# Redefining Uncertainty: Deep Learning with Probabilistic Regression Losses

Franco Marchesoni-Acland, You Too

February 2024

**Abstract**

Regression tasks, traditionally aimed at predicting precise outcomes from inputs, often overlook the uncertainty inherent in real-world data. This paper advocates for probabilistic regression within deep learning frameworks, emphasizing its criticality in fields where understanding uncertainty changes outcomes—such as finance and healthcare. By integrating deep learning with probabilistic regression through diverse differentiable cost functions, we enable models to predict outcome distributions, capturing the underlying data complexity. Our exploration through a synthetic one-dimensional experiment validates the approaches, showcasing its potential to not only forecast but also quantify uncertainty. This fusion of deep learning and probabilistic regression heralds a significant shift towards models that mirror the probabilistic nature of reality, promising advancements in predictive analytics. Code is available at **CITE**

## 1 Introduction

Deep learning has revolutionized the landscape of regression analysis, offering robust solutions where traditional methods falter **CITE** The quest for uncertainty quantification in predictions has led to the exploration of Bayesian neural networks **CITE** Monte Carlo dropout **CITE** and sampling from generative models **CITE** Despite their efficacy, these methods often introduce complexity and computational overhead. This work posits a more straightforward approach: leveraging deep learning with specialized loss functions to directly achieve probabilistic regression. This philosophy—optimizing directly for the desired outcome—streamlines the process, enhancing both efficiency and interpretability.

We begin by situating probabilistic regression within the traditional context of deep learning, demonstrating how deterministic predictions can serve as the nucleus for probability distributions, so that mean or median predicting methods can be evaluated probabilistically. We evaluate these simplistic probabilistic models, introducing metrics such as the Continuous Ranked Probability Score (CRPS) **CITE** reliability diagrams **CITE** log score **CITE** and Probability Integral Transform (PIT) histograms **CITE** each serving to dissect different facets of probabilistic accuracy and calibration.

Our main contribution is a collection of differentiable losses that foster probabilistic regression, encompassing likelihood optimization for Gaussian and Laplace distributions **CITE** histogram-based distributions based on bin mass **CITE**r support predictions **CITE** and the readaptation of implicit quantile networks **CITE** These methods collectively offer a versatile toolkit for embedding uncertainty directly within deep learning frameworks.

The efficacy of these approaches is empirically validated through an experiment modeling a bimodal distribution as a function of input, challenging the models to capture complex, input-dependent uncertainty. This synthetic setup not only tests the models' predictive accuracy but also their capacity to faithfully represent the underlying probabilistic nature of the data.

# 2 Background

Probabilistic regression seeks to model the conditional distribution $p(y|x)$ of a continuous outcome variable $y$, given an input vector $x$. For vector targets $\mathbf{y}$, the different $p(y = \mathbf{y}_i)$ are estimated, although posibly not independently. Unlike deterministic regression, which outputs a single point estimate $\hat{y} = f(x)$, probabilistic regression provides a full distribution that reflects the uncertainty and variability of the outcome. This approach leverages the cumulative distribution function (CDF), $F(y|x) = P(Y \leq y|x)$, a monotonically increasing function that maps any real-valued outcome $y$ to a probability $\alpha$ in the unit interval $[0, 1]$. The CDF is also right-continuous with well defined limiting behavior, and behaves better than its derivative the probability density function (PDF). The probability of the target variable to be included in any finite union of intervals can be computed by simply adding and substracting the CDF at the interval borders, making the CDF a very useful tool for decision making.

Mathematically, the CDF $F$ and its derivative the PDF $f$, are defined as follows:

$$F(y|x) = P(Y \leq y|x), \quad f(y|x) = \frac{d}{dy}F(y|x) \approx \frac{F(y+h|x) - F(y|x)}{h} \tag{1}$$

where $h$ is a small step size, facilitating the approximation of the PDF from the CDF. We will also write $F_x : \mathcal{Y} \to \mathrm{R}$ for the predicted CDF.

## 2.1 Neural Networks

Neural networks are of interest because given large amounts of data and computing power they can accurately approximate very complicated functions. We will deal with networks that given an input $x$ can estimate the parameters $\hat{y}$ of a CDF $F_{\hat{y}}(y|x)$.

