CARD: Classification and Regression Diffusion Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning the distribution of a continuous or categorical response variable $\boldsymbol{y}$ given its covariates $\boldsymbol{x}$ is a fundamental problem in statistics and machine learning. Deep neural network-based supervised learning algorithms have made great progress in predicting the mean of $\boldsymbol{y}$ given $\boldsymbol{x}$, but they are often criticized for their ability to accurately capture the uncertainty of their predictions. In this paper, we introduce classification and regression diffusion (CARD) models, which combine a denoising diffusion-based conditional generative model and a pre-trained conditional mean estimator, to accurately predict the distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$. We demonstrate the outstanding ability of CARD in conditional distribution prediction with both toy examples and real-world datasets, the experimental results on which show that CARD in general outperforms state-of-the-art methods, including Bayesian neural network-based ones that are designed for uncertainty estimation, especially when the conditional distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ is multi-modal.

1 Introduction

A fundamental problem in statistics and machine learning is to predict the response variable $\boldsymbol{y}$ given a set of covariates $\boldsymbol{x}$. Generally speaking, $\boldsymbol{y}$ is a continuous variable for regression analysis and a categorical variable for classification. Denote $f(\boldsymbol{x}) \in \mathbb{R}^C$ as a deterministic function that transforms $\boldsymbol{x}$ into a C dimensional output. Denote $f_c(\boldsymbol{x})$ as the c-th dimension of $f(\boldsymbol{x})$. Existing methods typically assume an additive noise model: for regression analysis with $\boldsymbol{y} \in \mathbb{R}^C$, one often assumes $\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$, while for classification with $y \in \{1, \ldots, C\}$, one often assumes $y = \arg\max\big(f_1(\boldsymbol{x}) + \boldsymbol{\epsilon}_1, \ldots, f_C(\boldsymbol{x}) + \boldsymbol{\epsilon}_C\big)$, where $\boldsymbol{\epsilon}_c \overset{iid}{\sim} \mathrm{EV}_1(0, 1)$, a standard type-1 extreme value distribution. Thus we have the expected value of $\boldsymbol{y}$ given $\boldsymbol{x}$ as $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}] = f(\boldsymbol{x})$ in regression and $P(y = c \,|\, \boldsymbol{x}) = \mathbb{E}[y = c \,|\, \boldsymbol{x}] = \mathrm{softmax}_c\big(f(\boldsymbol{x})\big) = \frac{\exp(f_c(\boldsymbol{x}))}{\sum_{c'=1}^{C} \exp(f_{c'}(\boldsymbol{x}))}$ in classification.

These additive-noise models are primarily focusing on accurately estimating the conditional mean $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$, while paying less attention to whether the noise distribution can accurately capture the uncertainty of $\boldsymbol{y}$ given $\boldsymbol{x}$. For this reason, they may not work well if the distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ clearly deviates from the additive-noise assumption. For example, if $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ is multi-modal, which commonly happens when there are missing categorical covariates in $\boldsymbol{x}$, then $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$ may not be close to any possible true values of $\boldsymbol{y}$ given that specific $\boldsymbol{x}$. More specifically, consider a person whose weight, height, blood pressure, and age are known but gender is unknown, then the testosterone or estrogen level of this person is likely to follow a bi-modal distribution and the chance of developing breast cancer is also likely to follow a bi-modal distribution. Therefore, these widely used additive-noise models, which use a deterministic function $f(\boldsymbol{x})$ to characterize the conditional mean of $\boldsymbol{y}$, are inherently restrictive in their ability for uncertainty estimation.

In this paper, our goal is to accurately recover the full distribution of $\boldsymbol{y}$ conditioning on $\boldsymbol{x}$ given a set of N training data points, denoted as $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{1,N}$. To realize this goal, we consider the diffusion-based (a.k.a. score-based) generative models [50, 20, 51, 53] and inject covariate-dependence into

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

both the forward (inference) and reverse (generative) diffusion chains. Our method can model the conditional distribution of both continuous and categorical $\boldsymbol{y}$ variable, and the algorithms developed under this method will be collectively referred to as **C**lassification **A**nd **R**egression **D**iffusion (CARD) models.

Diffusion-based generative models have received significant recent attention due to not only their ability to generate high-dimensional data, such as high-resolution photo-realistic images, but also their training stability. They can be understood from the perspective of score matching [22, 56] and Langevin dynamics [39, 58], as pioneered by Song and Ermon [50]. They can also be understood from the perspective of diffusion probabilistic models [48, 20], which first define a forward diffusion to transform the data into noise and then an inverse diffusion to regenerate the data from noise.

These previous methods mainly focus on unconditional generative modeling. While there exist guided-diffusion models [50, 53, 7, 40, 43] that target on generating high-resolution photo-realistic images that match the semantic meanings or content of the label, text, or corrupted-images, we focus on studying diffusion-based conditional generative modeling at a more fundamental level. In particular, our goal is to thoroughly investigate whether CARD can help accurately recovery $p(\boldsymbol{y} \,|\, \boldsymbol{x}, \mathcal{D})$, the predictive distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ after observing data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1,N}$. In other words, our focus is on regression analysis of continuous or categorical response variables given their corresponding covariates.

We summarize our main contributions as follows: 1) We show CARD, which injects covariate-dependence and a pre-trained conditional mean estimator into both the forward and reverse diffusion chains to construct a denoising diffusion probabilistic model, provides an accurate estimation of $p(\boldsymbol{y} \,|\, \boldsymbol{x}, \mathcal{D})$. 2) We provide a new metric to better evaluate how well a regression model captures the full distribution $p(\boldsymbol{y} \,|\, \boldsymbol{x}, \mathcal{D})$. 3) Experiments on standard benchmarks for regression analysis show that CARD achieves state-of-the-art results, using both existing metrics and the new one.

2 Methods and Algorithms for CARD

2.1 Problem Statement

Given the ground-truth response variable $\boldsymbol{y}_0$ and its covariates $\boldsymbol{x}$, and assuming a sequence of intermediate uncertain prediction $\boldsymbol{y}_{1:T}$ made by the diffusion model, the goal of supervised learning is to learn a good model such that the log-likelihood is maximized by optimizing the following ELBO:

$$\log p_\theta(\boldsymbol{y}_0 \,|\, \boldsymbol{x}) = \log \int p_\theta(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x}) d\boldsymbol{y}_{1;T} \geq \mathbb{E}_{q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x})} \left[\log \frac{p_\theta(\boldsymbol{y}_{0:T} \,|\, \boldsymbol{x})}{q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x})} \right], \tag{1}$$

where $q(\boldsymbol{y}_{1:T} \,|\, \boldsymbol{y}_0, \boldsymbol{x})$ is called the forward process or diffusion process in the concept of diffusion models [20]. Denoting $D_{\mathrm{KL}}(q \,||\, p)$ as the Kullback–Leibler (KL) divergence from distributions p to q. The above objective can be rewritten as

$$\mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{y}_0, \boldsymbol{x}) \coloneqq \mathcal{L}_0(\boldsymbol{y}_0, \boldsymbol{x}) + \sum_{t=2}^{T} \mathcal{L}_{t-1}(\boldsymbol{y}_0, \boldsymbol{x}) + \mathcal{L}_T(\boldsymbol{y}_0, \boldsymbol{x}), \tag{2}$$

$$\mathcal{L}_0(\boldsymbol{y}_0, \boldsymbol{x}) \coloneqq \mathbb{E}_q \left[-\log p_\theta(\boldsymbol{y}_0 \,|\, \boldsymbol{y}_1, \boldsymbol{x}) \right], \tag{3}$$

$$\mathcal{L}_{t-1}(\boldsymbol{y}_0, \boldsymbol{x}) \coloneqq \mathbb{E}_q \left[D_{\mathrm{KL}}\big(q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) \,\big\|\, p_\theta(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{x})\big) \right], \tag{4}$$

$$\mathcal{L}_T(\boldsymbol{y}_0, \boldsymbol{x}) \coloneqq \mathbb{E}_q \left[D_{\mathrm{KL}}\big(q(\boldsymbol{y}_T \,|\, \boldsymbol{y}_0, \boldsymbol{x}) \,\big\|\, p(\boldsymbol{y}_T \,|\, \boldsymbol{x})\big) \right]. \tag{5}$$

Here we follow the convention to assume $\mathcal{L}_T$ does not depend on any parameter and it will be close to zero by carefully diffusing the observed response variable $\boldsymbol{y}_0$ towards a pre-assumed distribution $p(\boldsymbol{y}_T \,|\, \boldsymbol{x})$. The remaining terms will make sure the model $p_\theta(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{x})$ predicts at different diffusion steps and we hope this gives us different scales in uncertainty measure. Different from the vanilla diffusion model, we assume the endpoint of the diffusion as

$$p(\boldsymbol{y}_T \,|\, \boldsymbol{x}) = \mathcal{N}(f_\phi(\boldsymbol{x}), \boldsymbol{I}), \tag{6}$$

where $f_\phi(\boldsymbol{x})$ is the pre-knowledge of the relation between $\boldsymbol{x}$ and $\boldsymbol{y}_0$, *e.g.*, pre-trained with $\mathcal{D}$ to approximate $\mathbb{E}[\boldsymbol{y} \,|\, \boldsymbol{x}]$, or $\mathbf{0}$ if we assume the relation is unknown. With a diffusion schedule $\{\beta_t\}_{t=1:T} \in (0,1)^T$, we specify the forward diffusion process conditional distributions in a similar

fashion as Pandey et al. [41], but for all timesteps including $t = 1$:

$$q\big(\boldsymbol{y}_t \,|\, \boldsymbol{y}_{t-1}, f_\phi(\boldsymbol{x})\big) = \mathcal{N}\big(\boldsymbol{y}_t; \sqrt{1-\beta_t}\boldsymbol{y}_{t-1} + (1-\sqrt{1-\beta_t})f_\phi(\boldsymbol{x}), \beta_t \boldsymbol{I}\big), \tag{7}$$

which admits a closed-form sampling distribution with arbitary timestep t:

$$q\big(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, f_\phi(\boldsymbol{x})\big) = \mathcal{N}\big(\boldsymbol{y}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{y}_0 + (1-\sqrt{\bar{\alpha}_t})f_\phi(\boldsymbol{x}), (1-\bar{\alpha}_t)\boldsymbol{I}\big), \tag{8}$$

where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_t \alpha_t$. Note that Eq. (7) can be viewed as an interpolation between true data $\boldsymbol{y}_0$ and the predicted conditional expectation $f_\phi(\boldsymbol{x})$, which gradually changes from the former to the latter throughout the forward process.

Such formulation corresponds to a tractable forward process posterior:

$$q(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) = q\big(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{y}_0, f_\phi(\boldsymbol{x})\big) = \mathcal{N}\Big(\boldsymbol{y}_{t-1}; \tilde{\boldsymbol{\mu}}\big(\boldsymbol{y}_t, \boldsymbol{y}_0, f_\phi(\boldsymbol{x})\big), \tilde{\beta}_t \boldsymbol{I}\Big), \tag{9}$$

where

$$\tilde{\boldsymbol{\mu}} := \underbrace{\frac{\beta_t\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}}_{\gamma_0}\boldsymbol{y}_0 + \underbrace{\frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}}_{\gamma_1}\boldsymbol{y}_t + \underbrace{\left(1 + \frac{(\sqrt{\bar{\alpha}_t}-1)(\sqrt{\alpha_t}+\sqrt{\bar{\alpha}_{t-1}})}{1-\bar{\alpha}_t}\right)}_{\gamma_2} f_\phi(\boldsymbol{x}),$$

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t.$$

We provide the derivation in Appendix A.1. The labels under the terms are used in Algorithm 2.

2.2 CARD for Regression

For regression problems, the goal of the reverse diffusion process is to gradually recover the distribution of the noise term, the aleatoric or local uncertainty inherent in the observations [25, 57], enabling us to generate samples that match the true conditional $p(\boldsymbol{y} \,|\, \boldsymbol{x})$.

Following the reparameterization introduced by denoising diffusion probabilistic models (DDPM) [20], we construct $\boldsymbol{\epsilon}_\theta\big(\boldsymbol{x}, \boldsymbol{y}_t, f_\phi(\boldsymbol{x}), t\big)$, which is a function approximator parameterized by a deep neural network that predicts the forward diffusion noise $\boldsymbol{\epsilon}$ sampled for $\boldsymbol{y}_t$. The training and inference procedure can be carried out in a standard DDPM manner.

Algorithm 1 Training (Regression)

1: Pre-train $f_\phi(\boldsymbol{x})$ that predicts $\mathbb{E}(\boldsymbol{y} \,|\, \boldsymbol{x})$ with MSE
2: **repeat**
3: Draw $y_0 \sim q(\boldsymbol{y}_0 \,|\, \boldsymbol{x})$
4: Draw $t \sim \text{Uniform}(\{1 \dots T\})$
5: Draw $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
6: Compute noise estimation loss

$$\mathcal{L}_{\boldsymbol{\epsilon}} = ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}, \sqrt{\bar{\alpha}_t}\boldsymbol{y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} + \sqrt{1-\bar{\alpha}_t}f_\phi(\boldsymbol{x}), f_\phi(\boldsymbol{x}), t)||^2$$

7: Take numerical optimization step on:

$$\nabla_\theta \mathcal{L}_{\boldsymbol{\epsilon}}$$

8: **until** Convergence

2.3 CARD for Classification

We formulate the classification tasks in a similar fashion as in Section 2.2, where we:

1. Replace the continuous response variable with a one-hot encoded label vector for $\boldsymbol{y}_0$;
2. Replace the mean estimator with a pre-trained classifier that outputs softmax probabilities of the class labels for $f_\phi(\boldsymbol{x})$.

