Sample-based Uncertainty Quantification with a Single Deterministic Neural Network

Takuya Kanazawa and Chetan Gupta
Industrial AI Lab, Hitachi America, Ltd. R&D, Santa Clara, CA, 95054, U.S.A.

Keywords: Uncertainty Quantification, Ensemble Forecasting, CRPS, Aleatoric Uncertainty, Epistemic Uncertainty, Energy Score, DISCO Nets.

Abstract: Development of an accurate, flexible, and numerically efficient uncertainty quantification (UQ) method is one of fundamental challenges in machine learning. Previously, a UQ method called DISCO Nets has been proposed (Bouchacourt et al., 2016) that trains a neural network by minimizing the so-called energy score on training data. This method has shown superior performance on a hand pose estimation task in computer vision, but it remained unclear whether this method works as nicely for regression on tabular data, and how it competes with more recent advanced UQ methods such as NGBoost. In this paper, we propose an improved neural architecture of DISCO Nets that admits a more stable and smooth training. We benchmark this approach on miscellaneous real-world tabular datasets and confirm that it is competitive with or even superior to standard UQ baselines. We also provide a new elementary proof for the validity of using the energy score to learn predictive distributions. Further, we point out that DISCO Nets in its original form ignore epistemic uncertainty and only capture aleatoric uncertainty. We propose a simple fix to this problem.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

In real-world applications of artificial intelligence (AI) and machine learning (ML), it is becoming essential to estimate uncertainty of predictions made by AI/ML models. This is especially true in high-stakes areas such as health care and autonomous driving, where inadvertent decisions can cause fatal damages. While traditional methods for uncertainty quantification (UQ) such as bootstrapping and quantile regression can be partly applied to modern AI/ML models, the rapid progress especially in the field of deep neural networks calls for a development of novel UQ methodologies (Abdar et al., 2021; Gawlikowski et al., 2021).

In this paper, we revisit a UQ method for neural networks (NN) on regression tasks. This method, called DISCO Nets (DISsimilarity COefficient Networks) (Bouchacourt et al., 2016), enables us to estimate uncertainty of a prediction in a fully nonparametric manner by using just a single deterministic NN (see also (Harakeh and Waslander, 2021; Pacchiardi et al., 2021) for related studies). Unlike Bayesian NN and Gaussian processes, DISCO Net does not encounter computational bottlenecks when it is scaled to a large problem. In addition, it can model a posterior distribution in more than one dimension straightforwardly, in contrast to conventional quantile regression-based methods that do not trivially generalize to higher dimensions.

Despite its flexibility and versatility, however, DISCO Net has not gained popularity comparable to other UQ methods such as Monte Carlo dropout (Gal and Ghahramani, 2016). There could be multiple reasons for that. First, DISCO Net belongs to a class of NN called implicit generative networks, which are generally difficult to train (Tagasovska and Lopez-Paz, 2019). Second, understanding the theoretical underpinning of DISCO Net requires sophisticated mathematics and statistics of scoring rules of distributions, which makes the method unfamiliar and less approachable for data science practitioners in industry. Third, while it is widely known that there are two types of uncertainty in ML called *aleatoric uncertainty* and *epistemic uncertainty* (Abdar et al., 2021; Gawlikowski et al., 2021; Hüllermeier and Waegeman, 2021), it is not totally obvious which of these uncertainties is estimated by DISCO Net. Fourth, DISCO Net has so far been primarily benchmarked on a limited range of tasks in computer vision, and evidence of its favorable performance on learning tasks on tabular data has been missing in the literature, despite abundance of tabular data in industry.

Kanazawa, T. and Gupta, C.
Sample-based Uncertainty Quantification with a Single Deterministic Neural Network.
DOI: 10.5220/0011546800003332
In *Proceedings of the 14th International Joint Conference on Computational Intelligence (IJCCI 2022)*, pages 292-304
ISBN: 978-989-758-611-8; ISSN: 2184-3236
Copyright © 2023 by SCITEPRESS – Science and Technology Publications, Lda. Under CC license (CC BY-NC-ND 4.0)

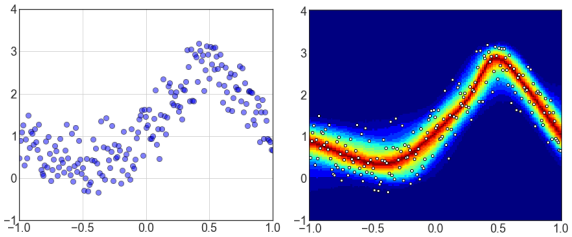

Figure 1: Left: random data points. Right: the result of applying DN+ to the same data. Aleatoric uncertainty is quantified appropriately. See section 5 for more detail.

We wish to address all these points in this paper. First, we propose an improved NN architecture of DISCO Net. The original DISCO Net assumed that a noise vector was simply concatenated to the input vector before being fed to the NN. However, empirically, the training of NN with this method is found to be difficult. The original work (Bouchacourt et al., 2016) has overcome this difficulty by using a very high (~ 200) dimensional noise vector, but it consequently requires a quite large NN and thus increases the training time. We rather propose to simply insert an embedding layer of a noise vector. This trick leads to stable learning *even for a one-dimensional noise vector* and boosts computational efficiency substantially. Second, we provide a rudimentary analytical proof that training of NN using the energy score allows to learn the correct predictive distribution, at least when the batch size in stochastic gradient descent training is large enough. We hope this will make DISCO Net more accessible and trustworthy for practical data scientists. Third, we numerically verify that DISCO Net misses epistemic uncertainty, although it is capable of capturing aleatoric uncertainty quite well. We propose to use an oscillatory activation function in DISCO Net to capture epistemic uncertainty. Fourth, we conduct extensive numerical experiments to test the effectiveness of the proposed approach for tabular datasets. We use 10 real-world datasets and show that the UQ capability of the present approach is competitive with or even superior to popular baseline methods.

The enhanced DISCO Net approach proposed in this work will be referred to as DN+ in the remainder of this paper.

In section 2 we summarize preceding works on UQ in machine learning. In section 3 we provide the background of this research, introducing concepts such as the energy score and CRPS. In section 4 we review DISCO Nets. Section 5 is about our main contributions, i.e., improvements on DISCO Nets, discussions on numerical experiments, and comparison with baselines. We conclude in section 6. A theoretical discussion on the validity of UQ training using the energy score is relegated to the appendix, owing to its technical nature.

2 RELATED WORK

In a classification problem, the uncertainty of an ML model may be concisely represented by class probabilities that sum to unity. In contrast, the uncertainty in regression is more complicated. Many existing UQ methods for regression calculate only the $X\%$ confidence interval, where $X \in \{95, 99\}$ are among the common choices. Although such a single interval estimate is quite useful from a practical point of view, it lacks detailed information on the shape of the posterior distribution $p(\mathbf{y}|\mathbf{x})$, where $\mathbf{x}$ denotes the input variable and $\mathbf{y}$ the output variable. Classical approaches such as the Gaussian process regression (Rasmussen and Williams, 2006) can represent the posterior as a multivariate Gaussian distribution, but fails when the noise is heteroscedastic and is unable to express multimodal posterior distributions.

A variety of improved UQ methods for deep NN have been proposed in the literature (Abdar et al., 2021; Gawlikowski et al., 2021).[1] Pivotal examples include Bayesian NN (Lampinen and Vehtari, 2001), quantile regression NN (Cannon, 2011), NN that model Gaussian mixtures (Bishop, 1994), ensembles of NN (Lakshminarayanan et al., 2017; Pearce et al., 2020), Monte Carlo dropout (Gal and Ghahramani, 2016), and direct UQ approaches (Lahlou et al., 2021). These methods effectively work to quantify either aleatoric uncertainty, epistemic uncertainty, or both (Hüllermeier and Waegeman, 2021). Aleatoric uncertainty represents inherent stochasticity of the response variable, whereas epistemic uncertainty comes from limitation of knowledge and can be decreased by gathering more data.

Recently, normalizing flows (NF) (Kobyzev et al., 2021; Papamakarios et al., 2021) have emerged as a versatile and flexible tool for UQ. NF is a generative model that uses a composition of multiple differentiable bijective maps modeled by NN to transform a simple (such as uniform or Gaussian) distribution to a more complex distribution of real data. Examples of probabilistic forecast based on NF can be found in (Sick et al., 2020; Charpentier et al., 2020; Dumas et al., 2021; Sendera et al., 2021; Jamgochian et al., 2022; Rittler et al., 2022; März and Kneib, 2022; Arpogaus et al., 2022; Cramer et al., 2022).

