# Neural Spline Search for Quantile Probabilistic Modeling

**Ruoxi Sun[1]*, Chun-Liang Li[1]*, Sercan Ö. Arık[1], Michael W. Dusenberry[2], Chen-Yu Lee[1], Tomas Pfister[1]**

[1]Google Cloud AI
[2]Google Research, Brain Team
{ruoxis, chunliang, soarik, dusenberrymw, chenyulee, tpfister}@google.com

## Abstract

Accurate estimation of output quantiles is crucial in many use cases, where it is desired to model the range of possibility. Modeling target distribution at arbitrary quantile levels and at arbitrary input attribute levels are important to offer a comprehensive picture of the data, and requires the quantile function to be expressive enough. The quantile function describing the target distribution using quantile levels is critical for quantile regression. Although various parametric forms for the distributions (that the quantile function specifies) can be adopted, an everlasting problem is selecting the most appropriate one that can properly approximate the data distributions. In this paper, we propose a non-parametric and data-driven approach, Neural Spline Search (NSS), to represent the observed data distribution without parametric assumptions. NSS is flexible and expressive for modeling data distributions by transforming the inputs with a series of monotonic spline regressions guided by symbolic operators. We demonstrate that NSS outperforms previous methods on synthetic, real-world regression and time-series forecasting tasks.

## Introduction

For many machine learning applications, modeling the prediction intervals (e.g. estimating the ranges all individual predictions observation fall), beyond point estimates, is crucial (Salinas et al. 2020; Wen et al. 2017; Tagasovska and Lopez-Paz 2019; Gasthaus et al. 2019; Pearce et al. 2018). The prediction intervals can help with decision making for retail sales optimization (Simchi-Levi et al. 2008), medical diagnoses (Begoli, Bhattacharya, and Kusnezov 2019; Mhaskar, Pereverzyev, and van der Walt 2017; Jiang et al. 2012), information safety (Smith, Dinev, and Xu 2011), financial investment management (Engle 1982), robotics and control (Buckman et al. 2018), autonomous transformation (Xu et al. 2014) and many others.

To estimate prediction intervals, we would need to estimate different levels of quantiles for the target distribution using quantile regression (Koenker and Regression 2005; Waldmann 2018). A real-world challenge is to select the parametric forms of target distributions, which is specified by the quantile function (also known as the inverse CDF function), to properly align with observed data distribution. Different choices for the target distribution (Gaussian, Poisson, Negative Binomial, Student-t etc.) may yield different quantile predictions, and misalignment of the assumption with the real distribution may hinder the performance of the model. Therefore, such heuristic or empirical hand-picking based parametric assumptions for the distribution can be sub-optimal. An approach based on learning from the data in an automated way, would be highly desirable, from both foundational and practical perspectives.

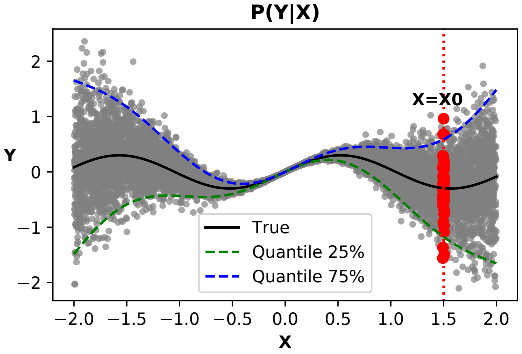

Figure 1: Modeling multiple quantiles at different condition-levels with a universal quantile function. The goal is to model target data distribution $y$ at any arbitrary quantile level and attribute level $X$, using one versatile quantile function. Gray dots are observed data points, while green and blue lines indicate 25% and 75% quantile levels. The data distribution $y$ varies at different levels of X, say variance of $y$ increases when $X$ is away from zero. Red dots are data points at $X = X_0$, $p(Y|X_0)$).

