DISTRIBUTION-FREE DATA UNCERTAINTY FOR NEURAL NETWORK REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantifying uncertainty is an essential part of predictive modeling, especially in the context of high-stakes decision-making. While classification output includes data uncertainty by design in the form of class probabilities, the regression task generally aims only to predict the expected value of the target variable. Probabilistic extensions often assume parametric distributions around the expected value, optimizing the likelihood over the resulting explicit densities. However, using parametric distributions can limit practical applicability, making it difficult for models to capture skewed, multi-modal, or otherwise complex distributions. In this paper, we propose optimizing a novel nondeterministic neural network regression architecture for loss functions derived from a sample-based approximation of the continuous ranked probability score (CRPS), enabling a truly distribution-free approach by learning to sample from the target's aleatoric distribution, rather than predicting explicit densities. Our approach allows the model to learn well-calibrated, arbitrary uni- and multivariate output distributions. We evaluate the method on a variety of synthetic and real-world tasks, including uni- and multivariate problems, function inverse approximation, and standard regression uncertainty benchmarks. Finally, we make all experiment code publicly available[1].

1 INTRODUCTION

With the continued development of machine learning, more and more industries consider using advanced predictive modeling to influence automated or human made decisions. Modern methodologies such as deep learning offer unparalleled accuracy for modeling complex real-world processes, with the potential to greatly enhance efficiency in many applications. Uncertainty quantification (UQ) is an essential part of predictive modeling, especially in applications where the prediction has the potential to influence high-stakes decision making. A large body of research deals with predicting the target value along with its uncertainty, such as Bayesian Neural Networks (BNN).

In a prediction task, uncertainty comes from multiple different sources. BNN-s are primarily concerned with *epistemic* (or knowledge) uncertainty, the uncertainty of how a model should predict the future based on the limited data we have at our disposal. In use-cases where data is limited, such approaches are invaluable for acknowledging potential blind spots in the modeling process.

However, in many industry applications, data is abundant. Still, a different kind of uncertainty remains even when data is plentiful: *aleatoric* (or data) uncertainty arises from constraints in the problem setup, where the model lacks information to make an exact prediction. Most tasks involve key pieces of information missing, factors which influence the target, yet ones we cannot provide to the model as inputs. Either due to inherent randomness or unobservable latent variables, in most real-world scenarios it is theoretically impossible to always predict the exact target value. However, as a constraint of the problem setup, aleatoric uncertainty cannot be reduced with additional data of the same kind and must be accounted for, even when data scarcity is not an issue.

Classification tasks are capable of expressing aleatoric uncertainty by design in the form of class probabilities. However, in the case of regression, the models traditionally only output a single value, or a simple parametric distribution at best. An *expectation* of 10mm of rainfall could indicate relative certainty of a rainy day with actual 10mm of rainfall, but could also indicate 10% chance of a deadly

[1] `https://anonymous.4open.science/r/crps_iclr-B270`

storm with 100mm of rain. In such cases, even a heteroscedastic distributional regression model with a simple unimodal output lacks the nuance required for informed decision making.

In this work, we propose a novel nonparametric way to model aleatoric uncertainty, relaxing the usual constraints of simple parametric distributions such as unimodality, and enabling the approximation of arbitrary distributions. We introduce new nondeterministic neural network architectures to generate samples from the aleatoric distribution of the target, and train the resulting models by drawing upon the theory of scoring rules, originally developed for evaluating probabilistic forecasts in fields such as meteorology and economics. We derive a novel, sample-based loss function based on an unbiased $\mathcal{O}(n \log n)$ formulation of the empirical *continuous ranked probability score* (CRPS), a strictly proper scoring rule for regression, describing both unweighted and weighted variants.

The resulting method is capable of learning arbitrary probability distributions from sample data in a well-calibrated manner, no longer being restricted to simple parametric uni-modal distributions, at the same time remaining conceptually simple and easy to implement. Further, it generalizes for multivariate regression, and is capable of learning the joint distribution of hundreds of output values. We evaluate our approach on multiple tasks, including synthetic uni- and multivariate examples, the inverse problem of MNIST classification, and also on the standard UCI regression uncertainty benchmark. The method consistently exhibits favorable behavior in practice, making it an appealing choice for modeling aleatoric uncertainty. Our contributions can be summarized as follows:

- We propose multiple novel architectures of nondeterministic neural networks for producing samples from the aleatoric target distribution with an efficient optimization procedure.
- We derive a loss function for optimizing neural networks using a novel unbiased empirical approximation of the CRPS, describing the loss for both unweighted and weighted samples.
- Finally, we extensively test the resulting method, comparing it against prior work such as BNN-s (Gawlikowski et al., 2023) or Mixture Density Networks (MDN) (Bishop, 1994).

Scaling. Our approach scales well to larger network sizes. While part of the model is evaluated multiple times, the added cost scales with target rather than input complexity, see Appendix C.8. Our experiments show that our models can represent a wide range of complex distributions, using a relatively small nondeterministic part considering modern deep learning hardware and model sizes.

2 Background

2.1 Types of Uncertainty

In prediction tasks, different sources of uncertainty can be distinguished. This section briefly summarizes the two main types we focus on; for a general overview, see (Gawlikowski et al., 2023).

Generally, the data we record and collect comes from a real-world process that is only partially observed. In regression, the target value (i.e., the value that we aim to predict) can be modeled as a random variable $Y : \Omega \to \mathbb{R}$, where elementary events $\omega \in \Omega$ are assumed to represent the true state of the world. Although we are unable to directly observe ω, we can observe a number of feature variables $X : \Omega \to \mathbb{R}^m$, which give us partial information about ω. The classic regression task is then defined as approximating the conditional expectation $\mathrm{E}[Y \mid X = x, D]$, where the data D is a sample of past values of (X, Y). In multivariate regression, Y takes on values from $\mathbb{R}^k$. Uncertainty quantification extends the regression task so that we are no longer simply interested in the conditional expectation, but rather the entire conditional distribution of y, i.e., $p(y \mid X = x, D)$.

Aleatoric uncertainty (Kendall & Gal, 2017) arises from X giving us limited information about ω. We can express aleatoric uncertainty through the conditional cumulative distribution function as

$$F_{Y|X}(y_0) = \int_\Omega \mathbb{1}\{Y(\omega) < y_0\}\ p(\omega \mid X)\, \mathrm{d}\omega. \tag{1}$$

Many constraints can be modeled through aleatoric uncertainty, for example measurement error, or uncertainty due to unobservable latent variables that affect Y. This kind of uncertainty *cannot* be reduced with more data (i.e., a larger sample D), as the uncertainty arises from constraints in the problem setup. In other words, aleatoric uncertainty is the uncertainty that remains even if we assume to have perfect knowledge of the relationships between X, Y, and ω.

Epistemic uncertainty on the other hand stems the fact that we have no perfect knowledge of the relationship of X, Y, and ω. Equation (1) is described in idealistic terms, and in fact neither term of the integration is known in practice, as our only information about X and Y comes from the sample D. Epistemic uncertainty is then modeled as the uncertainty of selecting f such that $f(y) \approx p(y \mid X = x, D)$, with larger dataset D implying less uncertainty. The majority of uncertainty research is focused on epistemic uncertainty, for example the research of BNN-s, see Section 2.2.

In classification, aleatoric uncertainty is expressed through class probabilities, while regression traditionally focuses on point estimates like the expectation. In probabilistic regression, aleatoric regression uncertainty is often modeled using parametric distributions, with the normal distribution being a particularly common choice (Jospin et al., 2022; Abdar et al., 2021; Magris & Iosifidis, 2023). However, parametric distributions are limited in their expressive power, and the wrong choice of distribution may undermine the entire learning process. Nevertheless, little attention is usually paid to the representation of the distribution of Y, with most uncertainty-related work either simply assuming the existence of a likelihood function or explicitly predicting the parameters of a normal distribution. This oversight persists despite the fact that that arbitrary-shaped aleatoric distributions can arguably be learned even when only a single value y is recorded for each input X, by leveraging the same implicit assumptions of smoothness that supervised learning generally relies on.

2.2 Bayesian Neural Networks

Bayesian neural networks (or BNN-s) (Blundell et al., 2015) are nondeterministic neural networks whose stated goal is to quantify epistemic uncertainty. Using the terminology of Section 2.1, BNN-s model the conditional density of Y through a neural network parameterized by $\theta \in \Theta$, i.e., $f_\theta(y) \approx p(y \mid X = x, D)$. Epistemic uncertainty is then expressed as the uncertainty of Θ, i.e.,

$$p(y \mid X = x, D) = \int_\Theta p(y \mid \theta, X = x)\, p(\theta \mid D)\, \mathrm{d}\theta, \tag{2}$$

where the first term being integrated is the aleatoric likelihood, in practice output by the neural network after having sampled θ according to the second term. A large body of research deals with the theory of the optimization of BNN-s, see (Gawlikowski et al., 2023) for an overview.

Each evaluation of a BNN yields a separate aleatoric uncertainty distribution, while epistemic uncertainty is captured by the nondeterminism of the network, with subsequent runs yielding different aleatoric uncertainty distributions sampled from the epistemic uncertainty distribution. In other words, epistemic uncertainty can be interpreted as the uncertainty of aleatoric uncertainty. Accordingly, they are sometimes referred to as first- and second-order uncertainties (Bengs et al., 2023).

Generally, aleatoric uncertainty is expressed in the form of a parametric distribution, while epistemic uncertainty is usually treated as nonparametric: while the distribution of the weights θ is individually assumed to be parametric, the resulting distribution of the output of the whole network can take arbitrary shapes. In this paper, we propose modeling both kinds as nonparametric distributions.

2.3 Proper scoring rules

When a model outputs a distribution, scoring rules can be used to quantify the accuracy of predictions based on a sample from the target, i.e., they are analogous to metrics like RMSE and AUC but specifically for probabilistic predictions. Generally, a scoring rule S takes a predicted distribution P and a sample y from the target, and assigns a score $\mathrm{S}(P, y)$. Denoting $\mathrm{E}_{Y \sim Q}\left[\mathrm{S}(P, Y)\right]$ as $\mathrm{S}(P, Q)$, a scoring rule is *proper* if $\mathrm{S}(Q, Q) \leq \mathrm{S}(P, Q)$ for all P, and *strictly proper* if equality implies $P = Q$ and vice versa. Strictly proper scoring rules encourage predictors to provide honest predictive distributions that accurately reflect the target's distribution (Gneiting & Raftery, 2007). An example is the logarithmic score (Good, 1952), defined as $\mathrm{LogS}(F, y) = -\log(f(y))$, where f is the density of F, more commonly known in machine learning literature as the negative log-likelihood.

A strictly proper scoring rule for regression is the *continuous ranked probability score* (CRPS):

$$\mathrm{CRPS}(F, y) = \int_{\mathbb{R}} (F(z) - \mathbb{1}\{y \leq z\})^2\, \mathrm{d}z, \tag{3}$$

where F is the cumulative distribution function (CDF) of the predicted distribution and y is the target sample. The CRPS can be analytically computed for a number of parametric distributions, see Jordan et al. (2019) for a comprehensive list.

2.4 EMPIRICAL CRPS

To approximate Equation (3) in practice, we can compute the formula substituting F for the empirical CDF of a sample. As described by (Gneiting & Raftery, 2007), an equivalent formulation (Baringhaus & Franz, 2004; Székely & Rizzo, 2005) of Equation (3), assuming $\mathrm{E}[F] < \infty$, is

$$\mathrm{CRPS}(F, y) = \mathrm{E}\,|Y - y| - \frac{1}{2}\,\mathrm{E}\,|Y - Y'|, \text{ where } Y, Y' \sim F,\ Y \perp\!\!\!\perp Y'. \tag{4}$$

When evaluating Equation (4) for the empirical CDF (F_{ECDF}) (Krüger et al., 2021) of a $(\hat{y}_1, \ldots, \hat{y}_n)$ sample distributed according to F, the resulting empirical score $\mathrm{CRPS}(F_{\text{ECDF}}, y)$ is given by

$$\frac{1}{n}\sum_{i=1}^{n} |y - \hat{y}_i| - \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} |\hat{y}_i - \hat{y}_j| = \frac{2}{n^2}\sum_{i=1}^{n} \left(\hat{y}_{(i)} - y\right)\left(n\mathbb{1}\{y < \hat{y}_{(i)}\} - i + \frac{1}{2}\right). \tag{5}$$

While the naive formula on the left hand side is $\mathcal{O}(n^2)$ complexity, the right hand side gives a relatively well-known (Murphy, 1970; Jordan, 2016; Laio & Tamea, 2007; Jordan et al., 2019) algebraically equivalent $\mathcal{O}(n \log n)$ formula, with $\left(\hat{y}_{(1)}, \ldots, \hat{y}_{(n)}\right)$ denoting the ordered sample statistic.

2.5 MIXTURE DENSITY NETWORKS

While most uncertainty-related work does not investigate distribution-free approaches for aleatoric regression uncertainty, one exception is Mixture Density Networks (MDN) (Bishop, 1994). MDN expresses the density as a mixture of a fixed number Gaussians. As the number of components tends to infinity, a Gaussian mixture is capable of approximating any distribution, therefore we consider MDN one of the strongest prior art in this field, also belonging to the class of models capable of learning aleatoric uncertainty in a distribution-free manner. However, MDN models often have practical limitations. In our experiments, using our own robust implementation we achieve much stronger scores for MDN then previously reported, see Appendix C.2 and the source code for details.

3 RELATED WORK

The literature of uncertainty quantification is vast, with the majority of works focusing on quantifying epistemic uncertainty, for example the BNN (Blundell et al., 2015), or its many enhancements (Sun et al., 2017; Gawlikowski et al., 2023). The UCI Benchmark (Hernández-Lobato & Adams, 2015) has become the standard benchmark to measure regression uncertainty, with many papers reporting RMSE and (negative) log-likelihood (NLL) results along with standard error. See Appendix A.1 for a list. While certain nonparametric regression approaches, e.g., some variants of as Gaussian Process Regression (Rasmussen, 2003; Sendera et al., 2021), are capable of modeling the output in a distribution-free manner, the area of distribution-free approaches for neural network-based aleatoric regression uncertainty is relatively under-researched. Such methods, for example MDN (Bishop, 1994), seem to be under-utilized and under-reported in uncertainty-related literature, with most surveys barely mentioning or omitting the topic, see Appendix A.2 for a small survey. MDN itself rarely appears as baseline. We are only aware of (El-Laham et al., 2023) on the UCI benchmark, however with substantially lower scores. See Appendix C.2 for more information.

