusion timestep, rather than a single amortized model for all timesteps, we are able to develop a unique method to investigate the impact of each input feature on the prediction at different diffusion timesteps.

Using DBT's trained models $\{f_{\boldsymbol{\theta}_t}\}_{t=1}^{T}$ on the *Mercedes* dataset (Table 1), we generated beeswarm summary plots of SHAP values (Lundberg and Lee, 2017) at six timesteps: $t = 1000, 800, 600, 400, 200, 1$, as shown in Figure 3. In each plot, the features are sorted by their magnitude of impact on model output, measured by the sum of SHAP values over all training samples.

The input to each $f_{\boldsymbol{\theta}_t}$ is the concatenated vector $\big(\hat{\boldsymbol{y}}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\big)$. Since $\boldsymbol{x}$ is 359-dimensional, "Feature 0" represents the noisy sample $\hat{\boldsymbol{y}}_t$, and "Feature 360" is the estimation of $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$, $f_\phi(\boldsymbol{x})$. We observe that "Feature 360" remains the most impactful feature at the four intermediate timesteps ($t = 800, 600, 400, 200$), highlighting the pivotal role of the pre-trained mean estimator $f_\phi(\boldsymbol{x})$ in guiding the sampling process. For the final model during sampling, $f_{\boldsymbol{\theta}_1}$, the most influential feature has changed to "Feature 0": the output is almost solely affected by the sample $\hat{\boldsymbol{y}}_1$, which should be very close to the true $\boldsymbol{y}_0$ if the model is well-trained.

Additionally, we observe changes in the ranking of other important features, indicating that the relative impact of each feature on the model's predictions varies over the course of generation.

We also provide a set of feature importance plots at the same six timesteps in Figure 4, along with an analysis comparing SHAP values and feature importance, in Appendix A.6.

5.1.4 UCI Datasets

We apply the same paradigm as Han et al. (2022) to benchmark DBT on real-world regression datasets. A detailed description of the experimental setup can be found in Appendix A.7. In addition to training DBT on the full dataset, we also evaluate its performance on incomplete data, where 10% of the covariate values

Table 1: OpenML regression tasks.

Dataset	DBT	CARD-T	CARD
	RMSE ↓		
Mercedes	**8.19 ± 1.50**	8.37 ± 1.75	8.80 ± 0.88
Allstate	**0.55 ± 0.00**	0.56 ± 0.00	0.60 ± 0.00
	NLL ↓		
Mercedes	**3.40 ± 0.04**	3.52 ± 0.05	7.85 ± 1.81
Allstate	**0.93 ± 0.00**	1.19 ± 0.00	1.07 ± 0.03
	QICE ↓		
Mercedes	**1.11 ± 0.36**	1.35 ± 0.27	6.36 ± 0.32
Allstate	0.34 ± 0.05	**0.25 ± 0.07**	3.18 ± 0.10

Table 2: UCI regression tasks.

Dataset	PBP	MC Dropout	Deep Ensembles	GCDS	CARD	**DBT**	**DBT** (10% **MCAR**)
	RMSE ↓						
Boston	2.89 ± 0.74	3.06 ± 0.96	3.17 ± 1.05	2.75 ± 0.58	**2.61 ± 0.63**	**2.73 ± 0.62**	3.30 ± 0.89
Concrete	5.55 ± 0.46	5.09 ± 0.60	4.91 ± 0.47	5.39 ± 0.55	**4.77 ± 0.46**	**4.56 ± 0.50**	5.17 ± 0.58
Energy	1.58 ± 0.21	1.70 ± 0.22	2.02 ± 0.32	0.64 ± 0.09	**0.52 ± 0.07**	**0.52 ± 0.07**	0.62 ± 0.14
Kin8nm	9.42 ± 0.29	7.10 ± 0.26	8.65 ± 0.47	8.88 ± 0.42	**6.32 ± 0.18**	**7.04 ± 0.23**	13.60 ± 0.46
Naval	0.41 ± 0.08	0.08 ± 0.03	0.09 ± 0.01	0.14 ± 0.05	**0.02 ± 0.00**	**0.07 ± 0.01**	0.25 ± 0.01
Power	4.10 ± 0.15	4.04 ± 0.14	4.02 ± 0.15	4.11 ± 0.16	**3.93 ± 0.17**	3.95 ± 0.16	**3.72 ± 0.16**
Protein	4.65 ± 0.02	4.16 ± 0.12	4.45 ± 0.02	4.50 ± 0.02	**3.73 ± 0.01**	**3.81 ± 0.04**	4.35 ± 0.04
Wine	0.64 ± 0.04	**0.62 ± 0.04**	0.63 ± 0.04	0.66 ± 0.04	0.63 ± 0.04	**0.61 ± 0.04**	0.65 ± 0.04
Yacht	0.88 ± 0.22	0.84 ± 0.27	1.19 ± 0.49	**0.79 ± 0.26**	**0.65 ± 0.25**	1.08 ± 0.39	1.12 ± 0.34
Year	8.86± NA	**8.77**± NA	8.79± NA	9.20± NA	**8.70**± NA	8.81± NA	9.23± NA
# Top 2	0	2	0	1	**9**	**7**	1
	NLL ↓						
Boston	2.53 ± 0.27	2.46 ± 0.12	2.35 ± 0.16	18.66 ± 8.92	**2.35 ± 0.12**	**2.33 ± 0.12**	3.77 ± 1.34
Concrete	3.19 ± 0.05	3.21 ± 0.18	**2.93 ± 0.12**	13.64 ± 6.88	2.96 ± 0.09	**2.91 ± 0.08**	3.06 ± 0.09
Energy	2.05 ± 0.05	1.50 ± 0.11	1.40 ± 0.27	1.46 ± 0.72	**1.04 ± 0.06**	**0.91 ± 0.21**	3.58 ± 10.69
Kin8nm	−0.83 ± 0.02	**−1.14 ± 0.05**	−1.06 ± 0.02	−0.38 ± 0.36	**−1.32 ± 0.02**	−1.09 ± 0.02	−0.56 ± 0.02
Naval	−3.97 ± 0.10	−4.45 ± 0.38	**−5.94 ± 0.10**	−5.06 ± 0.48	**−7.54 ± 0.05**	−4.31 ± 0.05	−3.84 ± 0.02
Power	2.92 ± 0.02	2.90 ± 0.03	2.89 ± 0.02	2.83 ± 0.06	**2.82 ± 0.02**	2.90 ± 0.02	**2.81 ± 0.02**
Protein	3.05 ± 0.00	2.80 ± 0.08	2.89 ± 0.02	2.81 ± 0.09	**2.49 ± 0.03**	**2.64 ± 0.02**	2.78 ± 0.02
Wine	1.03 ± 0.03	0.93 ± 0.06	0.96 ± 0.06	6.52 ± 21.86	**0.92 ± 0.05**	**0.88 ± 0.04**	13.94 ± 10.06
Yacht	1.58 ± 0.08	1.73 ± 0.22	1.11 ± 0.18	**0.61 ± 0.34**	0.90 ± 0.08	**0.59 ± 0.24**	0.91 ± 0.20
Year	3.69± NA	**3.42**± NA	3.44± NA	3.43± NA	**3.34**± NA	3.44± NA	3.52± NA
# Top 2	0	2	2	1	**8**	**6**	1
	QICE (in %) ↓						
Boston	3.50 ± 0.88	3.82 ± 0.82	**3.37 ± 0.00**	11.73 ± 1.05	**3.45 ± 0.83**	4.19 ± 1.18	9.26 ± 1.36
Concrete	2.52 ± 0.60	4.17 ± 1.06	2.68 ± 0.64	10.49 ± 1.01	**2.30 ± 0.66**	**2.52 ± 0.51**	4.21 ± 0.92
Energy	6.54 ± 0.90	5.22 ± 1.02	**3.62 ± 0.58**	7.41 ± 2.19	4.91 ± 0.94	**3.78 ± 0.91**	5.48 ± 1.15
Kin8nm	1.31 ± 0.25	1.50 ± 0.32	**1.17 ± 0.22**	7.73 ± 0.80	**0.92 ± 0.25**	1.31 ± 0.29	1.30 ± 0.27
Naval	4.06 ± 1.25	12.50 ± 1.95	6.64 ± 0.60	5.76 ± 2.25	**0.80 ± 0.21**	11.16 ± 1.66	**1.78 ± 0.26**
Power	**0.82 ± 0.19**	1.32 ± 0.37	1.09 ± 0.26	1.77 ± 0.33	0.92 ± 0.21	1.24 ± 0.20	**0.91 ± 0.23**
Protein	1.69 ± 0.09	2.82 ± 0.41	2.17 ± 0.16	2.33 ± 0.18	**0.71 ± 0.11**	0.95 ± 0.10	**0.73 ± 0.18**
Wine	**2.22 ± 0.64**	2.79 ± 0.56	**2.37 ± 0.63**	3.13 ± 0.79	3.39 ± 0.69	8.18 ± 1.17	13.91 ± 0.58
Yacht	6.93 ± 1.74	10.33 ± 1.34	7.22 ± 1.41	**5.01 ± 1.02**	8.03 ± 1.17	**5.96 ± 1.51**	6.26 ± 1.53
Year	2.96± NA	2.43± NA	2.56± NA	1.61± NA	**0.53**± NA	1.07± NA	**0.72**± NA
# Top 2	2	0	**4**	1	**6**	3	4