[1] Here we will focus on work other than DISCO Net (Bouchacourt et al., 2016; Harakeh and Waslander, 2021; Pacchiardi et al., 2021) because the latter was discussed in great detail in Introduction.

Ensembles from a single NN (known as *implicit NN ensembles*) have been studied in (Huang et al., 2016; Huang et al., 2017; Tagasovska and Lopez-Paz, 2019; Maddox et al., 2019; Antoran et al., 2020). While (Huang et al., 2016; Antoran et al., 2020) propose a NN with a probabilistic depth, (Huang et al., 2017; Maddox et al., 2019) suggest to use information in a stochastic gradient descent trajectory of a single NN to construct ensembles.

Gradient boosting decision trees are among the most popular ML models, which often outperform NN as a point forecaster on benchmark tests with tabular data. Recently, probabilistic forecasts in gradient boosting decision trees have been studied in (Duan et al., 2020; Sprangers et al., 2021; Brophy and Lowd, 2022). While (Duan et al., 2020; Sprangers et al., 2021) require fitting a parametric distribution (such as Gaussian or Weibull) to the data, (Brophy and Lowd, 2022) allows to produce a more flexible, nonparametric distributional forecast.

3 BACKGROUND

3.1 Distance between Probability Distributions

The discrepancy (distance) between two continuous probability distributions can be quantified using a variety of metrics such as f-divergences. One of the popular metrics in machine learning is the *maximum mean discrepancy* (MMD) (Gretton et al., 2012)

$$\mathrm{MMD}[\mathcal{F},p,q] := \sup_{f\in\mathcal{F}} \left(\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)]\right) \quad (1)$$

where p and q are distributions and $\mathcal{F}$ is a class of real-valued functions $f : \mathcal{X} \to \mathbb{R}$. When $\mathcal{F}$ is a reproducing kernel Hilbert space, there exists a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

$$\mathrm{MMD}^2[\mathcal{F},p,q] = \mathbb{E}_{x,x'\sim p}[k(x,x')] - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)] + \mathbb{E}_{y,y'\sim q}[k(y,y')]\,. \quad (2)$$

MMD has been recently used in deep reinforcement learning to model the distribution of future returns (Nguyen-Tang et al., 2021; Zhang et al., 2021).

A closely related quantity that measures the statistical distance between two distributions in Euclidean space is the so-called *energy distance* (Szekely, 2003; Szekely and Rizzo, 2004; Szekely and Rizzo, 2013; Szekely and Rizzo, 2017), defined as

$$\mathcal{D}_E(p,q) := 2\mathbb{E}_{\mathbf{x}\sim p,\mathbf{y}\sim q}\|\mathbf{x}-\mathbf{y}\| - \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim p}\|\mathbf{x}-\mathbf{x}'\| - \mathbb{E}_{\mathbf{y},\mathbf{y}'\sim q}\|\mathbf{y}-\mathbf{y}'\| \quad (3)$$

where $\|\cdot\|$ stands for the Euclidean L_2 norm. Equation (3) bears close similarity to (2). In fact, under suitable conditions, they are proven to be equivalent (Sejdinovic et al., 2013; Shen and Vogelstein, 2018).

A useful finite-sample version of (3) is given by

$$\mathcal{D}_E(p,q) = \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\|\mathbf{x}_i-\mathbf{y}_j\| - \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\|\mathbf{x}_i-\mathbf{x}_j\| - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{y}_i-\mathbf{y}_j\|\,. \quad (4)$$

3.2 How to Measure the Reliability of a Probabilistic Prediction?

Uncertainty of a prediction can be expressed in miscellaneous ways. Confidence intervals are a popular and simple example, whereas the shape of the probability density can also be specified. How to assess the reliability of such forecasts? If we had the knowledge of the data-generating process and knew the exact form of the true conditional probability $p(\mathbf{y}|\mathbf{x})$, it would be straightforward to assess the accuracy of a probabilistic forecast $p(\widehat{\mathbf{y}}|\mathbf{x})$ by using distance metrics such as MMD and the energy distance; however, this is not possible in general, as we only have access to a single realization $\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^N$ of the data-generating process. This issue was investigated in detail by (Gneiting and Raftery, 2007), who introduced the important concept of *proper scoring rules*. When the response variable $\mathbf{y}$ is a scalar, a widely used *strictly proper* scoring rule for probabilistic forecast is the continuous ranked probability score (CRPS) defined as

$$\mathrm{CRPS}(F,y) := \int_{-\infty}^{\infty} \mathrm{d}\widehat{y}\,\left[F(\widehat{y}) - \mathbb{1}\{\widehat{y}\geq y\}\right]^2\,, \quad (5)$$

where F is the cumulative distribution function of a prediction, and $\mathbb{1}\{\diamondsuit\} := 1$ if $\diamondsuit$ is true and $= 0$ otherwise. When the prediction is deterministic (i.e., a point forecast), CRPS coincides with the mean absolute error (MAE), so CRPS can be considered as a probabilistic generalization of MAE. Interestingly, CRPS may be cast into the form (Gneiting and Raftery, 2007)

$$\mathrm{CRPS}(F,y) = \mathbb{E}_F|\widehat{y}-y| - \frac{1}{2}\mathbb{E}_F|\widehat{y}-\widehat{y}'|\,, \quad (6)$$

where $\widehat{y}$ and $\widehat{y}'$ are independent copies of a random variable with the cumulative distribution function F and $\mathbb{E}_F$ is the expectation value with regard to F. Note that (6) agrees with $1/2$ of the energy distance (3) with a single sample of y. It is important that (6) is *strictly proper*, namely, its expectation value w.r.t. the distribution of y, i.e. $\mathbb{E}_y[\mathrm{CRPS}(F,y)]$, is minimized if and only if F coincides with the true cumulative distribution of y.

The expression (6) readily lends itself to a multi-dimensional generalization

$$\mathrm{ES}(P,\mathbf{y}) = \frac{1}{2}\mathbb{E}_{\widehat{\mathbf{y}},\widehat{\mathbf{y}}'\sim P}\|\widehat{\mathbf{y}}-\widehat{\mathbf{y}}'\|^{\alpha} - \mathbb{E}_{\widehat{\mathbf{y}}\sim P}\|\widehat{\mathbf{y}}-\mathbf{y}\|^{\alpha}, \quad (7)$$

which is called the *energy score* (Gneiting et al., 2008; Pinson and Girard, 2012) and has been heavily used in meteorology. (Note that *lower* CRPS is better and *higher* energy score is better.) The energy score is strictly proper for $0 < \alpha < 2$ (Szekely, 2003).

4 GENERATIVE ENSEMBLE PREDICTION BASED ON ENERGY SCORES

In this section, we give an overview of DISCO Nets (Bouchacourt et al., 2016), which make a density forecast based on sample generation.

The first step in this method is to enlarge the feature space, from the original one $\mathcal{X}$ to $\mathcal{X}\times\mathcal{X}_\mathrm{b}$, where $\mathcal{X}_\mathrm{b}$ is an arbitrary base space. The dimension of $\mathcal{X}_\mathrm{b}$ must be greater than or equal to the dimension of the target variable $\mathbf{y}$. A convenient choice for $\mathcal{X}_\mathrm{b}$ would be $[0,1]^d$. The next step is to select a base distribution P_b over $\mathcal{X}_\mathrm{b}$, which may be simply a uniform distribution. The training of NN proceeds along a standard stochastic gradient descent style. We take a minibatch of samples from the training dataset and "augment" every sample with a random vector sampled from P_b. That is, each input vector $\mathbf{x}_i$ is first duplicated N_b times, and then each copy is paired with an independently sampled random vector from $\mathcal{X}_\mathrm{b}$. As a result, the minibatch size increases from b_batch to $b_\mathrm{batch}\times N_\mathrm{b}$. The resulting "elongated" input vectors are fed into the NN and the outputs $\{\widehat{\mathbf{y}}_i^{(n)}\}_{n=1}^{N_\mathrm{b}}$ are obtained. Finally, the loss function $\mathcal{L}$ is computed by using the true regression target $\mathbf{y}_i$ and the model predictions $\{\widehat{\mathbf{y}}_i^{(n)}\}_{n=1}^{N_\mathrm{b}}$ according to the formula

$$\mathcal{L}\left(\mathbf{y},\{\widehat{\mathbf{y}}^{(n)}\}_{n=1}^{N_\mathrm{b}}\right) := \frac{1}{N_\mathrm{b}}\sum_{i=1}^{N_\mathrm{b}}\|\mathbf{y}-\widehat{\mathbf{y}}^{(i)}\| - \frac{1}{2N_\mathrm{b}^2}\sum_{i=1}^{N_\mathrm{b}}\sum_{j\neq i}\|\widehat{\mathbf{y}}^{(i)}-\widehat{\mathbf{y}}^{(j)}\|. \quad (8)$$

This is a negated finite-sample energy score (7) with $\alpha = 1$. The first term of (8) encourages all samples $\widehat{\mathbf{y}}^{(i)}$ to come closer to $\mathbf{y}$, whereas the second term of (8) induces a repulsive force between samples so that the samples' cloud swells. This loss function is summed across all data in the minibatch and the NN parameters are updated in the direction of the gradient descent of the loss. Since the energy score is a *strictly proper* scoring rule, its expectation value is maximized if and only if the evaluated density forecast agrees with the true density of $\mathbf{y}$. Hence it can be expected that the above training will let the NN predictions converge to the true data-generating distribution. For a formal justification on this point, see Appendix A. To the best of our knowledge, the elementary argument presented in Appendix A is new.