Minimizing proper scoring rules to learn is common practice, either implicitly or explicitly. Both maximum likelihood for regression and cross entropy for binary classification coincide with minimizing the logarithmic score (Good, 1952), also a main component of variational inference for BNN-s (Blundell et al., 2015). Maximum likelihood estimation with an assumed Gaussian likelihood coincides with the Dawid-Sebastiani score (Dawid & Sebastiani, 1999; Gneiting & Raftery, 2007; Czado et al., 2009). Examples of explicitly using scoring rules to learn include Lakshminarayanan et al. (2017); Bengs et al. (2023); Vahidi et al. (2023); Bouchacourt et al. (2016). Prior work using the CRPS for learning often does so to learn combination weights (Thorey et al., 2017; Berrisch & Ziel, 2023; van der Meer et al., 2024). An analytic expression of the CRPS for the Gaussian is used for optimization in Rasp & Lerch (2018); Gebetsberger et al. (2018).

The biased sample-based $\mathcal{O}(n \log n)$ CRPS formula is well-known (Murphy, 1970; Jordan, 2016; Laio & Tamea, 2007; Jordan et al., 2019), and the bias of the $\mathcal{O}(n^2)$ formula has also been noted before (Ferro et al., 2008; Thorey et al., 2017), however to our knowledge we are the first to describe

the unbiased $\mathcal{O}(n \log n)$ formula. Weighted CRPS versions exist (Gneiting & Ranjan, 2011; Allen, 2023; Taggart, 2023; Pic et al., 2023), however in the sense of weights over the range of Y to emphasize certain target values. In contrast, we introduce a *weighted-sample*-based approximation for the CRPS and, to our knowledge, are first to describe both the $\mathcal{O}(n^2)$ and the $\mathcal{O}(n \log n)$ formulas.

Non-gaussian output distributions beyond MDN have been studied before, e.g., the method of (Depeweg et al., 2018) is capable of representing complex distributions, including multi-modal ones. In the context of generative modeling, related concepts are, e.g., Normalizing Flows (Ardizzone et al., 2018), and *Maximum Mean Discrepancy* (MMD) (Li et al., 2015; Dziugaite et al., 2015; Du et al., 2023; Hertrich et al., 2024; Cui et al., 2020), which has a close connection to *kernel scoring rules* (such as the CRPS) (Zawadzki & Lahaie, 2015; Schölkopf, 2000; Sejdinovic et al., 2012).

Closest to our approach is Bouchacourt et al. (2016) (DISCO Nets), a probabilistic method for hand-pose estimation. They also use injected random noise for nondeterminism (as in Section 4.1), and derive a dissimilarity coefficient (Rao, 1982) based loss function that reduces to the Energy Score as special case (see Section 4.5), which the authors also highlight. Developed independently, our work focuses on optimizing the CRPS, whose multivariate generalization also coincides with the Energy Score. Our work extends beyond DISCO Nets through an $\mathcal{O}(n \log n)$ loss formula instead of $\mathcal{O}(n^2)$, weighted and multi-head architectures, explicit handling of epistemic uncertainty, and further adaptations for traditional regression. We believe the connections between the two approaches serve to further emphasize the broader applicability of the shared ideas behind both.

4 Learning to Sample from the Aleatoric Distribution

We propose optimizing neural networks using $\mathrm{CRPS}(F_{\text{ECDF}}, y)$ from Equation (5) as a loss function. By taking multiple samples from a nondeterministic neural network, we leverage automatic differentiation to backpropagate the loss across multiple evaluations at the same time. Note that since the ordered sample is a sufficient statistic, the *sort* operation does not need to be backpropagated.

The effects of Equation (5) interpreted as a loss function are quite intuitive. The first term is equivalent to the mean absolute error (MAE), forcing the network to center its predictions around the median of the target distribution. The second term is being maximized due to its negative sign, forcing the distribution to spread out, increasing its variance. With the $\frac{1}{2}$ coefficient, balance is found exactly when the predicted distribution follows the target distribution (Gneiting & Raftery, 2007).

4.1 Fast Unbiased Sample-based CRPS Loss

The formula for the $\mathcal{O}(n \log n)$ sample-based CRPS of Equation (5) is relatively well-known. Less widely known is the fact that the formulas of Equation (5) are biased, not unlike the naive estimate for empirical variance. As described by Ferro et al. (2008), Equation (5) underestimates the second term because the diagonal is always constant zero, unlike in the original of Equation (4). As a result, the naive formula underestimates prediction variance, making the model underconfident. An unbiased version (Ferro et al., 2008; Thorey et al., 2017) can be calculated by using the coefficient $\frac{1}{2n(n-1)}$ instead of $\frac{1}{2n^2}$. We derive the corresponding modification of the $\mathcal{O}(n \log n)$ formula as

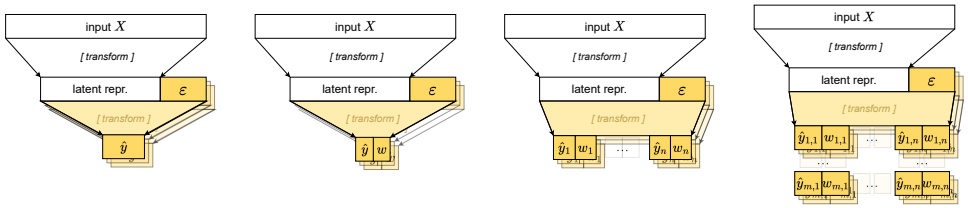
$$\mathrm{CRPS}'(F_{\text{ECDF}}, y) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \left(\tilde{y}_{(i)} - y\right) \left((n-1)\mathbb{1}\{y < \tilde{y}_{(i)}\} - i + 1\right), \tag{6}$$

where CRPS' denotes the unbiased version of the empirical CRPS, see Appendix B.4 for proof.

4.2 Optimizing Neural Networks for the Empirical CRPS

While any kind of nondeterministic neural network with sufficient expressive power should work in theory, we see no point in, e.g., sampling different weights for the network with each evaluation, as BNN-s do. Instead, we propose injecting the randomness through independent standard normal random samples ε concatenated to the input or a hidden layer activation, see Figure 1a. Exploring other nondeterminism injection methods is an interesting research topic, however is out of scope.

Using the terminology of Section 2.1, the resulting network can be interpreted as follows. First, the network transforms X into a latent representation ω_x revealing partial information about ω.



(a) Simple (b) Weighted (c) Multiple heads (d) Layered heads

Figure 1: Illustration of nondeterministic neural network architectures. The top (empty) part of each network is evaluated once, responsible for transforming the input. The yellow (solid) parts of each network are evaluated multiple times, with ε being randomly sampled each time.

Second, we sample ε in place of the unobservable latent variables that, together with X, are assumed to identify ω with certainty. Finally, the second part learns the deterministic mapping $(\omega_x, \varepsilon) \mapsto y$ which draws samples from the target distribution, with random ε values sampled in each run. In practice, only the second part needs to be evaluated multiple times. For further efficiency, the network can be implemented to compute multiple samples in a single batch, with output of shape $[b \times n \times k]$, where b is batch size, n is the number of samples, and k is the dimension of Y.

In theory, such a network can learn to produce samples from arbitrary probability distributions, and CRPS being a strictly proper scoring rule guarantees that the global optimum corresponds to the real aleatoric distribution under ideal conditions (i.e., assuming infinite data and MLP-s to be universal function approximators). Ultimately, the described setup provides a truly nonparametric way to learn aleatoric uncertainty for neural network regression.

4.3 Weighted CRPS Loss

The simple architecture described in Section 4.2 needs to model distributions by not only covering the full range of possible outputs depending on the injected randomness, but also making sure that potential outputs get sampled proportionally from the target distribution. Consequently, output regions with small probability get sampled infrequently during training, possibly leading to the model being unable to learn them in detail. Further, neural networks have an inherent bias towards learning smooth mappings from input to output, leading to the learned distribution possibly becoming overly smoothed when the network is limited in its capacity or cannot be trained until convergence (i.e., early stopping is used), both of which are common in practical applications.

An implicit bias towards smooth functions can be useful in certain applications, however it is also a limitation of the simple unweighted architecture. To improve the expressivity of the network, we introduce the weighted-sample-based CRPS loss (weighted CRPS for short), along with a modified architecture which produces a weighted sample as its output. By allowing a weighted sample, we are essentially allowing the model to draw samples $(\hat{y}_i)$ from distribution Q' instead of Q, along with the change-of-measure coefficients $w = \frac{\mathrm{d}Q}{\mathrm{d}Q'}$ (also known as likelihood-ratio or importance weight).

We derive the unbiased approximation of the CRPS in Appendix B.1 from a weighted sample as

$$\mathrm{CRPS}_w(F^w_{\mathrm{ECDF}}, y) = \frac{1}{n}\sum_{i=1}^{n} w_i|y-\hat{y}_i| - \frac{1}{2n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j|\hat{y}_i-\hat{y}_j|, \tag{7}$$

where $w_i = \frac{\mathrm{d}Q}{\mathrm{d}Q'}(\hat{y}_i)$, and F^w_{ECDF} denotes the empirical CDF of the weighted sample $(\hat{y}_i, w_i)$. We also derive an algebraically equivalent unbiased $\mathcal{O}(n \log n)$ formula, analogous to Equation (6), as

$$\mathrm{CRPS}'_w(F^w_{\mathrm{ECDF}}, y) = \frac{2}{n(n-1)}\sum_{i=1}^{n} w_{(i)}\big(\hat{y}_{(i)}-y\big)\left((n-1)\,\mathbb{1}\{y<\hat{y}_{(i)}\}-s_i+\frac{W-n+w_{(i)}+1}{2}\right), \tag{8}$$

denoting $W = \sum_{i=1}^{n} w_{(i)}$ and $s_i = \sum_{j=1}^{i} w_{(j)}$. See Appendix B.2 and B.3 for proof.

4.4 Weighted sampling architectures

While the weighted loss may in itself enhance model expressivity, its more significant impact is enabling advanced network architectures. A simple architecture must evaluate the non-deterministic

network component to generate new samples, requiring non-negligible compute time for larger sample sizes during both training and prediction. The weighted formula improves sampling efficiency by allowing the network to produce multiple weighted samples in each evaluation (see Figure 1c). Without weighting, such a network would be problematic, as all samples from the same evaluation would share the same frequency in the overall sample.

The weighted formula can leverage multiple such *sampling heads* while still ultimately being capable of representing arbitrary distributions. This increases expressivity, as each head samples from distinct parts of the output distribution, modeling simpler distributions individually. However, early stopping can cause transitions between heads to be non-seamless. A convenient solution is to apply the loss function separately to multiple sets of heads (see Figure 1d), resulting in heads with overlapping ranges. This approach resembles an in-network ensemble, however to avoid confusion with traditional ensembling, we refer to it as *layered sampling heads*. The synthetic task in Section 5.1 serves as a good visual illustration of the effects of different sampling architectures.

4.5 Multivariate Regression

The formulas described so far only apply to univariate targets. Fortunately, a multivariate generalization of the CRPS exists, called the *energy score* (Gneiting et al., 2008; Jordan et al., 2019):

$$\mathrm{ES}(F, y) = \frac{1}{n} \sum_i \|y - \tilde{y}_i\|_2 - \frac{1}{2n^2} \sum_i \sum_j \|\tilde{y}_i - \tilde{y}_j\|_2, \tag{9}$$

with the same caveats as Equation (5) regarding complexity and bias. While unbiased and weighted versions can be calculated analogously, the $\mathcal{O}(n \log n)$ formula is only valid for the univariate case. Potential ways to speed up the calculation do exist, see for example Hertrich et al. (2024). However, investigating the multivariate case in detail is out of scope for this work. Nevertheless, we do conduct experiments on multivariate problems using Equation (9) in its quadratic form, see Section 5.2.

4.6 Epistemic Uncertainty

The primary goal of our approach is to learn the aleatoric uncertainty of the target variable Y in a frequentist manner, without the use of Bayesian techniques. Although our method could be extended by mixing samples from a prior distribution into the training data as a way to mimic Bayesian priors, as it is sometimes done for example with out-of-distribution data (Malinin, 2019; Lakshminarayanan et al., 2017; Chen et al., 2018; Hafner et al., 2020), we do not attempt such approaches in this paper. Still, the benchmark used in Section 5.3 involves a large number of relatively small datasets, making it essential to account for epistemic uncertainty. Fortunately, the ensemble-based method of Lakshminarayanan et al. (2017) combines well with our proposed approach, as the output of multiple independently trained models can be mixed by simply pooling the samples.

4.7 Likelihood Estimation

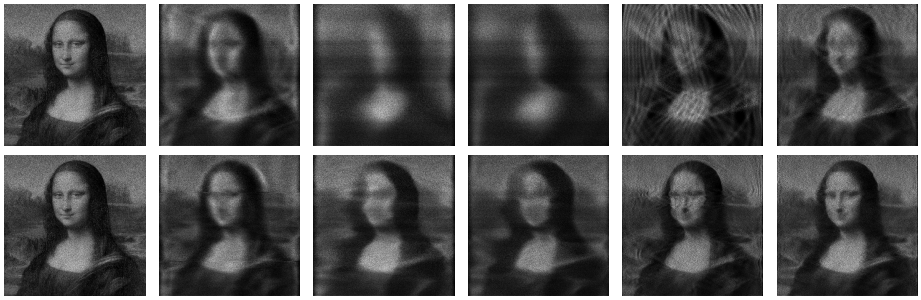
Since our method represents the output distribution through producing samples, it is very easy to estimate, e.g., percentiles. On the other hand, without an explicit density, it is nontrivial to evaluate likelihood at a given point. However, we still need to do so to compare against prior work, which generally evaluate methods using the negative log-likelihood metric (see Section 5).

In general, a sample-based unbiased estimator for the likelihood can not exist (Rosenblatt, 1956). However, the literature on *nonparametric density estimation* (Izenman, 1991) is vast, with many methods not only guaranteeing convergence, but providing useful estimates at smaller sample sizes. Given the ease of sampling in our case, the lack of explicit density in theory becomes a non-issue. In practice, we found Wang & Wang (2007) to work well together with out method, see Appendix C.1.