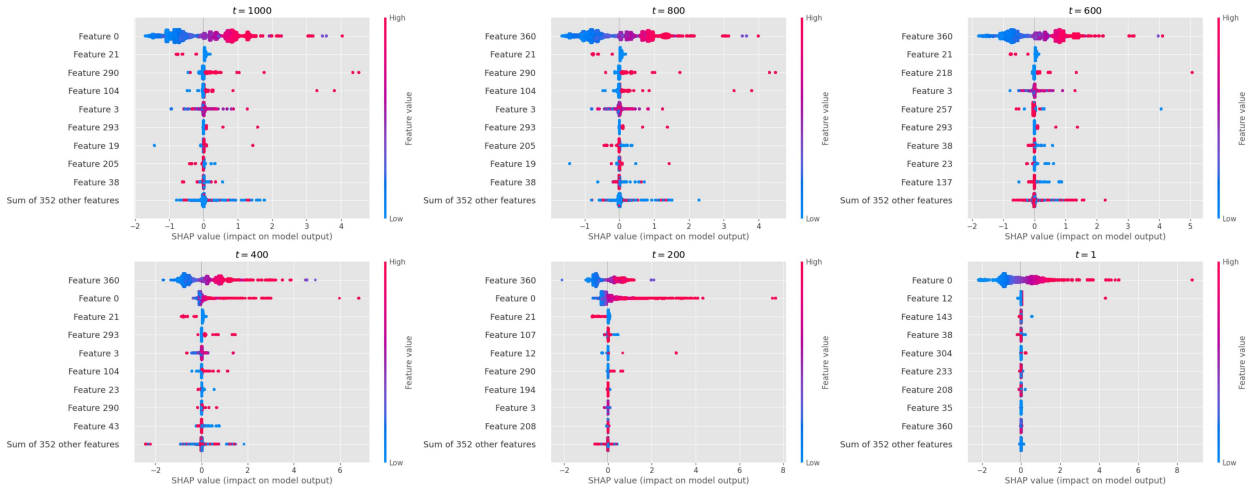

Figure 3: Beeswarm summary plots of SHAP values at six diffusion timesteps.

are randomly removed (Missing Completely at Random; MCAR). The evaluation metrics are reported in Table 2, along with the number of times each model achieves a Top-2 ranking based on each metric.

Our observations show that DBT trained on the full dataset achieves performance on par with CARD, while still outperforming other baseline methods. This demonstrates the effectiveness of our proposed method in modeling conditional distributions in real-world settings. It is important to note that all our experiments are based on inherently heterogeneous tabular data, characterized by variability in feature types, as well as in the number of features and samples. Therefore, introducing a new method for modeling such data should aim to provide a competitive alternative among state-of-the-art approaches, offering practitioners an additional tool for tackling new datasets. More discussion on this point can be found in Appendix A.8.

A distinct advantage of DBT over CARD becomes apparent when dealing with data containing missing values. DBT handles missing data without the need for imputation and demonstrates robust performance: it rarely records the worst metric when compared to other baseline models trained on complete data, occasionally achieves state-of-the-art results in terms of RMSE and NLL, and performs well in terms of QICE. This robustness makes DBT particularly useful in real-world applications where missing data is prevalent, such as healthcare, finance, and survey analysis, providing a reliable and efficient solution for handling incomplete tabular datasets.

5.2 Classification

For classification, we contextualize the diffusion boosting framework through a practical business application: credit card fraud detection.

5.2.1 The Story of OTA Fraud Detection

As generative AI has been advancing at a staggering speed in the past few years in terms of the fidelity of content generation, its negative social impact has also become increasingly concerning (Kenthapadi et al., 2023; Grace et al., 2024). Against this backdrop, this section endeavors to pivot the discourse towards the beneficial potential of generative AI: we introduce a fraud detection paradigm that leverages the stochasticity of generative models as its foundational mechanism. Our aim is to illuminate one positive application of generative AI, demonstrating its potential to contribute significantly to societal well-being.

We introduce the use case of fraud detection as a business component within an online travel agency (OTA), where transactions, including payments and bookings, are conducted digitally. This environment places OTAs at risk of credit card fraud, since it is hard to verify that the credit card user is indeed its rightful owner. Traditionally, companies have relied on rule-based models and human agents to identify suspicious transactions. However, the vast daily volume of transactions that came with complicated fraud patterns, coupled with the significant costs associated with employing a large team of agents, renders this approach impractical for scrutinizing every transaction.

In response to these challenges, OTA companies have gradually integrated machine learning techniques into their fraud detection ecosystem in recent years. These methods involve deploying classifiers that assess each transaction in real-time, and pass the dubious ones to human agents for further review. This hybrid approach significantly alleviates the burden on human agents by automating the initial screening process.

This operational model exemplifies a strategy known as *learning to defer (L2D)* (Madras et al., 2018; Narasimhan et al., 2022; Verma and Nalisnick, 2022), where AI systems recognize *when* to rely on human expertise for decision-making, thus providing a more efficient workflow while ensuring more reliable outcomes. This idea is pivotal in the contexts where AI's *decision confidence* is crucial.

5.2.2 Binary Classification on Real-World Tabular Data

We demonstrate how DBT conducts binary classification on tabular data using the evaluation framework described in Section 3.3.2. We train the DBT model on a credit card default dataset (Yeh and hui Lien, 2009), which is another benchmark dataset from Grinsztajn et al. (2022). This dataset contains 21 covariates with both numerical and categorical features. The model is trained on $11,944$ instances and evaluated on the remaining $1,328$ cases.

A pre-trained neural network classifier $f_\phi(\boldsymbol{x})$ predicts the test set with an accuracy of 57.68%. For evaluation, we generate only 10 samples for each test instance. Firstly, we make predictions using the majority-voted label, achieving an improved accuracy of 69.58%. We then compute the PIW for all test instances and summarize the results in Table 3. We observe that for the group of test instances predicted as Class 1, higher accuracy is accompanied by a narrower mean PIW. Furthermore, within each predicted class, instances with correct predictions have a narrower mean PIW compared to those with incorrect predictions. Additionally, we group the test instances by increasing PIW values and compute the accuracy within each bin, as shown in Table 4[1]. We observe that as the mean PIW increases from Bin 1 to Bin 4, the accuracy consistently decreases. The results from these two tables suggest that less variation in generated samples is associated with better performance in classification.

We now conduct the t-test on each test instance at two significance levels, 0.05 and 0.005. For each significance level, we observe in Table 5 that the accuracy for test instances with rejected t-tests is considerably higher than for those where the t-tests fail to reject. This observation holds at each predicted class level, as shown in Table 6.

By comparing the accuracy and t-test reject rates between the two predicted classes, we further conclude that the more accurate class exhibits a higher rate of t-test null hypotheses being rejected. This validates the t-test as an effective method for measuring model confidence.

This evaluation design aligns seamlessly with the requirements of a learning-to-defer method: we can interpret cases where the t-tests fail to reject as uncertain predictions made by the DBT model, which can then be

[1]The bins are sorted in ascending order of PIW. There are only four distinct PIW values, since a tree model has a limited number of possible outputs (*i.e.*, the number of leaves). Note that Bin 2 has a smaller PIW than Bin 3, although this is not reflected due to rounding to two decimal places.

Table 3: PIW for both majority-vote predicted class labels.

predicted class	accuracy	mean PIW		
		overall (count)	correct pred. (count)	incorrect pred. (count)
0	66.14%	110.03 (762)	108.10 (504)	113.79 (258)
1	74.20%	86.50 (566)	79.77 (420)	105.86 (146)

Table 4: Accuracy across different PIW bins.

Bin	1	2	3	4
mean PIW	0.00	94.44	94.44	121.86
accuracy (count)	87.36% (182)	82.31% (130)	70.00% (120)	64.06% (896)

Table 5: Accuracy by t-test outcomes.

t-test outcome	accuracy (count)	
	$\alpha = 0.05$	$\alpha = 0.005$
reject	77.79% (707)	81.02% (432)
fail to reject	60.23% (621)	64.06% (896)

Table 6: Accuracy by predicted class labels.

predicted class	accuracy	$\alpha = 0.05$			$\alpha = 0.005$		
		t-test reject rate	accuracy		t-test reject rate	accuracy	
			reject (count)	fail to reject (count)		reject (count)	fail to reject (count)
0	66.14%	43.96%	71.94% (335)	61.59% (427)	21.92%	73.05% (167)	64.20% (595)
1	74.20%	65.72%	83.06% (372)	57.22% (194)	46.82%	86.04% (265)	63.79% (301)

deferred to human agents for further evaluation. Assuming human agents can achieve the same level of accuracy as the cases with rejected t-tests, we can improve the overall accuracy from 69.58% to 76.68% with $\alpha = 0.05$, and to 78.59% with $\alpha = 0.005$.