In the test phase, for a given input vector $\mathbf{x}$, we can generate as many ensemble predictions as we like from the NN by first duplicating $\mathbf{x}$, then augmenting each copy with an independent random vector from P_b, and finally feeding them into the NN. Statistical quantities such as the mean, standard deviation and quantile points can be readily computed from the resulting ensemble of predictions.

DISCO Net (and DN+) differs from recent approaches based on NF (Sick et al., 2020; Charpentier et al., 2020; Dumas et al., 2021; Sendera et al., 2021; Jamgochian et al., 2022; Rittler et al., 2022; März and Kneib, 2022; Arpogaus et al., 2022; Cramer et al., 2022) in a number of ways. First, NF estimates the joint density $p(\mathbf{x},\mathbf{y})$ to derive the conditional density $p(\mathbf{y}|\mathbf{x})$ while DISCO Net directly models the latter with no recourse to the former. Second, NF maximizes the log-likelihood to optimize parameters while DISCO Net does not use the log-likelihood at all during training. Third, and related to the second point, NF needs a differentiable and *invertible* mapping, whereas DISCO Net is free from such restrictions, which leads to more flexible modeling and considerable simplification of implementation.

When compared with Monte Carlo dropout (Gal and Ghahramani, 2016), which is one of the most popular UQ methods, the main difference is that the NN in DISCO Net/DN+ is deterministic and no probabilistic sampling of network structures is performed.

It is worthwhile to note that the size of the prediction ensemble in DISCO Net/DN+ is arbitrary, and can be increased at no extra cost in the test phase, in contrast to the existing methods (Nguyen-Tang et al., 2021; Zhang et al., 2021) that have a fixed number of network outputs to model the ensemble.

Finally, we stress that DISCO Net/DN+ is non-parametric and, in principle, can model an arbitrary distribution in arbitrary dimensions, whereas networks that learn quantile points of a distribution using a quantile loss (Dabney et al., 2018b; Dabney et al., 2018a; Yang et al., 2019; Singh et al., 2022) generally fail to model a distribution in more than one dimension.

5 EXPERIMENTS

In this section we introduce improvements to DISCO Nets and conduct numerical experiments on synthetic and real tabular datasets to assess the reliability of our ensemble forecasts.

5.1 Evaluation Metrics

When the true underlying data-generating distribution is known, we can use the following metrics to directly measure the discrepancy between the predicted density and the true density.

- The Jensen–Shannon distance (JSD) (Endres and Schindelin, 2003): This quantity is the square root of the Jensen–Shannon divergence (JSdiv) defined as

$$\mathrm{JSD}(P\|Q)^2 = \mathrm{JSdiv}(P\|Q) \tag{9}$$

$$:= \frac{1}{2}D_{\mathrm{KL}}(P\|M) + \frac{1}{2}D_{\mathrm{KL}}(Q\|M)\,, \tag{10}$$

  where $D_{\mathrm{KL}}(P\|Q)$ is the Kullback–Leibler divergence (Kullback and Leibler, 1951) between the distributions P and Q, and $M := (P+Q)/2$. When the logarithm with base 2 is used for computation, $0 \le \mathrm{JSD}(P\|Q) \le 1$ holds.

- The Hellinger distance (Nikulin, 2001): This quantity is given in terms of probability densities as

$$\mathrm{HD}(P,Q) := \sqrt{\frac{1}{2}\int \mathrm{d}\mathbf{x}\left(\sqrt{p(\mathbf{x})} - \sqrt{q(\mathbf{x})}\right)^2}\,. \tag{11}$$

  It satisfies the bound $0 \le \mathrm{HD}(P,Q) \le 1$.

- The first Wasserstein distance: In terms of the cumulative distribution functions $F_p(x)$ and $F_q(x)$ of P and Q, we have, for one-dimensional distributions,

$$\mathrm{WD}(P,Q) = \int_{-\infty}^{\infty} \mathrm{d}x\, |F_p(x) - F_q(x)|\,. \tag{12}$$

- The square root of the energy distance (3): Similarly to the Wasserstein distance, in terms of one-dimensional cumulative distribution functions we have (Szekely, 2003)[2]

$$\mathrm{ED}(P,Q) := \sqrt{\mathcal{D}_E(P,Q)} \tag{13}$$

$$= \sqrt{2\int_{-\infty}^{\infty} \mathrm{d}x\, \left(F_p(x) - F_q(x)\right)^2}\,. \tag{14}$$

5.2 Neural Network Architecture

When the response variable $\mathbf{y}$ to be predicted is a scalar, we adopt a NN architecture depicted in figure 2 with $d_{\mathrm{a}} = 1$. Inspired by (Dabney et al., 2018a) we convert a scalar a sampled from a uniform distribution over $[0,1]$ to a 128-dimensional vector $(1, \cos(\pi a), \cos(2\pi a), \cdots, \cos(127\pi a))$, which is later merged into the layer of $\mathbf{x}$ via a dot product. For the MLP part of the network, we used 2 hidden layers of width 128, each followed by an activation layer. The network parameters are optimized with Adam.

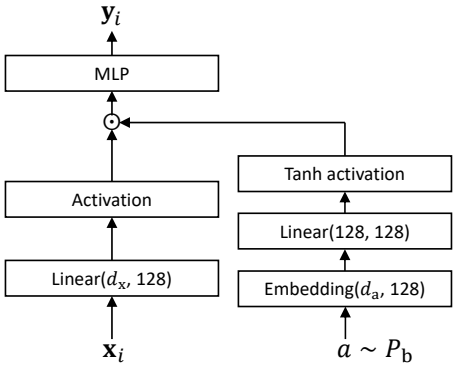

Figure 2: Architecture of a NN used to make distributional predictions by DN+. Here $d_{\mathbf{x}}$ is the feature dimension of $\mathbf{x}$ and d_{a} is the dimension of a.

When $\mathbf{y}$ is a two-dimensional vector, we sample a vector $a = (a_1, a_2)$ from a uniform distribution over $[0,1]^2$ and expand it to a 128-dimensional vector $(1, \cos(\pi a_1), \cos(2\pi a_1), \cdots, \cos(63\pi a_1)) \oplus (1, \cos(\pi a_2), \cos(2\pi a_2), \cdots, \cos(63\pi a_2))$.

The activation function for the embedding layer is changed from ReLU in (Dabney et al., 2018a) to Tanh. This choice was partly motivated by (Ma et al., 2020, Appendix B.3), which recommended using Sigmoid instead of ReLU. On the other hand, activation functions for the MLP part are not fixed and can be changed flexibly.

As we will see later, this surprisingly simply trick of inserting an embedding layer of an external noise works perfectly well for stable and efficient NN training in DN+.

5.3 Test on a Unimodal Synthetic Dataset

We tested the proposed method on a simple one-dimensional dataset (figure 1, left). The data comprising 200 points were generated from the distribution

$$y = \exp(\sin(\pi x)) + \varepsilon, \tag{15}$$

$$\varepsilon \sim \mathcal{N}\left(0, 0.4^2\right), \quad -1 \le x \le 1\,. \tag{16}$$

[2]The square root is taken here to match the definition of the energy distance function in Scipy: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.energy_distance.html.

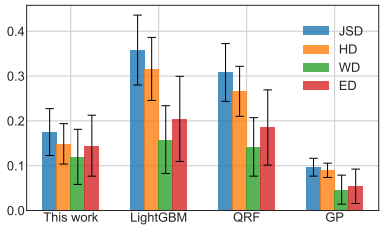

Figure 3: Average distance between the true distribution $p(y|x)$ and the output of each distributional predictors. The error bars represent the mean $\pm$ one standard deviation.