Algorithm 2 Inference (Regression)

1: $\boldsymbol{y}_T \sim \mathcal{N}(f_\phi(\boldsymbol{x}), \boldsymbol{I})$
2: **for** $t = T$ to 1 **do**
3: Draw $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$if $t > 1$
4: Calculate reparameterized $\hat{\boldsymbol{y}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\Big(\boldsymbol{y}_t - (1-\sqrt{\bar{\alpha}_t})f_\phi(\boldsymbol{x}) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta\big(\boldsymbol{x}, \boldsymbol{y}_t, f_\phi(\boldsymbol{x}), t\big)\Big)$
5: Let $\boldsymbol{y}_{t-1} = \gamma_0\hat{\boldsymbol{y}}_0 + \gamma_1\boldsymbol{y}_t + \gamma_2 f_\phi(\boldsymbol{x}) + \sqrt{\tilde{\beta}_t}\boldsymbol{z}$ if $t > 1$, else set $\boldsymbol{y}_{t-1} = \hat{\boldsymbol{y}}_0$
6: **end for**
7: **return** $\boldsymbol{y}_0$

This construction no longer assumes $\boldsymbol{y}_0$ to be drawn from a categorical distribution, but instead treating each one-hot label as a class prototype. The sampling procedure would output reconstructed $\boldsymbol{y}_0$ in the range of real numbers for each dimension, instead of a vector in the probability simplex. We convert such output to a probability vector in a softmax form of a temperature-weighted the Brier score [5], which computes the squared error between the prediction and a vector of 1s with length C, the number of classes. Mathematically, the probability of predicting the kth class $y = k$ and the final point prediction $\hat{y}$ can be expressed as

$$\Pr(y = k) = \frac{\exp(-(\boldsymbol{y}_0 - \boldsymbol{1}_C)_k^2/\tau)}{\sum_{i=1}^{C}\exp(-(\boldsymbol{y}_0 - \boldsymbol{1}_C)_i^2/\tau)};\ \hat{y} = \arg\max_k \big(-(\boldsymbol{y}_0 - \boldsymbol{1}_C)_k^2\big). \tag{10}$$

where $\tau > 0$ is the temperature parameter, and $(\boldsymbol{y}_0 - \boldsymbol{1}_C)_k^2$ indicates the kth dimension of the vector of element-wise square error between $\boldsymbol{y}_0$ and $\boldsymbol{1}_C$, *i.e.*, $(\boldsymbol{y}_0 - \boldsymbol{1}_C)_k^2 = \|\boldsymbol{y}_{0_k} - 1\|^2$. Intuitively, this construction would assign the class whose raw output in the sampled $\boldsymbol{y}_0$ is closest to the true class, encoded by the value of 1 in the one-hot label, with the highest probability.

Given the same covariates $\boldsymbol{x}$, the stochasticity of the generative model would give us a different class prototype reconstruction after each reverse process sampling, which enables us to construct predicted probability intervals for all class labels. Such stochastic reconstruction is in a similar fashion as DALL-E 2 [43] that applies a diffusion prior to reconstruct the image embedding by conditioning on the text embedding during the reverse diffusion process, which is a key step in the diversity of generated images.

3 Experiments

For the hyperparameters of CARD in both regression and classification tasks, we set the number of timesteps as $T = 1000$, a linear noise schedule with $\beta_1 = 10^{-4}$ and $\beta_T = 0.02$, same as Ho et al. [20]. We provide a more detailed walk-through of the experimental setup, including training and network architecture, in Appendix A.7.

3.1 Regression

Putting aside its statistical interpretation, the word *regress* indicates a direction opposite to *progress*, suggesting a less developed state. Such semantics in fact translates well into the statistical domain, in the sense that traditional regression analysis methods often only focus on estimating $\mathbb{E}(\boldsymbol{y} \mid \boldsymbol{x})$, while leaving out all the other details about $p(\boldsymbol{y} \mid \boldsymbol{x})$. In recent years, Bayesian neural networks (BNNs) have emerged as a class of models that aims at estimating the uncertainty [19, 11, 30, 55], providing a more complete picture of $p(\boldsymbol{y} \mid \boldsymbol{x})$. The metric that they use to quantify uncertainty estimation, negative log-likelihood (NLL), is computed with a Gaussian density, implying their assumption such that the conditional distributions $p(\boldsymbol{y} \mid \boldsymbol{x} = x)$ for all x are Gaussian. However, this assumption is very difficult to verify for real-world datasets: the covariates can be arbitrarily high-dimensional, making the feature space increasingly sparse with respect to the number of collected observations.

To accommodate the need for uncertainty estimation without imposing such restriction for the parametric form of $p(\boldsymbol{y} \mid \boldsymbol{x})$, we apply the following two metrics, both of which are designed to empirically evaluate the level of similarity between the learned and the true conditional distributions:

1. Prediction Interval Coverage Probability (PICP);

2. Quantile Interval Coverage Error (QICE).

PICP has been described in Yao et al. [61], whereas QICE is a new metric proposed by us. We describe both of them in what follows.

3.1.1 PICP and QICE

The PICP is computed as

$$\text{PICP} \coloneqq \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{y_n \geq \hat{y}_n^{\text{low}}} \cdot \mathbb{1}_{y_n \leq \hat{y}_n^{\text{high}}}, \tag{11}$$

where $\hat{y}_n^{\text{low}}$ and $\hat{y}_n^{\text{high}}$ represents the low and high percentiles, respectively, of our choice for the predicted $\boldsymbol{y}$ outputs given the same $\boldsymbol{x}$ input. This metric measures the proportion of true observations that fall in the percentile range of the generated $\boldsymbol{y}$ samples given each $\boldsymbol{x}$ input. Intuitively, when the learned distribution represents the true distribution well, this measurement should be close to the difference between the selected low and high percentiles. In this paper, we choose the 2.5^{th} and 97.5^{th} percentile, thus an ideal PICP value for the learned model should be 95%.

Meanwhile, there is a caveat for this metric: for example, imagine a situation where the 2.5^{th} to 97.5^{th} percentile of the learned distribution happens to cover the data between the 1st and 96th percentiles from the true distribution. Given enough samples, we shall still obtain a PICP value close to 95%, but clearly there is a mismatch between the learned distribution and the true one.

Based on such reasoning, we propose a new empirical metric QICE, which by design can be viewed as PICP with finer granularity. To compute QICE, we first generate enough $\boldsymbol{y}$ samples given each $\boldsymbol{x}$, and divide them into M bins with roughly equal sizes. We would obtain the corresponding quantile values at each boundary. In this paper, we set $M = 10$, and obtain the following 10 quantile intervals (QIs) of the generated $\boldsymbol{y}$ samples: below the 10th percentile, between the 10th and 20th percentiles, $\ldots$, between the 80th and 90th percentiles, and above the 90th percentile. Optimally, when the learned conditional distribution is identical to the true one, given enough samples from both learned and true distribution we shall observe about 10% of true data falling into each of these 10 QIs.

We define *QICE* to be the mean absolute error between the proportion of true data contained by each QI and the optimal proportion, which is $1/M$ for all intervals:

$$\text{QICE} \coloneqq \frac{1}{M}\sum_{m=1}^{M}\left| r_m - \frac{1}{M}\right|, \text{ where } r_m = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}_{y_n \geq \hat{y}_n^{\text{low}_m}} \cdot \mathbb{1}_{y_n \leq \hat{y}_n^{\text{high}_m}}. \tag{12}$$

Intuitively, under optimal scenario with enough samples, we shall obtain a QICE value of 0. Note that each r_m is indeed the PICP for the corresponding QI with boundaries at $\hat{y}_n^{\text{low}_m}$ and $\hat{y}_n^{\text{high}_m}$. Since the true $\boldsymbol{y}$ for each $\boldsymbol{x}$ is guaranteed to fall into one of these QIs, we are thus able to overcome the mismatch issue described in the above example for PICP: fewer true instances falling into one QI would result in more instances captured by another QI, thus increasing the absolute error for both QIs.

QICE is similar to NLL in the sense that it also utilizes the summary statistics of the samples from the learned distribution conditional on each new $\boldsymbol{x}$ to empirically evaluate how well the model fits the true data. Meanwhile, it does not assume any parametric form on the conditional distribution, making it a much more generalizable metric to measure the level of distributional match between the learned and the underlying true conditional distributions, especially when the true conditional distribution is known to be multi-modal. We will demonstrate this point through the regression toy examples.

3.1.2 Toy Examples

To demonstrate the effectiveness of CARD in regression tasks for not only learning the conditional mean $\mathbb{E}(\boldsymbol{y} \mid \boldsymbol{x})$, but also recreating the ground truth data generating mechanism, we first apply CARD on 8 toy examples, whose data generating functions are designed to possess different statistical characteristics: some have a uni-modal symmetric distribution for their error term (linear regression, quadratic regression, sinusoidal regression[1]), others have heteroscedasticity (log-log linear regression,

[1]The data generating function was originally proposed in [3].

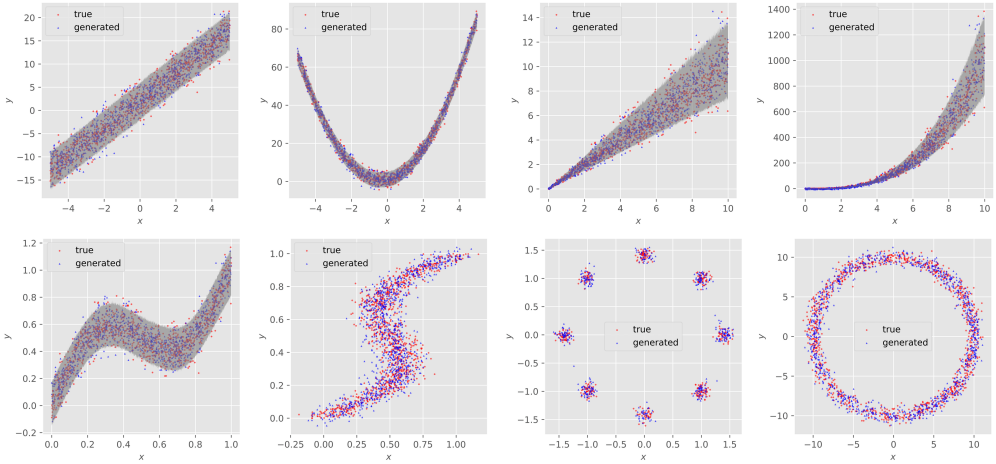

Figure 1: Regression toy example scatter plots. (**Top**) left to right: linear regression, quadratic regression, log-log linear regression, log-log cubic regression; (**Bottom**) left to right: sinusoidal regression, inverse sinusoidal regression, 8 Gaussians, full circle.

log-log cubic regression) or multi-modality (inverse sinusoidal regression[2], 8 Gaussians[3], full circle). We show that CARD can generate samples that are visually indistinguishable from the true response variables of the new covariates, as well as quantitatively match the true distribution in terms of some summary statistics. [rev.: We present the scatter plots of both true and generated data for all 8 tasks in Figure 1. For tasks with uni-modal conditional distribution, we fill the region between the 2.5^{th} and 97.5^{th} percentile of the generated $\boldsymbol{y}$'s. We observe that within each task, the generated samples blend remarkably well with the true test instances, suggesting the capability of reconstructing the underlying data generation mechanism by CARD. A more detailed description of the toy examples, including more quantitative analyses, is presented in Appendix A.12.]

3.1.3 UCI Regression Tasks

We continue to investigate our model through experiments on real-world datasets. We adopt the same set of 10 UCI regression benchmark datasets [8] as well as the experimental protocol proposed in Hernández-Lobato and Adams [19] and followed by Gal and Ghahramani [11] and Lakshminarayanan et al. [30]. The dataset information is provided in Table 11.

We apply multiple train-test splits with $90\%/10\%$ ratio in the same way as [19] (20 folds for all datasets except 5 for Protein and 1 for Year), and report the metrics by their mean and standard deviation across all splits. We compare our method to all aforementioned BNN frameworks: PBP, MC Dropout, and Deep Ensembles, as well as another deep generative model that estimates a conditional distribution sampler, GCDS [64]. Similar to BNNs, we evaluate the accuracy and predictive uncertainty estimation of CARD by reporting RMSE and NLL. Furthermore, we also report QICE for all methods to evaluate distributional matching. Since this new metric was not applied in previous methods, we re-ran the experiments for all BNNs and obtained comparable or slightly better results reported in their literature. Further details about the experiment setup for these models can be found in Appendix A.8. The experiment results with corresponding metrics are shown in Tables 1, 2, and 3, with the number of times each model achieves the best corresponding metric reported in the last row.

We observe that CARD outperforms existing methods, often by a considerable margin (especially on larger datasets), in all metrics for most of the datasets, and is competitive with the best method for the remaining ones: we obtain state-of-the-art results in 9 out of 10 datasets in terms of RMSE, 8 out of 10 for NLL, and 5 out of 10 for QICE. It is worth noting that although we do not explicitly optimize our model by MSE or by NLL, we still obtain better results than models trained with these objectives.

[2]Swap x and the generated y from the sinusoidal regression task.

[3]We set the coordinates of 8 modes at $(\sqrt{2}, 0)$, $(-\sqrt{2}, 0)$, $(0, \sqrt{2})$, $(0, -\sqrt{2})$, $(1, 1)$, $(1, -1)$, $(-1, 1)$, $(-1, -1)$, and add a noise sample to x and y coordinate of a mode to generate one instance.

Table 1: RMSE of UCI regression tasks. For both [1]Kin8nm and [2]Naval dataset, we multiply the response variable by 100 to match the scale of others.

Dataset	RMSE ↓				
	PBP	MC Dropout	Deep Ensembles	GCDS	CARD (ours)
Boston	2.89 ± 0.74	3.06 ± 0.96	3.17 ± 1.05	2.75 ± 0.58	$\mathbf{2.61 \pm 0.63}$
Concrete	5.55 ± 0.46	5.09 ± 0.60	4.91 ± 0.47	5.39 ± 0.55	$\mathbf{4.77 \pm 0.46}$
Energy	1.58 ± 0.21	1.70 ± 0.22	2.02 ± 0.32	0.64 ± 0.09	$\mathbf{0.52 \pm 0.07}$
Kin8nm[1]	9.42 ± 0.29	7.10 ± 0.26	8.65 ± 0.47	8.88 ± 0.42	$\mathbf{6.32 \pm 0.18}$
Naval[2]	0.41 ± 0.08	0.08 ± 0.03	0.09 ± 0.01	0.14 ± 0.05	$\mathbf{0.02 \pm 0.00}$
Power	4.10 ± 0.15	4.04 ± 0.14	4.02 ± 0.15	4.11 ± 0.16	$\mathbf{3.93 \pm 0.17}$
Protein	4.65 ± 0.02	4.16 ± 0.12	4.45 ± 0.02	4.50 ± 0.02	$\mathbf{3.73 \pm 0.01}$
Wine	0.64 ± 0.04	$\mathbf{0.62 \pm 0.04}$	0.63 ± 0.04	0.66 ± 0.04	0.63 ± 0.04
Yacht	0.88 ± 0.22	0.84 ± 0.27	1.19 ± 0.49	0.79 ± 0.26	$\mathbf{0.65 \pm 0.25}$
Year	$8.86 \pm$ NA	$8.77 \pm$ NA	$8.79 \pm$ NA	$9.20 \pm$ NA	$\mathbf{8.70} \pm$ NA
# best	0	1	0	0	**9**

Table 2: NLL of UCI regression tasks.