For learnable parametric modeling, one challenge is how to model all quantiles for all input attributes level in a computationally efficient way. First, modeling an any arbitrary quantile, as opposed to a couple of pre-defined quantile levels, offers a more comprehensive view on the target distribution, and provides convenience to use the quantile model (e.g. no need to re-train the model when quantiles at testing are different from the ones at training). Second, real-world data can have complex distributions beyond what simple assumptions can model. Fig. 1 shows different input attribute $X$ levels

*These authors contributed equally.

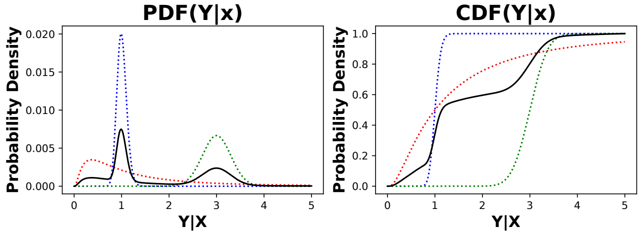

Figure 2: An example target distribution with a complex shape, in PDF and CDF space. Black lines are observed target distributions, in the form of mixture of the other three distributions shown with color. Fitting the black line accurately would be extremely difficult for most of the commonly-used single parametric splines, motivating for the use of learnable spline family composed of multiple splines.

have different dependency dynamics with target $y$ level (i.e. the variance of $y$ increases when $X$ apart from 0). Fig. 2 shows that the observed distribution cannot trivially fit well with one single distribution. Therefore, in order to model all quantiles at all $X$, we need a quantile function with a complexity that does not increase significantly with number of input attributes and the number of quantiles. This necessitates a versatile and highly-expressive quantile function.

There has been many efforts on improving various aspects of quantile regression. Gasthaus et al. (2019) proposes linear spline interpolation between knots in the inverse CDF space to model the target distribution in time-series forecasting setup. This is proposed to avoid the assumption on parametric form of the target distribution. Park et al. (2022) and Moon et al. (2021) focus on learning a valid quantile function without quantile crossing (e.g. quantiles violate monotonically increasing property), via special design of the neural network architecture or first-order inequality constraint optimization. Despite being distribution agnostic, these approaches for describing the target distribution (specified by quantile function) are restricted to one function family (e.g. linear spline), which may limit the expressiveness to represent the target distribution. In this paper, with the goal of designing an expressive quantile function for various quantiles and input levels, we propose a data-driven approach Neural Spline Search (NSS), which transforms the inputs with a series of monotonic spline regressions guided by symbolic operators. The contributions of our paper can be summarized as:

1. We propose an efficient search space and mechanism to find an expressive quantile function to model the data distribution, avoiding specifying a parametric form of the observed distribution as prior.
2. We propose a novel approach to generate an expressive quantile function using a combination of different distributions and operators guided by symbolic operators.
3. The proposed method can be incorporated into other tasks (including but not limited to time series forecasting) as their quantile function.
4. We demonstrate significant accuracy improvements across numerous regression or time series forecasting tasks. For example, on UCI benchmarks, we show 3.5%-7.0% improvement compared to next best methods.

# Related Work

**Quantile Regression** is used to estimate the target distribution at different quantile levels. The $\alpha$-quantile estimator is the solution when minimizing quantile loss at level $\alpha$ (Koenker and Bassett Jr 1978). Another quantile regression related loss is continuous ranked probability score (CRPS) (Gneiting and Raftery 2007), which is the averaging over all quantile levels, instead of one single quantile.

**Neural Network Quantile Forecasting**. To model sequential dependency of time series, several forecasting models propose a hidden state-emission framework ((Salinas et al. 2020; Wen et al. 2017; Gasthaus et al. 2019; de Bézenac et al. 2020; Wang et al. 2019)), where the dynamics of hidden states are modeled by auto-regressive recurrent neural works (e.g. LSTM), which takes previous hidden states and current observations as input and outputs current observation. Different from modeling the likelihood with parametric distributions (e.g. Gaussian (Salinas et al. 2020)), emission models for quantile estimation is to learn the parameters of quantile function. The overall framework is optimized by employing a quantile (Wen et al. 2017) or CRPS (Gasthaus et al. 2019) loss.