5 Experiments

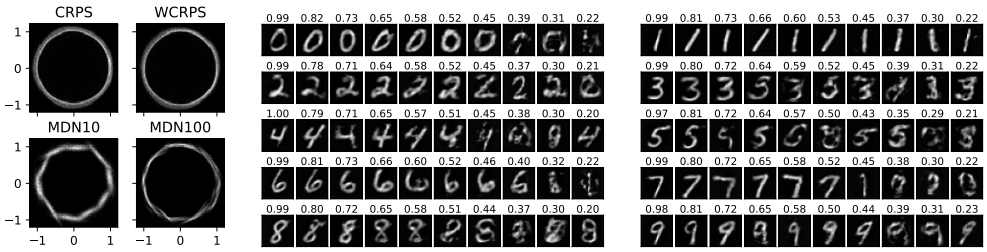
5.1 Univariate Regression on Synthetic Tasks

In theory, both MDN and our new CRPS-based networks can learn to represent arbitrary univariate distributions. To test their expressiveness visually, we train both methods to learn a synthetic



(a) Original (b) MDN (c) Simple (d) Weighted (e) MH (f) LMH

Figure 2: Learning the Mona Lisa as output distribution after 40 epochs (top) and after 2000 epochs (bottom). Each row of pixels in the images displays the heatmap of a predicted density based on the row number as input. MH stands for *multiple heads*, while LMH stands for *layered multiple heads*.



(a) Inverting $z^2 = x^2 + y^2$ (b) Modeling the distribution of MNIST pixel data from the label.

Figure 3: Multivariate tasks. Image (a) shows 2D densities, while (b) displays random samples.

distribution $x \to p(y)$, where we sample (x, y) points based on the luminance of an image of the Mona Lisa with x as row, and y as column index. As a result, the methods are required to learn many different, but continuously changing distributions, for each row of the image. Note that during training, the models only ever see separate (x, y) pairs at a time, i.e., not $(x; y_1, \ldots, y_n)$ samples.

We train multiple models of the same size, including an MDN model with 100 components, along with multiple kinds of CRPS-based models, and plot the output distributions as 2D heatmaps on Figure 2 after training for 40 and 2000 epochs. The MDN model learns quickly, however ends up with an overly smooth representation. Both single-head CRPS models fail to ultimately represent the details of the image, with perhaps the weighted version performing slightly better. The multi-head and layered-multi-head model both learn fast and manage to learn the baseline distribution in high-detail. However, without layering, visible artifacts remain even after 2000 epochs. Ultimately, the layered-multi-head variant learns fast and ends up representing the distribution with high-accuracy at the same time. See Appendix C.3 for samples during training and experimental setup.

5.2 Multivariate Regression

For multivariate regression, we run two experiments, one with synthetic and one with real-world data. For exact experimental setups, please refer to Appendix C.4, or the code repository. First, we train the networks to produce solutions to the equation $z^2 = x^2 + y^2$ with z as input and the joint distribution of (x, y) as output. We plot the resulting distributions for $z = 1$ in Figure 3a for four models of the same size: a simple (CRPS) and a weighted (WCRPS) CRPS-based model, and MDN models with 10 (MDN10) and 100 components (MDN100). As we can observe, all models learn to invert the equation, however the CRPS-based models provide a much more robust contour of the resulting circle-shaped distribution.

Having successfully learned the joint distribution of 2 variables, we now turn to a much more challenging task: learning the joint distribution of 784 variables. We reverse the input-output structure of the MNIST classification task (LeCun, 1998), turning it into a multivariate regression problem, with

the label as input and image pixels as output, similar to generative modeling, however without the use of locality structures. Note that we model MNIST solely to stress-test the multivariate formula, not for actual generative modeling purposes. In Figure 3b, we display samples from the predicted distributions. Since multivariate likelihood is hard to estimate, we instead indicate the likelihood of the sampled latent noise corresponding to the sampled outputs. Remarkably, the model is able to learn the joint distribution of the 784 output variables, producing coherent hand-written digits.

5.3 UCI Regression Uncertainty Benchmark and Baselines

The UCI Benchmark (Hernández-Lobato & Adams, 2015) has become the standard benchmark to measure regression uncertainty, with many papers reporting RMSE and (negative) log-likelihood (NLL) results along with standard error. We run our experiments on the exact train-test splits defined by Hernández-Lobato & Adams (2015), using the setup and code from the repository of Gal & Ghahramani (2016). See Appendix C.5 for discussion of hyperparameters and optimization.

Strongest baselines are Sun et al. (2017) (**PBP-MV**) for BNN, and Gal & Ghahramani (2016) (**Dropout**), covering best overall NLL from the literature (Mukhoti et al., 2018), except for boston ($\mathbf{2.33}$, El-Laham et al. (2023)), naval ($-\mathbf{5.64}$, Lakshminarayanan et al. (2017)) and yacht ($\mathbf{1.10}$, Springenberg et al. (2016)). We run our experiments with the network sizes of Sun et al. (2017) (PBP-MV), using two hidden layers with 50 neurons each. We reimplement the method of Lakshminarayanan et al. (2017) (**Ensembles**) and evaluate it at the same network size. For **MDN**, we use our own robust implementation, achieving much stronger scores then previously reported, see Appendix C.2. Baselines compute NLL analytically, while CRPS-based methods are evaluated as described in Section 4.7. For more discussion of baselines, please refer to Appendix C.6. Further comparison against simpler models is presented in Appendix C.11. We compare against Bouchacourt et al. (2016) in Appendix C.13 and against Depeweg et al. (2018) in Appendix C.14.

5.3.1 Results

We benchmark multiple variants of our CRPS-based networks, including versions with unweighted, weighted, multi-head, layered-multi-head versions, with and without ensembling. The overall best-performing variant turns out to be the single-head weighted version with ensembling (WCRPS_e) on Tables 1 and 2, where we present negative log-likelihood (NLL) and root mean square error (RMSE) scores along with baseline results. Scores without ensembling are also included (WCRPS). The most direct competition for WCRPS_e and WCRPS are MDN_{bnn} and MDN respectively, i.e., non-Gaussian methods with and without techniques for epistemic uncertainty. Best scores are highlighted in **bold**, however one should also note the error ranges included with scores when reading the results.

Overall, it is clear that relaxing the Gaussian assumption generally helps methods achieve higher scores. In NLL, only Dropout manages to score a victory out of the classic baselines. The rest of the strongest scores are shared between MDN_{bnn} and WCRPS_e, with the latter ultimately scoring best overall. We present evaluations for variants of CRPS-based methods in an ablation study in Appendix C.7, revealing that the method is also capable of achieving the best NLL result on the *concrete* dataset, with a score of $\mathbf{2.73}$ (MH variant with ensembling). Multi-head variants underperform WCRPS_e on many (though not all) datasets, indicating that the single-head architecture's strong implicit bias towards smooth distributions appears to be an advantage in most cases.

The *wine* dataset is unique as it is in fact a classification task with 6 discrete target values (see Appendix C.10). MDN outperforms others by 5 orders of magnitude in likelihood by using separate mixture components with increasingly small variances on each value. Note that this approach can theoretically produce NLL values converging to $-\infty$ even while assigning equal probabilities to all outputs, thus offering no useful prediction. Hence, on discrete datasets, NLL is an unreliable metric.

In RMSE, WCRPS_e dominates, with only a small advantage of MDN_{bnn} on *yacht*. As an explanation for the superior performance, while CRPS includes the mean absolute error (MAE) explicitly, all other methods optimize for NLL, which only indirectly optimizes the expectation of the prediction.

Total running time depends not only on raw numeric computation, but also on factors like batch size, data access patterns, convergence speed, and evaluation speed for early stopping. To provide a sense of relative speed, we report running time on the full UCI benchmark suite, which includes datasets with varying characteristics. With MDN at just a bit over 1.5 hours, WCRPS completes in about

Table 1: NLL results on the UCI Datasets benchmark (lower is better)

dataset	n	Dropout	PBP-MV	Ensembles	MDN	MDN_{bnn}	WCRPS	WCRPS_{e}
boston	0.5k	2.40±0.04	2.54±0.08	2.44±0.05	2.52±0.04	2.36±0.03	2.40±0.05	**2.32**±0.05
concrete	1.0k	**2.94**±0.02	3.04±0.03	2.97±0.04	3.13±0.04	2.99±0.03	3.09±0.04	2.95±0.05
energy	0.8k	1.21±0.01	1.01±0.01	0.62±0.08	0.77±0.04	0.65±0.02	0.47±0.04	**0.31**±0.03
kin8nm	8.2k	-1.14±0.01	-1.28±0.01	-1.34±0.00	-1.22±0.01	-1.30±0.01	-1.32±0.01	**-1.38**±0.01
naval	11.9k	-4.45±0.00	-4.85±0.06	-6.49±0.02	-6.24±0.04	**-6.72**±0.01	-6.65±0.05	-6.51±0.03
power	9.6k	2.81±0.01	2.78±0.01	2.69±0.01	2.65±0.01	2.62±0.01	2.66±0.01	**2.57**±0.01
protein	45.7k	2.87±0.00	2.77±0.01	2.66±0.02	2.00±0.01	**1.96**±0.01	2.22±0.01	2.05±0.01
wine	1.6k	0.93±0.01	0.97±0.01	0.91±0.01	-3.48±0.03	**-5.11**±0.09	0.90±0.03	0.80±0.03
yacht	0.3k	1.25±0.02	1.64±0.02	-0.02±0.04	0.26±0.09	0.63±0.04	0.07±0.06	**-0.16**±0.05

Table 2: RMSE results on the UCI Datasets benchmark (lower is better)

dataset	n	Dropout	PBP-MV	Ensembles	MDN	MDN_{bnn}	WCRPS	WCRPS_{e}
boston	0.5k	3.61±0.23	3.11±0.15	3.37±0.17	3.73±0.25	2.93±0.20	3.07±0.23	**2.91**±0.18
concrete	1.0k	5.45±0.19	5.08±0.14	5.19±0.19	5.99±0.12	5.25±0.11	5.51±0.14	**4.94**±0.17
energy	0.8k	0.97±0.06	0.45±0.01	0.86±0.12	1.17±0.09	0.51±0.01	0.45±0.02	**0.41**±0.01
kin8nm	8.2k	0.09±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	**0.06**±0.00
naval[2]	11.9k	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
power	9.6k	4.18±0.04	3.91±0.04	3.73±0.03	3.93±0.04	3.86±0.04	3.79±0.04	**3.62**±0.04
protein	45.7k	4.39±0.02	3.94±0.02	4.20±0.03	3.96±0.02	4.09±0.02	3.76±0.02	**3.47**±0.02
wine	1.6k	0.66±0.01	0.64±0.01	**0.63**±0.00	0.66±0.01	0.66±0.01	0.64±0.01	**0.63**±0.01
yacht	0.3k	1.23±0.37	0.81±0.06	0.92±0.08	1.97±0.25	**0.77**±0.07	0.96±0.13	0.78±0.09

2.1×, Ensembles in 3.4×, $\text{WCRPS}_{\text{ens}}$ in 7×, and MDN_{bnn} in 17.1× time, the latter being impacted by both slower per-epoch computation and slow convergence. See Appendix C.8 for details.

5.3.2 Calibration

Calibration is measured as described in Kuleshov et al. (2018). We take 1000 samples from the predicted distribution, take every 10th as an estimate of percentiles, and measure the ratio of test samples for which the ground truth falls below them. See Figure 4, with the ideal calibration as a dashed red line.

The method displays good calibration on most datasets, except yacht and naval, where it is underconfident. However, on both datasets, the underconfidence is also reflected on the train set (see Appendix C.9), implying that it is most likely a shortcoming of training, e.g., too large learning rate.

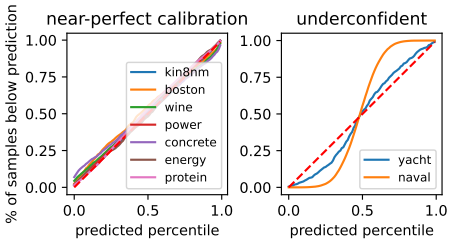


Figure 4: Calibration of WCRPS_{e} on the UCI Datasets benchmark.

6 Conclusions

In this paper, we proposed optimizing nondeterministic neural networks using loss functions derived from the continuous ranked probability score (CRPS). We derived unbiased $\mathcal{O}(n \log n)$ formulas for both unweighted and weighted sample-based approximations of the CRPS, and proposed multiple architectures for sampling from the aleatoric distribution of the target, enabling truly distribution-free learning. Through extensive experiments on both univariate and multivariate synthetic and real-world datasets, our method consistently demonstrates superior performance compared to baseline models, making it an attractive choice for capturing aleatoric uncertainty. Ultimately, we believe CRPS optimization is an effective and novel way of training neural networks for predicting regression uncertainty in a distribution-free manner, while at the same time remaining conceptually simple and easy to implement. Finally, we make our implementations and experiment code freely available.

[2]We have no exact baseline RMSE scores for *naval*, as prior work reports only 2 decimals (Sun et al., 2017).

REPRODUCIBILITY

We make all code and hyperparameters used to run the experiments in the paper publicly available in our source code repository at https://anonymous.4open.science/r/crps_iclr-B270.

REFERENCES

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

Sam Allen. Weighted scoringrules: Emphasising particular outcomes when evaluating probabilistic forecasts. *arXiv preprint arXiv:2305.07312*, 2023.

Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *Advances in neural information processing systems*, 33:14927–14937, 2020.

Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Ludwig Baringhaus and Carsten Franz. On a new multivariate two-sample test. *Journal of multivariate analysis*, 88(1):190–206, 2004.

Viktor Bengs, Eyke Hüllermeier, and Willem Waegeman. On second-order scoring rules for epistemic uncertainty quantification. In *International Conference on Machine Learning*, pp. 2078–2091. PMLR, 2023.

Jonathan Berrisch and Florian Ziel. Crps learning. *Journal of Econometrics*, 237(2, Part C):105221, 2023. ISSN 0304-4076. doi: https://doi.org/10.1016/j.jeconom.2021.11.008. URL https://www.sciencedirect.com/science/article/pii/S0304407621002724.

Christopher M Bishop. Mixture density networks. 1994.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Diane Bouchacourt, Pawan K Mudigonda, and Sebastian Nowozin. Disco nets: Dissimilarity coefficients networks. *Advances in Neural Information Processing Systems*, 29, 2016.

Axel Brando. Mixture density networks (mdn) for distribution and uncertainty estimation, 2017. URL https://github.com/axelbrando/Mixture-Density-Networks-for-distribution-and-uncertainty-estimation/blob/master/ABrando-MDN-MasterThesis.pdf. Report of the Master's Thesis: Mixture Density Networks for distribution and uncertainty estimation.

Wenhu Chen, Yilin Shen, William Yang Wang, and Hongxia Jin. A variational dirichlet framework for out-of-distribution detection. *arXiv: Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:67856169.

Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh. Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6915–6922. IEEE, 2018.

Adrian Corduneanu and Christopher M Bishop. Variational bayesian model selection for mixture distributions. In *Artificial intelligence and Statistics*, volume 2001, pp. 27–34. Morgan Kaufmann Waltham, MA, 2001.

Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 international conference on robotics and automation (icra)*, pp. 2090–2096. IEEE, 2019.

Peng Cui, Wenbo Hu, and Jun Zhu. Calibrated reliable regression using maximum mean discrepancy. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17164–17175. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c74c4bf0dad9cbae3d80faa054b7d8ca-Paper.pdf.

Joseph Curro and John Raquet. Deriving confidence from artificial neural networks for navigation. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pp. 1351–1361, 2018. doi: 10.1109/PLANS.2018.8373526.

Claudia Czado, Tilmann Gneiting, and Leonhard Held. Predictive model assessment for count data. *Biometrics*, 65(4):1254–1261, 2009. ISSN 0006341X, 15410420. URL http://www.jstor.org/stable/20640646.

A Philip Dawid and Paola Sebastiani. Coherent dispersion criteria for optimal experimental design. *Annals of Statistics*, pp. 65–81, 1999.

Bhargob Deka, Luong Ha Nguyen, and James-A. Goulet. Analytically tractable heteroscedastic uncertainty quantification in bayesian neural networks for regression tasks. *Neurocomputing*, 572:127183, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127183. URL https://www.sciencedirect.com/science/article/pii/S0925231223013061.

Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International conference on machine learning*, pp. 1184–1193. PMLR, 2018.

Chao Du, Tianbo Li, Tianyu Pang, Shuicheng Yan, and Min Lin. Nonparametric generative modeling with conditional sliced-wasserstein flows, 2023.

Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization, 2015.

Yousef El-Laham, Niccolò Dalmasso, Elizabeth Fons, and Svitlana Vyetrenko. Deep gaussian mixture ensembles. In *Uncertainty in Artificial Intelligence*, pp. 549–559. PMLR, 2023.

Christopher AT Ferro, David S Richardson, and Andreas P Weigel. On the effect of ensemble size on the discrete and continuous ranked probability scores. *Meteorological Applications: A journal of forecasting, practical applications, training techniques and modelling*, 15(1):19–24, 2008.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56(Suppl 1):1513–1589, 2023.

Manuel Gebetsberger, Jakob W. Messner, Georg J. Mayr, and Achim Zeileis. Estimation methods for nonhomogeneous regression models: Minimum continuous ranked probability score versus maximum likelihood. *Monthly Weather Review*, 146(12):4323 – 4338, 2018. doi: 10.1175/MWR-D-17-0364.1. URL https://journals.ametsoc.org/view/journals/mwre/146/12/mwr-d-17-0364.1.xml.

Soumya Ghosh, Jiayu Yao, and Finale Doshi-Velez. Model selection in bayesian neural networks via horseshoe priors. *Journal of Machine Learning Research*, 20(182):1–46, 2019.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Tilmann Gneiting and Roopesh Ranjan. Comparing density forecasts using threshold-and quantile-weighted scoring rules. *Journal of Business & Economic Statistics*, 29(3):411–422, 2011.

Tilmann Gneiting, Larissa I Stanberry, Eric P Grimit, Leonhard Held, and Nicholas A Johnson. Assessing probabilistic forecasts of multivariate quantities, with an application to ensemble predictions of surface winds. *Test*, 17:211–235, 2008.

Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018*, pp. 45–87, 2020.

Irving John Good. Rational decisions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 14(1):107–114, 1952.

James-A Goulet, Luong Ha Nguyen, and Saeid Amiri. Tractable approximate gaussian inference for bayesian neural networks. *Journal of Machine Learning Research*, 22(251):1–23, 2021.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL `http://arxiv.org/abs/1308.0850`.

Danijar Hafner, Dustin Tran, Timothy Lillicrap, Alex Irpan, and James Davidson. Noise contrastive priors for functional uncertainty. In *Uncertainty in Artificial Intelligence*, pp. 905–914. PMLR, 2020.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pp. 1861–1869. PMLR, 2015.

Johannes Hertrich, Christan Wald, Fabian Altekrüger, and Paul Hagemann. Generative sliced MMD flows with Riesz kernels. In *The Twelfth International Conference on Learning Representations*, 2024.

Lars U Hjorth and Ian T Nabney. Regularisation of mixture density networks. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470)*, volume 2, pp. 521–526. IET, 1999.

Christian Hubschneider, Robin Hutmacher, and J Marius Zöllner. Calibrating uncertainty models for steering angle estimation. In *2019 IEEE intelligent transportation systems conference (ITSC)*, pp. 1511–1518. IEEE, 2019.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506, 2021.

Alan Julian Izenman. Review papers: Recent developments in nonparametric density estimation. *Journal of the american statistical association*, 86(413):205–224, 1991.

Alexander Jordan, Fabian Krüger, and Sebastian Lerch. Evaluating probabilistic forecasts with scoringrules. *Journal of Statistical Software*, 90(12):1–37, 2019. doi: 10.18637/jss.v090.i12. URL `https://www.jstatsoft.org/index.php/jss/article/view/v090i12`.

Alexander I Jordan. *Facets of forecast evaluation*. PhD thesis, Dissertation, Karlsruhe, Karlsruher Institut für Technologie (KIT), 2016, 2016.

Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

H M Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE Access*, PP, 06 2018. doi: 10.1109/access.2018.2836917.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5580–5590, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Fabian Krüger, Sebastian Lerch, Thordis Thorarinsdottir, and Tilmann Gneiting. Predictive inference based on markov chain monte carlo output. *International Statistical Review*, 89(2):274–301, 2021.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pp. 2796–2804. PMLR, 2018.

Francesco Laio and Stefania Tamea. Verification tools for probabilistic forecasts of continuous hydrological variables. *Hydrology and Earth System Sciences*, 11(4):1267–1277, 2007.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Nils Lehmann, Jakob Gawlikowski, Adam J Stewart, Vytautas Jancauskas, Stefan Depeweg, Eric Nalisnick, and Nina Maria Gottschling. Lightning uq box: A comprehensive framework for uncertainty quantification in deep learning. *arXiv preprint arXiv:2410.03390*, 2024.

Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks, 2015.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1708–1716, New York, New York, USA, 20–22 Jun 2016. PMLR.

Martin Magris and Alexandros Iosifidis. Bayesian learning for neural networks: an algorithmic survey. *Artificial Intelligence Review*, 56(10):11773–11823, 2023.

Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7144–7153, 2019.

Andrey Malinin. *Uncertainty estimation in deep learning with application to spoken language assessment*. PhD thesis, 2019.

Safa Messaoud, David Forsyth, and Alexander G Schwing. Structural consistency and controllability for diverse colorization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 596–612, 2018.

Jishnu Mukhoti, Pontus Stenetorp, and Yarin Gal. On the importance of strong baselines in bayesian deep learning. *arXiv preprint arXiv:1811.09385*, 2018.

Allan H Murphy. The ranked probability score and the probability score: A comparison. *Monthly Weather Review*, 98(12):917–924, 1970.

Thomas Muschinski, Georg J. Mayr, Thorsten Simon, Nikolaus Umlauf, and Achim Zeileis. Cholesky-based multivariate gaussian regression. *Econometrics and Statistics*, 29:261–281, 2024. ISSN 2452-3062. doi: https://doi.org/10.1016/j.ecosta.2022.03.001. URL `https://www.sciencedirect.com/science/article/pii/S2452306222000168`.

Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.

Romain Pic, Clément Dombry, Philippe Naveau, and Maxime Taillardat. Distributional regression and its evaluation with the crps: Bounds and convergence of the minimax risk. *International Journal of Forecasting*, 39(4):1564–1572, October 2023. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2022.11.001. URL `http://dx.doi.org/10.1016/j.ijforecast.2022.11.001`.

C Radhakrishna Rao. Diversity and dissimilarity coefficients: a unified approach. *Theoretical population biology*, 21(1):24–43, 1982.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003.

Stephan Rasp and Sebastian Lerch. Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review*, 146(11):3885–3900, 2018.

Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956. doi: 10.1214/aoms/1177728190. URL https://doi.org/10.1214/aoms/1177728190.

Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE international conference on computer vision*, pp. 3591–3600, 2017.

Bernhard Schölkopf. The kernel trick for distances. In T. Leen, T. Dietterich, and V. Tresp (eds.), *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/4e87337f366f72daa424dae11df0538c-Paper.pdf.

Dino Sejdinovic, Arthur Gretton, Bharath Sriperumbudur, and Kenji Fukumizu. Hypothesis testing using pairwise distances and associated kernels (with appendix), 2012.

Marcin Sendera, Jacek Tabor, Aleksandra Nowak, Andrzej Bedychaj, Massimiliano Patacchiola, Tomasz Trzcinski, Przemysław Spurek, and Maciej Zieba. Non-gaussian gaussian processes for few-shot regression. *Advances in Neural Information Processing Systems*, 34:10285–10298, 2021.

Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

SiriTeam. Deep Learning for Siri's Voice: On-device Deep Mixture Density Networks for Hybrid Unit Selection Synthesis, 2017. URL https://machinelearning.apple.com/research/siri-voices.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning Structured Weight Uncertainty in Bayesian Neural Networks. In Aarti Singh and Jerry Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1283–1292. PMLR, 20–22 Apr 2017.

Gábor J Székely and Maria L Rizzo. A new test for multivariate normality. *Journal of Multivariate Analysis*, 93(1):58–80, 2005.

Robert Taggart. *Estimation of CRPS for precipitation forecasts using weighted sums of quantile scores and Brier scores*. Bureau of Meteorology, 2023.

Jean Thorey, Vivien Mallet, and Paul Baudin. Online learning with the continuous ranked probability score for ensemble forecasting. *Quarterly Journal of the Royal Meteorological Society*, 143(702): 521–529, 2017.

Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):1–10, 2014.

Hristos Tyralis and Georgia Papacharalampous. A review of predictive uncertainty estimation with machine learning. *Artificial Intelligence Review*, 57(4):94, 2024.

Amirhossein Vahidi, Simon Schoßer, Lisa Wimmer, Yawei Li, Bernd Bischl, Eyke Hüllermeier, and Mina Rezaei. Probabilistic self-supervised learning via scoring rules minimization. *arXiv preprint arXiv:2309.02048*, 2023.

Dennis van der Meer, Pierre Pinson, Simon Camal, and Georges Kariniotakis. Crps-based online learning for nonlinear probabilistic forecast combination. *International Journal of Forecasting*, 2024.

Bin Wang and Xiaofeng Wang. Bandwidth selection for weighted kernel density estimation. *arXiv preprint arXiv:0709.1616*, 2007.

Erik Zawadzki and Sebastien Lahaie. Nonparametric scoring rules. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Mar. 2015. doi: 10.1609/aaai.v29i1.9696. URL https://ojs.aaai.org/index.php/AAAI/article/view/9696.

A BACKGROUND

A.1 LIST OF PRIOR WORK EVALUATING ON THE UCI REGRESSION UNCERTAINTY BENCHMARK

The UCI Benchmark was originally defined by Hernández-Lobato & Adams (2015). We are aware of results being reported for the benchmark in Louizos & Welling (2016); Springenberg et al. (2016); Gal & Ghahramani (2016); Sun et al. (2017); Lakshminarayanan et al. (2017); Mukhoti et al. (2018); Ghosh et al. (2019); Amini et al. (2020); Goulet et al. (2021); El-Laham et al. (2023); Gawlikowski et al. (2023); Deka et al. (2024).

A.2 SMALL SURVEY OF UNCERTAINTY SURVEYS

To provide evidence for our claim that the distribution-free modeling of aleatoric uncertainty for neural network regression is a relatively under-researched and under-reported area, we conduct a small survey, where we list uncertainty-adjacent survey papers and examine them based on four criteria:

- **MDN** Whether the survey mentions the use of mixture models for aleatoric uncertainty in regression, cites MDN (Bishop, 1994), or other works using MDN
- **NF** Whether the survey mentions the use of Normalizing Flows for aleatoric uncertainty
- **MMD** Whether the survey mentions the Maximum Mean Discrepancy loss in the context of learning aleatoric uncertainty
- **ES** Whether the survey mentions the Energy score / CRPS in the context of aleatoric uncertainty or cites DISCO Nets (Bouchacourt et al., 2016)

We summarize our findings in Table 3. To elaborate further on each:

- Goan & Fookes (2020) is a survey of Bayesian Neural Networks, and as such is primarily concerned with epistemic uncertainty. However, it does cite SiriTeam (2017) which uses MDN.
- Abdar et al. (2021) is a general survey of uncertainty techniques in deep learning. In the main text, it describes MDN under the name *Gaussian Mixture Model* and cites Choi et al. (2018) which uses MDN.
- Kabir et al. (2018) is survey of uncertainty quantification in deep learning. Does not mention or cite any of the above.
- Hüllermeier & Waegeman (2021) is a general introduction to epistemic and aleatoric uncertainty in machine learning. Does not mention or cite any of the above.
- Jospin et al. (2022) is a tutorial on Bayesian Neural Networks, and (somewhat appropriately in this case) does not mention or cite any of the above
- Magris & Iosifidis (2023) is a survey on Bayesian learning for neural networks in general. Does not mention or cite any of the above.
- Gawlikowski et al. (2023) is a survey of uncertainty in deep neaural networks. Cites Corduneanu & Bishop (2001), therefore indirectly references MDN.
- Tyralis & Papacharalampous (2024) is a review of predictive uncertainty in machine learning. It does describe the CRPS, but mentions uses for learning only in context of ensembles or with a Gaussian assumption. It also mentions the Energy Score, but does not describe any uses. Does not cite Bouchacourt et al. (2016).

Many of these works describe using non-Gaussian distributions, however usually in the context of variational distributions, (i.e., the distribution of weights in a BNN). Some mention the use of MMD for quantile regression or calibration, but not for distributional regression.

Table 3: A small survey of uncertainty surveys.