Dataset	NLL ↓				
	PBP	MC Dropout	Deep Ensembles	GCDS	CARD (ours)
Boston	2.53 ± 0.27	2.46 ± 0.12	2.35 ± 0.16	18.66 ± 8.92	$\mathbf{2.35 \pm 0.12}$
Concrete	3.19 ± 0.05	3.21 ± 0.18	$\mathbf{2.93 \pm 0.12}$	13.64 ± 6.88	2.96 ± 0.09
Energy	2.05 ± 0.05	1.50 ± 0.11	1.40 ± 0.27	1.46 ± 0.72	$\mathbf{1.04 \pm 0.06}$
Kin8nm	-0.83 ± 0.02	-1.14 ± 0.05	-1.06 ± 0.02	-0.38 ± 0.36	$\mathbf{-1.32 \pm 0.02}$
Naval	-3.97 ± 0.10	-4.45 ± 0.38	-5.94 ± 0.10	-5.06 ± 0.48	$\mathbf{-7.54 \pm 0.05}$
Power	2.92 ± 0.02	2.90 ± 0.03	2.89 ± 0.02	2.83 ± 0.06	$\mathbf{2.82 \pm 0.02}$
Protein	3.05 ± 0.00	2.80 ± 0.08	2.89 ± 0.02	2.81 ± 0.09	$\mathbf{2.49 \pm 0.03}$
Wine	1.03 ± 0.03	0.93 ± 0.06	0.96 ± 0.06	6.52 ± 21.86	$\mathbf{0.92 \pm 0.05}$
Yacht	1.58 ± 0.08	1.73 ± 0.22	1.11 ± 0.18	$\mathbf{0.61 \pm 0.34}$	0.90 ± 0.08
Year	$3.69 \pm$ NA	$3.42 \pm$ NA	$3.44 \pm$ NA	$3.43 \pm$ NA	$\mathbf{3.34} \pm$ NA
# best	0	0	1	1	**8**

Table 3: QICE (in %) of UCI regression tasks.

Dataset	QICE ↓				
	PBP	MC Dropout	Deep Ensembles	GCDS	CARD (ours)
Boston	3.50 ± 0.88	3.82 ± 0.82	$\mathbf{3.37 \pm 0.00}$	11.73 ± 1.05	3.45 ± 0.83
Concrete	2.52 ± 0.60	4.17 ± 1.06	2.68 ± 0.64	10.49 ± 1.01	$\mathbf{2.30 \pm 0.66}$
Energy	6.54 ± 0.90	5.22 ± 1.02	$\mathbf{3.62 \pm 0.58}$	7.41 ± 2.19	4.91 ± 0.94
Kin8nm	1.31 ± 0.25	1.50 ± 0.32	1.17 ± 0.22	7.73 ± 0.80	$\mathbf{0.92 \pm 0.25}$
Naval	4.06 ± 1.25	12.50 ± 1.95	6.64 ± 0.60	5.76 ± 2.25	$\mathbf{0.80 \pm 0.21}$
Power	$\mathbf{0.82 \pm 0.19}$	1.32 ± 0.37	1.09 ± 0.26	1.77 ± 0.33	0.92 ± 0.21
Protein	1.69 ± 0.09	2.82 ± 0.41	2.17 ± 0.16	2.33 ± 0.18	$\mathbf{0.71 \pm 0.11}$
Wine	$\mathbf{2.22 \pm 0.64}$	2.79 ± 0.56	2.37 ± 0.63	3.13 ± 0.79	3.39 ± 0.69
Yacht	6.93 ± 1.74	10.33 ± 1.34	7.22 ± 1.41	$\mathbf{5.01 \pm 1.02}$	8.03 ± 1.17
Year	$2.96 \pm$ NA	$2.43 \pm$ NA	$2.56 \pm$ NA	$1.61 \pm$ NA	$\mathbf{0.53} \pm$ NA
# best	2	0	2	1	**5**

3.2 Classification

Similar to Lakshminarayanan et al. [30], our motivation for classification is not to achieve the state-of-the-art performance in terms of mean accuracy on the benchmark datasets, which is strongly related to network architecture design. Our goal is two-fold:

1. We aim to solve classification problems via a generative model, emphasizing its capability to improve the performance of a base classifier with deterministic outputs [rev.: in terms of accuracy];
2. [rev.: We intent to provide an alternative sense of uncertainty, by introducing the idea of model confidence at instance level, *i.e.*, how sure the model is about *each* of its predictions, through the stochasticity of outputs from a generative model.]

As another type of supervised learning problems, classification is different than regression mainly for the response variable being discrete class labels instead of continuous values. [rev.: The conventional operation is to cast the classifier output as a point estimate, with a value between 0 and 1. Such design is intended for prediction interpretability: since humans already have a cognitive intuition for probabilities [6], the output from a classification model is intended to convey a sense of likelihood for a particular class label. In other words, the predicted probability should reflect its confidence,

i.e., a level of certainty, in predicting such label. Guo et al. [17] provide the following example of a good classifier, whose outputs align with human intuition for probabilities: if the model outputs a probability prediction of 0.8, we hope it indicates that the model is 80% sure that its prediction is correct; given 100 predictions of 0.8, one shall expect roughly 80 of them to be classified correctly.

In that sense, a good classification algorithm not only can predict the correct label, but also can reflect the true correctness likelihood through its probability predictions, *i.e.*, providing calibrated confidence [17]. To evaluate the level of miscalibration by a model, metrics like Expected Calibration Error (ECE) and Maximum Calibration Error (MCE) [38] have been adopted in recent literature [29, 47] for image classification tasks, and calibration methods like Platt scaling and isotonic regression have been developed to improve such alignment [17].

Note that these methods are all based on point estimate predictions by the classifier. Furthermore, these alignment metrics can only be computed at a subgroup level in practice, instead of at instance level. In other words, one may not be able to make the claim with the existing classification framework such that *given a particular test instance*, how confident the classifier is in its prediction to be correct. We discuss our analysis in ECE with more details in Appendix A.15, which may help justify our motivation in introducing an alternative way to measure model confidence at the level of *individual test instances* in Sec. 3.2.1.]

3.2.1 Making Predictions with Instance Level Model Confidence via Generative Models

[rev.: We propose the following framework to access model confidence for its predictions *at instance level*: for each test instance, we first sample N class prototype reconstructions by CARD through the classification version of Algorithm 2, and then perform the following computations:

1. We directly calculate the prediction interval width (PIW) between the 2.5^{th} and 97.5^{th} percentiles of the N reconstructed values for all classes, *i.e.*, with C different classes in total, we would obtain C PIWs for each instance;
2. We then convert the samples into probability space with Eq. (10), and apply paired two-sample t-test as an uncertainty estimation method proposed in Fan et al. [9]: we obtain the most and second most predicted classes for each instance, and test whether the difference in their mean predicted probability is statistically significant.

This framework would require the classifier to not produce the exact same output each time, since the goal is to construct prediction intervals for each of the class labels. Therefore, the class of generative models is a preferable modeling choice due to its ability to produce stochastic outputs, instead of just a point estimate as traditional classifiers.

In practice, we view each one-hot label as a class prototype in real continuous space (introduced in Section 2.3), and we use a generative model to reconstruct this prototype in a stochastic fashion. The intuition is that if the classifier is sure about the class that a particular instance belongs to, it would precisely reconstruct the original prototype vector without much uncertainty; otherwise, different prototype reconstructions of the same test instance tend to have more variations: under the context of denoising diffusion models, given different samples from the prior distribution at timestep T, the class prototype reconstructions would appear rather different from each other.]

3.2.2 Classification with Prediction Uncertainty Evaluation on CIFAR-10 Dataset

[rev.: We demonstrate our experimental results on the CIFAR-10 dataset. We first contextualize the performance of CARD in conventional metrics including accuracy and NLL with other BNNs in ResNet-18 architecture in Table 4, with metrics reported in [55], a recent work in BNNs that proposes tighter ELBOs to improve variational inference performance and prior hyperparameter optimization. Following the recipe in Section 2.3, we first pre-train a deterministic classifier with the same ResNet-18 architecture, and achieve a test accuracy of 90.39%, with which we proceed to train CARD. We then obtain our instance prediction through majority vote, *i.e.*, the most predicted class label among its N samples, and achieve an improved test accuracy with a mean of 90.93% across 10 runs, showing its ability to improve test accuracy from the base classifier. Our NLL result is competitive among the best ones, even though the model is not optimized with a cross-entropy objective function, as we assume the class labels to be in the real continuous space.

Table 4: Comparison of accuracy (in %) and NLL for CIFAR-10 classification task with other BNNs.

Model	CMV-MF-VI	CM-MF-VI	CV-MF-VI	MF-VI	MC Dropout	MAP	CARD
Accuracy	86.25 ± 0.06	86.66 ± 0.24	79.78 ± 0.30	77.08 ± 1.14	83.64 ± 0.28	84.69 ± 0.35	$\mathbf{90.93 \pm 0.02}$
NLL	0.41 ± 0.00	$\mathbf{0.39 \pm 0.00}$	0.59 ± 0.00	0.68 ± 0.02	0.49 ± 0.00	0.93 ± 0.02	0.46 ± 0.00

We now present the results from one model run with the proposed framework in Sec. 3.2.1 for evaluating instance level prediction confidence. After obtaining the PIW and paired two-sample t-test ($\alpha = 0.05$) result from each test instance, we first split the test instances into two groups with correct and incorrect majority-vote predictions, obtain only the PIW corresponding to the true class for each instance, and compute the mean PIW of the true class within each group. We also split them by t-test rejection status, and compute the mean accuracy in each group. We report the results in Table 5, where the metrics are computed across all test instances and at the level of each class label.]

Table 5: PIW (multiplied by 100) and t-test results for CIFAR-10 classification task.

Class	Accuracy	PIW		Accuracy by t-test Status	
		Correct	Incorrect	Rejected	Not-Rejected (Count)
All	90.95%	2.37	21.52	91.25%	42.86% (63)
1	91.00%	3.28	18.83	91.51%	45.45% (11)
2	96.00%	0.55	29.27	96.19%	33.33% (3)
3	87.30%	2.65	24.40	87.55%	25.00% (4)
4	81.90%	5.48	21.45	82.10%	63.64% (11)
5	93.30%	2.41	30.02	93.67%	20.00% (5)
6	84.70%	4.16	19.57	85.21%	46.15% (13)
7	94.20%	1.84	26.01	94.38%	33.33% (3)
8	92.80%	1.96	19.35	93.07%	25.00% (4)
9	95.30%	0.56	15.75	95.49%	33.33% (3)
10	93.00%	1.50	14.04	93.26%	50.00% (6)

We observe from Table 5 that under the scope of the entire test set, mean PIW of the true class label among the correct predictions is narrower than that of the incorrect predictions by an order of magnitude, [rev.: indicating that when CARD is making correct predictions, its class label reconstructions have much smaller variations]. [rev.: We may interpret such results as that CARD can reveal what it does not know through the relativity in reconstruction variations. Furthermore,] when comparing mean PIWs across different classes, we observe that a class with higher prediction accuracy tends to have a sharper contrast in true label PIW between correct and incorrect predictions; additionally, the PIW values of both types of predictions tend to be larger in a less accurate class. [rev.: Meanwhile, it is worth noting that if we predict the class label by the one with the narrowest PIW for each instance, we can already obtain a test accuracy of 87.84%, suggesting a strong correlation between the prediction correctness and model confidence (in terms of label reconstruction variability).] Meanwhile, we observe that the accuracy of test instances rejected by the t-test is much higher than that of the not-rejected ones, both across the entire test set and within each class.

We point out that these metrics can reflect how sure CARD is about the correctness of its predictions, and can thus be used as an important indicator of whether the model prediction can be trusted or not. Therefore, it has the potential to be further applied in the human-machine collaboration domain [36, 42, 59, 13], such that one can apply such uncertainty measurement to decide if we can directly accept the model prediction, or we need to allocate the instance to humans for further evaluation.

4 Conclusion

In this paper, we propose Classification And Regression Diffusion (CARD) models, a class of conditional generative models that approaches supervised learning problems from a conditional generation perspective. Without training with objectives directly related to the evaluation metrics, we achieve state-of-the-art results on benchmark regression tasks. Furthermore, CARD exhibits a strong ability to represent the conditional distribution with multiple density modes. We also propose a new metric Quantile Interval Coverage Error (QICE), which can be viewed as a generalized version of negative log-likelihood in evaluating how well the model fits the data. Lastly, we introduce a framework to evaluate prediction uncertainty at the instance level for classification tasks.

References

[1] Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. Training normalizing flows with the information bottleneck for competitive generative classification. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[2] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2021. URL `https://arxiv.org/pdf/2107.03006`.

[3] Christopher Bishop. Mixture density networks. In *Aston University Neural Computing Research Group Report*, 1994.

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

[5] Glenn W. Brier. Verification of forecasts expressed in terms of probability. In *Monthly weather review*, 1950.

[6] Leda Cosmides and John Tooby. Are humans good intuitive statisticians after all? Rethinking some conclusions from the literature on judgment under uncertainty. In *cognition*, volume 58(1), pages 1–73, 1996.

[7] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=AAWuCvzaVt`.

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

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

[10] Ethan Fetaya, Joern-Henrik Jacobsen, Will Grathwohl, and Richard Zemel. Understanding the limitations of conditional generative models. In *International Conference on Learning Representations (ICLR 2020)*, 2020.

[11] 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*, 2016.

[12] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems 30*, 2017.

[13] Ruijiang Gao, Maytal Saar-Tsechansky, Maria De-Arteaga, Ligong Han, Min Kyung Lee, and Matthew Lease. Human-AI collaboration with bandit feedback. In *International Joint Conferences on Artificial Intelligence*, 2021.

[14] Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[15] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S.M. Ali Eslami, and Yee Whye Teh. Neural processes. In *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[16] Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations (ICLR 2020)*, 2020.

[17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[19] 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 International Conference on Machine Learning*, 2015.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

[21] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[22] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[23] Per Joachims. nnuncert: Uncertainty quantification with BNNs, 2021. URL `https://github.com/nnuncert/nnuncert`.

[24] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *arXiv preprint arXiv:2201.11793*, 2022.

[25] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *31st Conference on Neural Information Processing System*, 2017.

[26] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations (ICLR 2019)*, 2019.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[28] Durk P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28*, 2015.

[29] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being a bit frequentist improves Bayesian neural networks. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, 2022.

[30] 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.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[32] Shiao Liu, Xingyu Zhou, Yuling Jiao, and Jian Huang. Wasserstein generative learning of conditional distribution, 2021. URL `https://arxiv.org/abs/2112.10039`.