We applied DN+ to this data with the hyperparameters $n_{\text{epoch}} = 300, N_{\text{b}} = 50$ and $b_{\text{batch}} = 20$, using Tanh layer for the activation function. All parameters of the linear layers were initialized with random variables drawn from $\mathcal{N}\left(0, 0.1^2\right)$. The obtained distributional forecast (figure 1, right) captures aleatoric uncertainty of the data reasonably well. The color in the figure represents $0.5 - |Q - 0.5|$, where $Q \in [0,1]$ denotes the quantile of the predictive density.

To quantitatively assess the result, we took 200 equidistant points over the interval $[-1,1]$ of the x-axis and, for each x in this set, sampled 3000 points from the ensemble prediction of DN+. Then we computed the discrepancy between the obtained distribution and the ground-truth distribution (15) and took the average of the distance metric over all 200 points.

For comparison, we performed similar analyses with three popular UQ methods: LightGBM with a quantile loss (Ke et al., 2017), quantile regression forests (QRF) (Meinshausen, 2006), and Gaussian process regression (GP) (Rasmussen and Williams, 2006). For QRF and GP we have used the implementation of (Scikit-Garden, 2017) and (Pedregosa et al., 2011), respectively. Hyperparameters of LightGBM and QRF were tuned with Optuna (Akiba et al., 2019). We computed 51 quantiles with LightGBM and QRF, and therefrom constructed the conditional probability density $p(y|x)$ approximately. The result was so spiky that we had to smooth it with a Gaussian filter.

The benchmark result is summarized in figure 3. (For the definition of four distance metrics, we refer to section 5.1.) The result clearly shows that DN+ yields better distributional forecasts than LightGBM and QRF. However, DN+ is outperformed by GP, which does not come as a surprise because the ground-truth distribution (15) is Gaussian. We once again emphasize that the result for DN+ has been obtained in a fully nonparametric manner without assuming any parametric distribution such as Gaussian.

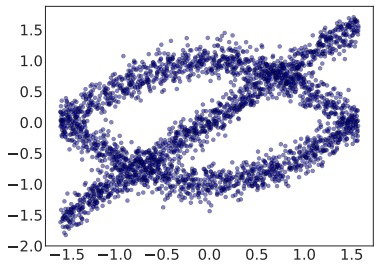

Figure 4: Average distance between the predictive distribution and the true distribution after training with four kinds of activation functions. The error bars are one standard deviation above and below the mean.

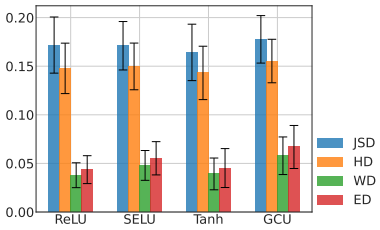

Figure 5: Data comprising 3000 points generated by adding a Gaussian noise to three crossing curves.

5.4 Test on a Multimodal Synthetic Dataset

Next, we test the proposed method on a multimodal synthetic dataset. The data were generated from the distributions over $-\pi/2 \le x \le \pi/2$,

$$\begin{cases} y = x + \varepsilon, \\ y = \cos x + \varepsilon', \\ y = -\cos x + \varepsilon'', \end{cases} \qquad \varepsilon, \varepsilon', \varepsilon'' \sim \mathcal{N}(0, 0.15^2). \quad (17)$$

The scatter plot of the data is shown in figure 5. The dataset includes 3000 points in total, 1000 for each distribution. As a function of x, y is multimodal and in general there are three peaks in the conditional probability density $p(y|x)$.

To test the effectiveness of DN+, we trained a NN with the hyperparameters $n_{\text{epoch}} = 300$, $N_{\text{b}} = 50$ and $b_{\text{batch}} = 150$. We conducted training with four distinct activation functions: Rectified Linear Unit (ReLU), Scaled Exponential Linear Unit (SELU) (Klambauer et al., 2017), Tanh activation, and Growing Cosine Unit (GCU) (Noel et al., 2021). To measure the performance, we took 100 equidistant points $x \in [-\pi/2, \pi/2]$ and, for each of them, computed the estimate $p(\widehat{y}|x)$ via $M = 10^4$ ensemble prediction. The distance between $p(\widehat{y}|x)$ and the true conditional $p(y|x)$ was measured with the metrics in section 5.1 and was then averaged over all 100 points. The result is shown in figure 5. It is observed that the scores of all four activation functions are similar, although Tanh activation

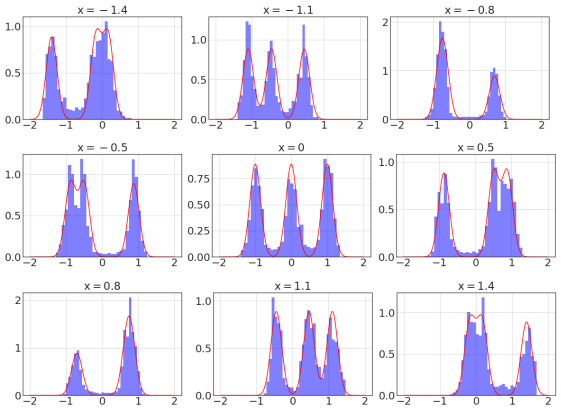

Figure 6: The histogram of the predicted conditional distribution $p(\hat{y}|x)$ (blue) overlayed with the true distribution $p(y|x)$ (red) for the dataset in figure 5. Tanh activation was used.

seems to perform slightly better than the others and GCU slightly worse than the others.

To gain more insights into the approximation quality of our method, we show the predicted distribution versus the ground truth for various values of x in figure 6. Clearly the predicted distribution reproduces the complex multimodal shape of the true distribution in a satisfactory manner. This is not possible with conventional methods such as the Gaussian processes.

We have also applied plain vanilla DISCO Nets (with no embedding layer of a random noise) to this dataset, but it only yielded seriously corrupted estimates of the predictive distribution. Actually it was this failure that forced us to seriously consider modifications to DISCO Nets.

While all activation functions give comparable accuracy of approximation for the range of data $-\pi/2 \leq x \leq \pi/2$, what happens outside this range? Do they still maintain similar functional forms? We have numerically studied this and found, as shown in figure 7, that the four neural networks give drastically different extrapolations. Since no data is available for $|x| > \pi/2$, the epistemic uncertainty must be high and a predictive distribution of y should have a broad support. This seems to hold true only for the network with GCU activation. We therefore conclude that, while the choice of an activation function hardly affects the network's capability to model aleatoric uncertainty for in-distribution data, it *does* affect the ability to model epistemic uncertainty for out-of-distribution data and hence great care must be taken.

To check the hyperparameter dependence of results, we have repeated numerical experiments with the same dataset for various sets of b_{batch} and N_{b}, using ReLU activation. The number of epochs n_{epoch}

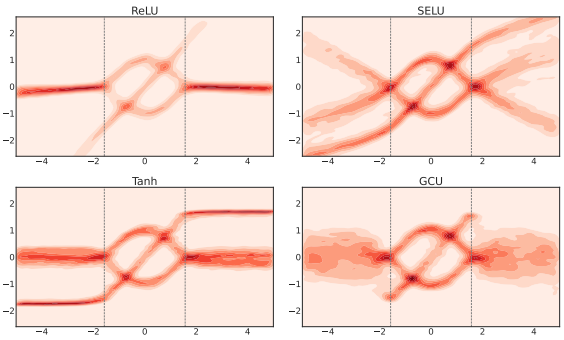

Figure 7: Density plots of the predictive distributions for $-5 \leq x \leq 5$ with four activation functions. Two vertical dashed lines at $x = \pm\pi/2$ indicate the boundary of the training dataset.

(a) Batch size

N_{b} \	50	100	150	200	250
10	0.1946	0.1814	0.1700	0.1681	0.1705
25	0.1837	0.1506	0.1575	0.1431	0.1447
50	0.1730	0.1625	0.1337	0.1318	0.1362
100	0.1624	0.1450	0.1371	0.1408	0.1354

(b) Batch size

N_{b} \	50	100	150	200	250
10	0.0663	0.0652	0.0561	0.0554	0.0605
25	0.0647	0.0569	0.0623	0.0532	0.0486
50	0.0593	0.0578	0.0464	0.0450	0.0449
100	0.0620	0.0538	0.0485	0.0539	0.0487

Figure 8: (a) Hellinger distance and (b) Wasserstein distance between the true distribution $p(y|x)$ and the predictive distribution $\widehat{p}(y|x)$ computed using ReLU activation for varying b_{batch} and N_{b}.

was fixed to $2 \times b_{\text{batch}}$ to keep the number of gradient updates the same across experiments, ensuring a fair comparison. The result is shown in figure 8. It is observed that larger N_{b} and/or larger b_{batch} generally leads to better (lower) scores. Note, however, that larger N_{b} entails higher numerical cost and longer computational time. It is therefore advisable to make these parameters large enough within the limit of available computational resources, although it is likely that the best hyperparameter values will in general depend on the characteristics of individual datasets under consideration.