Paper	MDN	NF	MMD	ES
Goan & Fookes (2020)	✓	-	-	-
Abdar et al. (2021)	✓	✓	-	-
Kabir et al. (2018)	-	-	-	-
Hüllermeier & Waegeman (2021)	-	-	-	-
Jospin et al. (2022)	-	-	-	-
Magris & Iosifidis (2023)	-	-	-	-
Gawlikowski et al. (2023)	✓	-	-	-
Tyralis & Papacharalampous (2024)	-	-	-	✓

B Proofs and Derivations

B.1 Weighted CRPS formula

The classic formula estimates the first term of Equation (4) by taking a samples $(\hat{y}_i)$ from $\hat{Y} \sim Q$ and calculating

$$\mathrm{E}\left|y - \hat{Y}_1\right| = \int_\Omega |y - \hat{y}| \,\mathrm{d}Q(\hat{y}) \approx \frac{1}{n}\sum_{i=1}^n |y - \hat{y}_i|. \tag{10}$$

If Q is absolutely continuous w.r.t. Q' (or $Q \ll Q'$), then given $Y' \sim Q'$ and its sample (y'_i),

$$\int_\Omega |y - \hat{y}| \,\mathrm{d}Q(\hat{y}) = \int_\Omega |y - \hat{y}| \tfrac{\mathrm{d}Q}{\mathrm{d}Q'}(\hat{y}) \,\mathrm{d}Q'(\hat{y}) = \mathrm{E}[|y - Y'|w(Y')] \approx \frac{1}{n}\sum_{i=1}^n |y - y'_i| w(y'_i), \tag{11}$$

where $w(y') = \frac{\mathrm{d}Q}{\mathrm{d}Q'}(y')$. Doing the same for the second term:

$$\mathrm{E}\left|\hat{Y}_1 - \hat{Y}_2\right| = \int_\Omega\int_\Omega |\hat{y}_1 - \hat{y}_2| \,\mathrm{d}Q(\hat{y}_1)\,\mathrm{d}Q(\hat{y}_2) \tag{12}$$

$$= \int_\Omega\int_\Omega |\hat{y}_1 - \hat{y}_2| \tfrac{\mathrm{d}Q}{\mathrm{d}Q'}(\hat{y}_1)\,\mathrm{d}Q'(\hat{y}_1) \tfrac{\mathrm{d}Q}{\mathrm{d}Q'}(\hat{y}_2)\,\mathrm{d}Q'(\hat{y}_2) \tag{13}$$

$$= \mathrm{E}[|Y'_1 - Y'_2| w(Y'_1) w(Y'_2)] \approx \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n |y'_i - y'_j| w(y'_i) w(y'_j), \tag{14}$$

with $Y'_1, Y'_2 \sim Q'$ iid. Finally,

$$\mathrm{CRPS}(Q, y) \approx \mathrm{CRPS}_w(F^w_{\mathrm{ECDF}}, y) = \frac{1}{n}\sum_{i=1}^n w_i|y - \hat{y}_i| - \frac{1}{2n^2}\sum_{i=1}^n\sum_{j=1}^n w_i w_j |\hat{y}_i - \hat{y}_j|, \tag{15}$$

where w_i further denotes $w(\hat{y}_i)$, and F^w_{ECDF} denotes the empirical cumulative distribution function of the weighted sample $(\hat{y}_i, w_i)$. An unbiased version can be computed as in Section 4.1 by replacing the $\frac{1}{2n^2}$ coefficient by $\frac{1}{2n(n-1)}$.

B.2 Fast biased version

First we are going to prove a formula for the biased version for the sake of simplicity, and describe how to get an unbiased version in Appendix B.3. The following formula can be calculated in $\mathcal{O}(n \log n)$ time by first ordering the sample to get $\hat{y}_{(1)}, \ldots, \hat{y}_{(n)}$ and corresponding weights $w_{(1)}, \ldots, w_{(n)}$.

Proposition B.1. *Let $W = \sum_{i=1}^n w_{(i)}$ and $s_i = \sum_{j=1}^i w_{(j)}$. Then*

$$\mathrm{CRPS}_w(F^w_{ECDF}, y) = \frac{2}{n^2}\sum_{i=1}^n w_{(i)}(\hat{y}_{(i)} - y)\left(n\mathbb{1}\{y < \hat{y}_{(i)}\} - s_i + \frac{W - n + w_{(i)}}{2}\right) \tag{16}$$

Proof. Using counting method. First observe that

$$\frac{1}{n}\sum_{i=1}^{n} w_{(i)}|y-\hat{y}_{(i)}| - \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{(i)}w_{(j)}|\hat{y}_{(i)}-\hat{y}_{(j)}| \tag{17}$$

$$= -\frac{1}{n^2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{1}\{i<j\}\, w_{(i)}w_{(j)}|\hat{y}_{(i)}-\hat{y}_{(j)}| - n\sum_{i=1}^{n} w_{(i)}|y-\hat{y}_{(i)}|\right) \tag{18}$$

Next, we are going to count the $|\hat{y}_{(i)} - y|$ distances on Figure 5, where we visually display the summands and their weights from Equation (18) (omitting the preceding $(-\frac{1}{n^2})$ coefficient)

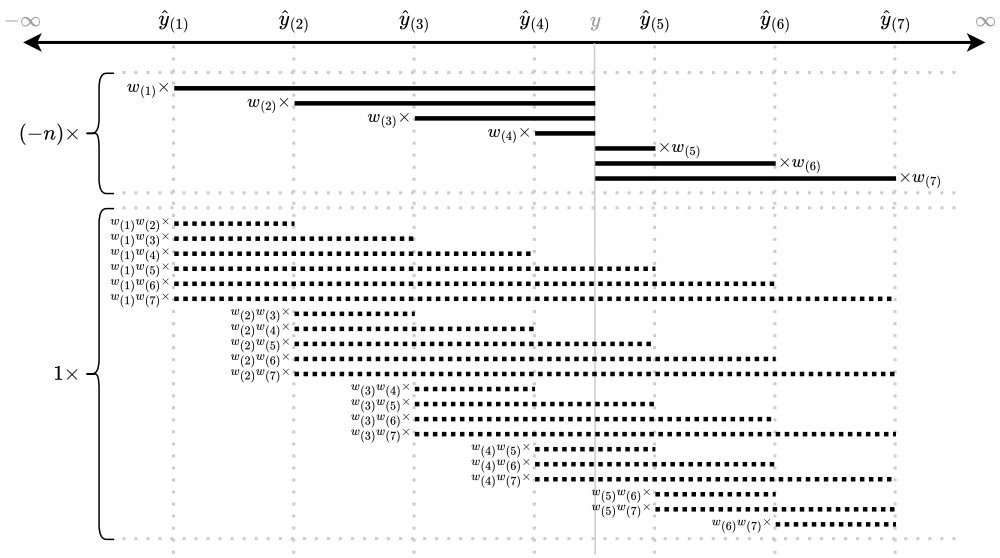


Figure 5: Counting method example with $n = 7$. We are counting the sum of coefficients for each $|\hat{y}_{(i)} - y|$ term. On the figure, solid lines are to be counted with a coefficient of $(-n)$ and dotted lines with a coefficient of 1. Further, individual change-of-measure coefficients are indicated next to each line in the form $w_{(i)}$ or $w_{(i)}w_{(j)}$. The global coefficient $(-\frac{1}{n^2})$ is omitted for simplicity.

Starting from the left with the example of $i = 1$, for $(y - \hat{y}_{(1)})$, we have the coefficients

$$-nw_{(1)} + w_{(1)}\sum_{j=2}^{n} w_{(j)}. \tag{19}$$

However, this way we over-count each $(y-\hat{y}_{(j)})$ term for $\{j > i,\ \hat{y}_{(j)} < y\}$ by a factor of $w_{(1)}w_{(j)}$, which we are going to correct for when counting the coefficients for $(y - \hat{y}_{(j)})$.

Accordingly, for the i-th term $(y - \hat{y}_{(i)})$ in general, assuming $\hat{y}_{(i)} < y$, we have the coefficients

$$-nw_{(i)} + w_{(i)}\sum_{j=i+1}^{n} w_{(j)} - w_{(i)}\sum_{j=1}^{i-1} w_{(j)} \tag{20}$$

$$= w_{(i)}\left(-n - \sum_{j=1}^{i-1} w_{(j)} + \left(W - \sum_{j=1}^{i} w_{(j)}\right)\right) \tag{21}$$

$$= w_{(i)}\left(W - n - 2s_i + w_{(i)}\right). \tag{22}$$

We can proceed to count the coefficients of $(\hat{y}_{(i)} - y)$ with $y < \hat{y}_{(i)}$ by mirroring the above logic. Denoting $k_i = W - \sum_{j=1}^{i-1} w_{(j)} = W - s_i + w_{(i)}$, we have the coefficients (cf. Equation (22))

$$w_i \left(W - n - 2k_i + w_{(i)}\right) = w_{(i)} \left(W - n - 2(W - s_i + w_{(i)})) + w_{(i)}\right) \tag{23}$$

$$= -w_{(i)}(W + n - 2s_i + w_{(i)}). \tag{24}$$

Finally, we have

$$\mathrm{CRPS}_w(F^w_{\mathrm{ECDF}}, y) \tag{25}$$

$$= -\frac{1}{n^2}\left(\sum_{\hat{y}_{(i)}<y} w_{(i)}\big(W-n-2s_i+w_{(i)}\big)(y-\hat{y}_{(i)}) - \sum_{y<\hat{y}_{(i)}} w_{(i)}\big(W+n-2s_i+w_{(i)}\big)\,(\hat{y}_{(i)} - y)\right) \tag{26}$$

$$= \frac{2}{n^2}\sum_{i=1}^{n} w_{(i)}(\hat{y}_{(i)}-y)\left(n\mathbb{1}\{y < \hat{y}_{(i)}\} - s_i + \frac{W - n + w_{(i)}}{2}\right). \tag{27}$$

□

B.3 Fast unbiased version

Proposition B.2. *An unbiased version of Equation* (16) *can also be calculated in* $\mathcal{O}(n \log n)$.

Proof.

$$\mathrm{CRPS}_w(F^w_{\mathrm{ECDF}}, y) = \frac{2}{n^2}\sum_{i=1}^{n} w_{(i)}(\hat{y}_{(i)} - y)\left(n\mathbb{1}\{y < \hat{y}_{(i)}\} - s_i + \frac{W - n + w_{(i)}}{2}\right) \tag{28}$$

$$= \frac{1}{n^2}\left(\sum_{y<\hat{y}_{(i)}} w_{(i)}\big(\hat{y}_{(i)} - y\big)\big(n-2s_i+W+w_{(i)}\big) + \sum_{\hat{y}_{(i)}<y} w_{(i)}\big(y - \hat{y}_{(i)}\big)\big(n+2s_i-W-w_{(i)}\big)\right) \tag{29}$$

Notice that

$$\frac{1}{n}\sum_{i=1}^{n} w_{(i)}|y - \hat{y}_{(i)}| = \frac{1}{n^2}\left(\sum_{y<\hat{y}_{(i)}} w_{(i)}\left(\hat{y}_{(i)} - y\right)n + \sum_{\hat{y}_{(i)}<y} w_{(i)}\left(y - \hat{y}_{(i)}\right)n\right), \tag{30}$$

and that Equation (30) appears in Equation (29). Therefore, we can modify Equation (29) to express the unbiased coefficient for the second term of Equation (7) without affecting the first term by taking

$$\frac{1}{n(n-1)}\Bigg(\sum_{y<\hat{y}_{(i)}} w_{(i)}\left(\hat{y}_{(i)} - y\right)\left((n-1) - 2s_i + W + w_{(i)}\right) + \tag{31}$$

$$+ \sum_{\hat{y}_{(i)}<y} w_{(i)}\left(y - \hat{y}_{(i)}\right)\left((n-1) + 2s_i - W - w_{(i)}\right)\Bigg) \tag{32}$$

$$= \frac{2}{n(n-1)}\Bigg(\sum_{y<\hat{y}_{(i)}} w_{(i)}\left(\hat{y}_{(i)} - y\right)\left(n - 1 - s_i + \frac{W - n + w_{(i)} + 1}{2}\right) + \tag{33}$$

$$+ \sum_{\hat{y}_{(i)}<y} w_{(i)}\left(\hat{y}_{(i)} - y\right)\left(-s_i + \frac{W - n + w_{(i)} + 1}{2}\right)\Bigg) \tag{34}$$

Transforming back to short form and denoting the unbiased version with CRPS'_w, we get

$$\mathrm{CRPS}'_w(F^w_{\mathrm{ECDF}}, y) = \frac{2}{n(n-1)}\sum_{i=1}^{n} w_{(i)}\big(\hat{y}_{(i)}-y\big)\left((n-1)\,\mathbb{1}\{y<\hat{y}_{(i)}\} - s_i + \frac{W-n+w_{(i)}+1}{2}\right). \tag{35}$$

□

B.4 Fast unbiased unweighted version

Proposition B.3.

$$\mathrm{CRPS}'(F_{ECDF}, y) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \left(\tilde{y}_{(i)} - y\right) \left((n-1)\mathbb{1}\{y < \tilde{y}_{(i)}\} - i + 1\right), \tag{36}$$

Proof. Taking $w_{(i)} \equiv 1$, we get $W = n$ and $s_i = i$. Substituting into Equation (35), the proposition immediately follows. □

C Implementation details and results

C.1 Kernel Density Estimation

As described, we use the adaptive-bandwidth kernel density estimator of Wang & Wang (2007) for likelihood-estimation in our experiments. We implement a GPU-based version of the algorithm, based on the C++ based cpu-implementation of the awKDE[3] Python library.

The method works by first creating a naive pilot likelihood estimate in each sampled point based on the full sample using a traditional constant bandwidth KDE, and then setting the local bandwith to be inversely proportional to the pilot-estimate. Calculating the pilot estimates requires calculating the pairwise distance matrix for all sampled points, making the algorithm quadratic in the number of points. Since calculating pairwise distances on a GPU is fast, the quadratic runtime proved to be feasible in our experiments. We run the algorithm with the default parameters for the original implementation of awKDE, i.e., using Silverman's estimate for the global bandwidth parameter (Silverman, 2018).

As Wang & Wang (2007) note, the quality of the pilot estimate is ultimately not critical for the quality of the final estimate. Accordingly, we have seen some success with using a fixed number of samples to produce the pilot-estimate, making the runtime linear instead of quadratic. However, proper investigation of such methods is out of scope for this work.

C.2 MDN Implementation

MDN models seem to be relatively under-utilized in uncertainty-related literature, with some surveys barely mentioning the method (Abdar et al., 2021; Goan & Fookes, 2020), and others omitting any mention at all (Gawlikowski et al., 2023; Jospin et al., 2022; Magris & Iosifidis, 2023; Kabir et al., 2018; Hüllermeier & Waegeman, 2021; Tyralis & Papacharalampous, 2024). Further, it appears that MDN has very rarely been evaluated as baseline in prior work. We are aware of a single instance where it is used as a baseline method (El-Laham et al., 2023) for the UCI benchmark, however with the authors reporting substantially lower scores than our experiments.