[33] Shiwei Liu, Tianlong Chen, Zahra Atashgahi, Xiaohan Chen, Ghada Sokar, Elena Mocanu, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Deep ensembling with no overhead for either training or testing: The all-round blessings of dynamic sparsity. In *Proceedings of the International Conference on Learning Representations (ICLR 2022)*, 2022.

[34] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[35] Radek Mackowiak, Lynton Ardizzone, Ullrich Köthe, and Carsten Rother. Generative classifiers as a basis for trustworthy image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

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

[37] Jishnu Mukhoti and Yarin Gal. Evaluating Bayesian deep learning methods for semantic segmentation. In *arXiv preprint arXiv: 1811.12709*, 2018.

[38] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *AAAI*, page 2901, 2015.

[39] Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, page 113, 2011.

[40] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[41] Kushagra Pandey, Avideep Mukherjee, Piyush Rai, and Abhishek Kumar. DiffuseVAE: Efficient, controllable and high-fidelity generation from low-dimensional latents. *arXiv preprint arXiv:2201.00308*, 2022.

[42] Maithra Raghu, Katy Blumer, Greg Corrado, Jon Kleinberg, Ziad Obermeyer, and Sendhil Mullainathan. The algorithmic automation problem: Prediction, triage, and human effort. In *arXiv preprint arXiv:1903.12220*, 2019.

[43] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

[44] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[45] Hongyu Ren, Shengjia Zhao, and Stefano Ermon. Adaptive antithetic sampling for variance reduction. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[46] Michael Revow, Christopher KI Williams, and Geoffrey E Hinton. Using generative models for handwritten digit recognition. *IEEE transactions on pattern analysis and machine intelligence*, 18(6):592–606, 1996.

[47] Tim G. J. Rudner, Zonghao Chen, Yee Whye Teh, and Yarin Gal. Tractable function-space variational inference in Bayesian neural networks. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

[48] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[49] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2020.

[50] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11918–11930, 2019.

[51] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020.

[52] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *arXiv e-prints*, pages arXiv–2101, 2021.

[53] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

[54] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting, 2014.

[55] Marcin B. Tomczak, Siddharth Swaroop, Andrew Y. K. Foong, and Richard E. Turner. Collapsed variational bounds for Bayesian neural networks. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, 2021.

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

[57] Zhendong Wang and Mingyuan Zhou. Thompson sampling via local uncertainty. In *International Conference on Machine Learning*, pages 10115–10125. PMLR, 2020.

[58] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[59] Bryan Wilder, Eric Horvitz, and Ece Kamar. Learning to complement humans. In *International Joint Conferences on Artificial Intelligence*, 2020.

[60] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. *arXiv preprint arXiv:2112.07804*, 2021.

[61] Jiayu Yao, Weiwei Pan, Soumya Ghosh, and Finale Doshi-Velez. Quality of uncertainty quantification for Bayesian neural network inference. In *ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning*, 2019.

[62] Huangjie Zheng, Xu Chen, Jiangchao Yao, Hongxia Yang, Chunyuan Li, Ya Zhang, Hao Zhang, Ivor Tsang, Jingren Zhou, and Mingyuan Zhou. Contrastive attraction and contrastive repulsion for representation learning. *arXiv preprint arXiv:2105.03746*, 2021.

[63] Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Truncated diffusion probabilistic models. *arXiv preprint arXiv:2202.09671*, 2022.

[64] Xingyu Zhou, Yuling Jiao, Jin Liu, and Jian Huang. A deep generative approach to conditional sampling. *Journal of the American Statistical Association*, pages 1–28, 2021.

[65] Roland S. Zimmermann, Lukas Schott, Yang Song, Benjamin A. Dunn, and David A. Klindt. Score-based generative classifiers. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix B
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix B
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplement
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Supplement
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Theoretical Derivations and Additional Experiment Results

A.1 Theoretical Derivations

Derivation for Forward Process Posteriors: In this section, we derive the mean and variance of the forward process posteriors $q(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x})$ in Eq. 9, which are tractable when conditioned on $\boldsymbol{y}_0$:

$$q(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_t, \boldsymbol{y}_0, \boldsymbol{x}) \propto \underbrace{q\big(\boldsymbol{y}_t \mid \boldsymbol{y}_{t-1}, f_\phi(\boldsymbol{x})\big)}_{\text{Eq. (7)}} \underbrace{q\big(\boldsymbol{y}_{t-1} \mid \boldsymbol{y}_0, f_\phi(\boldsymbol{x})\big)}_{\text{Eq. (8)}} \tag{13}$$

$$\propto \exp\Bigg(-\frac{1}{2}\Bigg(\frac{\big(\boldsymbol{y}_t - (1-\sqrt{\alpha_t})f_\phi(\boldsymbol{x}) - \sqrt{\alpha_t}\boldsymbol{y}_{t-1}\big)^2}{\beta_t} + \frac{\big(\boldsymbol{y}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{y}_0 - (1-\sqrt{\bar{\alpha}_{t-1}})f_\phi(\boldsymbol{x})\big)^2}{1-\bar{\alpha}_{t-1}} \Bigg)\Bigg) \tag{14}$$

$$\propto \exp\Bigg(-\frac{1}{2}\Bigg(\frac{\alpha_t \boldsymbol{y}_{t-1}^2 - 2\sqrt{\alpha_t}\big(\boldsymbol{y}_t - (1-\sqrt{\alpha_t})f_\phi(\boldsymbol{x})\big)\boldsymbol{y}_{t-1}}{\beta_t} + \frac{\boldsymbol{y}_{t-1}^2 - 2\big(\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{y}_0 + (1-\sqrt{\bar{\alpha}_{t-1}})f_\phi(\boldsymbol{x})\big)\boldsymbol{y}_{t-1}}{1-\bar{\alpha}_{t-1}} \Bigg)\Bigg) \tag{15}$$

$$= \exp\Bigg(-\frac{1}{2}\Bigg(\Big(\underbrace{\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}}_{1} \Big)\boldsymbol{y}_{t-1}^2 - 2\Big(\underbrace{\frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\boldsymbol{y}_0 + \frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{y}_t + \big(\frac{\sqrt{\alpha_t}(\sqrt{\alpha_t}-1)}{\beta_t} + \frac{1-\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\big)f_\phi(\boldsymbol{x})}_{2} \Big)\boldsymbol{y}_{t-1} \Bigg)\Bigg), \tag{16}$$

where

$$1 = \frac{\alpha_t(1-\bar{\alpha}_{t-1}) + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})} = \frac{1-\bar{\alpha}_t}{\beta_t(1-\bar{\alpha}_{t-1})}, \tag{17}$$

and we have the posterior variance

$$\tilde{\beta}_t = \frac{1}{1} = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t. \tag{18}$$

Meanwhile, the following coefficients of the terms in the posterior mean through dividing each coefficient in 2 by 1:

$$\gamma_0 = \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} / 1 = \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\beta_t, \tag{19}$$

$$\gamma_1 = \frac{\sqrt{\alpha_t}}{\beta_t} / 1 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\sqrt{\alpha_t}, \tag{20}$$

and

$$\begin{aligned} \gamma_2 &= \Bigg(\frac{\sqrt{\alpha_t}(\sqrt{\alpha_t}-1)}{\beta_t} + \frac{1-\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} \Bigg) / 1 \\ &= \frac{\alpha_t - \bar{\alpha}_t - \sqrt{\alpha_t}(1-\bar{\alpha}_{t-1}) + \beta_t - \beta_t\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t} \\ &= 1 + \frac{(\sqrt{\bar{\alpha}_t}-1)(\sqrt{\alpha_t}+\sqrt{\bar{\alpha}_{t-1}})}{1-\bar{\alpha}_t}, \end{aligned} \tag{21}$$

which together give us the posterior mean

$$\tilde{\boldsymbol{\mu}}\big(\boldsymbol{y}_t, \boldsymbol{y}_0, f_\phi(\boldsymbol{x})\big) = \gamma_0\boldsymbol{y}_0 + \gamma_1\boldsymbol{y}_t + \gamma_2 f_\phi(\boldsymbol{x}).$$

Derivation for Forward Process Sampling Distribution with Arbitrary Timesteps: For completeness, we include the derivation for the parameters of the forward diffusion process sampling distribution with arbitrary t steps.

The expectation term is based on Eqs. (36)–(38) in Pandey et al. [41]. From Eq. (7), we have that for all $t = 1, \ldots, T$,

$$\boldsymbol{y}_t = \sqrt{1-\beta_t}\boldsymbol{y}_{t-1} + (1-\sqrt{1-\beta_t})f_\phi(\boldsymbol{x}) + \sqrt{\beta_t}\boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{22}$$

Taking expectation of both sides, we have

$$\begin{aligned}
\mathbb{E}(\boldsymbol{y}_t) &= \sqrt{1-\beta_t}\mathbb{E}(\boldsymbol{y}_{t-1}) + (1-\sqrt{1-\beta_t})f_\phi(\boldsymbol{x}) && (23)\\
&= \sqrt{1-\beta_t}\big(\sqrt{1-\beta_{t-1}}\mathbb{E}(\boldsymbol{y}_{t-2}) + (1-\sqrt{1-\beta_{t-1}})f_\phi(\boldsymbol{x})\big) + (1-\sqrt{1-\beta_t})f_\phi(\boldsymbol{x})\\
&= \sqrt{(1-\beta_t)(1-\beta_{t-1})}\mathbb{E}(\boldsymbol{y}_{t-2}) + \big(1-\sqrt{(1-\beta_t)(1-\beta_{t-1})}\big)f_\phi(\boldsymbol{x}) && (24)\\
&\vdots\\
&= \sqrt{\prod_{i=2}^{t}(1-\beta_i)}\mathbb{E}(\boldsymbol{y}_1) + \left(1-\sqrt{\prod_{i=2}^{t}(1-\beta_i)}\right)f_\phi(\boldsymbol{x}) && (25)\\
&= \sqrt{\prod_{i=2}^{t}(1-\beta_i)}\big(\sqrt{1-\beta_1}\boldsymbol{y}_0 + (1-\sqrt{1-\beta_1})f_\phi(\boldsymbol{x})\big) + \left(1-\sqrt{\prod_{i=2}^{t}(1-\beta_i)}\right)f_\phi(\boldsymbol{x}) && (26)\\
&= \sqrt{\prod_{i=1}^{t}(1-\beta_i)}\boldsymbol{y}_0 + \left(1-\sqrt{\prod_{i=1}^{t}(1-\beta_i)}\right)f_\phi(\boldsymbol{x}) && (27)\\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{y}_0 + (1-\sqrt{\bar{\alpha}_t})f_\phi(\boldsymbol{x}). && (28)
\end{aligned}$$

Meanwhile, since the addition of two independent Gaussians with different variances, $\mathcal{N}(\mathbf{0}, \sigma_a^2\boldsymbol{I})$ and $\mathcal{N}(\mathbf{0}, \sigma_b^2\boldsymbol{I})$, is $\mathcal{N}(\mathbf{0}, (\sigma_a^2+\sigma_b^2)\boldsymbol{I})$, we can derive the variance term accordingly,

$$\sigma^2(\boldsymbol{y}_t) = (1-\bar{\alpha}_t)\boldsymbol{I}. \tag{29}$$

A.2 Related Work

[rev.: Under the supervised learning settings, to model the conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ besides just the conditional mean $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}]$ through deep neural networks, existing works have been focusing on quantifying predictive uncertainty, and several lines of work have been proposed. Bayesian neural networks (BNNs) model such uncertainty by assuming distributions over network parameters, capturing the plausibility of the model given the data [4, 19, 11, 28, 55]. Kendall and Gal [25] also model the uncertainties in the model outputs besides model parameters, by including the additive noise term as part of the neural network output. Meanwhile, ensemble-based methods [30, 33] have been proposed to model predictive uncertainty by combining multiple neural networks with stochastic outputs. Furthermore, the neural processes family [15, 14, 26, 16] has introduced a series of models that capture predictive uncertainty in an out-of-distribution fashion, particularly designed for few-shot learning settings.

Above mentioned models have all assumed a parametric form in $p(\boldsymbol{y} \mid \boldsymbol{x})$, namely Gaussian distribution, or a mixture of Gaussians, and optimize the network parameters based on a Gaussian negative log-likelihood objective function. Deep generative models on the other hand, have been known for modeling implicit distributions without parametric distributional assumptions, but very few works have been proposed to utilize such feature to tackle regression tasks. GAN-based models are introduced in [64, 32] for conditional density estimation and predictive uncertainty quantification. For classification tasks on the other hand, generative classifiers [46, 10, 1, 35] is a class of models that also perform classification with generative models; among them, Zimmermann et al. [65] propose score-based generative classifiers to tackle classification tasks with score-based generative models [52, 53]. They model $p(\boldsymbol{x} \mid \boldsymbol{y})$ and predict the label with the largest conditional likelihood in $\boldsymbol{x}$, while CARD models $p(\boldsymbol{y} \mid \boldsymbol{x})$ instead.

In recent years, the class of diffusion-based (or score-based) deep generative models has demonstrated its outstanding performance in modeling high-dimensional multi-modal distributions [20, 49, 24, 60, 7, 50, 51], with most work focusing on Gaussian diffusion processes operating in continuous state spaces. Hoogeboom et al. [21] introduce extensions of diffusion models for categorical data, and Austin et al. [2] have proposed diffusion models for discrete data as a generalization of the multinomial diffusion models, which could provide an alternative way of performing classification with diffusion-based models.]

A.3 Classification with Prediction Uncertainty Evaluation on FashionMNIST Dataset

[rev.: We perform classification on FashionMNIST dataset with CARD, and present the results in a similar fashion as Sec. 3.2.2. We first contextualize the performance of CARD through the mean accuracy of other BNNs with the LeNet CNN [31] architecture in Table 6, where the metrics were also reported in Tomczak et al. [55]. For this dataset, our pre-trained classifier has the same LeNet architecture as the baselines, which achieves a test accuracy of 91.12%. CARD improves the mean test accuracy to 91.79%.

Table 6: Comparison of mean accuracy (in $\%$) on FashionMNIST dataset with other BNNs.

Model	CMV-MF-VI	CM-MF-VI	CV-MF-VI	CM-MF-VI OPT	MF-VI	MAP	MC Dropout	MF-VI EB	f_ϕ (LeNet)	CARD
Accuracy	91.10 ± 0.22	90.95 ± 0.31	88.53 ± 0.13	90.67 ± 0.07	87.04 ± 0.28	88.06 ± 0.22	87.99 ± 0.17	87.04 ± 0.08	91.12	$\mathbf{91.79 \pm 0.09}$

We then present Table 7, from which we can draw the same conclusions as Table 5 on the CIFAR-10 dataset. We set $\alpha = 0.01$ for the paired two-sample t-test. When making the prediction for each instance merely by the class with the narrowest PIW, we obtain a test accuracy of 89.36%.]