**Symbolic Regression** has shown great success in many fields, including program synthesis (Parisotto et al. 2016), mathematical expressions extraction (Cranmer et al. 2020), physics-based learning (Li et al. 2019; Petersen et al. 2019). As the search space is enormous and scaled exponentially with the length of operators, symbolic regression rule operators are usually set to be a small number and are learned by Monte Carlo Tree Search guided evolutionary strategies (Li et al. 2019) or reinforcement learning (Petersen et al. 2019).

# Methods

## Learning Quantile Function in Quantile Regression

Let the input data attributes $X$ and the target variable $y$ are jointly distributed as $p(X, y)$. The conditional cumulative distribution function (CDF) is $F(Y = y|X) = P(Y \leq y|X)$. The quantile function, which is also called the inverse CDF function, takes quantile level as inputs and returns a threshold value $Y$ below which random draws from the given CDF would fall quantile percent of the time. Specifically, the $\alpha$-th quantile function of $y|X = x$ is denoted as:

$$q(\alpha, x) = F^{-1}_{y|X=x}(\alpha) = \inf\{y : F(y|X = x) \geq \alpha\} \quad (1)$$

Here we can think the quantile function is to perform a transformation on a uniform-distributed random variable $\alpha \sim U(0, 1)$ to the target distribution $p(y|X)$. Quantile function is able to fully specify a distribution. So specifying the quantile function is describing the target distribution $p(y|X)$.

Quantile regression estimates different conditional quantile levels of the target variable given a certain level of input