MDN is sometimes also referred to simply as Gaussian mixture model (GMM) (Hubschneider et al., 2019; Abdar et al., 2021), however in general these two terms do not mean the same thing, as GMM includes a larger class of methods with various modeling and optimization strategies. It is also important to note that not every Gaussian mixture model is made equal: in order to approximate even simple asymmetric densities, a high number of Gaussians might be required. Using methods relying on, e.g., ensembling (Jospin et al., 2022), prohibits the use of a high number of mixture components due to computational constraints.

The reason for its neglect is perhaps the fact that MDN appears to be notoriously challenging to implement, often suffering from numerical instability, failure to converge, and mode collapse (Makansi et al., 2019; Brando, 2017; Rupprecht et al., 2017; Cui et al., 2019; Curro & Raquet, 2018; Messaoud et al., 2018; Graves, 2013; Hjorth & Nabney, 1999). Nevertheless, we do implement and evaluate a robust variant of MDN, capable of optimizing the mixture of 100 Gaussian components in our experiments.

[3] https://github.com/mennthor/awkde

Our implementation takes a number of steps to avoid numerical instabilities and convergence issues that tend to plague implementations of MDN, this way achieving much stronger scores on the UCI benchmark then previously reported. We apply the following techniques:

- We standardize both the input variables X and the target Y to have 0 mean and a variance of 1.
- We apply a linear learning rate warm-up schedule, where the learning-rate is set to 0.1 of its value at the beginning and increases over 10 epochs.
- We apply gradient clipping at 10 before taking a step along the loss gradient.
- We calculate the log-likelihood in the log-domain as

$$l_i = -\frac{1}{2}\left(\frac{y-\mu_i}{\sigma_i}\right)^2 - \ln\left(\sigma_i\sqrt{2\pi}\right) \tag{37}$$

$$w_i = \pi_i - \mathrm{LSE}(\pi_1, \ldots, \pi_n) \tag{38}$$

$$\text{log-likelihood} = \mathrm{LSE}\left(l_1 + w_1, \ldots, l_n + w_n\right) \tag{39}$$

  where LSE denotes the numerically stable implementation of $\mathrm{LogSumExp}$ of the PyTorch framework[4], i.e.,

$$\mathrm{LSE}(x_1, \ldots, x_n) = \log \sum_{i=1}^{n} e^{x_i}. \tag{40}$$

- During training, we clamp the log-likelihood loss to over -20, as even a single evaluation where the likelihood at y is effectively zero can result in the experiment producing ∞ and nan values.
- We clamp the inputs of the $\mathrm{softplus}$ to over -15 and the input logits of the $\mathrm{softmax}$ between -15 and 15, as both tend to converge to very low and hight values:
  - as the method turns off certain components, their logits π_i tend towards $-\infty$;
  - if the method needs to represent an unimodal distribution, its logit π_i tends toward ∞;
  - on discrete datasets and in the case of unused components, the logit π_i as well as the input of the $\mathrm{softplus}$ s_i tends toward $-\infty$.

See Appendix C.4 for multivariate details.

C.3 Learning the Mona Lisa

For training, we sample 5000 samples for each row from the distribution defined by the luminance of an 500×500 image of the Mona Lisa, such that the input of the network is the row index, and the expected output is the column index. We train $\mathbb{R} \to \mathbb{R}$ networks with layers of size $[128, 512, 512, 1024]$, GELU activation, SGD optimizer with lr $= 10^{-4}$ and a momentum value of 0.8 for 2000 epochs. See Figure 6 for distributions at 40, 100, 200, 500, 1000 and 2000 epochs. The WCRPS-based method uses 50 heads for both multi-head variants, and 5 layers for the layered-multi-head variant.

C.4 Multivariate experimental setup

For the paraboloid task $z^2 = x^2 + y^2$ we generate training data by drawing 2M samples from $X, Y \sim \mathcal{U}(-2, 2)$, filter them to $x^2 + y^2 <= 4$, and calculate the corresponding z^2 values. We train $\mathbb{R} \to \mathbb{R}^2$ networks with layers of size $[64, 128, 128]$, GELU activation and LayerNorm, and use an SGD optimizer with 1k epochs and lr $= 10^{-4}$.

We implement a multivariate version of MDN with full covariance matrix modeling based on the Cholesky-decomposition based parameterization of the multivariate normal distribution (Muschinski et al., 2024), while also employing techniques from Appendix C.2 where applicable.

For the MNIST task, we train a network with layers of size $[16, 1024, 2048, 4096]$ with ReLU activation on the train set of MNIST for 1k epochs with lr $= 10^{-5}$ and a momentum value of 0.9

[4]https://pytorch.org/docs/stable/generated/torch.logsumexp.html

(a) Original (b) MDN (c) Simple (d) Weighted (e) MH (f) LMH

Figure 6: Learning the Mona Lisa as output distribution after 40, 100, 200, 500, 1000 and 2000 epochs from top to bottom respectively.

C.5 Hyperparameters on the UCI benchmark

We do not optimize hyperparameters individually for each dataset, however do use adaptive batch sizes, and larger L2 regularization for the 4 smallest datasets. Early stopping is used with 20% of the training set as validation.

In our experiments we use the *AdamW* optimizer with a learning-rate of 0.001 and a linear learning rate warm-up schedule starting with the coefficient 0.1 and ending with 1 after 10 epochs. We use early stopping with a patience value of 50 on validation sets consisting of 20% of the training set for each split. We set L2 regularization over the weights to 10^{-6} on all datasets except the four smallest ones (boston, concrete, energy, yacht), where we use 10^{-4}. Batch size is chosen to be $\max(\sqrt{m} - 5, 1)$ with m being the training set size, for a balance of computational efficiency and predictive accuracy.

We run Ensembles and ensembling-based variants of WCRPS using 5 networks, as in the original paper of Lakshminarayanan et al. (2017). We use 20 heads for all multi-head variants of WCRPS on the UCI benchmark, and 10 layers for all layered-multi-head variants. For evaluating the CRPS formula variants, we take 100 samples during training and 1000 samples for evaluation during early stopping. See Appendix C.12 for a comparison of different choices for the samples drawn during training.

C.6 Baselines

Most prior work run experiments by using 100 instead of 50 neurons for each hidden layer for the largest (protein) dataset. For simplicity, we use 50 neurons only, as the models seem to perform well regardless.

For our reimplementation of Ensembles, we also employ techniques from Appendix C.2 where applicable. Our results are generally in line with or slightly better than those reported in the original paper at the same network sizes, except for the naval dataset, where our implementation performs significantly better (-5.64 originally, -6.52 in our testing for the single-layer model). Our improvement may be due to one or more enhancements discussed in Appendix C.2, possibly influenced by the unusual data range, as also indicated by the RMSE values in Table 2.

Note, that the standard network size in most prior work is a single layer of 50 neurons. However, the best prior scores in Sun et al. (2017) (PBP-MV) use a network size of two hidden layers with 50 neurons each, which leads us to re-evaluate Ensembles at the same size. Further, we also evaluate MDN and CRPS-based models at the same network size for fairness.

For the CRPS-based models we use GELU (Hendrycks & Gimpel, 2016) activation together with layer normalization (Ba et al., 2016), as it appears to help the network. For correctness and transparency, we also re-evaluate Ensembles, MDN, and MDN_{bnn} with the same setup, and include stronger versions in the main text of the paper in Table 4. See Tables 4 and 5 for pairwise comparisons in NLL and RMSE. According to the results, we include the ReLU versions for Ensembles (5/9 for NLL and 4/7 for RMSE), the ReLU versions for MDN_{bnn} (4/8 for NLL with 3 overall strongest results and 6/8 for RMSE), and the GELU version for MDN (6/9 for NLL and 5/8 for RMSE).

The source-code for the method Dropout is publicly available, however proper evaluation requires grid-search optimization of two separate hyperparameters over every single train-test split of 9 datasets separately, which proved to be prohibitively expensive for the two-layer variant. Therefore, we report the original scores for the single-layer variant in case of Dropout.

Due to methodological differences, we cannot compare against Amini et al. (2020). Amini et al. (2020) propose modeling aleatoric uncertainty using the Normal distribution, and epistemic uncertainty as a parametric meta-distribution over the parameters of the aleatoric normal distribution (Normal-Inverse Gamma). They report good performance on the UCI benchmark. However, we could not include their results: reproducing their experiments and investigating the code provided[5] reveals that

[5] https://github.com/aamini/evidential-deep-learning/tree/main/neurips2020

Table 4: Pairwise comparison of ReLU and GELU based baseline variants, evaluated in NLL. Pairwise best scores are highlighted with underline. Scores that perform at least on par when compared to the results in Table 1 are highlighted in **bold**.

dataset	Ensembles	Ensembles^{gelu}	MDN	MDN^{gelu}	MDN$_{\text{bnn}}$	MDN$^{\text{gelu}}_{\text{bnn}}$	WCRPS$_{\text{e}}$
boston	2.44$\pm$0.05	2.41$\pm$0.05	2.45$\pm$0.06	2.52$\pm$0.04	2.36$\pm$0.03	**2.32**$\pm$0.04	**2.32**$\pm$0.05
concrete	2.97$\pm$0.04	3.14$\pm$0.07	3.16$\pm$0.03	3.13$\pm$0.04	2.99$\pm$0.03	2.99$\pm$0.03	2.95$\pm$0.05
energy	0.62$\pm$0.08	0.75$\pm$0.15	1.09$\pm$0.06	0.77$\pm$0.04	0.65$\pm$0.02	0.58$\pm$0.03	**0.31**$\pm$0.03
kin8nm	-1.34$\pm$0.00	-1.37$\pm$0.01	-1.15$\pm$0.01	-1.22$\pm$0.01	-1.30$\pm$0.01	-1.37$\pm$0.01	**-1.38**$\pm$0.01
naval	-6.49$\pm$0.02	-6.12$\pm$0.02	-6.55$\pm$0.03	-6.24$\pm$0.04	**-6.72**$\pm$0.01	-6.38$\pm$0.02	-6.51$\pm$0.03
power	2.69$\pm$0.01	2.74$\pm$0.02	2.67$\pm$0.00	2.65$\pm$0.01	2.62$\pm$0.01	2.64$\pm$0.01	**2.57**$\pm$0.01
protein	2.66$\pm$0.02	2.64$\pm$0.05	2.02$\pm$0.01	2.00$\pm$0.01	**1.96**$\pm$0.01	1.97$\pm$0.01	2.05$\pm$0.01
wine	0.91$\pm$0.01	0.96$\pm$0.02	-3.75$\pm$0.06	-3.48$\pm$0.03	**-5.11**$\pm$0.09	-3.91$\pm$0.04	0.80$\pm$0.03
yacht	-0.02$\pm$0.04	**-0.32**$\pm$0.06	0.79$\pm$0.09	0.26$\pm$0.09	0.63$\pm$0.04	0.55$\pm$0.03	**-0.16**$\pm$0.05

Table 5: Pairwise comparison of ReLU and GELU based baseline variants, evaluated in RMSE. Pairwise best scores are highlighted with underline.

dataset	Ensembles	Ensembles^{gelu}	MDN	MDN^{gelu}	MDN$_{\text{bnn}}$	MDN$^{\text{gelu}}_{\text{bnn}}$	WCRPS$_{\text{e}}$
boston	3.37$\pm$0.17	3.27$\pm$0.24	3.54$\pm$0.27	3.73$\pm$0.25	2.93$\pm$0.20	3.10$\pm$0.22	2.91$\pm$0.18
concrete	5.19$\pm$0.19	5.47$\pm$0.16	6.34$\pm$0.11	5.99$\pm$0.12	5.25$\pm$0.11	5.36$\pm$0.14	4.94$\pm$0.17
energy	0.86$\pm$0.12	0.52$\pm$0.02	1.89$\pm$0.13	1.17$\pm$0.09	0.51$\pm$0.01	0.61$\pm$0.03	0.41$\pm$0.01
kin8nm	0.07$\pm$0.00	0.07$\pm$0.00	0.08$\pm$0.00	0.07$\pm$0.00	0.07$\pm$0.00	0.06$\pm$0.00	0.06$\pm$0.00
naval	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00
power	3.73$\pm$0.03	3.83$\pm$0.04	3.96$\pm$0.04	3.93$\pm$0.04	3.86$\pm$0.04	3.95$\pm$0.04	3.62$\pm$0.04
protein	4.20$\pm$0.03	4.09$\pm$0.03	4.07$\pm$0.03	3.96$\pm$0.02	4.09$\pm$0.02	4.11$\pm$0.02	3.47$\pm$0.02
wine	0.63$\pm$0.00	0.64$\pm$0.01	0.65$\pm$0.01	0.66$\pm$0.01	0.66$\pm$0.01	0.64$\pm$0.01	0.63$\pm$0.01
yacht	0.92$\pm$0.08	1.24$\pm$0.15	1.68$\pm$0.18	1.97$\pm$0.25	0.77$\pm$0.07	0.83$\pm$0.08	0.78$\pm$0.09

Table 6: NLL results on the UCI Datasets benchmark (lower is better). Scores that perform at least on par when compared to the results in Table 1 are highlighted in **bold**.

dataset	CRPS	WCRPS	WCRPS^{mh}	$\text{WCRPS}^{\text{lmh}}$	WCRPS_{e}	$\text{WCRPS}_{\text{e}}^{\text{mh}}$	$\text{WCRPS}_{\text{e}}^{\text{lmh}}$
boston	2.38$\pm$0.06	2.40$\pm$0.05	2.40$\pm$0.04	2.40$\pm$0.06	**2.32**$\pm$0.05	**2.32**$\pm$0.05	**2.30**$\pm$0.04
concrete	3.11$\pm$0.04	3.09$\pm$0.04	2.97$\pm$0.04	**2.89**$\pm$0.03	2.95$\pm$0.05	**2.73**$\pm$0.04	**2.74**$\pm$0.03
energy	0.46$\pm$0.04	0.47$\pm$0.04	0.64$\pm$0.02	0.79$\pm$0.03	**0.31**$\pm$0.03	0.67$\pm$0.01	0.78$\pm$0.01
kin8nm	-1.34$\pm$0.01	-1.32$\pm$0.01	-1.30$\pm$0.00	-1.29$\pm$0.01	**-1.38**$\pm$0.01	-1.36$\pm$0.01	-1.33$\pm$0.01
naval	-6.61$\pm$0.03	-6.65$\pm$0.05	-5.93$\pm$0.04	-5.83$\pm$0.03	-6.51$\pm$0.03	-5.95$\pm$0.03	-5.80$\pm$0.01
power	2.66$\pm$0.01	2.66$\pm$0.01	2.62$\pm$0.01	2.62$\pm$0.01	**2.57**$\pm$0.01	**2.56**$\pm$0.00	2.60$\pm$0.00
protein	2.29$\pm$0.00	2.22$\pm$0.01	2.22$\pm$0.01	2.31$\pm$0.01	2.05$\pm$0.01	2.18$\pm$0.01	2.29$\pm$0.01
wine	0.89$\pm$0.04	0.90$\pm$0.03	-0.36$\pm$0.03	0.15$\pm$0.02	0.80$\pm$0.03	-0.19$\pm$0.02	0.11$\pm$0.02
yacht	0.07$\pm$0.08	0.07$\pm$0.06	0.62$\pm$0.04	0.93$\pm$0.04	**-0.16**$\pm$0.05	0.80$\pm$0.03	0.92$\pm$0.03