Table 7: PIW (multiplied by 100) and t-test results for FashionMNIST classification task.

Class	Accuracy	PIW		Accuracy by t-test Status	
		Correct	Incorrect	Rejected	Not-Rejected (Count)
All	91.79%	0.67	3.20	92.07%	55.84% (77)
1	88.10%	0.96	3.40	88.45%	61.54% (13)
2	98.50%	0.39	2.08	98.60%	66.67% (3)
3	87.70%	0.84	3.42	88.00%	50.00% (8)
4	91.10%	0.76	2.97	91.59%	53.85% (13)
5	87.90%	0.89	2.91	88.40%	33.33% (9)
6	97.20%	0.41	2.89	97.29%	66.67% (3)
7	74.80%	1.37	3.26	74.90%	70.00% (20)
8	97.40%	0.49	1.60	97.50%	50.00% (2)
9	98.40%	0.34	1.93	98.50%	0.00% (1)
10	96.80%	0.46	5.59	97.09%	40.00% (5)

A.4 Test for Normality Assumption of Paired Two-Sample t-test

To access the normality assumption of the paired two-sample t-test, we inspect the Q-Q plots of the differences in probability between the most and the second most predicted classes within each test instance. We include 16 instances in Figure 2 from CARD predictions on FashionMNIST dataset, each with 100 samples. We observe that in all plots, the points align closely with the 45-degree line, indicating that the normality assumption is valid.

A.5 Patch Accuracy vs Patch Uncertainty (PAvPU)

Besides the methods of accessing model confidence at instance level introduced in the paper, we also consider an additional uncertainty evaluation metric in this section, *Patch Accuracy vs Patch Uncertainty* (PAvPU) [37], which measures the proportion of predictions that are either correct while the model is confident about them, or incorrect while the model is ambiguous. Based on the t-test results, we can easily compute PAvPU, which is defined as

$$\text{PAvPU} := \frac{n_{ac} + n_{iu}}{n_{ac} + n_{au} + n_{ic} + n_{iu}}, \tag{30}$$

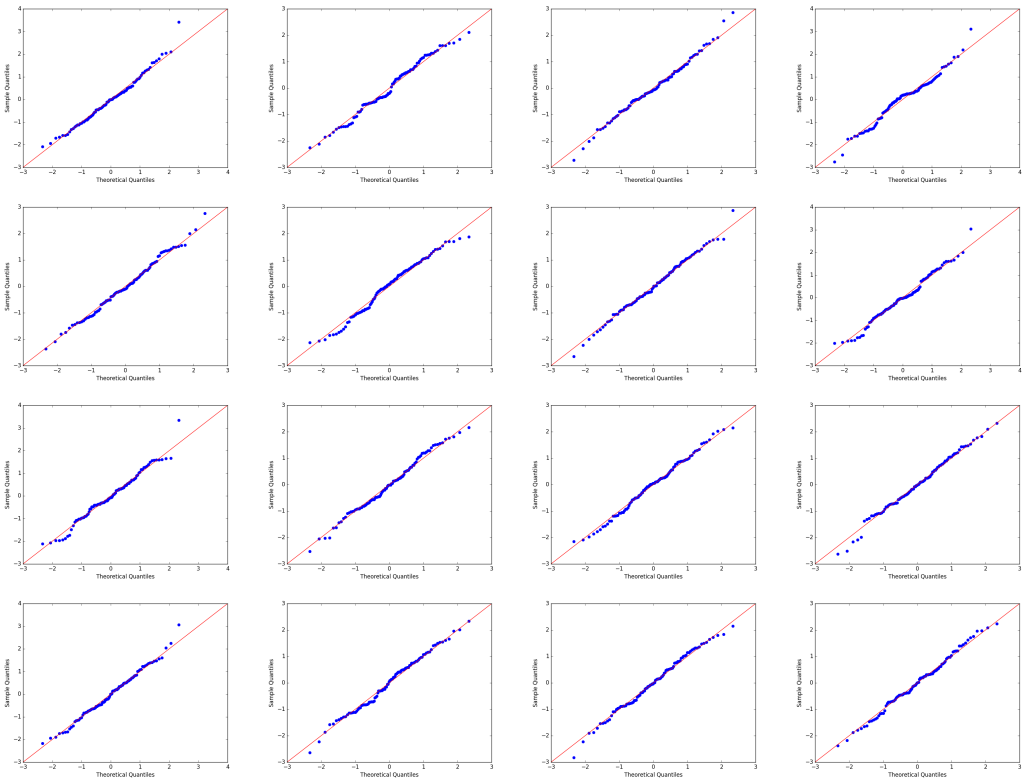

Figure 2: Q-Q plots for the differences in probability between the most and 2nd most predicted class.

where $n_{ac}, n_{au}, n_{ic}, n_{iu}$ represent the number of accurate (correct) predictions when the model is certain (confident) about them, accurate but uncertain, inaccurate but certain, as well as inaccurate and uncertain. We apply the t-test results as the proxy for the model's confidence level on each of its predictions. A higher value indicates that the model tends to be correct when being confident, and to make mistakes when being vague — a characteristic that we want our model to possess. On a related note, accuracy can be computed by replacing n_{iu} with n_{au} in Eq. 30. In the following section, we provide additional experimental results including this metric.

A.6 Classification on Noisy MNIST dataset

To further demonstrate the effectiveness of the CARD model for classification, especially in expressing model confidence, we run additional experiments on Noisy MNIST dataset (adding a Gaussian noise with mean 0 and variance 1 to each pixel). Besides reporting PIW and t-test results similar to Table 5, we also compute PAvPU, and compare the results with baseline models appearing in Fan et al. [9], *i.e.*, MC Dropout [11], Gaussian Dropout [54], Concrete Dropout [12], Bayes by Backprop [4], and variants of Contextual Dropout [9].

Following the same experimental settings in Fan et al. [9], we apply a batch size of 128 for model training, and adopt the same MLP architecture with two hidden layers of 300 and 100 hidden units, respectively, each followed by a ReLU non-linearity, as the network architecture for the pre-trained classifier $f_\phi(\boldsymbol{x})$. We simplify the diffusion network architecture (detailed in Section A.7) by reducing feature dimension from $2\,048$ to 128 to match the model parameter scale for a fair comparison. We pre-train $f_\phi(\boldsymbol{x})$ for 100 epochs, and obtain a test set accuracy of 85.50%. We train the diffusion model for $1\,000$ epochs. Unlike on CIFAR-10 or FashionMNIST dataset, we do not apply data normalization on the noisy MNIST dataset, following the settings in Fan et al. [9].

Similar to Table 5, we report the mean PIW among both correct and incorrect predictions, and the mean accuracy among both groups rejected and not-rejected by the paired two-sample t-test, in Table

8. Same as Fan et al. [9], we set $\alpha = 0.05$. We report these metrics for all test instances and for each class label along with their group accuracy.

Table 8: PIW (multiplied by 100) and t-test ($\alpha = 0.05$) results for Noisy MNIST classification task.

Class	Accuracy	PIW		Accuracy by t-test Status	
		Correct	Incorrect	Rejected	Not-Rejected (Count)
All	88.26%	39.76	76.25	91.24%	43.04% (618)
1	94.69%	15.93	68.67	96.63%	35.48% (31)
2	96.48%	20.48	74.73	97.66%	46.15% (26)
3	88.47%	29.92	74.98	91.25%	44.26% (61)
4	87.23%	41.38	78.58	90.85%	45.68% (81)
5	85.34%	48.87	78.61	87.65%	53.73% (67)
6	85.76%	51.35	76.43	89.85%	40.54% (74)
7	92.28%	22.92	74.04	94.49%	28.13% (32)
8	86.97%	39.80	74.42	90.17%	37.10% (62)
9	80.90%	62.48	77.73	84.25%	45.88% (85)
10	83.25%	76.92	76.67	87.69%	42.42% (99)

We are able to draw similar conclusions from Table 8 as in Table 5: across the entire test set, mean PIW of the true class label among the correct predictions is much narrower than that of the incorrect predictions, implying that CARD is confident in its correct predictions, and more likely to make mistakes when being vague. When comparing mean PIWs at class level, we observe that a more accurate class is inclined to have a larger difference between correct and incorrect predictions. Meanwhile, we have almost 10 times of not-rejected cases as those for CIFAR-10 task shown in Table 5. Similarly, the accuracy of test instances rejected by the t-test is much higher than that of the not-rejected ones, both across the entire test set and at the class level. This result could have a more significant practical impact when applying human-machine collaboration: we are able to identify more than 6% of the data with a mean accuracy of less than 50%. If we pass these cases to human agents, we would be able to remarkably improve the classification accuracy, while still enjoying the automation by the machine classifier in the vast majority.

Furthermore, we contextualize the performance of CARD by reporting accuracy, PAvPU and NLL, along with those from the baseline models mentioned at the beginning of Section A.6, in Table 9. The metrics of the other models are from Table 1 in Fan et al. [9].

Table 9: PAvPU (in $\%$) along with accuracy (in $\%$) and NLL for Noisy MNIST classification task.

Method	Accuracy	PAvPU (0.05)	NLL
MC - Bernoulli	86.36	85.63	1.72
MC - Gaussian	86.31	85.64	1.72
Concrete	86.52	86.77	1.68
Bayes by Backprop	86.55	87.13	2.30
Contextual Gating	86.20	-	1.81
Contextual Gating + Dropout	86.70	87.01	1.71
Bernoulli Contextual	87.43	87.81	1.41
Gaussian Contextual	87.35	87.72	1.43
CARD (ours)	**88.26**	**89.12**	**0.39**

We observe that CARD obtains an accuracy of 88.26% (improves from 85.50% by the pre-trained classifier f_ϕ), and a PAvPU of 89.12%, both are the best among all models. This implies that our model is making not only accurate classifications, but also high-quality predictions in terms of model confidence. Lastly, we apply temperature scaling in Guo et al. [17] to calibrate the predicted probability for the computation of NLL, where the temperature parameter is tuned with the training set, and again obtain the best metric among all models.

A.7 General Experiment Setup Details

In this section, we provide the experimental setup for the CARD model in both regression and classification tasks.

Training: As mentioned at the beginning of Section 3, we set the number of timesteps to $1\,000$, and adopt a linear β_t schedule same as Ho et al. [20]. We set the learning rate to 0.001 for all tasks. We use the AMSGrad [44] variant of the Adam optimizer [27] for all regression tasks. We use the Adam optimizer for the classification tasks on all presented datasets, and adopt cosine learning rate decay [34]. We use EMA on model parameters with a decay factor of 0.9999. We adopt antithetic sampling [45] to draw correlated timesteps during training. We set the number of epochs at $5\,000$ by default for regression tasks, and $1\,000$ by default for classification tasks, to sufficiently cover the timesteps with $T = 1000$. The batch size for each UCI regression task are reported in Table 12. We also apply a batch size of 256 for all toy regression tasks and classification tasks. For all UCI regression tasks, we follow the convention in Hernández-Lobato and Adams [19] and standardize both the input features and the response variable to have zero mean and unit variance, and remove the standardization to compute the metrics. For all toy regression tasks, we do not standardize the input feature; we only standardize the response variable on the log-log cubic regression task. For classification on CIFAR-10 and FashionMNIST, we normalize the dataset with training set mean and standard deviation.

Network Architecture: For the diffusion model, we adopt a simpler network architecture than [60, 63], by first changing the Transformer sinusoidal position embedding to linear embedding for the timestep. As the network $\boldsymbol{\epsilon}_\theta\big(\boldsymbol{x}, \boldsymbol{y}_t, f_\phi(\boldsymbol{x}), t\big)$ has 3 other inputs besides the timestep t, we integrate them in different ways for regression and classification tasks:

- For regression, we first concatenate $\boldsymbol{x}$, $\boldsymbol{y}_t$, and $f_\phi(\boldsymbol{x})$, then send the resulting vector through 3 fully-connected layers, all with an output dimension of 128. We perform Hadamard product between each of the output vector with the corresponding timestep embedding, followed by a Softplus non-linearity, before sending the resulting vector to the next fully-connected layer. Lastly, we apply a 4-th fully-connected layer to map the vector to one with a dimension of 1, as the output forward diffusion noise prediction. We summarize the architecture in Table 10 (a).
- For classification on CIFAR-10, we first apply an encoder on the flattened input image (originally $32 \times 32 \times 3$) to obtain a representation with 4096 dimensions. The encoder consists of 3 fully-connected layers with an output dimension of 4096. Meanwhile, we concatenate $\boldsymbol{y}_t$ and $f_\phi(\boldsymbol{x})$, and apply a fully-connected layer to obtain an output vector of 4096 dimensions. We perform a Hadamard product between such vector and a timestep embedding to obtain a response embedding conditioned on the timestep. We then perform Hadamard product between image embedding and response embedding to integrate these variables, and send the resulting vector through 2 more fully-connected layers with 4096 output dimensions, each would first followed by a Hadamard product with a timestep embedding, and lastly a fully-connected layer with an output dimension of 1 as the noise prediction. Note that all fully-connected layers are also followed by a batch normalization layer and a Softplus non-linearity, except the output layer. We summarize the architecture in Table 10 (b).

For the pre-trained model $f_\phi(\boldsymbol{x})$, we adjust the functional form and training scheme based on the task. For regression, we adopt a feed-forward neural network with two hidden layers, each with 100 and 50 hidden units, respectively. We apply a Leaky ReLU non-linearity with a 0.01 negative slope after each hidden layer. In practice, we find that when the dataset size is small (most of the UCI tasks have less than $10\,000$ data points, and several around or less than $1\,000$), a deep neural network $f_\phi(\boldsymbol{x})$ is prone to overfit. We thus set the default number of epochs to $1\,000$, and adopt early stopping with a patience of 50 epochs, *i.e.*, we terminate the training if there is no improvement in validation set MSE for 50 consecutive epochs. We split the original training set with $60\%/40\%$ ratio and apply early stopping to find the optimal number of epochs, then train $f_\phi(\boldsymbol{x})$ on the full training set. For classification, we apply a pre-trained ResNet-18 network for CIFAR-10 dataset. We only train the model with 10 epochs, as more would lead to overfit. We apply the Adam optimizer for $f_\phi(\boldsymbol{x})$ in all tasks, and we only pre-train the model and freeze it during the training of the diffusion model, instead of fine-tuning it.