We assume that a neural network is a function $g_\theta : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. The network is parameterized by $\theta$, and it is trained to minimize a loss function $L(\theta, \mathcal{D})$, which measures the discrepancy between the predicted $\hat{y} = g(x)$ and the true target $y$ for every input-output pair $(x, y)$ in the dataset $\mathcal{D}$. The loss function and the $g_\theta$ are differentiable with respect to $\theta$, and these gradients are used to update the parameters $\theta$ through variants of stochastic gradient descent. In what follows neural networks will be called $g$.

Note that for binary classification usually $g : \mathcal{X} \to \mathrm{R}$ and $\hat{y} = 1_{\sigma(g(x))>0.5}$ where $\sigma$ is the sigmoid function that maps R to $[0, 1]$ **CITE** Many times $\sigma(g(x))$ is interpreted as a probability making the prediction probabilistic (although not necessarily calibrated) **CITE** This is not the case of regression, where most commonly $g : \mathcal{X} \to \mathrm{R}$ and $\hat{y} = g(x)$, and the output is interpreted as a point estimate. The usual loss functions for regression are the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) **CITE**

Bayesian neural networks use probability distributions for the network parameters $\theta$ and produce probabilistic predictions. Although specialized libraries exist, they are not included by default in mainstream deep learning frameworks **CITE**orch keras. We refer the interested reader to **CITE**

## 2.2 Ensemble Methods

One common way of attacking probabilistic regression is through ensembles. A collection of $M$ deterministic models $\{g_i : \mathcal{X} \to \mathrm{R}\}_{i\in[1,M]}$ produces a collection of predictions $\{g_i(x)\}_{i\in[0,M]}$ and these predictions are used to build a CDF as in 2. Ensemble methods as defined here include collecting predictions from many non-identical models. Each of these models can be

- obtained separately, which is the general version of ensembling
- produced by a dropout vector that zeroes neurons of a network trained with dropout, which is called Monte-Carlo (MC) dropout **CITE**
- be produced by a noise vector that parameterizes the computation of a network, which is equivalent to sampling from a generative model **CITE**

or obtained in other particular ways (e.g. using XGBoost **CITE**.

Once the models predict the points $\{g_i(x)\}_{i\in[1,M]}$, and assuming without loss of generality that $g_i(x) \leq g_j(x); i < j$, the CDF is built by using the rank as the cumulative probability:

$$F(y|x) = \frac{1}{M} \sum_{i=1}^{M} 1_{g_i(x)\leq y}. \tag{2}$$

Even though this approach is flexible and provides uncertainty quantification and enhanced accuracy, it is not optimized for probabilistic regression. In this work, we propose ways to directly optimize for probabilistic regression.

# 3 Metrics for Evaluating Probabilistic Regression

## 3.1 Scoring rules

Scoring rules are used to evaluate the quality of probabilistic predictions. They are functions $S : \mathcal{F} \times \mathcal{Y} \to \mathrm{R}$ that measure the discrepancy between the predicted distribution $F(y|x)$ and the true outcome $y$. A scoring rule is proper if the expected value of the scoring rule is minimized when the predicted distribution matches the true distribution of the data.

## 3.2 Log Score

The **Log Score** provides a measure of the predictive accuracy of a probabilistic model, penalizing both overconfident and underconfident predictions. Defined as the negative logarithm of the PDF evaluated at the observed value, the Log Score encourages models to assign high probability density to the true outcomes. It is a proper scoring rule.

$$LS(x, y) = -\log(f(y|x)) \tag{3}$$

## 3.3 Continuous Ranked Probability Score

The **Continuous Ranked Probability Score** evaluates the entire predicted distribution, measuring the integrated squared difference between the predicted CDF and a step function centered at the observed value. It captures both the accuracy and sharpness of probabilistic predictions, rewarding models that provide both precise and calibrated probability distributions.

$$CRPS(x, y) = \int_{-\infty}^{\infty} (F(y'|x) - 1_{y \leq y'})^2 dy' \tag{4}$$

CRPS is the most expressive metric for probabilistic regression as it captures both accuracy and sharpness and it is a proper scoring rule. Sometimes it is the only metric used to evaluate probabilistic regression models **CITE**

## 3.4 Calibration and Sharpness

Calibration refers to the alignment between predicted probabilities and observed frequencies, a critical aspect of probabilistic models' reliability. Sharpness, on the other hand, pertains to the concentration of predictive distributions, with sharper predictions indicating higher confidence. Together, these metrics offer a holistic view of a model's performance, balancing the precision of probability estimates with their accuracy and reliability.