In order to benchmark DN+, we again compared it with LightGBM, QRF and GP. We have used ReLU activation for DN+. Some examples of the posterior

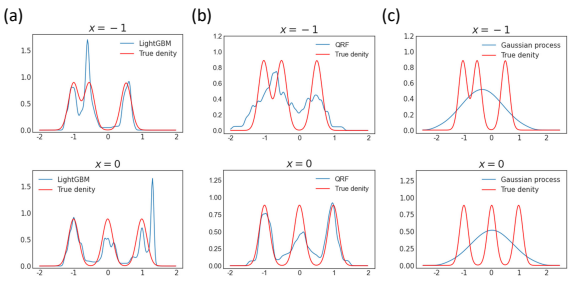

Figure 9: The true density $p(y|x)$ (red curve) and the estimated density $\widehat{p}(y|x)$ (blue curve) obtained at $x = -1$ and 0 for the dataset in figure 5 using the three methods: (a) LightGBM with quantile losses, (b) QRF, and (c) GP.

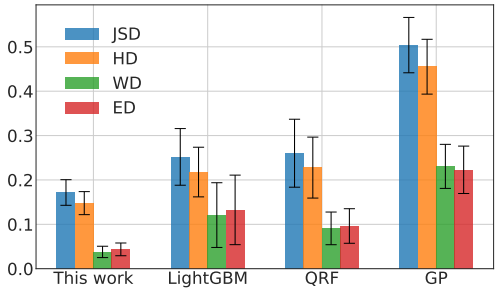

Figure 10: Comparison of DN+ against three popular UQ methods (lower scores are better). Error bars represent the mean ± one standard deviation.

distributions obtained with each method are displayed in figure 9. While LightGBM and QRF seem to capture qualitative features of the posterior well, GP entirely misses local structures due to its inherent Gaussian nature of the approximation.

For a quantitative comparison of the methods, we computed the discrepancy between the estimated distribution and the true distribution and took the average over $-\pi/2 \leq x \leq \pi/2$. The result is displayed in figure 10. Not surprisingly, GP marked the worst score. DN+ attained the lowest score in all the four metrics, thus underlining the effectiveness of this approach for modeling complex multimodal density distributions.

5.5 Sampling of Functions after Fitting to Data

Drawing samples from a posterior predictive distribution is central to Bayesian inference (Bishop, 2006). In GP the cost of naive generation of samples scales cubically with the number of observations, and historically a variety of approximation schemes have been proposed (Lázaro-Gredilla et al., 2010; Wilson et al., 2020). In DISCO Nets and DN+, it is in fact trivial to sample functions from the posterior after training on a dataset. While we make an ensemble forecast at given $\mathbf{x}$ by running a forward pass with a large number of distinct inputs $a \sim P_\mathrm{b}$ sampled from the base space $\mathcal{X}_\mathrm{b}$, sample functions can be obtained by simply fixing $a \sim P_\mathrm{b}$ and running a forward pass with varying $\mathbf{x}$. For illustration, we generated a test dataset as shown in the top panel of figure 11. In total 600 points were generated uniformly inside three squares of size 1. We have applied DN+ to this data with hyperparameters $n_\mathrm{epoch} = 50$, $N_\mathrm{b} = 20$ and $b_\mathrm{batch} = 80$ using four activation functions. For each of them we sampled 10 functions, as shown in the bottom panel of figure 11. They cover the range of data in a similar way but perform qualitatively different extrapolations outside the range of data. These characteristics that are specific

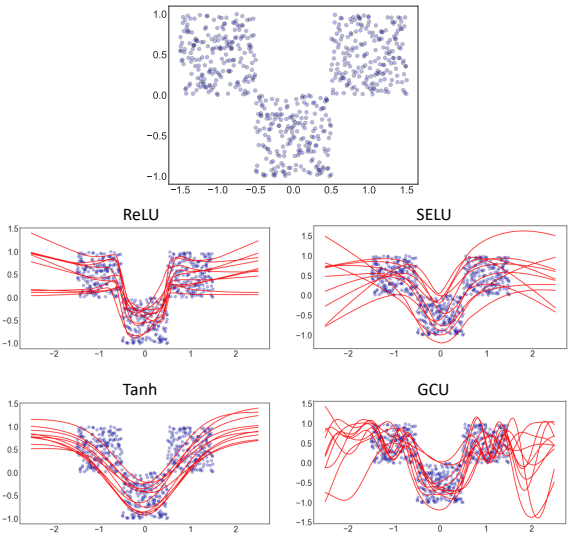

Figure 11: Data distribution (top) and samples drawn from the fitted NN (bottom) for the four activation functions.

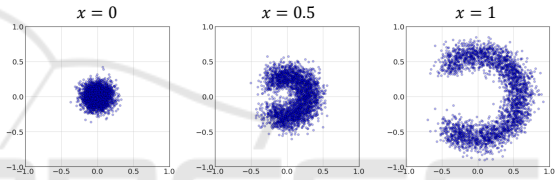

Figure 12: The dataset generated for the experiment with 2D output $\mathbf{y} = (y_1, y_2)$.

to activation functions, akin to the choice of kernel functions in GP, must be taken into consideration carefully when using this sampling technique for practical purposes.

5.6 Two-dimensional Prediction

Next, we proceed to considering the case where the response variable $\mathbf{y}$ is two-dimensional. As described in section 5.2, we sample a two-dimensional random vector $a \in [0,1]^2$ to make ensemble predictions. As a testbed of the proposed method, we generated a dataset from the distribution below.

$$\mathbf{y} = \begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix}, \quad \begin{cases} y_1(x) = 0.6\cos\theta \times x + \varepsilon \\ y_2(x) = 0.6\sin\theta \times x + \varepsilon' \end{cases} \tag{18}$$

$$\varepsilon, \varepsilon' \sim \mathcal{N}\left(0, 0.1^2\right), \quad \theta \sim U\left(\left[-\frac{3\pi}{4}, \frac{3\pi}{4}\right]\right) \tag{19}$$

for $0 \leq x \leq 1$. The scatter plots of the data at three values of x are displayed in figure 12. For small x, the data distribution is essentially isotropic, while for larger x a gap on the left gradually opens up. The task for DN+ is to learn this nontrivial evolution of the distribution over the entire unit interval $x \in [0,1]$. We

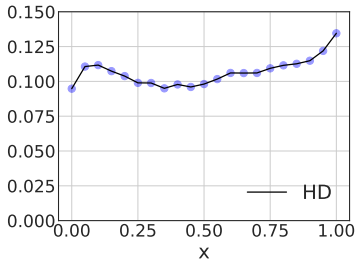

Figure 13: Density plots of the true distribution (left) and DN+'s prediction (right) in the (y_1, y_2)-plane.

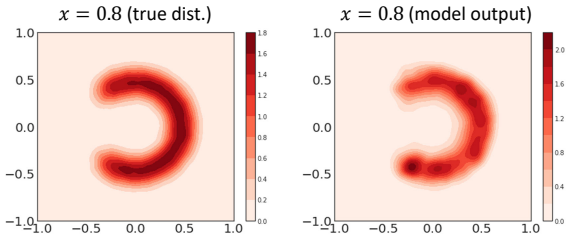

Figure 14: Discrepancy between the true distribution $p(\mathbf{y}|x)$ and the predicted distribution $p(\hat{\mathbf{y}}|x)$ measured by the Hellinger distance based on 10^5 ensemble forecast.

have taken an equidistant grid of 3000 points over $[0,1]$ and generated the dataset $\{(x_i, \mathbf{y}_i)\}_{i=1}^{3000}$, which was fed to our NN. We have used ReLU activation and trained the model with $b_{\text{batch}} = 300$, $N_{\text{b}} = 100$ and $n_{\text{epoch}} = 300$. For various x, we made an ensemble forecast with $M = 10^5$ and computed its Hellinger distance from the true distribution (18). The result is shown in figure 14. The worst score was 0.135 at $x = 1$ and the mean over all x was 0.106, hence we conclude that the model has correctly reproduced the data distribution for all x. For illustration, we show the true distribution and the learned distribution on the $\mathbf{y}$-plane at $x = 0.8$ in figure 14. Clearly, major features of the distribution are reproduced to a good accuracy. We anticipate that the result would be improved if the model's layer width is increased further and the hyperparameters of the training such as b_{batch} are optimized.