Furthermore, by comparing the results between the two significance levels in both Tables 5 and 6, we observe the role of the significance level as a measure of decision conservativeness: a lower significance level implies a more conservative decision strategy, resulting in fewer instances where the t-tests are rejected, *i.e.*, fewer predictions are made with confidence.

6 Conclusion

We propose the **Diffusion Boosting** paradigm as a new supervised learning algorithm, combining the merits of both *Classification and Regression Diffusion Models (CARD)* and *Gradient Boosting*. We implement **Diffusion Boosted Trees (DBT)**, which parameterizes the diffusion model by a single tree at each timestep. We demonstrate through experiments the advantages of DBT over CARD, and present a case study of fraud detection for DBT to perform classification on tabular data with the ability of learning to defer.

Acknowledgments

The authors acknowledge the Texas Advanced Computing Center (TACC) for providing HPC and storage resources that have contributed to the research results reported within this paper. The authors would also like to thank Ruijiang Gao and Huangjie Zheng for their discussions during the course of this project.

References

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-image diffusion models with an ensemble of expert denoisers. *ArXiv*, abs/2211.01324, 2022.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 29th Conference on Neural Information Processing Systems*, 2015.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational Inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 2015.

Shahin Boluki, Randy Ardywibowo, Siamak Zamani Dadaneh, Mingyuan Zhou, and Xiaoning Qian. Learnable Bernoulli dropout for Bayesian deep learning. In *Proceedings of the 23th International Conference on Artificial Intelligence and Statistics*, volume 108, pages 3905–3916. PMLR, 2020.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 99:1–21, 2022.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

A. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes rendus de l'Académie des Sciences*, 25:536–538, 1847.

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, pages 785–794, 2016.

Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. BART: Bayesian Additive Regression Trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, 2021.

Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Tony Duan, Anand Avati, Daisy Yi Ding, Khanh K. Thai, Sanjay Basu, Andrew Y. Ng, and Alejandro Schuler. NGBoost: Natural gradient boosting for probabilistic prediction. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020.

Xinjie Fan, Yizhe Zhang, Zhendong Wang, and Mingyuan Zhou. Adaptive correlated monte carlo for contextual categorical sequence generation. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Xinjie Fan, Shujian Zhang, Korawat Tanwisuth, Xiaoning Qian, and Mingyuan Zhou. Contextual dropout: An efficient sample-dependent dropout module. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*. PMLR, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Proceedings of the 27th Conference on Neural Information Processing Systems*, 2014.

Katja Grace, Harlan Stewart, Julia Fabienne Sandkühler, Stephen Thomas, Ben Weinstein-Raun, and Jan Brauner. Thousands of AI authors on the future of AI. *ArXiv*, abs/2401.02843, 2024.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? In *Proceedings of the 36th Conference on Neural Information Processing Systems*, 2022.

Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. CARD: Classification and Regression Diffusion Models. In *Proceedings of the 36th Conference on Neural Information Processing Systems*, 2022.

Jingyu He, Saar Yalov, and P. Richard Hahn. XBART: Accelerated Bayesian Additive Regression Trees. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, volume 89, pages 1130–1138. PMLR, 2019.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 2015.

Jennifer L. Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and Epistemic Uncertainty in Machine Learning: An introduction to concepts and methods. *Machine Learning*, 110:457–506, 2021.

Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, 2024.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proceedings of the 36th Conference on Neural Information Processing Systems*, 2022.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Machine Learning*, 2017.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

Krishnaram Kenthapadi, Himabindu Lakkaraju, and Nazneen Rajani. Generative AI meets responsible AI: Practical challenges and opportunities. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

Durk P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Proceedings of the 29th Conference on Neural Information Processing Systems*, 2015.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

Shiao Liu, Xingyu Zhou, Yuling Jiao, and Jian Huang. Wasserstein generative learning of conditional distribution. *arXiv*, abs/2112.10039, 2021.

Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

David Madras, Toniann Pitassi, and Richard S. Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, 2018.

Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko. Uncertainty in gradient boosting via ensembles. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Harikrishna Narasimhan, Wittawat Jitkrittum, Aditya K. Menon, Ankit Rawat, and Sanjiv Kumar. Post-hoc estimators for learning to defer to an expert. In *Proceedings of the 36th Conference on Neural Information Processing Systems*, 2022.

Mang Ning, Mingxiao Li, Jianlin Su, A. A. Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. *arXiv*, abs/2308.15321, 2023a.

Mang Ning, E. Sangineto, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Input perturbation reduces exposure bias in diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023b.

Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

Jorge Nocedal and Stephen J. Wright. Line search methods. In *Numerical Optimization*, chapter 3. Springer, 1999.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, 2018.

Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. Are neural rankers still outperformed by gradient boosted decision trees? In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Florian Schmidt. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, page 157–167, 2019.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Rodney Sparapani, Charles Spanbauer, and Robert McCulloch. Nonparametric machine learning and efficient computation with Bayesian Additive Regression Trees: The BART R package. *Journal of Statistical Software*, 97(1):1–66, 2021.

Jennifer E. Starling, Jared S. Murray, Carlos M. Carvalho, Radek K. Bukowski, and James G. Scott. BART with Targeted Smoothing: An analysis of patient-specific stillbirth risk. *The Annals of Applied Statistics*, 14(1):28–50, 2020.

Aleksei Ustimenko and Liudmila Prokhorenkova. SGLB: Stochastic gradient langevin boosting. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. URL `http://doi.acm.org/10.1145/2641190.2641198`.

Rajeev Verma and Eric T. Nalisnick. Calibrated learning to defer with one-vs-all classifiers. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23 (7):1661–1674, 2011.

Zhendong Wang and Mingyuan Zhou. Thompson sampling via local uncertainty. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020.

Jeremy Watt, Reza Borhani, and Aggelos Konstantinos Katsaggelos. Universal approximators. In *Machine Learning Refined: Foundations, Algorithms, and Applications*, chapter 11.2. Cambridge University Press, 2020.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

Yongxin Yang, Irene Garcia Morillo, and Timothy M. Hospedales. Deep neural decision trees. In *2018 ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*, 2018.

I-Cheng Yeh and Che hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2, Part 1):2473–2480, 2009.

Mingzhang Yin and Mingyuan Zhou. Semi-Implicit Variational Inference. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

Longlin Yu, Tianyu Xie, Yu Zhu, Tong Yang, Xiangyu Zhang, and Cheng Zhang. Hierarchical semi-implicit variational inference with application to diffusion model acceleration. In *Proceedings of the 37th Conference on Neural Information Processing Systems*, 2023.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity Distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Xingyu Zhou, Yuling Jiao, Jin Liu, and Jian Huang. A deep generative approach to conditional sampling. *Journal of the American Statistical Association*, 118(543):1837–1848, 2023.

A Appendix

A.1 Background: An In-Depth Version

In this section, we provide a more comprehensive version of Section 2, with a focus on establishing the objective functions of both gradient boosting and CARD.

A.1.1 Supervised Learning

We aim to tackle the problem of *supervised learning*: given a set of covariates $\boldsymbol{x} = \{x_1, \ldots, x_p\}$, and a response variable $\boldsymbol{y}$ — a numerical variable for regression, or a categorical one for classification — we seek to learn a mapping that takes the covariates as inputs and predicts the response variable as its output, with the hope that it can generalize to new and unseen data after observing some training data.

The mapping usually takes the form of a mathematical function, thus the supervised learning problem becomes a *function estimation* problem: the goal is to obtain an approximation $F^*(\boldsymbol{x})$ of the function $F(\boldsymbol{x})$ that maps $\boldsymbol{x}$ to $\boldsymbol{y}$, which minimizes the expectation of a loss function $L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)$ over the joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ (Friedman, 2001):

$$F^* = \arg\min_F \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\big[L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)\big]. \tag{21}$$

When imposing a parametric form $\boldsymbol{\theta}$ upon the function F as a common practice, the function is now read as $F(\boldsymbol{x}; \boldsymbol{\theta})$, and the function estimation problem becomes a *parameter optimization* problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\big[L\big(\boldsymbol{y}, F(\boldsymbol{x}; \boldsymbol{\theta})\big)\big], \tag{22}$$

thus $F^*(\boldsymbol{x}) = F(\boldsymbol{x}; \boldsymbol{\theta}^*)$.