A probabilistic regression model achieves good calibration when, for every predicted probability $p$, the frequency of the actual outcome falling within a specified predictive interval or quantile consistently aligns with $p$. This principle is pivotal for ensuring the reliability of the model's uncertainty quantification, and it is particularly elucidated through practical examples like prediction intervals or quantile predictions. Calibration must be assessed and potentially improved using techniques such as isotonic regression or Platt scaling.

Given a model that generates a cumulative distribution function $F_x$ for each input $x$, one can ascertain the cumulative probability of the target $y$, denoted as $\text{CDF}_x(y) = P(Y \leq y|x) = \alpha$. Applying this procedure across each data pair $(x_i, y_i)$ yields a collection of pairs $(\alpha_i, y_i)$. These pairs serve as the basis for calculating various calibration metrics and generating insightful plots, as discussed below:

### 3.4.1 PIT Histogram

The Probability Integral Transform (PIT) histogram is a crucial diagnostic tool for evaluating model calibration. By plotting a histogram of the $\alpha_i$ values, one anticipates a uniform distribution if the model is perfectly calibrated. This uniformity signifies that the model's predicted probabilities accurately reflect the empirical distribution of outcomes. It's a consequence of the Probability Integral Transform theorem **CITE** which states that if $Y$ is a continuous random variable with cumulative distribution function $F_Y$, and if $U = F_Y(Y)$, then $U$ follows a uniform distribution on the interval $[0, 1]$. In our notation, $U = F_x(Y)$.

### 3.4.2 Reliability Diagram

A reliability diagram offers another visual method to assess the calibration of probabilistic predictions. In this plot, the x-axis represents all levels of cumulative probability $P$ and the y-axis the proportion of predictions whose cumulative probability $\alpha = F_x(y)$ at the ground truth $y$ is lower than $P$. The intuition is that for every $P$ one can evaluate the predictions as if they were binary by asking: how many times was the value I assigned to the cumulative probability $P$ higher than the ground truth? If the model was well calibrated, the ratio should be $P$. The reliability diagram is a plot of this ratio for every $P$ and should approach the identity.

### 3.4.3 Expected Calibration Error (ECE)

The Expected Calibration Error provides a quantitative measure of a model's calibration by computing the mean absolute deviation between the predicted cumulative probabilities and the observed frequencies. For each cumulative probability prediction $\alpha_i$, the calibration error is calculated as

$$\text{ECE} = |\alpha_i - \frac{1}{n} \sum_j 1_{y_j \leq F_{x_j}^{-1}(\alpha)}|, \tag{5}$$

where $n$ is the total number of observations and $F_{x_j}^{-1}(\alpha)$ is such that $F_{x_j}(F_{x_j}^{-1}(\alpha)) = \alpha$. The ECE is then the average of these individual calibration errors across all predictions, offering a concise metric that summarizes the overall calibration of the model. This metric essentially quantifies the mean absolute error between the reliability diagram and the identity line, serving as a critical indicator of the model's calibration quality.

# 4 Differentiable Losses for Probabilistic Regression

To effectively train deep learning models on probabilistic regression tasks, we utilize differentiable loss functions that directly optimize the model parameters based on the discrepancies between the predicted distributions and observed data. These loss functions are designed to be computationally tractable and conducive to gradient-based optimization, facilitating the integration of probabilistic forecasting into the deep learning framework. All models (represented with an M) predict parameters that characterize a CDF. They only differ on the loss function and on how the CDF is defined.

## 4.1 Making Regression Probabilistic

When training with the MSE or the MAE loss, the model learns to predict the mean or the median of the target distribution respectively. We can form a predicted distribution by considering this point estimate and using it to parameterize the center of a distribution, e.g. Gaussian or Laplace. The other parameters of the distribution, namely the width, are fit to the training data. For the Gaussian and Laplacian cases these are simply the square root of the MSE or the MAE respectively. These will serve as baseline models M1 and M2. Any probabilistic regression model should aim to surpass these baselines.