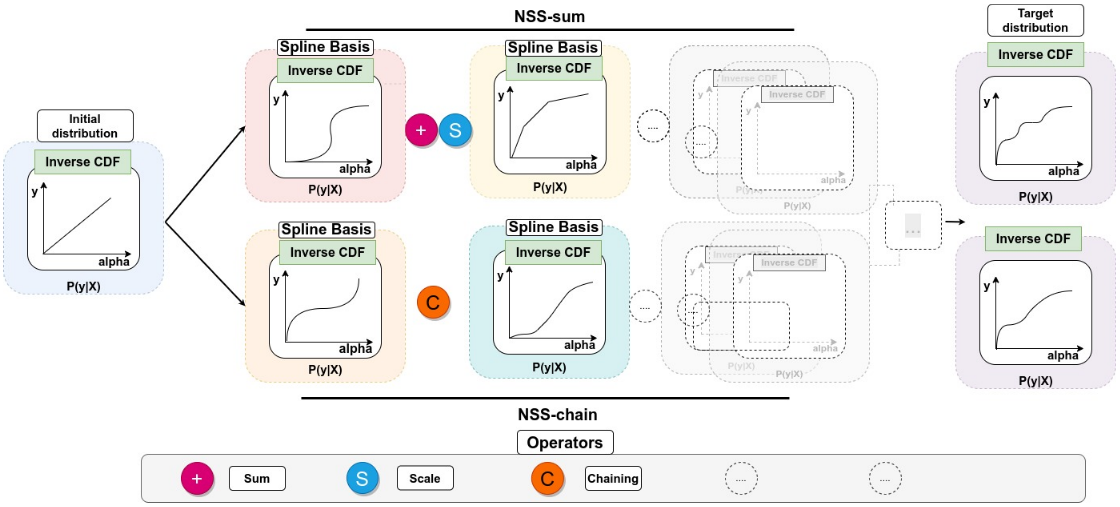

Figure 3: Overview of Neural Spline Search (NSS). Modeling the target data distribution can be done by learning the quantile function (e.g. inverse CDF), which maps a [0, 1]-variable (quantile) to a target value $y$. Unlike parametric methods which specify a distribution family and learn the parameters, NSS can generate the target distribution through a set of transformations on the inverse CDF space (quantile space), where the transformation is guided by a series of operators. Here, the bottom gray box shows possible operators (denoted as circles), including but not limited to summation ("+"), scale ("S"), and chaining ("C"). The basis splines are shown with color-shaded squares. The initial distribution is a uniform distribution, as shown in the leftmost panel (blue shaded), and the target distribution is the rightmost distribution (purple shaded). There is no obvious parametric distribution to achieve this transformation. Therefore, NSS is used to search for the suitable transformation through simple operators. In the first row of the middle panel, we show operators for *NSS-sum*, where the initial uniform distribution is transformed by the red- and the yellow-shaded splines (e.g. c-spline) through sum ("+") and scale ("S") operators. The second row shows the chaining transformation of the initial distribution, where the orange and cyan splines are used to transform the initial spline. The parameters of the splines are learned by a neural network. In general, the operators and transformations in NSS are not limited to two splines (we represent them as the gray splines next to the yellow and cyan shaded splines).

attributes, as opposed to regression, which estimates the conditional mean of the target variable. In quantile regression, a particular quantile level $\alpha$ of the conditional distribution of $y$ given $X = x$, $q(\alpha, x)$ is estimated by minimizing the *pinball loss* $\rho$ (or quantile loss), as the the quantile function $q$ is shown to be the minimizer of the expected pinball loss (Koenker and Bassett Jr 1978):

$$\rho^\alpha(y, q) = (y - q)(\alpha - \mathbf{1}(y < q)), \tag{2}$$

$$q(\alpha, x) = \arg\min_q \mathbb{E}_y[\rho^\alpha(y, q)]. \tag{3}$$

where $\mathbf{1}$ is the indicator function. One shortcoming of pinball loss is only measuring the loss at a single quantile level, which hinders the estimated $q$ for a global picture of the distribution (i.e. other $\alpha$ levels). On contrast, the *continuous ranked probability score* (CRPS) considers all quantile levels by integrating the pinball loss over $\alpha = [0, 1]$ (Matheson and Winkler 1976; Gneiting and Raftery 2007).

$$\text{CRPS}(y, q) = \int_0^1 2\rho^\alpha(y, q) d\alpha \tag{4}$$

As a proper scoring rule (Gneiting and Raftery 2007), CRPS is minimized when the quantile function is $q = F$. That is,

$$F_y^{-1} = \arg\min_q \mathbb{E}_y[\text{CRPS}(y, q)]. \tag{5}$$

Please refer (Koenker and Regression 2005) for detailed proof.

## Improving the Expressiveness of Quantile Function

Fig. 2 demonstrate the need of an expressive quantile function for modeling target distribution. Inspired from neural architecture search (NAS) (Elsken, Metzen, and Hutter 2019), we propose an approach to search for the suitable combination of distributions. The search is over different operations and basis distributions. We first introduce parametrization of quantile function, and the two non-parametric spline-based distributions.

**Parameterizing Quantile Functions** We propose to parameterize the quantile function $q_\theta(\alpha, x)$ using a deep neural network with parameters $\theta$. The quantile function is aimed to be accurate for any quantile levels $\alpha$ and input attributes level $X = x$. $X$ is high dimensional in real data, not as the one dimensional in the toy examples in Fig. 1 and Fig. 2.

**C-spline Distribution** The c-spline ($y^\alpha = q_\theta^{csplie}(\alpha, x)$) describes the CDF (Fig. 2, Right Panel) of a probability distribution $F_{y|X}$ by setting $K$ anchor points (denoted as knots) on the CDF curve and performing linear interpolation to fill in the gap between the knots. Specifically, the knots split CDF curve into bins and c-spline learns the width $w_i$ and height $h_i$ of bins by neural networks NN that depend on the input attributes level $X = x$.