Numerical optimization methods need to be applied to solve Eq. (22) for most $F(\boldsymbol{x}; \boldsymbol{\theta})$ and L (Friedman, 2001). The standard procedure for many of these methods is as follows: first, they determine the direction to improve the objective L, then compute the step size via line search (Nocedal and Wright, 1999) for the parameter $\boldsymbol{\theta}$ to move along this direction. Gradient descent (Cauchy, 1847) is one of the most well-known methods in the machine learning community for finding the descent direction, which computes *the negative gradient* as the steepest descent direction for an objective function differentiable in the neighborhood of the point of interest.

A.1.2 Gradient Descent

Denote the objective function in Eq. (22) as

$$\Phi(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\big[L\big(\boldsymbol{y}, F(\boldsymbol{x}; \boldsymbol{\theta})\big)\big],$$

the gradient descent update at any intermediate step m is

$$\boldsymbol{\theta}_m = \boldsymbol{\theta}_{m-1} + \rho_m \cdot \big(- \nabla_{\boldsymbol{\theta}_{m-1}} \Phi(\boldsymbol{\theta}_{m-1})\big), \tag{23}$$

where the step size

$$\rho_m = \arg\min_{\rho} \Phi\Big(\boldsymbol{\theta}_{m-1} + \rho \cdot \big(- \nabla_{\boldsymbol{\theta}_{m-1}} \Phi(\boldsymbol{\theta}_{m-1})\big)\Big). \tag{24}$$

Therefore, with M total update steps and $\boldsymbol{\theta}_0$ as the initialization, we have the optimized parameter via gradient descent as

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 + \sum_{m=1}^{M} \rho_m \cdot \big(- \nabla_{\boldsymbol{\theta}_{m-1}} \Phi(\boldsymbol{\theta}_{m-1})\big). \tag{25}$$

A.1.3 Gradient Boosting

While gradient descent can be described as a numerical optimization method *in the parameter space*, gradient boosting (Friedman, 2001) is essentially gradient descent *in the function space*: by considering $F(\boldsymbol{x})$ evaluated at each $\boldsymbol{x}$ to be a parameter, Friedman (2001) establishes the objective function at the joint distribution level as

$$\Phi(F) = \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\big[L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)\big] = \mathbb{E}_{p(\boldsymbol{x})}\Big[\mathbb{E}_{p(\boldsymbol{y}\,|\,\boldsymbol{x})}\big[L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)\big]\Big], \tag{26}$$

and equivalently, the objective function at the instance level:

$$\Phi\big(F(\boldsymbol{x})\big) = \mathbb{E}_{p(\boldsymbol{y}\,|\,\boldsymbol{x})}\big[L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)\big], \tag{27}$$

whose gradient can be computed as

$$\nabla_{F(\boldsymbol{x})}\Phi\big(F(\boldsymbol{x})\big) = \frac{\partial\Phi\big(F(\boldsymbol{x})\big)}{\partial F(\boldsymbol{x})} = \mathbb{E}_{p(\boldsymbol{y}\,|\,\boldsymbol{x})}\left[\frac{\partial L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)}{\partial F(\boldsymbol{x})}\right], \tag{28}$$

where the second equation results from assuming sufficient regularity to interchange differentiation and integration.

Following the gradient-based numerical optimization paradigm as in Eq. (25), we obtain the optimal solution in the function space:

$$F^*(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \sum_{m=1}^{M} \rho_m \cdot \big(-g_m(\boldsymbol{x})\big), \tag{29}$$

where $f_0(\boldsymbol{x})$ is the initial guess, and $g_m(\boldsymbol{x}) = \nabla_{F_{m-1}(\boldsymbol{x})}\Phi\big(F_{m-1}(\boldsymbol{x})\big)$ is the gradient at optimization step m.

Given a finite set of samples $\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_1^N$ from $p(\boldsymbol{x}, \boldsymbol{y})$, we have the data-based analogue of $g_m(\boldsymbol{x})$ defined only at these training instances:

$$g_m(\boldsymbol{x}_i) = \frac{\partial L\big(\boldsymbol{y}_i, \hat{F}_{m-1}(\boldsymbol{x}_i)\big)}{\partial \hat{F}_{m-1}(\boldsymbol{x}_i)}. \tag{30}$$

Since the goal of supervised learning is to generalize the predictive function to unseen data, Friedman (2001) proposes to use a parameterized class of functions $h(\boldsymbol{x}; \boldsymbol{\alpha})$ to learn the negative gradient term at every gradient descent step. Specifically, $h(\boldsymbol{x}; \boldsymbol{\alpha})$ is trained with the squared-error loss at optimization step m to produce $\{h(\boldsymbol{x}_i; \boldsymbol{\alpha}_m)\}_1^N$ most parallel to $\{-g_m(\boldsymbol{x}_i)\}_1^N$, and the solution $h(\boldsymbol{x}; \boldsymbol{\alpha}_m)$ can be applied to approximate $-g_m(\boldsymbol{x})$ for any $\boldsymbol{x}$, whose parameter

$$\boldsymbol{\alpha}_m = \arg\min_{\boldsymbol{\xi},\omega} \sum_{i=1}^{N} \big(-g_m(\boldsymbol{x}_i) - \omega \cdot h(\boldsymbol{x}_i; \boldsymbol{\xi})\big)^2, \tag{31}$$

while the multiplier ρ_m is optimized via line search,

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L\big(\boldsymbol{y}_i, \hat{F}_{m-1}(\boldsymbol{x}_i) + \rho \cdot h(\boldsymbol{x}_i; \boldsymbol{\alpha}_m)\big). \tag{32}$$

Therefore, with finite data, the gradient descent update in the function space at step m is

$$\hat{F}_m(\boldsymbol{x}) = \hat{F}_{m-1}(\boldsymbol{x}) + \rho_m \cdot h(\boldsymbol{x}; \boldsymbol{\alpha}_m), \tag{33}$$

and the prediction of $\boldsymbol{y}$ given any $\boldsymbol{x}$ can be obtained through

$$\hat{\boldsymbol{y}} = \hat{F}^*(\boldsymbol{x}) = \hat{F}_0(\boldsymbol{x}) + \sum_{m=1}^{M} \rho_m \cdot h(\boldsymbol{x}; \boldsymbol{\alpha}_m). \tag{34}$$

The function $h(\boldsymbol{x}; \boldsymbol{\alpha})$ is termed a *weak learner* or *base learner*, and is often parameterized by a simple Classification And Regression Tree (CART) (Breiman et al., 1984). Eq. (34) has the form of an ensemble of

weak learners, trained sequentially and combined via weighted sum[2].

Among all choices of the loss function $L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)$, the squared-error loss is of particular interest:

$$L\big(\boldsymbol{y}, F(\boldsymbol{x})\big) = \tfrac{1}{2}\big(\boldsymbol{y} - F(\boldsymbol{x})\big)^2. \tag{35}$$

In this case, the negative gradient is

$$-\frac{\partial L}{\partial F(\boldsymbol{x})} = \boldsymbol{y} - F(\boldsymbol{x}), \tag{36}$$

which is the residual[3]. As a result, each weak learner aims to predict the residual term at its corresponding optimization step. It is tempting to draw parallels between this residual predicting behavior by gradient boosting and the residual learning paradigm by ResNet (He et al., 2015) at face value, thus we intend to point out their difference here: the former is due to the particular choice of the squared-error loss function, Eq. (35), while the latter results from its computational excellencies in dealing with the vanishing/exploding gradient problem in very deep neural networks.

The squared-error loss is the default choice of loss function for regression tasks by popular gradient boosting libraries like XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018). It is worth mentioning that the optimal solution for minimizing the expected square-error loss is the conditional mean, $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$.

A.1.4 Classification and Regression Diffusion Models (CARD)

With the same goal as gradient boosting of taking on supervised learning problems, CARD (Han et al., 2022) approaches them from a very different angle: by adopting a generative modeling framework, a CARD model directly outputs the samples from the conditional distribution $p(\boldsymbol{y} \,|\, \boldsymbol{x})$, instead of some summary statistics such as the conditional mean $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$. A unique advantage of this class of models is that it is free of any assumptions on the parametric form of $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ — *e.g.*, the additive-noise assumption with a particular form of its noise distribution (a zero-mean Gaussian distribution for regression, or a standard Gumbel distribution for classification), which has been prevalently applied by the existing methods. A finer level of granularity in the outputs of CARD (*i.e.*, directly generating samples instead of predicting a summary statistic) helps to paint a more complete picture of $p(\boldsymbol{y} \,|\, \boldsymbol{x})$: with enough samples, the model can capture the variability and modality of the conditional distribution, besides accurately recovering $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$. The advantage of generative modeling becomes more evident when $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ is multimodal, or has heteroscedasticity, as shown by the toy examples in Han et al. (2022). Meanwhile, CARD consistently performs better in terms of the conventional metrics like RMSE and NLL on real-world datasets than other uncertainty-aware methods that are explicitly optimized for these metrics as their objectives.