Table 7: RMSE results on the UCI Datasets benchmark (lower is better). Scores that perform at least on par when compared to the results in Table 2 are highlighted in **bold**.

dataset	CRPS	WCRPS	WCRPS^{mh}	$\text{WCRPS}^{\text{lmh}}$	WCRPS_{e}	$\text{WCRPS}_{\text{e}}^{\text{mh}}$	$\text{WCRPS}_{\text{e}}^{\text{lmh}}$
boston	3.01$\pm$0.21	3.07$\pm$0.23	3.08$\pm$0.20	3.12$\pm$0.25	**2.91**$\pm$0.18	2.98$\pm$0.20	2.93$\pm$0.20
concrete	5.48$\pm$0.16	5.51$\pm$0.14	5.11$\pm$0.15	5.23$\pm$0.15	**4.94**$\pm$0.17	**4.87**$\pm$0.15	**4.85**$\pm$0.13
energy	0.44$\pm$0.02	0.45$\pm$0.02	0.52$\pm$0.02	0.48$\pm$0.02	**0.41**$\pm$0.01	0.45$\pm$0.02	0.43$\pm$0.01
kin8nm	0.07$\pm$0.00	0.07$\pm$0.00	0.07$\pm$0.00	0.07$\pm$0.00	**0.06**$\pm$0.00	**0.06**$\pm$0.00	0.07$\pm$0.00
naval	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00
power	3.79$\pm$0.05	3.79$\pm$0.04	3.74$\pm$0.04	3.71$\pm$0.04	**3.62**$\pm$0.04	3.65$\pm$0.04	3.64$\pm$0.04
protein	3.77$\pm$0.01	3.76$\pm$0.02	3.86$\pm$0.02	3.80$\pm$0.03	**3.47**$\pm$0.02	3.69$\pm$0.01	3.60$\pm$0.04
wine	0.64$\pm$0.01	0.64$\pm$0.01	0.64$\pm$0.01	0.65$\pm$0.01	**0.63**$\pm$0.01	**0.63**$\pm$0.01	0.64$\pm$0.01
yacht	0.95$\pm$0.11	0.96$\pm$0.13	0.89$\pm$0.09	1.05$\pm$0.12	0.78$\pm$0.09	0.99$\pm$0.10	0.94$\pm$0.11

1. They do not use the same train-test splits as other approaches mentioned in Section 5.3, rather random ones;
2. Their code appears to validate on the test set, and reports *the minimum of the achieved NLL results* over all epochs.

The setup seems consistent across all measured methods in the paper, therefore their results are internally consistent (although an argument could be made that it favors methods with good predictions *and large variance* across training epochs). However, unfortunately their results are not comparable with the rest of the literature as presented.

Implementing a proper train-validation-test setup for their model appears to result in subpar performance in our experiments, however this could be the case for a number of reasons, such as suboptimal hyperparameter choices on our part.

C.7 CRPS Model Variants on the UCI Benchmark

We present NLL scores in Table 6 and RMSE scores in Table 7 for various CRPS-based model variants. A subscript of *e* indicates ensemble models, superscript of *mh* and *lmh* indicates multi-head and layered-multi-head variants, respectively.

C.8 Running Times & Scaling

See Table 8 for running times in seconds on the UCI Benchmark. All of our implementations were run on an AMD EPYC 7F72 workstation with 2x24 CPU cores (48 cores, 96 threads in total) and 384 MiB of L3 cache. We used a single NVIDIA A100-SXM4-40GB GPU per experiment, with CUDA version 12.2 and driver version 535.183.01, ensuring consistent conditions across experiments.

To illustrate scaling, we compare the running time of training 10 epochs with MDN and CRPS on the largest *protein* dataset for increasingly large model sizes, however leaving the expressive power

Table 8: Running times on the full UCI Benchmark in seconds.

MDN	WCRPS	Ensembles	$\text{WCRPS}_{\text{ens}}$	MDN_{bnn}
5594	12074	19070	39624	95902

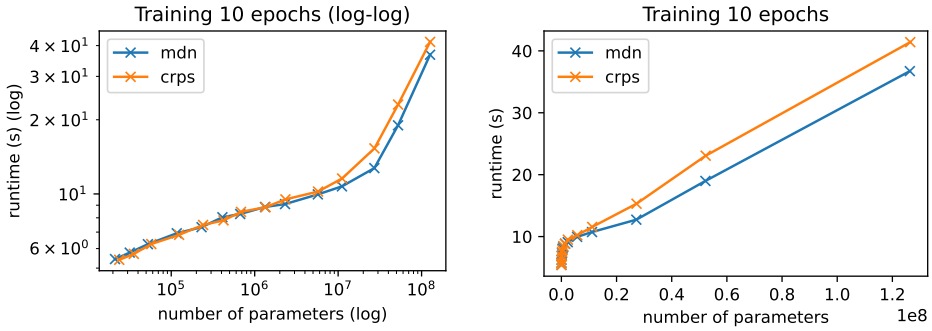


Figure 7: Scaling of a CRPS badsed models compared to a deterministic model.

w.r.t. the output distribution constant (100 components for MDN and two layers of size [100, 50] for the CRPS based model). See Figure 7 for results.

At smaller model sizes, the specifics of the GPU hardware make comparisons somewhat random, as devices can execute a large amount of computation in one step. However, as expected, at larger sizes, the nondeterministic part of the network incurs a constant amount of additional cost, leaving the running time approximately linear in the number of model parameters.

C.9 Calibration

Please refer to Figure 8 for calibration plots on all train and test datasets.

C.10 Dataset Statistics

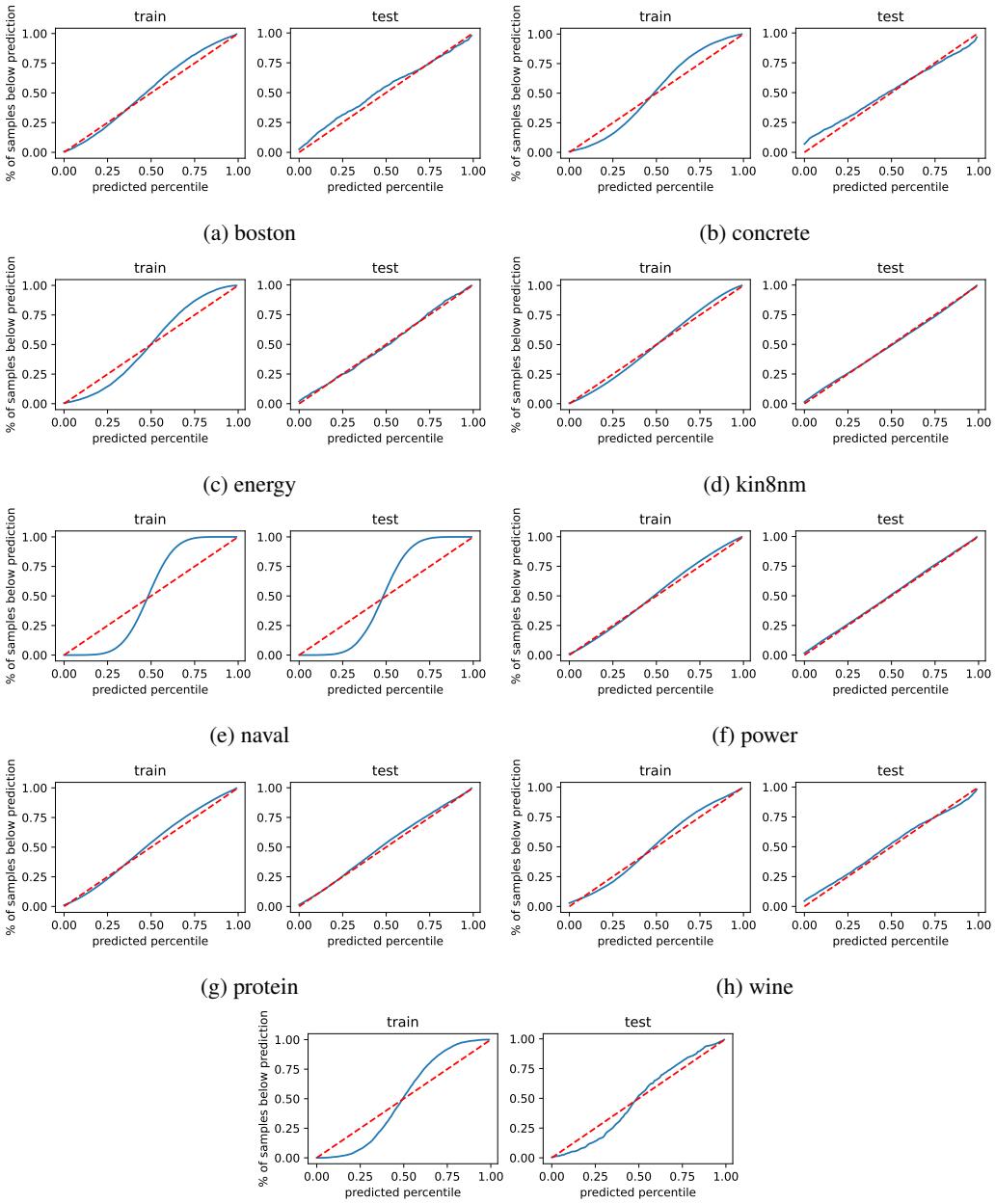
Please refer to Table 9 for dataset statistics.

C.11 Performance Against Simple Methods

While the primary concern in term of performance is comparison against advanced alternatives, such as the baselines of Section 5.3, there is value in observing the improvement when comparing against more traditional model types. In this section we present measurements comparing our

Table 9: Value frequencies in the UCI benchmark datasets

dataset	samples (m)	unique y	unique / m	most frequent y	most frequent / m
boston	506	229	45.26%	16	3.16%
concrete	1030	845	82.04%	6	0.58%
energy	768	586	76.30%	6	0.78%
kin8nm	8192	8191	99.99%	2	0.02%
naval	11934	51	0.43%	234	1.96%
power	9568	4836	50.54%	9	0.09%
protein	45730	15903	34.78%	272	0.59%
wine	1599	6	0.38%	681	42.59%
yacht	308	258	83.77%	3	0.97%



(a) boston (b) concrete (c) energy (d) kin8nm (e) naval (f) power (g) protein (h) wine (i) yacht

Figure 8: Calibration for the WCRPS_e model on all datasets separately for train and test sets

Table 10: NLL results on the UCI Datasets benchmark (lower is better). Scores that perform at least on par when compared to the results in Table 1 are highlighted in **bold**.

dataset	SHR	MDN	BNN	MDN_{bnn}	WCRPS_{e}
boston	2.74$\pm$0.11	2.45$\pm$0.06	2.36$\pm$0.04	2.36$\pm$0.03	**2.32**$\pm$0.05
concrete	3.30$\pm$0.09	3.16$\pm$0.03	2.98$\pm$0.03	2.99$\pm$0.03	2.95$\pm$0.05
energy	1.53$\pm$0.21	1.09$\pm$0.06	1.00$\pm$0.21	0.65$\pm$0.02	**0.31**$\pm$0.03
kin8nm	-1.20$\pm$0.01	-1.15$\pm$0.01	-1.28$\pm$0.01	-1.30$\pm$0.01	**-1.38**$\pm$0.01
naval	-6.59$\pm$0.03	-6.55$\pm$0.03	**-6.84**$\pm$0.03	**-6.72**$\pm$0.01	-6.51$\pm$0.03
power	2.78$\pm$0.02	2.67$\pm$0.00	2.77$\pm$0.01	2.62$\pm$0.01	**2.57**$\pm$0.01
protein	2.89$\pm$0.06	2.02$\pm$0.01	2.73$\pm$0.02	**1.96**$\pm$0.01	2.05$\pm$0.01
wine	0.98$\pm$0.03	-3.75$\pm$0.06	0.89$\pm$0.02	**-5.11**$\pm$0.09	0.80$\pm$0.03
yacht	0.67$\pm$0.08	0.79$\pm$0.09	0.26$\pm$0.07	0.63$\pm$0.04	**-0.16**$\pm$0.05

Table 11: RMSE results on the UCI Datasets benchmark (lower is better). Scores that perform at least on par when compared to the results in Table 2 are highlighted in **bold**.

dataset	SHR	MDN	BNN	MDN_{bnn}	WCRPS_{e}
boston	3.63$\pm$0.25	3.54$\pm$0.27	3.01$\pm$0.24	2.93$\pm$0.20	**2.91**$\pm$0.18
concrete	6.20$\pm$0.16	6.34$\pm$0.11	5.30$\pm$0.12	5.25$\pm$0.11	**4.94**$\pm$0.17
energy	1.51$\pm$0.20	1.89$\pm$0.13	0.54$\pm$0.03	0.51$\pm$0.01	**0.41**$\pm$0.01
kin8nm	0.08$\pm$0.00	0.08$\pm$0.00	0.07$\pm$0.00	0.07$\pm$0.00	**0.06**$\pm$0.00
naval	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00	0.00$\pm$0.00
power	3.94$\pm$0.04	3.96$\pm$0.04	3.96$\pm$0.04	3.86$\pm$0.04	**3.62**$\pm$0.04
protein	4.54$\pm$0.04	4.07$\pm$0.03	4.39$\pm$0.04	4.09$\pm$0.02	**3.47**$\pm$0.02
wine	0.64$\pm$0.01	0.65$\pm$0.01	**0.63**$\pm$0.01	0.66$\pm$0.01	**0.63**$\pm$0.01
yacht	1.39$\pm$0.17	1.68$\pm$0.18	0.88$\pm$0.08	**0.77**$\pm$0.07	0.78$\pm$0.09

method against a simple probabilistic neural-network based heteroscedastic regression model, and a classic BNN trained with variational inference.