A.8 UCI Baseline Model Experiment Setup Details and Dataset Information

In this section, we first provide the experimental setup for the UCI regression baseline models (PBP, MC Dropout, Deep Ensembles, and GCDS), including learning rate, batch size, network architecture,

Table 10: CARD $\boldsymbol{\epsilon}_\theta$ network architecture. We denote concatenation as $\oplus$, Hadamard product as $\odot$, and Softplus non-linearity as σ. We also denote a fully-connected layer as g, and a hidden layer output as l, with subscripts to differentiate them.

(a) Regression network architecture.

input: $\boldsymbol{x}, \boldsymbol{y}_t, f_\phi(\boldsymbol{x}), t$
$l_1 = \sigma\Big(g_{1,a}\big(\boldsymbol{x} \oplus \boldsymbol{y}_t \oplus f_\phi(\boldsymbol{x})\big) \odot g_{1,b}(t)\Big)$
$l_2 = \sigma\big(g_{2,a}(l_1) \odot g_{2,b}(t)\big)$
$l_3 = \sigma\big(g_{3,a}(l_2) \odot g_{3,b}(t)\big)$
output: $g_4(l_3)$

(b) Classification network architecture.

input: $\boldsymbol{x}, \boldsymbol{y}_t, f_\phi(\boldsymbol{x}), t$
$l_{1,x} = \sigma\Big(\mathrm{BN}\big(g_{1,x}(\boldsymbol{x})\big)\Big)$
$l_{2,x} = \sigma\Big(\mathrm{BN}\big(g_{2,x}(\boldsymbol{x})\big)\Big)$
$l_{3,x} = \mathrm{BN}\big(g_{1,x}(\boldsymbol{x})\big)$
$l_{1,y} = \sigma\Big(\mathrm{BN}\Big(g_{1,y}\big(\boldsymbol{y}_t \oplus f_\phi(\boldsymbol{x})\big) \odot g_{1,b}(t)\Big)\Big)$
$l_1 = l_{3,x} \odot l_{1,y}$
$l_2 = \sigma\Big(\mathrm{BN}\big(g_{2,a}(l_1) \odot g_{2,b}(t)\big)\Big)$
$l_3 = \sigma\Big(\mathrm{BN}\big(g_{3,a}(l_2) \odot g_{3,b}(t)\big)\Big)$
output: $g_4(l_3)$

number of epochs, etc. We applied the GitHub repo [23] to run BNN models, and implemented our own version of GCDS since the code has not been published.

Overall, we apply the same learning rate of 0.001 for all models on all datasets, except for PBP on the Boston dataset, which is 0.1. We also apply the Adam optimizer for all experiments. We follow the convention in Hernández-Lobato and Adams [19] and standardize both the input features and response variable for training, and remove the standardization for evaluation.

For batch size, we adjust on a case-by-case basis, taking the running time and dataset size into consideration. We provide the batch size in Table 12. For network architecture, we use ReLU non-linearities for all 3 BNNs, and Leaky ReLU with a 0.01 negative slope for GCDS; we choose the number of hidden layers with the number of hidden units per layer from the following 3 options: a) 1 hidden layer with 50 hidden units; b) 1 hidden layer with 100 hidden units; c) 2 hidden layers with 100 and 50 hidden units, respectively. We show the choice of hidden layer architecture in Table 13. For the number of training epochs, we also vary on a case-by-case basis: PBP and Deep Ensembles both applied 40 epochs, but we observed in many experiments that the model has not converged. We show the number of epochs in Table 14. Lastly, dataset Boston, Energy and Naval all contain one or more categorical variables, thus we ran the experiments both with and without conducting one-hot encoding on the data. We found that except the Naval dataset with PBP, all other cases had worse metrics when one-hot encoding was applied.

We summarize the dataset information in terms of their size and number of features in Table 11.

Table 11: Dataset size (N observations, P features) of UCI regression tasks.

Dataset	Boston	Concrete	Energy	Kin8nm	Naval	Power	Protein	Wine	Yacht	Year
(N, P)	$(506, 13)$	$(1030, 8)$	$(768, 8)$	$(8192, 8)$	$(11934, 16)$	$(9568, 4)$	$(45730, 9)$	$(1599, 11)$	$(308, 6)$	$(515345, 90)$

We reiterate that we re-ran the experiments with the baseline BNN models to compute the new metric QICE along with the conventional metrics RMSE and NLL. We carefully tuned the hyperparameters to obtain results better than or comparable with the ones reported in the original papers.

A.9 UCI Regression Tasks PICP across All Methods

[rev.: In this section, we report PICP for all methods in Table 15 from the same runs with the corresponding metrics in Tables 1, 2, and 3.]

Table 12: Batch size settings of UCI regression tasks across different models.

	PBP	MC Dropout	Deep Ensembles	GCDS	CARD (ours)
Boston	32	32	32	32	32
Concrete	32	32	32	32	32
Energy	32	32	32	32	32
Kin8nm	64	32	64	64	64
Naval	64	32	64	64	64
Power	64	64	64	64	64
Protein	100	256	100	256	256
Wine	32	32	32	32	32
Yacht	32	32	32	32	32
Year	256	256	100	256	256

Table 13: Network hidden layer architecture options of UCI regression tasks across different models.

	PBP	MC Dropout	Deep Ensembles	GCDS
Boston	a	c	a	c
Concrete	a	c	c	c
Energy	a	b	a	c
Kin8nm	a	c	a	a
Naval	a	c	a	a
Power	a	b	a	a
Protein	b	c	b	c
Wine	a	c	a	c
Yacht	a	a	a	a
Year	b	c	b	b

A.10 Ablation Study on Choice of Prior – UCI Boston Dataset

[rev.: In this section, we conduct ablation study for two model variants with different prior distribution settings on the UCI Boston dataset, under various settings of number of timesteps T with adjusted β_t linear schedule (to make sure that $\sqrt{\bar{\alpha}_1}$ is close to 1 and $\sqrt{\bar{\alpha}_T}$ is close to 0). We report the evaluation metrics in Table 16, where we compare the original CARD setting with $\mathcal{N}(f_\phi(x), \boldsymbol{I})$ as the prior distribution at timestep T, to the alternative setting with $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ as the prior distribution. We observe that as T decreases, RMSE and NLL do not deteriorate for $\mathcal{N}(f_\phi(x), \boldsymbol{I})$ prior (CARD setting), but those from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ prior become worse. The metrics that measure distributional fitting, QICE and PICP, gets worse $\mathcal{N}(f_\phi(x), \boldsymbol{I})$ prior as well, but such deterioration is not as much as $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ prior. The results indicate that our setting of an informative prior $\mathcal{N}(f_\phi(x), \boldsymbol{I})$ contribute to the regression performance of CARD. Furthermore, the setting of total number of timesteps T do not affect mean estimation for $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ prior, but would noticeably impact the distributional fitting (*i.e.*, the recovery of aleatoric uncertainty).]

A.11 Ablation Study on Diffusion Network Parameterization – CIFAR-10 Dataset

[rev.: In this section, we study the impact of different $\boldsymbol{\epsilon}_\theta$ network parameterization forms on the CIFAR-10 dataset, through model performance in terms of accuracy and PAvPU, as well as training efficiency at the first 100 epochs. We compare four model variants, each with a different prior and $\boldsymbol{\epsilon}_\theta$ network parameterization combination, in Table 17, by reporting accuracy and PAvPU on test set over 10 runs.

We observe that given the same prior distribution setting, both metrics do not differ much by whether or not we include $f_\phi(x)$ as $\boldsymbol{\epsilon}_\theta$ network input. Meanwhile, both model variants (V1, V2) with a prior of $\mathcal{N}(f_\phi(x), \boldsymbol{I})$ outperform the other two variants (V3, V4) of $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ prior, suggesting the application of an informative prior would benefit the performance. Furthermore, the variant (V4) without $f_\phi(x)$ as either the prior mean or $\boldsymbol{\epsilon}_\theta$ input has the worst performance, indicating the inclusion of a pre-trained classifier can improve the model performance in both accuracy and uncertainty estimation.

Furthermore, the choice of $\mathcal{N}(f_\phi(x), \boldsymbol{I})$ prior also helps with training efficiency: we observe in Figure 3 that the model performance improved faster for $\mathcal{N}(f_\phi(x), \boldsymbol{I})$ prior at the beginning of

Table 14: Number of training epochs of UCI regression tasks across different models.

	PBP	MC Dropout	Deep Ensembles	GCDS
Boston	100	1000	100	500
Concrete	100	500	40	500
Energy	100	500	100	500
Kin8nm	100	500	100	500
Naval	100	500	100	500
Power	100	500	100	500
Protein	100	500	100	500
Wine	100	500	100	500
Yacht	100	500	100	500
Year	100	100	100	500

Table 15: PICP (in $\%$) of UCI regression tasks.

Dataset	$\lvert \text{PICP} - 95 \rvert \downarrow$ PBP	MC Dropout	Deep Ensembles	GCDS	CARD (ours)
Boston	91.27 ± 4.82	$\mathbf{96.08 \pm 2.70}$	88.73 ± 5.68	31.37 ± 6.79	93.24 ± 3.59
Concrete	92.28 ± 2.87	$\mathbf{97.52 \pm 2.43}$	90.34 ± 3.64	39.85 ± 4.53	90.24 ± 3.45
Energy	93.18 ± 3.12	99.03 ± 1.08	$\mathbf{96.49 \pm 1.97}$	63.57 ± 10.26	98.70 ± 1.30
Kin8nm[1]	$\mathbf{95.06 \pm 0.77}$	95.37 ± 2.24	96.53 ± 0.67	59.06 ± 5.31	93.68 ± 0.79
Naval[2]	93.52 ± 4.40	100.00 ± 0.00	99.78 ± 0.28	83.71 ± 14.87	$\mathbf{95.35 \pm 0.60}$
Power	95.75 ± 0.69	96.28 ± 0.76	95.91 ± 0.71	89.13 ± 1.14	$\mathbf{94.87 \pm 0.65}$
Protein	$\mathbf{94.79 \pm 0.13}$	96.46 ± 0.77	96.08 ± 0.28	85.24 ± 0.86	95.38 ± 0.16
Wine	92.72 ± 1.80	91.41 ± 2.66	91.06 ± 1.64	86.37 ± 2.33	$\mathbf{93.88 \pm 2.10}$
Yacht	$\mathbf{96.94 \pm 2.60}$	100.00 ± 0.00	98.87 ± 1.54	83.23 ± 4.28	99.84 ± 0.70
Year	$93.04 \pm$ NA	$\mathbf{94.61} \pm$ NA	$95.44 \pm$ NA	$87.08 \pm$ NA	$93.35 \pm$ NA
# best	**3**	**3**	1	0	**3**

training, by measuring the accuracy on test set with 1 sample every 10 epochs for the first 100 epochs during training, and plotting the metric (as the mean across all runs) against the number of epochs. Due to the measurement similarity, we omit V2 and V4 and only plot the metrics from V1 (for $\mathcal{N}(f_\phi(x), \boldsymbol{I})$ prior) and V3 (for $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ prior). We observe that after only 20 epochs, the accuracy of 1 sample by CARD is already close to 90%, while the variant with a $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ prior is only around 75%, suggesting the advantage in training efficiency with an informative prior.]

A.12 Regression Toy Example Details

The 8 toy examples are summarized by Table 18. For each task, we create the dataset by sampling $10,240$ data points from the data generating function, and randomly split them into training and test sets with an $80\%/20\%$ ratio. For all uni-modal cases as well as the full circle task, the $\boldsymbol{x}$ variable is sampled from a uniform distribution. The noise variable $\boldsymbol{\epsilon}$ is sampled from a Gaussian distribution. The dataset of the inverse sinusoidal task is created by simply swapping $\boldsymbol{x}$ and $\boldsymbol{y}$ variable of the sinusoidal task (so that we have multi-modality when the new $\boldsymbol{x}$ is roughly between 0.25 and 0.75), thus the name of the task.

To quantitatively evaluate the performance of CARD, we generate 1000 $\boldsymbol{y}$ samples for each $\boldsymbol{x}$ in the test set, and compute the corresponding metrics. We conduct such procedure over 10 runs, each applying a different random seed to generate the dataset, and report the mean and standard deviation over all runs for each metric. For all tasks regardless of the form of $p(\boldsymbol{y} \mid \boldsymbol{x})$, we compute PICP and QICE. For tasks with uni-modal $p(\boldsymbol{y} \mid \boldsymbol{x})$ distributions, we summarize the 1000 samples for each test $\boldsymbol{x}$ by computing their mean, as an unbiased estimator to $\mathbb{E}(\boldsymbol{y} \mid \boldsymbol{x})$, and compute the root mean squared error (RMSE) between the estimated and true conditional means. For all tasks, we obtain a mean PICP very close to the optimal 95%, and most of the tasks have a mean QICE value far less than 0.01 except log-log cubic regression, which also has a mean RMSE noticeably larger by an order of magnitude among cases with uni-modal conditional distributions. Note that the $\boldsymbol{y}$ samples here have a much wider range: as $\boldsymbol{x}$ increases from 0 to 10, $\boldsymbol{y}$ increases from 0 to over 1200, resulting in a much more difficult task. Therefore, the metrics reported here can be viewed with relativity, and combined with the qualitative conclusions from Figure 1. The metrics of all tasks, including RMSE, QICE, and PICP, are recorded to Table 19. Note that QICE has been converted to percentage scale as we report two significant figures for all metrics.

Table 16: Ablation study on 2 prior distribution settings on UCI Boston dataset with different T.