5.7 Test on Real-World Datasets

Finally we compare the performance of various baselines with DN+ on 10 publicly available tabular datasets for regression tasks. For details of datasets we refer to Appendix B. This time we have included NGBoost (Duan et al., 2020) in the baselines, which fits parametric distributions to data via gradient boosting of decision trees. For each dataset, we have used 90% of the data for training and 10% for testing. Hyperparameters of the baselines are provided in Appendix B. In DN+, we used the hyperparameters $(n_{\text{epoch}}, b_{\text{batch}}, N_{\text{b}}) = (150, 200, 100)$ for all datasets except for the Concrete dataset for which $(n_{\text{epoch}}, b_{\text{batch}}, N_{\text{b}}) = (200, 50, 50)$. In the inference mode, $M = 500$ ensemble predictions were made. `QuantileTransformer(output_distribution='normal')` from Scikit-Learn was used to preprosess both the input and output of the NN.

The result on the point forecast performance measured by MAE and RMSE is displayed in table 1. Overall, LightGBM was the best performing model and DN+ was close to worst, although DN+ was best on the Kin8nm dataset. This is partly as expected, since our NN of DN+ is not optimized for achieving the best point forecast. As is widely recognized, highly tuned gradient boosting models often outperform plain vanilla MLP, though we stress that DN+ can integrate with more complex and deeper NN as well. We leave engineering efforts for achieving better point forecast accuracy to future work.

The probabilistic forecast performance is compared in table 2. As a metric we used negative log-likelihood (NLL) and CRPS. In order to compute NLL numerically, we have used Gaussian KDE for the four methods other than NGBoost. This time, DN+ was best on average on NLL, while LightGBM was best on average on CRPS, respectively. It should be noted that DN+ was best on three datasets for NLL. Overall, the probabilistic forecast performance of DN+ was at least quite competitive with the standard baselines of probabilistic regression. The fact that DN+ did not clearly outperform the baselines may be indicating that the probability distributions of the response variables in these datasets may have a simple unimodal structure that hardly requires nonparametric methods for complex multimodal distributions.

6 CONCLUSION

In this paper, we introduced a method DN+ built on top of DISCO Net (Bouchacourt et al., 2016) for uncertainty quantification with neural networks for regression tasks. DN+ is a nonparametric method that can model arbitrary complex multimodal distributions. In contrast to Bayesian neural networks and Gaussian processes, DN+ scales well to a large dataset without a computational bottleneck. It also offers an easy way to sample as many functions as desired from the posterior distribution. Due to its simplicity, DN+ can be combined with a variety of complex neural networks, including convolutional and recurrent neural networks. In numerical experiments, we have shown that DN+ can even model a two-dimensional distribution of a response variable successfully, in stark contrast to conventional quantile-based approaches that struggle in

Table 1: Point forecast performance of each method on each dataset. Best result is in bold face. The lower score is better.

Dataset	MAE (↓)					RMSE (↓)				
	QRF	LGBM	NGB	GP	This work	QRF	LGBM	NGB	GP	This work
California	0.345	0.328	**0.319**	0.368	0.328	0.522	0.474	**0.465**	0.545	0.517
Concrete	2.63	**2.60**	2.69	2.68	2.71	3.47	**3.35**	3.49	3.61	3.65
Kin8nm	0.116	0.099	0.116	**0.060**	0.061	0.143	0.124	0.150	0.080	**0.079**
Power Plant	2.37	**2.30**	2.37	2.60	2.73	3.40	**3.22**	3.39	3.63	3.76
House Sales	68.2k	**63.6k**	68.3k	86.0k	79.7k	123k	**111k**	117k	153k	166k
Elevators	1.87E-3	1.62E-3	1.58E-3	**1.55E-3**	1.61E-3	2.76E-3	2.36E-3	2.15E-3	**2.09E-3**	2.24E-3
Bank8FM	0.0227	**0.0221**	0.0227	0.0231	0.0223	0.0311	0.0303	**0.0301**	0.0307	0.0310
Sulfur	**0.0172**	0.0194	0.0224	0.0183	0.0197	0.0308	0.0348	0.0386	**0.0302**	0.0338
Superconduct	**5.06**	6.45	6.66	5.35	5.53	**8.96**	10.6	10.5	9.22	9.75
Ailerons	2.07E-4	**1.07E-4**	2.09E-4	1.15E-4	1.30E-4	2.64E-4	**1.52E-4**	2.73E-4	1.58E-4	1.84E-4
Average Rank (↓)	2.7	**2.1**	3.4	2.7	3.2	3.3	**2.4**	3.0	2.8	3.5

Table 2: Probabilistic forecast performance of each method on each dataset. Best result is in bold face. The lower score is better.

Dataset	NLL (↓)					CRPS (↓)				
	QRF	LGBM	NGB	GP	This work	QRF	LGBM	NGB	GP	This work
California	0.316	0.194	0.111	0.411	**0.097**	0.143	**0.130**	0.139	0.167	0.134
Concrete	2.74	2.44	2.39	2.48	**2.36**	1.80	**1.03**	1.39	1.33	1.38
Kin8nm	−0.590	−0.972	−0.869	**−1.35**	−0.964	0.065	0.048	0.058	**0.031**	0.036
Power Plant	2.27	2.16	**2.11**	2.39	2.33	1.16	**1.04**	1.14	1.24	1.27
House Sales	**12.2**	12.3	12.3	12.4	12.4	28.8k	**26.0k**	28.2k	35.9k	28.2k
Elevators	−4.89	−5.09	**−5.14**	−5.04	−5.11	8.76E-4	**7.56E-4**	7.80E-4	7.65E-4	8.09E-4
Bank8FM	−2.19	−2.29	**−2.45**	−2.39	−2.36	0.0128	0.0110	**0.0105**	0.0106	0.0107
Sulfur	**−2.89**	−2.79	−2.59	−2.52	−2.72	**6.30E-3**	7.55E-3	9.63E-3	8.90E-3	7.79E-3
Superconduct	**2.72**	3.09	3.14	3.32	2.87	**1.63**	2.16	2.67	2.76	1.87
Ailerons	−7.03	−7.72	−7.06	−7.65	**−7.95**	9.42E-5	**5.17E-5**	1.21E-4	5.55E-5	6.54E-5
Average Rank (↓)	3.5	2.6	2.5	3.7	**2.3**	3.6	**1.8**	3.3	3.1	2.9

higher than one dimensions.

In future work, it would be interesting to apply DN+ to uncertainty quantification problems where images or time-series data are given as inputs to the model.

REFERENCES

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297. arXiv:2011.06225.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631. arXiv:1907.10902.

Antoran, J., Allingham, J., and Hernández-Lobato, J. M. (2020). Depth Uncertainty in Neural Networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. arXiv:2006.08437.

Arpogaus, M., Voss, M., Sick, B., Nigge-Uricher, M., and Dürr, O. (2022). Short-Term Density Forecasting of Low-Voltage Load using Bernstein-Polynomial Normalizing Flows. arXiv:2204.13939.

Bishop, C. M. (1994). Mixture density networks. Technical Report. Aston University, Birmingham.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bouchacourt, D., Mudigonda, P. K., and Nowozin, S. (2016). DISCO Nets : DISsimilarity COefficients Networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. arXiv:1606.02556.

Brophy, J. and Lowd, D. (2022). Instance-Based Uncertainty Estimation for Gradient-Boosted Regression Trees. arXiv:2205.11412.

Cannon, A. J. (2011). Quantile regression neural networks: Implementation in R and application to precipitation downscaling. *Computers & Geosciences*, 37:1277–1284.

Charpentier, B., Zügner, D., and Günnemann, S. (2020). Posterior Network: Uncertainty Estimation without

OOD Samples via Density-Based Pseudo-Counts. arXiv:2006.09239.

Cramer, E., Witthaut, D., Mitsos, A., and Dahmen, M. (2022). Multivariate Probabilistic Forecasting of Intraday Electricity Prices using Normalizing Flows. arXiv:2205.13826.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018a). Implicit Quantile Networks for Distributional Reinforcement Learning. *Proceedings of the 35 th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80*. arXiv:1806.06923.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2018b). Distributional Reinforcement Learning with Quantile Regression. *The Thirty-Second AAAI Conferenceon Artificial Intelligence (AAAI-18)*. arXiv:1710.10044.