A simple heteroscedastic regression model predicts the mean along with the variance of the model, and computes the loss function accordingly (**SHR** for **S**imple **H**eteroscedastic **R**egression). Note that assuming normal distribution for the noise, this model coincides with MDN when setting the number of components to 1.

Classic Bayesian neural networks also predict the parameters of the aleatoric distribution, however also assume probability distributions over the weights of the network. The resulting network can be optimized using variational inference Blundell et al. (2015) (**BNN**). Again assuming normal distribution for the noise, this model coincides with a BNN-based version of MDN with 1 components.

We present results for SHR, MDN, BNN, MDN_{bnn}, and WCRPS_{e} in Tables 10 and 11. As we can observe, in most cases simple methods either benefit from relaxing the Gaussian assumption, or perform similarly to the corresponding variant of MDN. On some datasets, however, the Gaussian restriction *does* seem to benefit the method, most apparent on the *yacht* and *naval* datasets.

C.12 Performance Comparison for the Number of Samples

As the CRPS-based methods represent their distribution through producing samples, the number of samples drawn for calculating the loss is an important hyperparameter during training. In Table 12, we present performance as well as running times for various choices for the number of samples drawn during training.

A smaller number of samples drawn results in faster per-batch calculation time, but in turn the training process takes longer to converge. While the running time overall seems to vary randomly to a relatively large degree, still we can observe that smaller values result in faster running times.

Table 12: NLL results on the UCI Datasets benchmark (lower is better) for different values of samples drawn during training.

dataset	2	3	5	10	20	50	100	200
boston	2.33±0.04	2.30±0.04	2.40±0.04	2.39±0.05	2.35±0.05	2.33±0.06	2.40±0.05	2.40±0.05
concrete	3.02±0.03	3.04±0.04	3.06±0.05	3.04±0.04	3.08±0.03	3.09±0.04	3.09±0.04	3.11±0.05
energy	0.59±0.03	0.49±0.02	0.49±0.03	0.47±0.03	0.50±0.05	0.44±0.03	0.47±0.04	0.46±0.05
kin8nm	-1.24±0.02	-1.34±0.01	-1.34±0.01	-1.34±0.01	-1.34±0.01	-1.33±0.01	-1.32±0.01	-1.33±0.01
naval	-6.07±0.03	-6.23±0.03	-6.39±0.03	-6.41±0.03	-6.52±0.04	-6.57±0.03	-6.65±0.05	-6.58±0.03
power	2.81±0.01	2.80±0.01	2.78±0.00	2.73±0.01	2.66±0.01	2.66±0.01	2.66±0.01	2.65±0.01
protein	2.66±0.01	2.60±0.01	2.51±0.01	2.27±0.04	2.21±0.01	2.22±0.00	2.22±0.01	2.23±0.01
wine	0.93±0.02	0.91±0.03	0.92±0.02	0.91±0.03	0.92±0.03	0.92±0.02	0.90±0.03	0.87±0.03
yacht	0.23±0.05	0.13±0.05	0.05±0.06	-0.05±0.06	0.00±0.05	-0.08±0.07	0.07±0.06	0.08±0.08
time (h)	2.34	2.51	2.41	2.90	3.61	3.55	3.35	3.43

On the other hand, while some smaller datasets seem to favor small sample values (notably *boston* and *concrete*), larger sample sizes seem much more robust overall, especially on the larger datasets in the benchmark (*naval, power, protein*).

C.13 Performance Comparison Against DISCO Nets

As described before, the closest prior approach is DISCO Nets (Bouchacourt et al., 2016), where the authors introduce a multivariate distributional regression method for hand-pose estimation. Remarkably, they also use injected random noise for nondeterminism (as in Section 4.1), and derive a loss function which, with the correct parametrization, is very similar to a multivariate version of the basic non-weighted version of our method, essentially reproducing the Energy Score (see Section 4.5), which the authors also highlight.

As the DISCO Nets method is not primarily formulated for univariate regression, for fairness we first compare on the original task of (Bouchacourt et al., 2016). DISCO Nets is originally measured on the NYU Hand Pose Estimation dataset (Tompson et al., 2014) as preprocessed by Oberweger et al. (2015). The processed dataset contains 72,757 training and 8254 testing frames, with the sets consisting of RGBD images of hands, taken from 3 viewpoints. However, only the front view and the depth channel is used for the experiment. For each data point, the label consist of the 3d position of 14 finger joints.

To compare, we use the original code of DISCO Nets[6] to run the original preprocessing pipeline to acquire the train and test sets. We also use the same network size as Bouchacourt et al. (2016), consisting of 3 convolutional layers with 8 filters each, max pooling layers, and 3 linear layers of size 1024 with ReLU activation.

Performance is measured as in Bouchacourt et al. (2016), in terms of Energy Score (denoted as *ProbLoss*, smaller is better), Mean Joint Euclidean Error (MeJEE, smaller is better), Max Joint Euclidean Error (MaJEE, smaller is better) and Fraction of Frames within distance (FF, larger is better), including standard error of the mean (SEM) for the first three. For exact definitions, please refer to the original paper and its supplement. We take scores for DISCO Nets and baselines from the original paper, and present scores for WCRPS and WCRSP_e in Table 13. As we can observe, the improved models outperform DISCO Nets, with the WCRPS_e model performing really strongly in particular. While deterministic measures (MeJEE, MaJEE, FF) also show higher scores, the performance difference is most apparent in ProbLoss, hinting at much better predictive distributions.

Second, we implement the method on the UCI Benchmark and evaluate its performance under the same conditions as in Section 5.3.1. For hyperparameters not explicitly investigated in Bouchacourt et al. (2016), we adopt the same values as the original authors (e.g., $\beta = 1$). Since DISCO Nets also generates samples rather than directly modeling the density, we apply the KDE approach described in Appendix C.1 to compute the NLL.

[6]https://github.com/oval-group/DISCO Nets

Table 13: Performance comparison against Bouchacourt et al. (2016) and its baselines using the multivariate WCRPS and WCRPS_w models on the NYU Hand Pose dataset, ±SEM.

Model	ProbLoss (mm)	MeJEE (mm)	MaJEE (mm)	FF (80mm)
$\text{BASE}_{\beta=1,\sigma=1}$	103.8±0.627	25.2±0.152	52.7±0.290	86.040
$\text{BASE}_{\beta=1,\sigma=5}$	99.3±0.620	25.5±0.151	52.9±0.289	85.773
$\text{BASE}_{\beta=1,\sigma=10}$	96.3±0.612	25.7±0.149	53.2±0.288	85.664
$\text{DISCO}_{\beta=1,\gamma=0}$	92.9±0.533	21.6±0.128	46.0±0.251	92.971
$\text{DISCO}_{\beta=1,\gamma=0.25}$	89.9±0.510	21.2±0.122	46.4±0.252	93.262
$\text{DISCO}_{\beta=1,\gamma=0.5}$	83.8±0.503	20.9±0.124	45.1±0.246	94.438
WCRPS	72.1±0.447	19.5±0.114	39.0±0.211	96.510
WCRPS_e	**63.7±0.447**	**18.8±0.126**	**37.6±0.233**	**96.958**

Table 14: NLL results on the UCI Datasets benchmark (lower is better). Scores that perform at least on par when compared to the results in Table 1 are highlighted in **bold**.

dataset	dropout	CRPS	WCRPS_e	$\text{DISCO}^{\beta=1}_{\gamma=0}$	$\text{DISCO}^{\beta=1}_{\gamma=0.25}$	$\text{DISCO}^{\beta=1}_{\gamma=0.5}$	BNN-LV
boston	2.40±0.04	2.38±0.06	**2.32**±0.05	3.44±0.14	2.86±0.09	2.34±0.04	2.96±0.03
concrete	**2.94**±0.02	3.11±0.04	2.95±0.05	4.16±0.06	3.51±0.04	3.02±0.03	3.20±0.02
energy	1.21±0.01	0.46±0.04	**0.31**±0.03	2.61±0.11	0.82±0.12	0.65±0.03	2.37±0.04
kin8nm	-1.14±0.01	-1.34±0.01	**-1.38**±0.01	0.81±0.03	-0.83±0.01	-1.33±0.01	-1.24±0.01
naval	-4.45±0.00	-6.61±0.03	-6.51±0.03	-6.59±0.03	-6.47±0.03	-6.04±0.03	-4.96±0.25
power	2.81±0.01	2.66±0.01	**2.57**±0.01	3.54±0.03	3.13±0.01	2.71±0.01	2.81±0.02
protein	2.87±0.00	2.29±0.00	2.05±0.01	5.30±0.14	inf±nan[7]	2.34±0.01	2.68±0.07
wine	0.93±0.01	0.89±0.04	0.80±0.03	2.62±0.09	1.75±0.06	0.94±0.03	1.15±0.03
yacht	1.25±0.02	0.07±0.08	**-0.16**±0.05	0.29±0.07	0.10±0.07	0.37±0.07	3.33±0.06

In Bouchacourt et al. (2016), the authors propose using the MEU (Maximum Expected Utility) point, conceptually similar to the geometric median, and suggest employing MEU-based RMSE for early stopping by comparing it to the ground truth. However, in our experiments, we resort to using NLL for early stopping, as MEU-based RMSE proved ineffective; it often failed to decrease beyond a certain threshold or even increased from its starting point. In a univariate regression problem, the output distribution's geometric median is equivalent to the median, thus MEU is essentially defined as the median of the sample. Given that the median generally does not coincide with the mean which is known to be optimal for RMSE, this outcome is not unexpected.

We test three hyperparameter configurations for DISCO Nets, as presented in Tables 14 and 15. As expected, the theoretically optimal variant ($\gamma = 0.5$) performs well, while other configurations yield suboptimal results. However, even the correctly parameterized version is outperformed by our best method. This outcome is again unsurprising, since with the correct hyperparameter settings, adaptations for regression benchmarking, and other enhancements we apply for training, the method essentially replicates the unweighted CRPS approach trained using a small sample size for loss calculation.

C.14 Performance Comparison Against BNN-LV

The method introduced in Depeweg et al. (2018) (BNN-LV) is theoretically capable of representing complex distributions, such multi-modal ones. However, the original paper primarily evaluates the method on reinforcement learning tasks and lacks detailed quantitative analysis. Therefore, we choose not to benchmark against it using its original datasets. Fortunately, a recent uncertainty quantification framework (Lehmann et al., 2024) implements BNN-LV for regression. Given that the first author of Depeweg et al. (2018) is also a co-author of (Lehmann et al., 2024) and was

[7] When a ground truth point falls too far outside the predicted distribution, the likelihood function becomes essentially 0 numerically, causing the NLL metric to take on the value of ∞.

Table 15: RMSE results on the UCI Datasets benchmark (lower is better). Scores that perform at least on par when compared to the results in Table 2 are highlighted in **bold**.

dataset	dropout	CRPS	WCRPS_e	$\text{DISCO}^{\beta=1}_{\gamma=0}$	$\text{DISCO}^{\beta=1}_{\gamma=0.25}$	$\text{DISCO}^{\beta=1}_{\gamma=0.5}$	BNN-LV
boston	$3.61_{\pm 0.23}$	$3.01_{\pm 0.21}$	$\mathbf{2.91}_{\pm 0.18}$	$4.15_{\pm 0.27}$	$3.49_{\pm 0.21}$	$2.99_{\pm 0.20}$	$5.54_{\pm 0.40}$
concrete	$5.45_{\pm 0.19}$	$5.48_{\pm 0.16}$	$\mathbf{4.94}_{\pm 0.17}$	$6.95_{\pm 0.12}$	$6.34_{\pm 0.17}$	$5.36_{\pm 0.17}$	$6.15_{\pm 0.15}$
energy	$0.97_{\pm 0.06}$	$0.44_{\pm 0.02}$	$\mathbf{0.41}_{\pm 0.01}$	$2.87_{\pm 0.14}$	$0.71_{\pm 0.15}$	$0.56_{\pm 0.02}$	$3.76_{\pm 0.49}$
kin8nm	$0.09_{\pm 0.00}$	$0.07_{\pm 0.00}$	$\mathbf{0.06}_{\pm 0.00}$	$0.08_{\pm 0.00}$	$0.07_{\pm 0.00}$	$0.07_{\pm 0.00}$	$0.07_{\pm 0.00}$
naval	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
power	$4.18_{\pm 0.04}$	$3.79_{\pm 0.05}$	$\mathbf{3.62}_{\pm 0.04}$	$5.23_{\pm 0.12}$	$4.55_{\pm 0.04}$	$3.95_{\pm 0.04}$	$4.15_{\pm 0.04}$
protein	$4.39_{\pm 0.02}$	$3.77_{\pm 0.01}$	$\mathbf{3.47}_{\pm 0.02}$	$5.75_{\pm 0.01}$	$4.70_{\pm 0.08}$	$3.96_{\pm 0.03}$	$19.99_{\pm 4.20}$
wine	$0.66_{\pm 0.01}$	$0.64_{\pm 0.01}$	$\mathbf{0.63}_{\pm 0.01}$	$0.70_{\pm 0.01}$	$0.65_{\pm 0.01}$	$0.65_{\pm 0.01}$	$0.91_{\pm 0.11}$
yacht	$1.23_{\pm 0.37}$	$0.95_{\pm 0.11}$	$0.78_{\pm 0.09}$	$1.07_{\pm 0.11}$	$0.88_{\pm 0.10}$	$1.05_{\pm 0.10}$	$12.75_{\pm 1.64}$

substantially involved in the BNN-LV implementation[8], we consider the framework of (Lehmann et al., 2024) to be an authoritative reference implementation for BNN-LV.

Since the method requires relatively long per-epoch training times and converges slowly, taking approximately 26X as long as our fastest measured method MDN, we do not find it practical to optimize over multiple hyperparameter combinations. Instead, we employ the hyperparameters suggested in the framework's manual. For NLL calculations, we use the KDE approach outlined in Appendix C.1, and early stopping is performed based on measuring the training loss on the validation set. The results are summarized in Tables 14 and 15.

While on some datasets the BNN-LV method seems to consistently converge and produces respectable scores often comparable to classic baselines such as *dropout*, on others it occasionally seems to fail to converge (observe $\pm$ standard errors). However, in general it does not produce competitive scores when compared to the best measured non-Gaussian methods such as WCRPS_e or MDN_bnn.

[8] https://github.com/lightning-uq-box/lightning-uq-box/pull/21