T	β_t schedule (β_1, β_T)	Prior	RMSE	NLL	QICE	PICP
1000	(0.0001, 0.02)	$\mathcal{N}(f_\phi(x), \boldsymbol{I})$	$\mathbf{2.61 \pm 0.63}$	2.65 ± 0.12	$\mathbf{3.45 \pm 0.83}$	93.24 ± 3.59
		$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$	2.71 ± 0.69	$\mathbf{2.37 \pm 0.12}$	3.53 ± 0.99	$\mathbf{93.53 \pm 3.34}$
500	(0.0001, 0.04)	$\mathcal{N}(f_\phi(x), \boldsymbol{I})$	$\mathbf{2.63 \pm 0.72}$	$\mathbf{2.33 \pm 0.13}$	3.94 ± 1.05	$\mathbf{93.14 \pm 3.19}$
		$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$	2.70 ± 0.68	2.34 ± 0.12	$\mathbf{3.48 \pm 0.76}$	91.76 ± 3.75
100	(0.001, 0.175)	$\mathcal{N}(f_\phi(x), \boldsymbol{I})$	$\mathbf{2.65 \pm 0.67}$	$\mathbf{2.30 \pm 0.18}$	$\mathbf{4.09 \pm 1.13}$	$\mathbf{88.82 \pm 5.15}$
		$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$	2.69 ± 0.66	2.32 ± 0.21	4.19 ± 1.12	85.20 ± 6.34
50	(0.001, 0.35)	$\mathcal{N}(f_\phi(x), \boldsymbol{I})$	$\mathbf{2.61 \pm 0.71}$	$\mathbf{2.31 \pm 0.25}$	$\mathbf{5.06 \pm 1.46}$	$\mathbf{81.96 \pm 6.31}$
		$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$	2.76 ± 0.66	2.57 ± 0.39	5.38 ± 1.55	76.18 ± 7.13
10	(0.01, 0.95)	$\mathcal{N}(f_\phi(x), \boldsymbol{I})$	$\mathbf{2.63 \pm 0.58}$	$\mathbf{2.56 \pm 0.44}$	$\mathbf{5.34 \pm 1.24}$	$\mathbf{77.65 \pm 7.00}$
		$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$	2.80 ± 0.75	2.98 ± 0.85	5.52 ± 1.20	75.39 ± 7.58

Table 17: 4 Ablation study on 4 model variants on CIFAR-10 dataset.

Variant	Prior	$f_\phi(x)$ included as $\boldsymbol{\epsilon}_\theta$ input	Accuracy	PAvPU
V1	$\mathcal{N}(f_\phi(x), \boldsymbol{I})$	True	90.93 ± 0.02	$\mathbf{91.11 \pm 0.04}$
V2	$\mathcal{N}(f_\phi(x), \boldsymbol{I})$	False	$\mathbf{90.94 \pm 0.02}$	91.08 ± 0.03
V3	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$	True	90.88 ± 0.03	91.06 ± 0.03
V4	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$	False	90.82 ± 0.02	91.02 ± 0.03

The results in Table 19 implicitly suggest that our proposed metric QICE is reasonable: when RMSE is low (way below 1) and PICP is close to 95%, implying that CARD is performing well in terms of both mean estimation and distributional matching, QICE is also low (far less than 1%); for the most difficult task, log-log cubic regression, as RMSE is above 5 and PICP deviates relatively most from 95% (but not much), QICE also has the largest value (slightly alove 1%).

A.13 The Evolution of Samples through the Diffusion Process

[rev.: We present the evolution of both q and p distribution samples through the forward and reverse diffusion process, respectively. We first visualize the behaviors of these samples from the training on linear regression task in Figure 4, where we pick timesteps with an interval of 200 steps including the 1st and the last timestep, namely $t = 1, 200, 400, 600, 800, T$. The p samples presented are from near the end of training. We observe that $\boldsymbol{\epsilon}_\theta$ has been trained to match the q samples at different timesteps well, including the variance. Furthermore, note that the true variance from data generating function is set to 4, while the prior $p(\boldsymbol{y}_T \mid \boldsymbol{x})$ has a variance of 1. We can observe the gradual increase of variance in the reverse direction. This example helps to illustrate that when $f_\phi(\boldsymbol{x})$ can already estimate the mean accurately, it makes the task for the diffusion model easier: in this case to solely focus on recovering the aleatoric uncertainty.

Similarly, we present the samples from q and p distribution during training for the full circle regression task in Figure 5. Besides observing the matching in samples at all selected timesteps, we emphasize CARD's ability to model multi-modality at various intensity. As t increases, we observe the samples gradually evolve from a uni-modal distribution into a multi-modal one, and the diffusion model is able to capture such progress. To quantify such match, we plot the quantile coverage ratios for samples at $t = 0$ from one run, along with the optimal coverage ratio (0.1, for 10 bins), in Figure 6. Note that we obtain a QICE of 0.62 in this run. We observe that CARD samples cover the true data with a ratio close to the optimal across all bins; The 5th bin has relatively the most deviation from the optimal ratio (which is understandable as the full circle dataset has a bi-modal distribution across most $\boldsymbol{x}$, with no datapoints in the center portion), which is compensated by the 2nd, 4th and last bin with coverage slightly above the optimal ratio.

We continue with a plot of samples during test time from the UCI Boston dataset. In Figure 7, we plot the generated samples from p (in blue) along with q samples (in red) at various t. The x-axis represents the count of samples, instead of the actual $\boldsymbol{x}$ since the true covariate space is high-dimensional. Note that we generate 1 000 samples given each $\boldsymbol{x}$. While still observing the good mix between q and p samples for $t = 200, \ldots, T$ from 2nd plot to the right, we observe that for $t = 1$, all samples for each $\boldsymbol{x}$ forms a vertical region that covers the corresponding $\boldsymbol{y}_1$ (which shall be

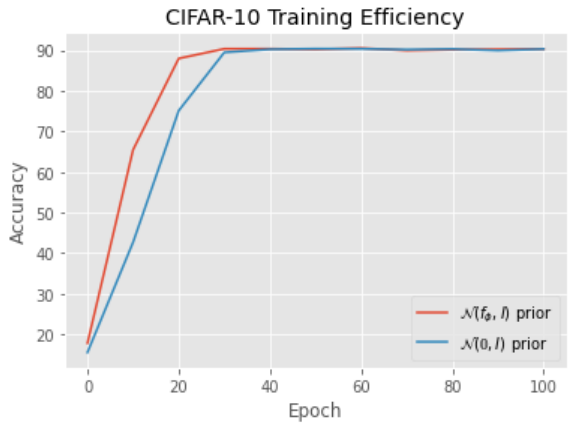

Figure 3: Performance from two prior settings on CIFAR-10 test set with 1 sample.

Table 18: Regression toy examples.

Regression Task	Data Generating Function	$\boldsymbol{x}$	$\boldsymbol{\epsilon}$
Linear	$\boldsymbol{y} = 2\boldsymbol{x} + 3 + \boldsymbol{\epsilon}$	$\mathrm{U}(-5, 5)$	$\mathcal{N}(0, 2^2)$
Quadratic	$\boldsymbol{y} = 3\boldsymbol{x}^2 + 2\boldsymbol{x} + 1 + \boldsymbol{\epsilon}$	$\mathrm{U}(-5, 5)$	$\mathcal{N}(0, 2^2)$
Log-Log Linear	$\boldsymbol{y} = \exp\big(\log(\boldsymbol{x}) + \boldsymbol{\epsilon}\big)$	$\mathrm{U}(0, 10)$	$\mathcal{N}(0, 0.15^2)$
Log-Log Cubic	$\boldsymbol{y} = \exp\big(3\log(\boldsymbol{x}) + \boldsymbol{\epsilon}\big)$	$\mathrm{U}(0, 10)$	$\mathcal{N}(0, 0.15^2)$
Sinusoidal	$\boldsymbol{y} = \boldsymbol{x} + 0.3\sin(2\pi\boldsymbol{x}) + \boldsymbol{\epsilon}$	$\mathrm{U}(0, 1)$	$\mathcal{N}(0, 0.08^2)$
Inverse Sinusoidal	swap $\boldsymbol{x}$ and $\boldsymbol{y}$ from Sinusoidal	—	—
8 Gaussians	8 modes	—	$\mathcal{N}(0, 0.1^2)$
Full Circle	$\boldsymbol{y} = (10 + \boldsymbol{\epsilon})\big(\cos(2\pi\boldsymbol{x}) + \sin(2\pi\boldsymbol{x})\big)$	$\mathrm{U}(0, 1)$	$\mathcal{N}(0, 0.5^2)$

very close to $\boldsymbol{y}_0$ due to the linear β_t schedule) from q distribution at various positions (*i.e.*, middle, upper half, lower half, near top, near bottom). We observe that the samples are representative of the true conditionals $p(\boldsymbol{y} \mid \boldsymbol{x} = x)$ for each x.]

A.14 The Development of Model Performance through the Reverse Diffusion Process

[rev.: Following the demonstration of behavior change in samples with respect to timestep t during both the training and test time, we present the change in model evaluation metrics on the test set as a function of t. We again use the UCI Boston dataset, and compute all evaluation metrics, including RMSE, NLL, QICE and PICP, at all timesteps from $t = T$ to $t = 0$. Since the standard procedure for running the task contains 20 different splits on the dataset, we compute these metrics at all timesteps for all splits, and take the mean across all splits at each timestep t. We combine these plots in Figure 8. For RMSE, we observe that the performance at $t = T$ is already quite good, due to the setting of $f_\phi(\boldsymbol{x})$ as the prior mean; as the reverse process continues, the metric gradually improves by decreasing. Both NLL and QICE behave in a similar fashion: as t decreases, the metric steadily improves. For PICP, note that it crosses the optimal value of 0.95 coverage ratio around the end of the reverse process; however, it did not deviate much from such value. This plot demonstrates the successive improvements across all metrics during the reverse diffusion process, starting from an already decent place (in terms of RMSE and NLL) due to the application of an informative prior.]

A.15 Improving the Granularity of ECE from Subgroup to Instance Level

[rev.: In this section, we first present the definition of ECE. This material is from Guo et al. [17], and we include it here for completeness. We then provide our analysis, specifically about its granularity to measure prediction confidence by the model, which motivates us to introduce an alternative way to measure model prediction confidence at a finer granularity (*i.e.*, at instance level) in our paper.

ECE is defined as:

$$\mathrm{ECE} := \mathbb{E}_{\hat{P}}\big[\big|\mathbb{P}\big(\hat{Y} = Y \big| \hat{P} = p\big) - p\big|\big], \tag{31}$$

Table 19: Regression toy example RMSE, QICE (in %), and PICP (in %).

Regression Task	RMSE ↓	QICE ↓	PICP
Linear	0.07 ± 0.02	0.54 ± 0.14	95.29 ± 0.53
Quadratic	0.21 ± 0.03	0.55 ± 0.12	95.12 ± 0.55
Log-Log Linear	0.07 ± 0.01	0.55 ± 0.15	95.17 ± 0.62
Log-Log Cubic	5.85 ± 1.38	1.31 ± 0.26	96.08 ± 0.62
Sinusoidal	0.01 ± 0.00	0.48 ± 0.11	94.81 ± 0.54
Inverse Sinusoidal	—	0.71 ± 0.18	95.89 ± 0.52
8 Gaussians	—	0.66 ± 0.19	95.92 ± 0.46
Full Circle	—	0.60 ± 0.05	95.52 ± 0.42

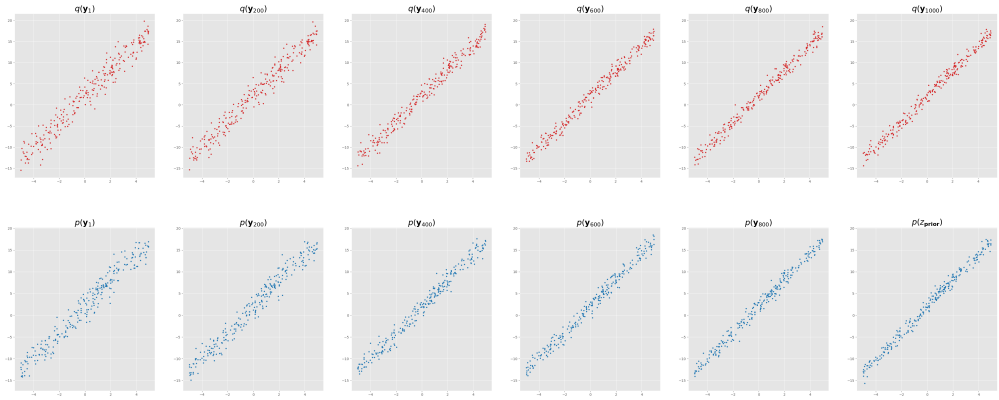

Figure 4: q and p distribution samples for linear regression task during training. (**Top**) left to right: $q\big(\boldsymbol{y}_t \,|\, \boldsymbol{y}_0, f_\phi(\boldsymbol{x})\big)$ for $t = 1, 200, \ldots, T$; (**Bottom**) right to left: $p_\theta(\boldsymbol{y}_{t-1} \,|\, \boldsymbol{y}_t, \boldsymbol{x})$ for $t = T, \ldots, 200, 1$.

where Y and $\hat{Y}$ are true and predicted class label, respectively; $\hat{P}$ is the predicted probability associated to $\hat{Y}$. A perfect calibration is defined as:

$$\mathbb{P}\big(\hat{Y} = Y | \hat{P} = p\big) = p, \ \forall p \in [0, 1]. \tag{32}$$

However, since the predicted probability $\hat{Y}$ is continuous in $[0, 1]$, we cannot compute ECE with finite instances, thus we approximate it by first **dividing the probability space into M bins with equal width**, then compute the confidence and accuracy within each bin. Each test instance is placed into one specific bin by the predicted probability value associated with the true class label. For the m-th bin B_m, we have accuracy (proportion of correct predictions)

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(\hat{y}_i = y_i) \tag{33}$$

and confidence (mean of predicted probabilities)

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i, \tag{34}$$

where $\hat{y}_i$ and $\hat{p}_i$ are the predicted label and its associated probability value, and y_i is the true label, for instance i. We thus have the empirical version of ECE as:

$$\text{ECE} \coloneqq \sum_{m=1}^{M} \frac{|B_m|}{n} \big|\text{acc}(B_m) - \text{conf}(B_m)\big|, \tag{35}$$

where $|B_m|$ and n are the cardinality of the m-th bin and the total number of instances, respectively.

To summarize, although we are interested in measuring the miscalibration through the difference between $p(y_i = \hat{y}_i \,|\, \boldsymbol{x}_i)$ and $\hat{p}_i$, we are only able to compute such miscalibration *at the granularity of subgroup level* – usually with the number of subgroups M set to 10 or less in practice. In other

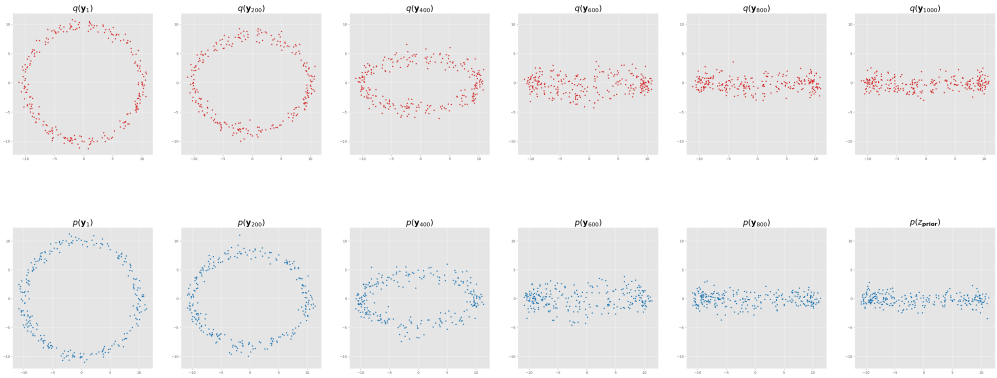

Figure 5: q (top) and p (bottom) distribution samples for full circle regression task during training.