Duan, T., Anand, A., Ding, D. Y., Thai, K. K., Basu, S., Ng, A., and Schuler, A. (2020). Ngboost: Natural gradient boosting for probabilistic prediction. *Proceedings of the 37th International Conference on Machine Learning, PMLR*, pages 2690–2700. arXiv:1910.03225.

Dumas, J., Lanaspeze, A. W. D., Cornélusse, B., and Sutera, A. (2021). A deep generative model for probabilistic energy forecasting in power systems: normalizing flows. arXiv:2106.09370.

Endres, D. M. and Schindelin, J. E. (2003). A New Metric for Probability Distributions. *IEEE Trans. Inf. Theory*, 49:1858–1860.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *Proceedings of The 33rd International Conference on Machine Learning, PMLR*, 48:1050–1059. arXiv:1506.02142.

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., and Zhu, X. X. (2021). A Survey of Uncertainty in Deep Neural Networks. arXiv:2107.03342.

Gneiting, T. and Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102:359–378.

Gneiting, T., Stanberry, L. I., Grimit, E. P., Held, L., and Johnson, N. A. (2008). Assessing probabilistic forecasts of multivariate quantities, with an application to ensemble predictionsof surface winds. *Test*, 17:211–235.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13:723–773.

Harakeh, A. and Waslander, S. L. (2021). Estimating and Evaluating Regression Predictive Uncertainty in Deep Object Detectors. In *ICLR 2021*. arXiv:2101.05036.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017). Snapshot Ensembles: Train 1, Get M for Free. In *5th International Conference on Learning Representations, ICLR 2017*. arXiv:1704.00109.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep Networks with Stochastic Depth. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *ECCV 2016*, volume 9908 of *Lecture Notes in Computer Science*. arXiv:1603.09382.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110:457–506. arXiv:1910.09457.

Jamgochian, A., Wu, D., Menda, K., Jung, S., and Kochenderfer, M. J. (2022). Conditional Approximate Normalizing Flows for Joint Multi-Step Probabilistic Forecasting with Application to Electricity Demand. arXiv:2201.02753.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 3149–3157.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-Normalizing Neural Networks. *31st Conference on Neural Information Processing Systems (NIPS 2017)*. arXiv:1706.02515.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2021). Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979. arXiv:1908.09257.

Kullback, S. and Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86.

Lahlou, S., Jain, M., Nekoei, H., Butoi, V., Bertin, P., Rector-Brooks, J., Korablyov, M., and Bengio, Y. (2021). DEUP: Direct Epistemic Uncertainty Prediction. arXiv:2102.08501.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. arXiv:1612.01474.

Lampinen, J. and Vehtari, A. (2001). Bayesian approach for neural networks—review and case studies. *Neural Networks*, 14:257–274.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. (2010). Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881.

Ma, X., Xia, L., Zhou, Z., Yang, J., and Zhao, Q. (2020). DSAC: Distributional Soft Actor Critic for Risk-Sensitive Reinforcement Learning. arXiv:2004.14547.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). A Simple Baseline for Bayesian Uncertainty in Deep Learning. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. arXiv:1902.02476.

Meinshausen, N. (2006). Quantile Regression Forests. *Journal of Machine Learning Research*, 7:983–999.

März, A. and Kneib, T. (2022). Distributional Gradient Boosting Machines. arXiv:2204.00778.

Nguyen-Tang, T., Gupta, S., and Venkatesh, S. (2021). Distributional Reinforcement Learning via Moment Matching. *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*. arXiv:2007.12354.

Nikulin, M. S. (2001). Hellinger distance. Encyclopedia of Mathematics, EMS Press.

Noel, M. M., L, A., Trivedi, A., and Dutta, P. (2021). Growing Cosine Unit: A Novel Oscillatory Activation Function That Can Speedup Training and Reduce Parameters in Convolutional Neural Networks.

Pacchiardi, L., Adewoyin, R., Dueben, P., and Dutta, R. (2021). Probabilistic Forecasting with Generative Networks via Scoring Rule Minimization. arXiv:2112.08217.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22:1–64. arXiv:1912.02762.

Pearce, T., Leibfried, F., and Brintrup, A. (2020). Uncertainty in Neural Networks: Approximately Bayesian Ensembling. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR*, 108:234–244. arXiv:1810.05546.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pinson, P. and Girard, R. (2012). Evaluating the quality of scenarios of short-term wind power generation. *Applied Energy*, 96:12–20.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Rittler, N., Graziani, C., Wang, J., and Kotamarthi, R. (2022). A Deep Learning Approach to Probabilistic Forecasting of Weather. arXiv:2203.12529.

Scikit-Garden (2017). https://github.com/scikit-garden/scikit-garden.

Sejdinovic, D., Sriperumbudur, B., Gretton, A., and Fukumizu, K. (2013). Equivalence of distance-based and RKHS-based statistics in hypothesis testing. *Ann. Statist.*, 41:2263–2291. arXiv:1207.6076.

Sendera, M., Tabor, J., Nowak, A., Bedychaj, A., Patacchiola, M., Trzcinski, T., Spurek, P., and Zieba, M. (2021). Non-Gaussian Gaussian Processes for Few-Shot Regression. *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*. arXiv:2110.13561.

Shen, C. and Vogelstein, J. T. (2018). The Exact Equivalence of Distance and Kernel Methods for Hypothesis Testing. arXiv:1806.05514.

Sick, B., Hothorn, T., and Dürr, O. (2020). Deep transformation models: Tackling complex regression problems with neural network based transformation models. arXiv:2004.00464.

Singh, R., Lee, K., and Chen, Y. (2022). Sample-based Distributional Policy Gradient. *Proceedings of The 4th Annual Learning for Dynamics and Control Conference, PMLR*, 168:676–688. arXiv:2001.02652.

Sprangers, O., Schelter, S., and de Rijke, M. (2021). Probabilistic gradient boosting machines for large-scale probabilistic regression. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. arXiv:2106.01682.

Szekely, G. J. (2003). E-statistics: Energy of Statistical Samples. Bowling Green State University, Department of Mathematics and Statistics Technical Report No. 03–05.

Szekely, G. J. and Rizzo, M. L. (2004). Testing for Equal Distributions in High Dimension. *InterStat*. November (5).

Szekely, G. J. and Rizzo, M. L. (2013). Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143:1249–1272.

Szekely, G. J. and Rizzo, M. L. (2017). The Energy of Data. *Annual Review of Statistics and Its Application*, 4:447–479.

Tagasovska, N. and Lopez-Paz, D. (2019). Single-Model Uncertainties for Deep Learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS 2019)*. arXiv:1811.00908.

Wilson, J., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. (2020). Efficiently sampling functions from Gaussian process posteriors. *Proceedings of the 37th International Conference on Machine Learning, PMLR*, 119:10292–10302. arXiv:2002.09309.

Yang, D., Zhao, L., Lin, Z., Qin, T., Bian, J., and Liu, T. (2019). Fully Parameterized Quantile Function for Distributional Reinforcement Learning. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. arXiv:1911.02140.

Zhang, P., Chen, X., Zhao, L., Xiong, W., Qin, T., and Liu, T.-Y. (2021). Distributional Reinforcement Learning for Multi-Dimensional Reward Functions. *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*. arXiv:2110.13578.

APPENDIX

A Minimizer of the Loss Function

As mentioned in section 3, the energy score (7) is a *strictly proper* scoring rule, making it an ideal tool for distributional learning. However, the underlying mathematics (Szekely, 2003) may not be easily accessible for practitioners of data science. In this appendix, we present a short and rudimentary argument highlighting that NN training based on the energy score does indeed allow to learn the correct predictive distribution.