The parameterization of CARD follows the Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020) framework, which is a generative model that aims to learn a function that maps a sample from a simple known distribution (often called the *noise distribution*) to a sample from the target distribution. However, instead of directly generating the sample with only one function evaluation like other classes of generative models — including GANs (Goodfellow et al., 2014) and VAEs (Kingma and Welling, 2014) — the function produces a *less noisy version* of its input after each evaluation, which is then fed into the *same* function to produce the next one. For CARD, the final output can be viewed as a noiseless sample of $\boldsymbol{y}$ from $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ after enough steps. This autoregressive fashion of computing can be described as *iterative refinement* or *progressive denoising*.

CARD adopts the DDPM framework by treating the noisy samples from the intermediate steps as latent

[2]Each weight ρ_m is conceptually the step size in numerical optimization. In practice, we often find it to be a constant that is preset or scheduled, instead of learned through line search: *e.g.*, in Ke et al. (2017), the weight of each weak learner is set to 1.

[3]We attach the algorithm of gradient boosting with the squared-error loss in Appendix A.3 for reference.

variables, and construct a Markov chain to link them together, so that the progressive data *generation* process can be modeled analytically, in the sense that an explicit distributional form (*i.e.*, Gaussian) can be imposed upon adjacent latent variables.

This Markov chain is formed in the direction *opposite* to the data *generation* process described above, with each variable subscripted by its chronological order: *e.g.*, the target response variable $\boldsymbol{y}$ is re-denoted as $\boldsymbol{y}_0$, and the noise variable is $\boldsymbol{y}_T$, where T is the total number of steps, or timesteps, for this Markov process. As the data generation process that goes from $\boldsymbol{y}_T$ to $\boldsymbol{y}_0$ is described as a denoising procedure above, this Markov chain that goes from $\boldsymbol{y}_0$ to $\boldsymbol{y}_T$ defines a noise-adding mechanism, where the stepwise transition $q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_{t-1}, \boldsymbol{x})$ is defined through a Gaussian distribution. The conditional distribution of all latent variables given the target variable (and the covariates) in the noise-adding direction,

$$q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x}) = \prod_{t=1}^{T} q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_{t-1}, \boldsymbol{x}), \tag{37}$$

is called the *forward diffusion process.*

Meanwhile, denoting the learnable parameter in the generative model as $\boldsymbol{\theta}$, the joint distribution (conditioning on the covariates) in the data generation direction is

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x}) = p(\boldsymbol{y}_T \,|\, \boldsymbol{x}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{x}), \tag{38}$$

in which $p(\boldsymbol{y}_T \,|\, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{I})$ is *the* noise distribution — a Gaussian distribution with a mean of $\boldsymbol{\mu}_T$ — and is also referred to as the *prior distribution.* This joint distribution is called the *reverse diffusion process.*

As a generative model, CARD is trained via an objective rooted in distribution matching: re-denoting the ground truth conditional distribution $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ as $q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$, we wish to learn $\boldsymbol{\theta}$ so that $p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$ approximates $q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$ well, *i.e.*,

$$D_{\mathrm{KL}}\big(q(\boldsymbol{y}_0 \,|\, \boldsymbol{x}) \,\big\|\, p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})\big) \approx 0, \tag{39}$$

where

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x}) d\boldsymbol{y}_{1:T}. \tag{40}$$

We have the following relationship:

$$H(q, p_{\boldsymbol{\theta}}) = H(q) + D_{\mathrm{KL}}(q \parallel p_{\boldsymbol{\theta}}), \tag{41}$$

in which $H(q, p_{\boldsymbol{\theta}}) = -\mathbb{E}_q[\log p_{\boldsymbol{\theta}}]$ is the cross entropy of $p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$, $H(q) = \mathbb{E}_q[-\log q]$ is the entropy of $q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$, and $D_{\mathrm{KL}}(q \parallel p_{\boldsymbol{\theta}}) = \mathbb{E}_q[\log q/p_{\boldsymbol{\theta}}]$ is the KL divergence of $q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$ from $p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$. Since $q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$ does not contain $\boldsymbol{\theta}$, $\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}\big(q(\boldsymbol{y}_0 \,|\, \boldsymbol{x}) \,\big\|\, p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})\big)$ is equivalent to $\min_{\boldsymbol{\theta}} H\big(q(\boldsymbol{y}_0 \,|\, \boldsymbol{x}), p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})\big)$. The variational bound (*i.e.*, the negative ELBO) can be derived from this cross entropy term (see Appendix A.4) as a standard objective function to be minimized for training CARD:

$$\begin{aligned} &\mathbb{E}_{q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})}\big[-\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})\big] \\ \leq &\mathbb{E}_{q(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x})}\left[\log \frac{q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x})}\right] =: L. \end{aligned} \tag{42}$$

By following the same procedure as Appendix A in Ho et al. (2020), the objective L in (42) can be rewritten as

$$L = \mathbb{E}_{q(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x})}\left[L_T + \sum_{t=2}^{T} L_{t-1} + L_0\right], \tag{43}$$

in which

$$
\begin{aligned}
L_T &\coloneqq D_{\mathrm{KL}}\big(q(\boldsymbol{y}_T \,|\, \boldsymbol{y}_0, \boldsymbol{x}) \,\big\|\, p(\boldsymbol{y}_T \,|\, \boldsymbol{x})\big), \\
L_{t-1} &\coloneqq D_{\mathrm{KL}}\big(q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) \,\big\|\, p_{\boldsymbol{\theta}}(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{x})\big), \\
L_0 &\coloneqq -\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{y}_1, \boldsymbol{x}).
\end{aligned}
$$

As what will be shown later, the forward process does not contain any learnable parameters, thus L_T is a constant with respect to $\boldsymbol{\theta}$. Meanwhile, the form of $p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{y}_1, \boldsymbol{x})$ is more of an application-dependent design choice. Therefore, the main focus for optimizing $\boldsymbol{\theta}$ is on the remaining L_{t-1} terms, for $t = 2, \ldots, T$.

The distribution $q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x})$ in L_{t-1} is called the *forward process posterior distribution*, which is tractable and can be derived by applying Bayes' rule:

$$
q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) \propto q\big(\boldsymbol{y}_t \,|\, \boldsymbol{y}_{t-1}, \boldsymbol{x}\big) \cdot q\big(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_0, \boldsymbol{x}\big), \tag{44}
$$

in which both $q\big(\boldsymbol{y}_t \,|\, \boldsymbol{y}_{t-1}, \boldsymbol{x}\big)$ and $q\big(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_0, \boldsymbol{x}\big)$ are Gaussian: as mentioned before, the former is the stepwise transition distribution in the forward process, defined as

$$
q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_{t-1}, \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{y}_t; \sqrt{\alpha_t}\boldsymbol{y}_{t-1} + (1 - \sqrt{\alpha_t})\boldsymbol{\mu}_T, \beta_t \boldsymbol{I}\big), \tag{45}
$$

in which β_t is the t-th term of a predefined noise schedule $\beta_1, \ldots, \beta_T$, and $\alpha_t \coloneqq 1 - \beta_t$. This design gives rise to a closed-form distribution to sample $\boldsymbol{y}_t$ at any arbitrary timestep t:

$$
q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{y}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{y}_0 + (1 - \sqrt{\bar{\alpha}_t})\boldsymbol{\mu}_T, (1 - \bar{\alpha}_t)\boldsymbol{I}\big), \tag{46}
$$

in which $\bar{\alpha}_t \coloneqq \prod_{j=1}^{t} \alpha_j$. Each of the forward process posteriors thus has the form of

$$
q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) = \mathcal{N}\Big(\boldsymbol{y}_{t-1}; \tilde{\boldsymbol{\mu}}(\boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{\mu}_T), \tilde{\beta}_t \boldsymbol{I}\Big), \tag{47}
$$

where the variance

$$
\tilde{\beta}_t \coloneqq \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t, \tag{48}
$$

and the mean

$$
\tilde{\boldsymbol{\mu}}(\boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{\mu}_T) \coloneqq \gamma_0 \cdot \boldsymbol{y}_0 + \gamma_1 \cdot \boldsymbol{y}_t + \gamma_2 \cdot \boldsymbol{\mu}_T, \tag{49}
$$

in which the coefficients are:

$$
\begin{aligned}
\gamma_0 &= \frac{\beta_t \sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t}, \\
\gamma_1 &= \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t}, \\
\gamma_2 &= \left(1 + \frac{(\sqrt{\bar{\alpha}_t} - 1)(\sqrt{\alpha_t} + \sqrt{\bar{\alpha}_{t-1}})}{1 - \bar{\alpha}_t}\right),
\end{aligned}
$$

as derived in Appendix A.1 of Han et al. (2022).