### 4.1.1 M1: Mean Squared Error

Use the MSE as a loss. After training, compute the square root of the MSE. This is the standard deviation of the Gaussian distribution whose mean the model is predicting.

$$MSE(x, y) = (y - \hat{y})^2 \tag{6}$$

### 4.1.2 M2: Mean Absolute Error

Use the MAE as a loss. After training, compute the MAE. This is the scale of the Laplace distribution whose median the model is predicting.

$$MAE(x, y) = |y - \hat{y}| \tag{7}$$

## 4.2 Likelihood-based losses

Instead of predicting the mean or median and then estimating the deviation or scale from the dataset, one can fit both center and width of the distributions to the data simultaneously using a likelihood-based loss. The model predicts the distribution parameters directly, and the loss is the negative log-likelihood of the observed data given the predicted distribution. This is a proper scoring rule, and it is the most direct way to train a probabilistic model that yields a CDF for each input.

### 4.2.1 M3: Gaussian Likelihood

Use as loss function:

$$L(\mu, \sigma^2) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{(y-\mu)^2}{2\sigma^2} \tag{8}$$

and predict for each input $x$ the mean $\mu$ and the variance $\sigma^2$ of a Gaussian distribution. Pytorch **CITE**as a built-in function to compute this loss.

### 4.2.2 M4: Laplace Likelihood

Use as loss function:

$$L(\mu, b) = \log(2b) + \frac{|y-\mu|}{b} \tag{9}$$

and analogously with M3 predict the mean $\mu$ and the scale $b$ of a Laplace distribution.

## 4.3 M5: Quantile regression

Quantile regression is a form of probabilistic regression that directly predicts quantiles of the target distribution. The loss function is the pinball loss set to different quantiles. This is a proper scoring rule. Use as losses:

$$L_q(\tau) = (\tau - 1_{y\leq\hat{y}})(y-\hat{y}) \tag{10}$$

where $\tau$ is the quantile level, and $1_{y\leq\hat{y}}$ is the indicator function that is 1 if $y \leq \hat{y}$ and 0 otherwise. For instance, if we predict 3 outputs, we can set $\tau = 0.25, 0.5, 0.75$ to predict the first, second and third quartiles of the target distribution. The quantiles define the bins of the predicted distribution, and the mass is asigned proportionally to each bin's width.

## 4.4 M6: Classification with cross-entropy loss

An approach used by some works **CITE** is to discretize the target distribution and predict the probability of each bin. This is a form of probabilistic regression, and the cross-entropy loss is used. The bins can be defined by the quantiles of the target distribution, or by any other method. This is the standard classification loss in deep learning. Unfortunately, it doesn't take into account the fact that the bins are ordered.

### 4.5 Note on histogram distributions

For predefined bins, care must be taken so that they include the support of the target distribution. One way to do this is to set a lower and upper bound and predict the probability of being outside this region to the left or right. More precisely, when considering bin edges $\{b_i\}_{i\in[0,B-1]}$ and probabilities for each bin $\{p_i : i \in [0, B-1], p_i = P(b_{i-1} < Y \leq b_i)\}$, $p_{-1} = 0$ and $p_B = P(b_{B-1} < Y)$, and assuming a piecewise linear CDF, one has $F(y) \to p_0$ for $y < b_0$ and $F(y) \to 1 - p_B$ for $b_{B-1} < y$ (this result is obtained by defining $b_{-1} = b_0 - T$ and $b_B = b_{B-1} + T$ and increasing T arbitrarily).

### 4.6 M7: Implicit quantile regression

Introduced in the paper "Implicit Quantile Networks for Distributional Reinforcement Learning" **CITE** this loss leverages some properties of the quantile functions to allow for a better approximation of the quantiles. Assume the network $g : \mathcal{X} \to \mathrm{R}^d$ maps the input $x$ to some features $g(x)$, and some other layers $h : \mathrm{R}^d \to \mathrm{R}$ map the features to an output value. An additional function $\phi : [0, 1] \to \mathrm{R}^d$ embeds sampled quantiles $\tau$ into the feature space. Then we have that the estimated value is:

$$\hat{y}(x, \tau) = h(g(x) \cdot \phi(\tau)) \tag{11}$$

with $\phi_j(\tau) = \mathrm{ReLU}\left(\sum_{i=0}^{n-1} \cos(\pi i \tau) w_{ij} + b_j\right)$, where $n = 64$, and $w_{ij}$ and $b_j$ are the parameters of the function $\phi$. The coordinate $j$ indexes each one of the $d$ dimensions. The loss in the paper is the Huber quantile regression loss, but we simplify it to the quantile regression loss. One samples $N$ quantiles $\tau_1, \ldots, \tau_N$ from a uniform distribution, and the loss is:

$$L(x, y) = \frac{1}{N} \sum_{i=1}^{N} (\tau_i - 1_{y \leq \hat{y}(\tau_i)})(y - \hat{y}_i(\tau_i)) \tag{12}$$