Let us begin by observing that, when the loss (8) is averaged over all data in the minibatch, it approximately holds that

$$\begin{aligned}&\frac{1}{b_{\text{batch}}}\sum_{k=1}^{b_{\text{batch}}}\mathcal{L}\left(\mathbf{y}_k,\{\widehat{\mathbf{y}}_k^{(n)}\}_{n=1}^{N_{\text{b}}}\right)\\ &\simeq\int \mathrm{d}\mathbf{x}\,p(\mathbf{x})\Bigg[\int \mathrm{d}\mathbf{y}\,p(\mathbf{y}|\mathbf{x})\int \mathrm{d}\widehat{\mathbf{y}}\,p(\widehat{\mathbf{y}}|\mathbf{x})\|\mathbf{y}-\widehat{\mathbf{y}}\|\\ &\quad-\frac{1}{2}\int \mathrm{d}\widehat{\mathbf{y}}\,p(\widehat{\mathbf{y}}|\mathbf{x})\int \mathrm{d}\widehat{\mathbf{y}}'\,p(\widehat{\mathbf{y}}'|\mathbf{x})\|\widehat{\mathbf{y}}-\widehat{\mathbf{y}}'\|\Bigg]\end{aligned}\tag{20}$$

where $p(\mathbf{y}|\mathbf{x})$ is the true conditional distribution and $p(\widehat{\mathbf{y}}|\mathbf{x})$ is the predicted distribution by the model. To stress that they are different functions, we will hereafter write $p(\mathbf{y}|\mathbf{x})$ as $p_{\mathbf{x}}(\mathbf{y})$ and $p(\widehat{\mathbf{y}}|\mathbf{x})$ as $f_{\mathbf{x}}(\mathbf{y})$, respectively. Our interest is in the stationary point(s) of the loss function as a functional of $f_{\mathbf{x}}(\mathbf{y})$. Before taking the functional derivative w.r.t. $f_{\mathbf{x}}$, we must take note of the fact that $\int \mathrm{d}\mathbf{y}\, f_{\mathbf{x}}(\mathbf{y}) = 1$. Introducing a Lagrangian multiplier function $\lambda(\mathbf{x})$, we obtain the loss functional

$$\int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \int \mathrm{d}\mathbf{y} \int \mathrm{d}\mathbf{y}' \left\{ p_{\mathbf{x}}(\mathbf{y}) f_{\mathbf{x}}(\mathbf{y}') - \frac{1}{2} f_{\mathbf{x}}(\mathbf{y}) f_{\mathbf{x}}(\mathbf{y}') \right\} \times \|\mathbf{y} - \mathbf{y}'\| - \int \mathrm{d}\mathbf{x}\, \lambda(\mathbf{x}) \left(\int \mathrm{d}\mathbf{y}\, f_{\mathbf{x}}(\mathbf{y}) - 1 \right). \tag{21}$$

Differentiation w.r.t. $f_{\mathbf{x}}(\mathbf{y})$ yields the saddle point equation

$$\int \mathrm{d}\mathbf{y}' \{ p_{\mathbf{x}}(\mathbf{y}') - f_{\mathbf{x}}(\mathbf{y}') \} \|\mathbf{y} - \mathbf{y}'\| = \frac{\lambda(\mathbf{x})}{p(\mathbf{x})}. \tag{22}$$

Let us consider the case where $\mathbf{y}$ is a scalar. Using $\partial_y^2 |y - y'| = 2\delta(y - y')$ we may take the second derivative of both sides of (22) and obtain $f_{\mathbf{x}}(\mathbf{y}) = p_{\mathbf{x}}(\mathbf{y})$. This means that the model's prediction agrees with the true conditional distribution, which is the desired result.

Actually this mathematical trick applies to arbitrary *odd* dimensions. For illustration, suppose $\mathbf{y}$ is in three dimensions. The radial component of the Laplacian in three dimensions is given by

$$\Delta g(r) = \frac{1}{r^2} \partial_r \big(r^2 \partial_r g(r) \big) \tag{23}$$

for $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$, so

$$\Delta^2 r = \Delta(\Delta r) \propto \Delta \frac{1}{r} \propto \delta(\mathbf{x}). \tag{24}$$

In the last step we have used the fact that $1/r$ is the Green's function of the Poisson equation in three dimensions. Therefore, acting $\Delta_{\mathbf{y}}$ on both sides of (22) yields $f_{\mathbf{x}}(\mathbf{y}) - p_{\mathbf{x}}(\mathbf{y}) = 0$.

The case of even dimensions needs a little more caution. Let $\kappa(\mathbf{a}, \mathbf{b}) := \|\mathbf{a} - \mathbf{b}\|$. Suppose $g(\mathbf{a}) = \int \mathrm{d}\mathbf{b}\, \kappa(\mathbf{a}, \mathbf{b}) f(\mathbf{b})$ holds for some functions f and g. Then it is easy to verify that $f(\mathbf{c}) = \int \mathrm{d}\mathbf{a}\, \kappa^{-1}(\mathbf{c}, \mathbf{a}) g(\mathbf{a})$, where the inverse of κ is defined by the relation

$$\int \mathrm{d}\mathbf{a}\, \kappa^{-1}(\mathbf{c}, \mathbf{a}) \kappa(\mathbf{a}, \mathbf{b}) = \delta(\mathbf{c} - \mathbf{b}). \tag{25}$$

Using this for (22) we formally obtain

$$p_{\mathbf{x}}(\mathbf{y}) - f_{\mathbf{x}}(\mathbf{y}) = \frac{\lambda(\mathbf{x})}{p(\mathbf{x})} \int \mathrm{d}\mathbf{y}'\, \kappa^{-1}(\mathbf{y}, \mathbf{y}'). \tag{26}$$

We compute κ^{-1} in the momentum space since convolution becomes an algebraic product. Let

$$\|\mathbf{a}\| = \int \frac{\mathrm{d}^d q}{(2\pi)^d} \mathrm{e}^{i\mathbf{a}\cdot\mathbf{q}} \mu(\mathbf{q}). \tag{27}$$

Then

$$\mu(\mathbf{q}) = \int \mathrm{d}\mathbf{b}\, \mathrm{e}^{-i\mathbf{q}\cdot\mathbf{b}} \|\mathbf{b}\| \tag{28}$$

$$= \lim_{\varepsilon \to +0} \int \mathrm{d}\mathbf{b}\, \mathrm{e}^{-\varepsilon\|\mathbf{b}\| - i\mathbf{q}\cdot\mathbf{b}} \|\mathbf{b}\|. \tag{29}$$

For $d = 2$, a careful calculation shows that

$$\mu(\mathbf{q}) = -\frac{2\pi}{\|\mathbf{q}\|^3}. \tag{30}$$

Therefore

$$\kappa^{-1}(\mathbf{y}, \mathbf{y}') = -\frac{1}{2\pi} \int \frac{\mathrm{d}^2 p}{(2\pi)^2} \mathrm{e}^{i(\mathbf{y}-\mathbf{y}')\cdot\mathbf{p}} \|\mathbf{p}\|^3 \tag{31}$$

$$= \frac{1}{2\pi} \Delta_{\mathbf{y}'} \int \frac{\mathrm{d}^2 p}{(2\pi)^2} \mathrm{e}^{i(\mathbf{y}-\mathbf{y}')\cdot\mathbf{p}} \|\mathbf{p}\| \tag{32}$$

$$= -\frac{1}{4\pi^2} \Delta_{\mathbf{y}'} \frac{1}{\|\mathbf{y} - \mathbf{y}'\|^3}. \tag{33}$$

Plugging this into (26) yields $p_{\mathbf{x}}(\mathbf{y}) = f_{\mathbf{x}}(\mathbf{y})$, as desired. Here we have used the fact that the integral of a total derivative vanishes due to the absence of a surface contribution. The argument above generalizes to higher even d.

B Summary of Real-World Datasets

Below is the summary of the datasets used for the benchmark test in section 5.7.

Dataset	Size	dim($\mathbf{x}$)	URL
California	20640	8	https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html
Concrete	1030	8	https://www.openml.org/d/4353
Kin8nm	8192	8	https://www.openml.org/d/189
Power Plant	9568	4	https://archive.ics.uci.edu/ml/datasets/combined+cycle+power+plant
House Sales	21613	19	https://www.openml.org/d/42635
Elevators	16599	18	https://www.openml.org/d/216
Bank8FM	8192	8	https://www.openml.org/d/572
Sulfur	10081	5	https://www.openml.org/d/23515
Superconduct	21263	81	https://www.openml.org/d/43174
Ailerons	13750	33	https://www.openml.org/d/44137

Remarks:

- The number of trees in QRF was set to 1000.
- The number of trees in LightGBM was set to 300. `learning_rate` and `min_child_weight` were tuned.
- The number of trees in NGBoost was set to 500. `learning_rate` was tuned. The Gaussian distribution was used for fitting.
- GP's kernel used the sum of the isotropic Matern kernel and the white kernel. `QuantileTransformer(output_distribution='normal')` from Scikit-Learn was used to preprosess the input, and `RobustScaler()` from Scikit-Learn was used to scale the output.
- To calculate CRPS we have used the library `properscoring` available at https://github.com/TheClimateCorporation/properscoring.