Now to minimize each L_{t-1}, $p_{\boldsymbol{\theta}}(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{x})$ needs to approximate the Gaussian distribution $q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x})$, whose variance (48) is already known. Therefore, the learning task is reduced to optimizing $\boldsymbol{\theta}$ for the estimation of the forward process posterior mean $\tilde{\boldsymbol{\mu}}(\boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{\mu}_T)$. CARD adopts the noise-prediction loss introduced in Ho et al. (2020), a simplification of L_{t-1}:

$$
\mathcal{L}_{\mathrm{CARD}} = \mathbb{E}_{p(t, \boldsymbol{y}_0 \,|\, \boldsymbol{x}, \boldsymbol{\epsilon})}\left[\big|\big|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\big(\boldsymbol{x}, \boldsymbol{y}_t, f_\phi(\boldsymbol{x}), t\big)\big|\big|^2\right], \tag{50}
$$

in which $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is sampled as the *forward* process noise term, $\boldsymbol{y}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{y}_0 + (1 - \sqrt{\bar{\alpha}_t})\boldsymbol{\mu}_T + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$ is the sample from the *forward* process distribution (46), and $f_\phi(\boldsymbol{x})$ is the point estimate of $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$ (usually parameterized by a pre-trained neural network, and to be used as an additional input to the diffusion model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$).

Note that the vanilla CARD framework directly sets the prior mean as $\boldsymbol{\mu}_T = f_\phi(\boldsymbol{x})$. Here we write the

generic form $\boldsymbol{\mu}_T$ instead, not only for better clarity in methodology demonstration, but also as a slight design change in the diffusion boosting framework: we introduce an extra degree of freedom into the vanilla CARD framework by allowing the choice of the prior mean to be different than the conditional mean estimation $f_\phi(\boldsymbol{x})$, while still using $f_\phi(\boldsymbol{x})$ as one input to the diffusion model at each timestep since it possesses the information of $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}]$.

A.2 In-Depth Analysis of Related Studies

In this section, we contextualize our work by exploring its relationships with several related studies referenced in Section 4.

A.2.1 Distinctions Between Diffusion Boosting and SGLB

Ustimenko and Prokhorenkova (2021) introduces SGLB, a gradient boosting algorithm based on the Langevin diffusion equation. Despite the similar terminology used for their names, SGLB and Diffusion Boosting fundamentally differ in their methodologies. To clarify these differences, we have summarized the key distinctions between SGLB and Diffusion Boosting in Table 7.

Table 7: Differences between SGLB and Diffusion Boosting.

	SGLB	Diffusion Boosting
Is the method a variant of gradient boosting?	yes	no
target of the weak learner h_t for different t's	different	same
input of the weak learner h_t for different t's	same	different
objective function for regression and for classification	different	same
presence of stochasticity	only during training	during both training and inference
Is the output of the trained model deterministic?	yes	no
output of the model	a point estimate of $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}]$ or $p(\boldsymbol{y} = 1 \mid \boldsymbol{x})$	a sample from $p(\boldsymbol{y} \mid \boldsymbol{x})$
context of the term "diffusion"	a special form of the Langevin diffusion equation	diffusion models as a class of generative models

We elaborate the differences listed in Table 7 as follows:

- SGLB is a variant of gradient boosting, while Diffusion Boosting is not:
  - SGLB builds upon the original gradient boosting framework by adding a Gaussian noise sample (whose variance is controlled by the inverse diffusion temperature hyperparameter) to the negative gradient to form the target of each weak learner, which helps the numerical optimization to achieve convergence to the global optimum, regardless of the convexity of the loss function. In other words, SGLB is a gradient boosting algorithm that achieves better global convergence than the original algorithm. In the vanilla gradient boosting, the objective function of the weak learner $h_t(\boldsymbol{x})$ at step t is
  $$|| - \nabla_{F_{(t-1)}(\boldsymbol{x})} L\big(\boldsymbol{y}, F_{(t-1)}(\boldsymbol{x})\big) - h_t(\boldsymbol{x})||^2,$$
  i.e., the weak learners across different t's have *different* targets $-\nabla_{F_{(t-1)}(\boldsymbol{x})} L\big(\boldsymbol{y}, F_{(t-1)}(\boldsymbol{x})\big)$ to predict, but share the *same* input $\boldsymbol{x}$.
  - In Diffusion Boosting, the objective function of the weak learner $h_t\big(\hat{\boldsymbol{y}}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\big)$ is
  $$||\boldsymbol{y} - h_t\big(\hat{\boldsymbol{y}}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\big)||^2,$$
  i.e., the weak learners across different steps share the *same* target $\boldsymbol{y}$ to predict, but have *different* inputs $\hat{\boldsymbol{y}}_t$ (See Algorithm 1 Line 11).
- Objective function:
  - SGLB uses different objective functions $L\big(\boldsymbol{y}, F(\boldsymbol{x})\big)$ for regression and for classification: L is usually chosen to be the squared-error loss for regression, and logistic loss for classification.
  - Diffusion Boosting uses the same objective function for both types of supervised learning tasks: $D_{\mathrm{KL}}\big(q(\boldsymbol{y} \mid \boldsymbol{x}) \,\big\|\, p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})\big)$ as in Eq. (39), or equivalently, $||\boldsymbol{y} - h_t\big(\hat{\boldsymbol{y}}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\big)||^2$ for each weak learner.

- Stochasticity is only present during the training of SGLB, but is present during both training and inference of Diffusion Boosting:
  - For SGLB, stochasticity is introduced during training in the form of a Gaussian noise sample added to each negative gradient, in order to facilitate parameter space exploration. Once the model is trained, the prediction is the same given the same covariates x: a *point estimate* of $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}]$ for regression when the objective is the squared-error loss, or of $p(\boldsymbol{y} = 1 \mid \boldsymbol{x})$ for classification when the objective is the logistic loss.
  - For Diffusion Boosting, stochasticity is present during training when sampling $\boldsymbol{y}_{t+1}$ via the forward process (Algorithm 1 Line 6) and $\hat{\boldsymbol{y}}_t$ via the posterior (Algorithm 1 Line 9), as well as inference (Algorithm 2 Line 1 and Line 5). Given the same covariates $\boldsymbol{x}$, the output is different across different draws, each represents a sample from the learned $p(\boldsymbol{y} \mid \boldsymbol{x})$.
- The context of "diffusion":
  - In SGLB, the word "diffusion" appeared via terms "Langevin diffusion" and "inverse diffusion temperature", both of which are related to the mathematical description of the evolution of particles over time, under the context of physics or stochastic processes.
  - In Diffusion Boosting, the word "diffusion" is short for "diffusion models" as a class of generative models proposed by Sohl-Dickstein et al. (2015).
  - Note that Song et al. (2021) provides an alternative formulation of diffusion models via stochastic SDE, whose forward SDE Eq. (5) resembles the one shown in Ustimenko and Prokhorenkova (2021) Eq. (6), but the data generation process refers to the reverse SDE, *i.e.*, Eq. (6) in Song et al. (2021).

A.2.2 Distinctions Between Diffusion Boosting and GBDT Ensembles

Malinin et al. (2021) propose GBDT ensembles, an ensemble-based framework parameterized by trees, designed to estimate predictive uncertainty in supervised learning tasks, specifically targeting epistemic uncertainty. We have summarized the key differences between GBDT ensembles and Diffusion Boosting in Table 8.

Table 8: Differences between GBDT ensembles and Diffusion Boosting.

	GBDT ensembles	Diffusion Boosting
Parametric assumption on $p(\boldsymbol{y}\|\boldsymbol{x})$?	yes	no
types of predictive uncertainty estimated	total uncertainty, including both aleatoric and epistemic uncertainty	only aleatoric uncertainty
OOD detection	yes	no

We elaborate the differences listed in Table 8 as follows:

- Assumption on the parametric form of the distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$:
  - To achieve uncertainty estimation in regression, GBDT ensembles assumes $p(\boldsymbol{y} \mid \boldsymbol{x})$ to have a Gaussian distribution, whose parameters (mean and log standard deviation) are optimized via the expected Gaussian NLL.
  - Diffusion Boosting does not assume any parametric form on $p(\boldsymbol{y} \mid \boldsymbol{x})$. This is a nontrivial paradigm shift from most existing supervised learning methods, which provides additional versatility in modeling conditional distributions, including the ones with multimodality and heteroscedasticity, as shown in our toy regression examples (Figure 2).
- Types of predictive uncertainty each method focuses on modeling:
  - GBDT ensembles model both aleatoric uncertainty (data uncertainty) and epistemic uncertainty (knowledge uncertainty) via the decomposition of uncertainty (Depeweg et al., 2018) for both classification and regression, with an emphasis on epistemic uncertainty since the experiments focus on OOD and error detection.