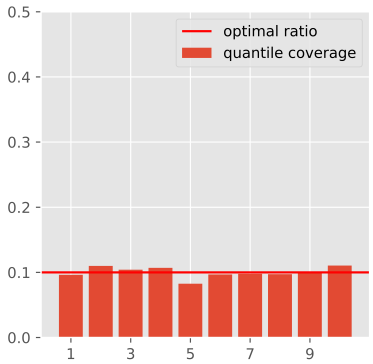

Figure 6: Sample coverage ratio by bins for full circle regression task (QICE 0.62).

words, we cannot make a statement with the existing classification framework like the following: given this new test instance, we predict the class label to be (some class), but we are very sure our prediction is correct. This observation motivates us to introduce an alternative way in measuring model confidence, *at the granularity of instance level*.]

A.16 Classification with Prediction Uncertainty Evaluation on CIFAR-100 Dataset

[rev.: We perform classification on CIFAR-100 dataset with CARD. We contextualize the performance of CARD through the mean accuracy of other BNNs with ResNet-18 architecture in Table 20, where the metrics were reported in Tomczak et al. [55]. Our pre-trained classifier has the same ResNet-18 architecture, which achieves a test accuracy of 71.37%. With 10 runs, CARD achieves a test accuracy of $71.42\% \pm 0.01\%$, along with a PAvPU of $71.48\% \pm 0.03\%$.

Table 20: Comparison of mean accuracy (in $\%$) on CIFAR-100 dataset with other BNNs.

Model	CMV-MF-VI	CM-MF-VI	CV-MF-VI	MF-VI	MC Dropout	MAP	f_ϕ (ResNet-18)	CARD
Accuracy	60.59 ± 0.39	59.61 ± 0.37	46.22 ± 0.54	40.54 ± 0.72	54.49 ± 0.36	52.08 ± 0.34	71.37	$\mathbf{71.42 \pm 0.01}$

We pick one run (accuracy 71.42%, PAvPU 71.53%) and report PIW among correct and incorrect predictions, at entire test set level and at class level. Since the dataset has 100 classes, we only report PIW within the class with the most and the least accurate predictions. We report the metrics along with those from the ImageNet-100 and the ImageNet dataset in Table 21. Furthermore, when making the prediction for each instance merely by the class with the narrowest PIW, we obtain a test accuracy of 60.53%. Meanwhile, we set $\alpha = 0.01$ to conduct the paired two-sample t-test. We only report mean accuracy given different t-test results at test set level due to the total number of classes: among

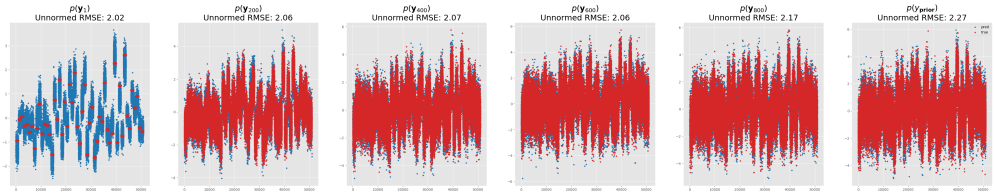

Figure 7: q (red) and p (blue, 1 000 samples per $\boldsymbol{x}$) samples from UCI Boston test set.

all 10 000 test instances, we reject the null hypothesis of the t-test on 9 961 instances, which have a mean accuracy of 71.56%; the remaining 39 not-rejected cases have a mean accuracy of 35.90%.]

Table 21: PIW (multiplied by 100) for the CIFAR-100, ImageNet-100 and ImageNet classification tasks. Due to the number of classes, we only report the PIW for all test instances, and within the most and least accurate class.

Class	Accuracy	PIW	
		Correct	Incorrect
CIFAR-100			
overall	71.42%	0.59	3.91
most acc.	95.00%	0.16	1.92
least acc.	44.00%	5.09	5.84
ImageNet-100			
overall	82.34%	2.06	13.73
most acc.	98.00%	0.72	8.06
least acc.	42.00%	6.79	14.15
ImageNet			
overall	74.28%	0.65	3.11
most acc.	98.00%	0.27	2.80
least acc.	8.00%	20.10	50.07

A.17 Classification with Prediction Uncertainty Evaluation on ImageNet-100 Dataset

[rev.: In this section, we report experiment results of classification on ImageNet-100 dataset with CARD, in a similar fashion as Sec. A.16. For this dataset, f_ϕ has a ResNet-50 architecture and uses parameter of linear classifier fine-tuned with the self-supervised recipe [18, 62], which achieves a test accuracy of 82.30%. With 10 runs, CARD achieves a test accuracy of $82.35\% \pm 0.03\%$, along with a PAvPU of $82.73\% \pm 0.07\%$.

We pick one run (accuracy 82.34%, PAvPU 82.70%) to report PIW among correct and incorrect predictions, at entire test set level and within class with the most and the least accurate predictions, in Table 21. When making the prediction for each instance merely by the class with the narrowest PIW, we obtain a test accuracy of 68.64%. Meanwhile, we set $\alpha = 0.01$ for the paired two-sample t-test, same as the CIFAR-100 experiment. Again due to the total number of classes, we only report mean accuracy given different t-test results at test set level: among all 5 000 test instances, we reject the null hypothesis of the t-test on 4 942 instances, which have a mean accuracy of 82.90%; the remaining 58 not-rejected cases have a mean accuracy of 34.48%.]

A.18 Classification with Prediction Uncertainty Evaluation on ImageNet Dataset

[rev.: In this section, we report experiment results of classification on ImageNet dataset with CARD, in a similar fashion as Sec. A.16. For this dataset, similar to ImageNet-100, f_ϕ uses a ResNet-50 architecture with parameters learned with self-supervised pipeline, which achieves a test accuracy of 73.87%. With 10 runs, CARD achieves a test accuracy of $74.28\% \pm 0.01\%$, along with a PAvPU of $74.63\% \pm 0.02\%$.

We pick one run (accuracy 74.28%, PAvPU 74.63%) to report PIW among correct and incorrect predictions, at entire test set level and within class with the most and the least accurate predictions, in Table 21. When making the prediction for each instance merely by the class with the narrowest

PIW, we obtain a test accuracy of 69.22%. Meanwhile, we set $\alpha = 0.01$ for the paired two-sample t-test, same as the CIFAR-100 experiment. Due to the total number of classes, we only report mean accuracy given different t-test results at test set level: among all $50\,000$ test instances, we reject the null hypothesis of the t-test on $49\,651$ instances, which have a mean accuracy of 74.63%; the remaining 349 not-rejected cases have a mean accuracy of 24.93%.]

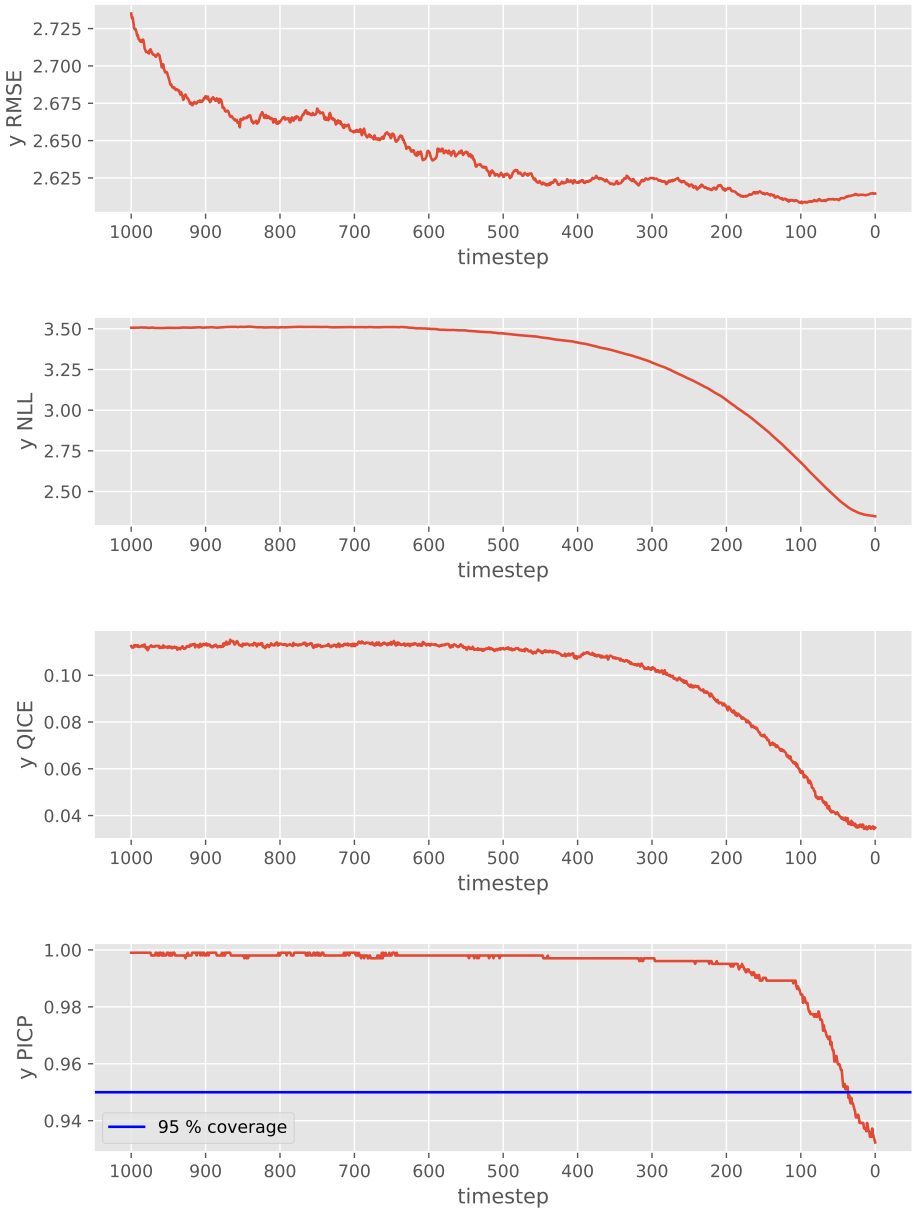

Figure 8: Change in regression evaluation metrics on UCI Boston dataset. The value at each timestep is the mean across all 20 splits.

Table 22: Model size and computation complexity of CARD and deterministic neural networks (architecture same as f_ϕ) on different datasets. Throughputs measures the number of samples calculated per second. For UCI tasks, we measure the parameter size and computation complexity on the subset with biggest data dimension.

Task	Regression				Classification					
Dataset	Toy		UCI		FMNIST		CIFAR-10/100		ImageNet-100/1K	
model	#Params	Throughputs	#Params	Throughputs	#Params	Throughputs	#Params	Throughputs	#Params	Throuputs
CARD	5.24e-2M	25334.72	6.55e-2M	244146.47	6.41e-2M	1033.24	16.52M	244.21	32.76M	137.13
D NN	5.22e-2M	25378.50	6.51e-2M	250555.79	6.02e-2M	1057.95	11.17M	249.56	25.62M	147.56

B Broader Impact and Limitations

We believe that our model has a practical impact in various industrial settings, where supervised learning methods have been increasingly applied to facilitate the decision making process. For regression tasks, *e.g.*, understanding the relationship between drug dosage and biographical features of a patient in the medical domain, and evaluating player performance given various on-court measurements, where the actual distribution of the real-world data is complicated and unknown, we are able to compute summary statistics with minimal amount of assumptions, as we do not assume the parametric form of the conditionals $p(\boldsymbol{y} \mid \boldsymbol{x})$. The analytical results from our framework thus have the potential to reach a broader audience, who in the past might be hindered by the sheer amount of jargon due to the lack of statistical training. For classification tasks, as mentioned in both of the experiments, we could easily adopt a human-machine collaboration framework: since our model is capable of conveying the prediction confidence along with the prediction itself, we could pass the cases where the model is less assertive to humans for further evaluation. This trait is especially valuable for classification tasks with exceptionally imbalanced data, *e.g.*, fraud detection and ad click-through rate prediction, where the volume of one class could be orders of magnitude more than the other. False negative errors for these applications are usually quite expensive, and simply adjusting classification threshold would often put too many positive predictions for human agent evaluation. As demonstrated in our experiments, CARD is capable of providing uncertain cases with a very reasonable ratio, keeping the workload for human agents at a sensible level.

Meanwhile, since CARD is capable of modeling multi-modality, we are concerned that it could be devised for malicious purposes, like revealing personalised information of the patient through reverse engineering the prediction results: as an extension to the medical example of breast cancer given in the introduction section, one could tell with high confidence the gender of the patient based on the predicted mode. For research purposes, in this work we only consider Gaussian diffusion and the reverse denoising in CARD, while there could be much more options when given different data. For example, in classification we can optimize the classification likelihood with cross-entropy instead of using simple MSE loss, directly perform diffusion in discrete spaces, as in [2]. We only take a pre-trained model f_ϕ as a deterministic neural network, while there could be more possibilities like combining BNN methods with CARD. Moreover, the computation efficiency of CARD may also be further investigated if the dataset size becomes larger. We encourage researchers in our community to further study those potential safety concerns and approaches for improvements in order to develop more mature supervised learning tools.

C Computational Resources

Our models are trained and evaluated with a single Nvidia GeForce RTX 3090 GPU. We use PyTorch 1.10.0 as the deep learning framework. Our CARD model trains between 100 and 200 steps per second at batch size specified in Table 12 for regression tasks, and at 44 steps per second at batch size 256 for classification on CIFAR-10 dataset. The sampling for a batch of 250 test instances, each with 100 samples, takes around 1.05 seconds at most for the classification task on FashionMNIST. The computation time could vary if we apply different network architectures for regression and classification tasks, and the architecture details are provided in Appendix A.7.

We summarize the model parameter size and measure the throughputs, *i.e.*, how many data samples can be proceeded per second, as computation complexity measure in Table 22. As our model consists of both a diffusion model θ in the label prediction and a pre-trained prior mean model f_ϕ, we compute

them respectively. From the table we can observe the parameter size of θ is heavier than f_ϕ, while the computation is more efficient than f_ϕ.