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 [49, 19, 50, 52] 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 [21, 55] and Langevin dynamics [38, 57], as pioneered by Song and Ermon [49]. They can also be understood from the perspective of diffusion probabilistic models [47, 19], 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 [49, 52, 7, 39, 42] 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 [19]. 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. [40], 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 [24, 56], 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) [19], 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 $t > 1$, else set $\boldsymbol{z} = \boldsymbol{0}$
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 $\boldsymbol{y}_{t-1} = \gamma_0\hat{\boldsymbol{y}}_0 + \gamma_1\boldsymbol{y}_t + \gamma_2 f_\phi(\boldsymbol{x}) + \sqrt{\tilde{\beta}_t}\boldsymbol{z}$
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 [42] 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. [19]. 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 [18, 11, 29, 54], 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. [60], 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].

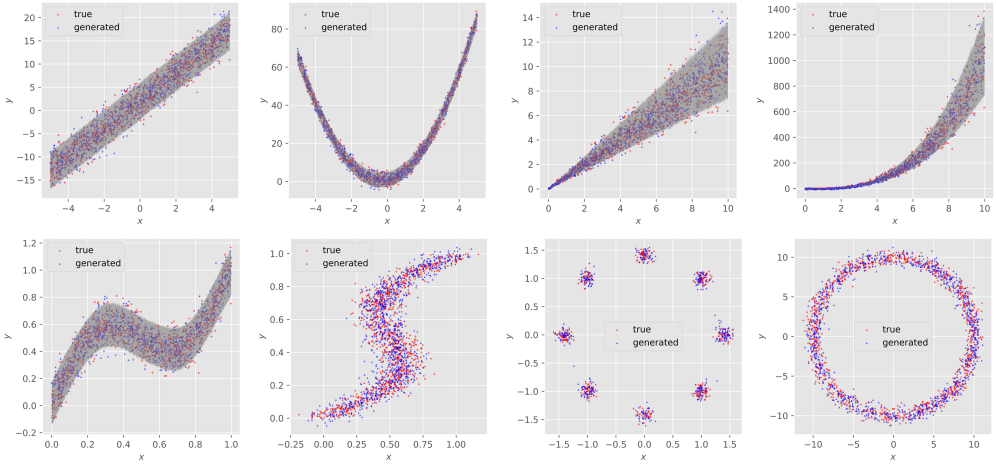

Figure 1: Regression toy example scatter plots. (**Top**) left to right: linear regression, quadratic regression, log-log linear regression, log-log cubic regression; (**Bottom**) left to right: sinusoidal regression, inverse sinusoidal regression, 8 Gaussians, full circle.

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 [18] and followed by Gal and Ghahramani [11] and Lakshminarayanan et al. [29]. The dataset information is provided in Table 11.

We apply multiple train-test splits with $90\%/10\%$ ratio in the same way as [18] (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 [62]. 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. [29], 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.

Therefore, 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) [37] have been adopted in recent literature [28, 46] 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 [54], 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	MAP	MC Dropout	ResNet-18+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 [35, 41, 58, 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] 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.

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

[20] 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.

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

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

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

[24] 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.

[25] 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.

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

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

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

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

[34] 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.

[35] 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.

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

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

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] 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.

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

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

[49] 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.

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

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

[52] 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`.

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

[54] 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.

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

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

[57] 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.

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

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

[60] 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.

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

[62] 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.

[63] 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]