– Diffusion Boosting follows the paradigm in CARD and only models the aleatoric uncertainty, *i.e.*, recovering the uncertainty inherent to the ground truth data generation mechanism. (We did experiment with OOD data: for toy examples with 1D x variable, we could only observe variations at the boundary of x, but the model outputs a constant value outside the boundary. This aligns with the description in Malinin et al. (2021): "... as decision trees are discriminative functions, if features have values outside the training domain, then the prediction is the same as for the 'closest' elements in the dataset. In other words, the models' behavior on the boundary of the dataset is further extended to the outer regions.")

A.3 Gradient Boosting for Least-Squares Regression

We include the algorithm from (Friedman, 2001) here as Algorithm 3 for reference, with a slight adjustment in notation.

Algorithm 3 Gradient Boosting on Squared-Error Loss

Input: Training samples $\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_1^N$
Output: Trained weak learners $\{h(\boldsymbol{x}; \boldsymbol{\alpha}_m)\}_1^M$
1: Initialize the function by $\hat{F}_0(\boldsymbol{x}) = \arg\min_s \sum_{i=1}^N L(\boldsymbol{y}_i, s) = \frac{1}{N}\sum_{i=1}^N \boldsymbol{y}_i$
2: **for** $m = 1$ to M **do**
3: Compute the negative gradient (a.k.a. pseudoresponses)

$$\tilde{\boldsymbol{y}}_i = -\left[\frac{\partial L(\boldsymbol{y}_i, F(\boldsymbol{x}_i))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=\hat{F}_{m-1}(\boldsymbol{x})} = \boldsymbol{y}_i - \hat{F}_{m-1}(\boldsymbol{x}_i), \qquad i = 1, \ldots, N$$

4: $(\rho_m, \boldsymbol{\alpha}_m) = \arg\min_{\boldsymbol{\alpha}, \rho} \sum_{i=1}^N \big(\tilde{\boldsymbol{y}}_i - \rho \cdot h(\boldsymbol{x}_i; \boldsymbol{\alpha})\big)^2$
5: $\hat{F}_m(\boldsymbol{x}_i) = \hat{F}_{m-1}(\boldsymbol{x}_i) + \rho_m \cdot h(\boldsymbol{x}_i; \boldsymbol{\alpha}_m), \qquad i = 1, \ldots, N$
6: **end for**

A.4 Derivation of the Variational Bound as the Objective to Train CARD Models

$$
\begin{aligned}
H\big(q(\boldsymbol{y}_0 \,|\, \boldsymbol{x}), p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})\big) &= -\mathbb{E}_{q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_0 \,|\, \boldsymbol{x})\big] && (51)\\
&= -\mathbb{E}_{q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})}\Big[\log\big(\int p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x}) d\boldsymbol{y}_{1:T}\big)\Big] && (52)\\
&= -\mathbb{E}_{q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})}\Big[\log\big(\int q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x}) \cdot \frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x})}{q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x})} d\boldsymbol{y}_{1:T}\big)\Big] && (53)\\
&= -\mathbb{E}_{q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})}\left[\log\left(\mathbb{E}_{q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x})}\left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x})}{q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x})}\right]\right)\right] && (54)\\
&\leq \underbrace{-\mathbb{E}_{q(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x})}{q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x})}\right]}_{\text{negative ELBO}}, && (55)
\end{aligned}
$$

in which we apply Jensen's inequality to go from (54) to (55).

A.5 Training and Sampling Algorithms of CARD-T

We present the training and sampling algorithms of CARD-T — which can be viewed as the non-amortized tree-based version of CARD — in Algorithms 4 and 5, respectively, for reference. Note that the training of each $f_{\boldsymbol{\theta}_t}$ in Algorithm 4 can be parallelized conceptually; the for loop is included for simplicity and clarity.

Algorithm 4 CARD-T Training

Input: Training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_{0,i})\}_{i=1}^N$
Output: Trained mean estimator $f_\phi(\boldsymbol{x})$ and tree ensemble $\{f_{\boldsymbol{\theta}_t}\}_{t=1}^T$

1: Pre-train $f_\phi(\boldsymbol{x})$ to estimate $\mathbb{E}[\boldsymbol{y}_0 \,|\, \boldsymbol{x}]$
2: **for** $t = T$ to 1 **do**
3: Sample $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
4: Obtain $\boldsymbol{y}_t$ sample via $q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, \boldsymbol{x})$ reparameterization:

$$\boldsymbol{y}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{y}_0 + (1 - \sqrt{\bar{\alpha}_t})\boldsymbol{\mu}_T + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_t$$

5: Train $f_{\boldsymbol{\theta}_t}$ with MSE loss to predict the forward process noise sample:

$$\mathcal{L}_{\boldsymbol{\theta}}^{(t)} = \mathbb{E}\left[\left|\left|\boldsymbol{\epsilon}_t - f_{\boldsymbol{\theta}_t}\left(\boldsymbol{y}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\right)\right|\right|^2\right]$$

6: **end for**

Algorithm 5 CARD-T Sampling

Input: Test data $\{\boldsymbol{x}_j\}_{j=1}^M$, trained $f_\phi(\boldsymbol{x})$ and $\{f_{\boldsymbol{\theta}_t}\}_{t=1}^T$
Output: Response variable prediction $\hat{\boldsymbol{y}}_{0,1}$

1: Draw $\hat{\boldsymbol{y}}_T \sim \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{I})$
2: **for** $t = T$ to 1 **do**
3: Predict the forward process noise term $\hat{\boldsymbol{\epsilon}}_t = f_{\boldsymbol{\theta}_t}\left(\hat{\boldsymbol{y}}_t, \boldsymbol{x}, f_\phi(\boldsymbol{x})\right)$
4: Compute $\hat{\boldsymbol{y}}_{0,t}$ via $q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, \boldsymbol{x})$ reparameterization:

$$\hat{\boldsymbol{y}}_{0,t} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\hat{\boldsymbol{y}}_t - (1 - \sqrt{\bar{\alpha}_t})\boldsymbol{\mu}_T - \sqrt{1 - \bar{\alpha}_t}\hat{\boldsymbol{\epsilon}}_t\right)$$

5: **if** $t > 1$ **then**
6: Draw the noisy sample $\hat{\boldsymbol{y}}_{t-1} \sim q\left(\boldsymbol{y}_{t-1} \,|\, \hat{\boldsymbol{y}}_t, \hat{\boldsymbol{y}}_{0,t}, f_\phi(\boldsymbol{x})\right)$
7: **end if**
8: **end for**
9: **return** $\hat{\boldsymbol{y}}_{0,1}$

A.6 Feature Importance Analysis on OpenML Dataset

We produced feature importance plots at the same timesteps as in Figure 3 for the trained DBT models on the *Mercedes* dataset (Table 1), as shown in Figure 4. In each plot, the features are sorted by their magnitude of feature importance. We observe that the most impactful feature in Figure 3 and 4 matches up at all selected timesteps. However, the remaining lists of influential features differ slightly. This discrepancy arises from the different methods used to measure feature impact:

- **SHAP values**: SHAP values indicate how much each feature contributes to the deviation of the prediction from the average prediction (baseline). In other words, SHAP values represent a feature's responsibility for a change in the model output.
- **Feature importance (gain)**: Feature importance based on "gain" measures the total improvement in the model's performance brought by a feature across all splits where the feature is used, where "gain" represents the reduction in the loss function when the feature is used to split the data. In other words, gain-based feature importance sums up the improvements in the loss function due to splits involving the feature.

In summary, SHAP values and feature importance provide two distinct ways of measuring the impact of features: SHAP values focus on changes in the model output, while feature importance based on gain considers improvements in the objective function.

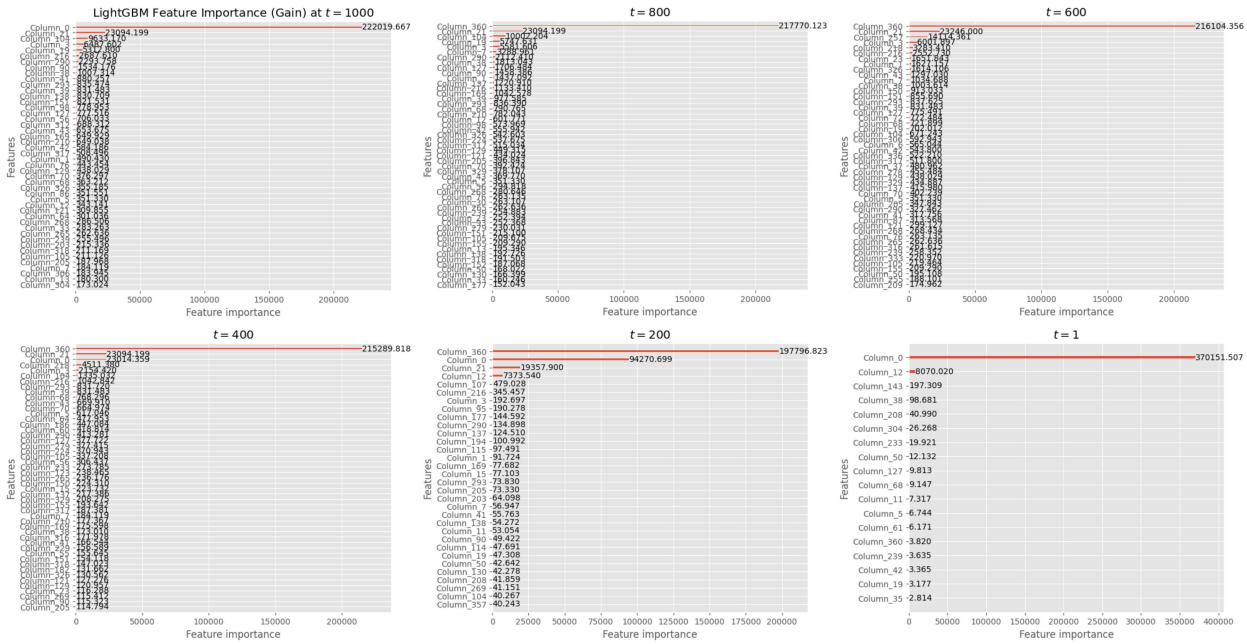

Figure 4: Feature importance plots at six diffusion timesteps.