Finally, the CDF $F_x(y) = \tau_y$ is obtained by finding the $\tau_y$ such that $\hat{y}(x, \tau_y) = y$.

## 5 Experiments

We conduct experiments on a synthetic dataset. The dataset is composed by samples $(x, y)$ where $x$ is a mixing parameter and $y$ is sampled from a mixture of two gaussians $\mathcal{M} \sim \mathcal{N}_1 x + \mathcal{N}_2(1 - x)$. This means that varying the input $x$ varies the target distribution associated to it.

We train the models on this dataset using as backbone a small multi-layer perceptron with 0.1 dropout **CITE** GELU activation **CITE**nd batch normalization **CITE** Unless otherwise specified, we train with Pytorch SGD optimizer with learning rate scheduled with one cosine cycle **CITE**ncluding 5% warmup iterations.

```
args = {’SEED’: 0, ’N’: 10000, ’N_test’: 100, ’mean1’: 0, ’mean2’: 1,
’std1’: 1.1, ’std2’: 0.1, ’n_epochs’: 144, ’n_factors’: 10, ’postfix’: ’_default’}
--------------------------------------------------
Quantile Regressor
Training took 27.65036106109619 seconds
Evaluating took 0.8435070514678955 seconds
----------------------------------------
CRP error: 7.633e-01
Calibration error: 3.233e-01
Centering error: 3.454e-01
MAE of the median predictor: 9.395e-01
----------------------------------------

--------------------------------------------------
MedianScale Regressor
Training took 32.226783990859985 seconds
Evaluating took 0.3915867805480957 seconds
----------------------------------------
CRP error: 5.939e-01
Calibration error: 4.209e-01
Centering error: 4.216e-01
MAE of the median predictor: 3.679e+00
----------------------------------------

--------------------------------------------------
Median Regressor
Training took 30.52402925491333 seconds
Evaluating took 0.4669013023376465 seconds
----------------------------------------
CRP error: 4.955e-01
Calibration error: 3.659e-01
Centering error: 3.194e-01
MAE of the median predictor: 7.377e-01
----------------------------------------

--------------------------------------------------
MeanStd Regressor
Training took 29.006088495254517 seconds
Evaluating took 0.3683760166168213 seconds
----------------------------------------
CRP error: 1.421e+00
Calibration error: 2.655e-01
Centering error: 3.849e-01
MAE of the median predictor: 5.451e+00
----------------------------------------
```

```
--------------------------------------------------
Mean Regressor
Training took 30.91608762741089 seconds
Evaluating took 0.6533441543579102 seconds
----------------------------------------
CRP error: 6.074e-01
Calibration error: 2.920e-01
Centering error: 2.865e-01
MAE of the median predictor: 1.013e+00
----------------------------------------

--------------------------------------------------
Implicit Quantile Regressor
Training took 665.1354336738586 seconds
Evaluating took 5.988877534866333 seconds
----------------------------------------
CRP error: 8.478e+02
Calibration error: 3.227e-01
Centering error: 2.011e-01
MAE of the median predictor: 1.023e+00
----------------------------------------

--------------------------------------------------
Bin Classifier Regressor
Training took 28.58774209022522 seconds
Evaluating took 0.4074678421020508 seconds
----------------------------------------
CRP error: 5.780e-01
Calibration error: 3.622e-01
Centering error: 2.906e-01
MAE of the median predictor: 7.120e-01
----------------------------------------
```

# 6 Conclusion

This work underscores the importance of probabilistic regression in modern predictive modeling, offering a pathway to more nuanced and informative predictions. Through the formulation of differentiable loss functions, we pave the way for the broader adoption of probabilistic approaches in deep learning, with promising implications for various applications that require robust uncertainty quantification.