A.7 UCI Regression Experiment Setup

The same 10 UCI regression benchmark datasets and the experimental protocol proposed in Hernández-Lobato and Adams (2015), and followed by Gal and Ghahramani (2016), Lakshminarayanan et al. (2017), and Han et al. (2022), is adopted. The dataset information in terms of the sample size and number of features is provided in Table 9. For both Kin8nm and Naval dataset, the response variable is scaled by 100.

The standard 90%/10% train-test splits in Hernández-Lobato and Adams (2015) (20 folds for all datasets except 5 for Protein and 1 for Year) is applied, and metrics are summarized by their mean and standard deviation (except Year) across all splits. We compare the performance of DBT to all aforementioned BNN frameworks: PBP, MC Dropout, and Deep Ensembles, plus another deep generative model for learning conditional distributions, GCDS (Zhou et al., 2023), as well as CARD. Following the same paradigm of BNN model assessment, we evaluate the accuracy and predictive uncertainty estimation of DBT, CARD and GCDS by reporting RMSE and NLL. Furthermore, we compute QICE for all methods to evaluate distribution matching.

Table 9: Dataset size (N observations, P features) of UCI regression tasks.

Dataset	Boston	Concrete	Energy	Kin8nm	Naval	Power	Protein	Wine	Yacht	Year
(N, P)	$(506, 13)$	$(1030, 8)$	$(768, 8)$	$(8192, 8)$	$(11,934, 16)$	$(9568, 4)$	$(45,730, 9)$	$(1599, 11)$	$(308, 6)$	$(515,345, 90)$

A.8 A Closer Look at DBT's Performance in UCI Regression Tasks

For the UCI regression tasks, DBT trained on the full dataset performs on par with CARD in most cases, while outperforming other baseline methods. We echo two insightful observations from the seminal paper on gradient boosting by Friedman (2001):

- "The performance of any function estimation method depends on the particular problem to which it is applied."
- "Every method has particular targets for which it is most appropriate and others for which it is not."

We also encourage readers to review Table 1 in Lakshminarayanan et al. (2017): the metrics are marked in bold by taking into account the error bars. By applying the convention in our work — where only **the metrics with the best mean** are marked in bold — only 2 out of 10 datasets would feature bold metrics for their proposed method (Deep Ensembles) in terms of RMSE, as opposed to the 8 out of 10 reported.

We emphasize that this is the inaugural work on **Diffusion Boosting**, and our primary goal is to introduce this framework as the first model that 1) is simultaneously a diffusion-based generative model and a boosting algorithm, and 2) can be parameterized by trees to model a conditional distribution, without any assumptions on its distributional form. We have highlighted DBT's advantage over CARD in modeling piecewise-defined functions (Section 5.1.1) and, although it secures slightly fewer Top-2 results than CARD on the UCI datasets, it already outperforms all other baseline methods. These outcomes convincingly demonstrate the potential of our proposed framework. We believe that the results endorse the framework's capabilities, and we look forward to further enhancements in future work.

A.9 Assessing the Efficacy of An Amortized GBT: One Model for All Timesteps

We replaced the noise-predicting network in CARD with an amortized GBT model and ran the experiment on UCI benchmark datasets. Despite extensive tuning, we observed consistently poor performance across all datasets. For example, on the Boston dataset, the best results we obtained were: RMSE 24.76 ± 1.59, NLL 6.65 ± 0.00, and QICE 16.98 ± 0.01, which are significantly worse than those reported in Table 2. For the GBT model, we used $1,000$ trees, $10,000$ noise samples for each instance, 31 leaves per tree, with timestep t as an additional model input.

We believe this discrepancy can be attributed to several key factors:

- Our experiment involved drawing $10,000$ noise samples for each instance, paired with a randomly sampled timestep. Consequently, on average, each instance was only paired with 10 noise samples per timestep — far fewer than our standard hyperparameter setting of 100 noise samples per tree. Increasing the number of noise samples is impractical, due to the substantial memory requirements incurred by duplicating the dataset multiple times. Notably, the UCI Boston dataset, one of the smallest datasets as shown in Table 9, contains fewer than 500 training samples.
- A tree model only has as many distinct outputs as the number of leaves, thus struggles to accommodate the diversity of outputs required across all $1,000$ diffusion timesteps to make good predictions.

These challenges highlight the limitations of using a single amortized GBT model under our experimental conditions, and substantiate our choice to employ a different tree at each timestep in our study.

A.10 Runtime Performance Analysis

We implemented DBT using the official PyTorch repository of CARD to directly leverage their model evaluation framework. (Therefore in our experiments, when the dataset does not contain missing values, the conditional mean estimator $f_\phi(\boldsymbol{x})$ is parameterized by deep neural networks out of convenience.)

During training, the time required to train each tree is relatively short. For instance, on our largest dataset, UCI Year, which has a training set dimensionality of $(46,371,500,90)$, it takes approximately 100 seconds to train each tree using an AMD EPYC 7513 CPU. In contrast, on a smaller dataset like UCI Boston with a dimensionality of $(45,500,13)$, each tree takes about 0.03 seconds to train. However, constructing the training

set for each tree consumes more time; for the UCI Year dataset, this process takes about 2.5 minutes.

During inference, the procedure is inherently sequential as each tree's output is required to construct the Gaussian mean for the sampling of $\boldsymbol{y}$ in the next diffusion timestep: see Eq. (47) and (49). This sequential nature prevents parallelization during inference, setting it apart from other contemporary gradient boosting algorithms.

Given our focus on methodology development in this work, we intend to explore system optimizations, including runtime performance, in our future research.

B Limitations

As discussed in Section 5.1.4 and Appendix A.8, our method aims to tackle tabular data, which is inherently heterogeneous. Consequently, it is challenging to identify a set of tabular datasets that adequately represent the diversity of such data. In this work, we follow the practice of Han et al. (2022) by testing our method on 10 UCI regression datasets (Table 9). These datasets are standard benchmarks in many works, including those focused on predictive uncertainty, facilitating easier benchmarking with existing methods. However, we note that half of these datasets are relatively small, which might not fairly reflect model performance in terms of quantile-based evaluation metrics that measure distribution matching, such as QICE.

Additionally, as mentioned at the beginning of Section 5, our model currently requires batch learning instead of mini-batch learning. The need for multiple noise samples per instance necessitates duplicating the entire training dataset multiple times, resulting in significant memory consumption compared to CARD or other neural network-based methods. We emphasize that this limitation arises from the choice of package (LightGBM) rather than our method itself. We have identified a potential solution using the data iterator functionality in XGBoost and plan to address this limitation in future work.

Finally, as discussed in Appendix A.10, unlike other contemporary gradient boosting libraries, our model's prediction computation cannot be parallelized due to the sequential nature of the reverse process sampling. This limitation constrains the evaluation speed during inference.

C Broader Impacts

We discussed in Section 5.2.1 the deployment of DBT as a model for learning to defer, which could potentially have positive societal impacts by enhancing decision-making through human-AI collaboration for anomaly detection. While the proposed framework offers promising solutions for tackling supervised learning problems, several potential negative societal impacts need to be considered. Privacy issues may arise if the response variable samples generated by the model inadvertently leak sensitive information or enable the re-identification of individuals in anonymized datasets. Additionally, although our proposed framework in Section 5.2 aims to promote human-in-the-loop business scenarios, the improved decision-making capabilities might still lead to increased automation in industry settings, potentially displacing workers in roles involving routine decision-making tasks. Finally, the environmental impact of training computationally expensive diffusion models should not be overlooked, as it contributes to increased energy consumption and a larger carbon footprint.