$$\{w_i, h_i\}^K = \mathrm{NN}_\theta(x)$$
$$y^\alpha = r(\{w_i, h_i\}^K, \alpha) \quad \forall \alpha \in [0:1]$$

where $h_i$ and $w_i$ are non-negative delta values imposed by non-negative activation (i.e. Relu or Sigmoid), and the location of each bin (e.g. Y|X) is $L_i = \sum_{k=0}^{i} w_k$ and quantile level $\alpha_i = \sum_{k=0}^{i} h_k$. The accumulation sum design is to ensure that quantile function is monotically increasing and there is no quantile crossing. $r$ is a function to convert knots to output of quantile function: for quantile level $\alpha_i$ that is on the knots, we can directly read from $l_i$ , for quantile levels that are off the knots, quantile values can be computed through linear algebra operations on the two nearby knots $r(\alpha) = \begin{cases} l_i + \frac{(\alpha - \alpha_i)(l_j - l_i)}{\alpha_j - \alpha_i}, & \text{if } \alpha_i \leq \alpha \leq \alpha_j \quad 0 \leq i, j \leq K \\ l_k, & \text{if } h_k = \alpha \end{cases}$

**P-spline Distribution** The difference between p-spline from c-spline is having anchor knots in PDF space, instead of CDF space. Similarly with C-spline, P-spline also perform linear interpolation over knots, and the quantile level is achieved by integration over pdf via polynomial operations.

## Neural Spline Search (NSS)

We describe our proposed method, Neural Spline Search (NSS), which is overviewed in Fig. 3. Similar to symbolic regression (Parisotto et al. 2016; Li et al. 2019), NSS effectively searches in the space of discrete symbolic operators and distribution space for a candidate that can better fit the target data distribution. Specifically, let $T(O, S, k)$ denote the space of all transformations, via operators $O$ on all distribution $S$ with a maximum sequence length $k$. NSS aims to find the function $f(x)$ selecting operators and distributions in the space $T$ such that $\{f(x) \in T(O, S, k) : \ell(f(x), x_{train}) \leq \delta$ }, where $\ell$ denotes loss function CRPS, $x_{train}$ is training data and $\delta$ is the acceptance threshold. Given the large search space composed of combinations of numerous splines and operators, we restrict to use spline-based distribution as the basis distribution, and limit the operator search space to summation and chaining operations upon the transformation basis spline regressions. Note that this work can be easily extend to other operations and distributions, which we leave to future work. We describe the following NSS transformations as they are observed to work well consistently across different datasets: NSS with summation (NSS-sum) and NSS with chaining (NSS-chain). Algorithm 1 and Fig. 4(b)

### NSS-sum

NSS-sum performs transformations using the scale and summation operators. We represent this scenario with two splines: Spline 1: c-spline and Spline 2: p-spline, and two operators: scale $O1 : O(a) = \lambda a$ and summation $O2 : O(a, b) : a + b$; therefore, the overall transformation is (Spline 1-Operator 1) - (Spline 2-Operator 2), which yields: $f$ = c-spline + $\lambda$ p-spline.

**Algorithm 1:** Neural Spline Search

```
Operators = {+, $\times$, Scale, Chain, ...}
Splines = {c-spline, p-spline, Gaussian, Cauchy ...}
Data: Quantile level $\alpha \in [0, 1]$, $N$ data points
      $\{X \in \mathbb{R}^d, y \in \mathbb{R}^1\}_N, d \geq 1$, with chain
      depth $k$. Transform indicates the
      transformation using the input spline $S_\theta$ and
      operator $O$.
Result: $p(y|X)$ and $F^{-1}_{y|X}(\alpha)$
$k \leftarrow 1$;
while $k \leq K$ do
    Select $O = \{O_i\}_{no} \in$ Operators ;
    Select $S = \{S_j\}_{ns} \in$ Splines ;
    $\theta \leftarrow$ MLP$(X)$ ;
    $y_{pred} \leftarrow$ Transform$(S_\theta, O, \alpha)$;
    if $\alpha$ NSS-chain then
        Normalize $y_{pred}$ to $[0, 1]$ as $y'_{pred}$ ;
        $\alpha \leftarrow y'_{pred}$;
    else
        $X \leftarrow Y$          ▷ if X-NSS-chain ;
    end
    $k \leftarrow k + 1$;
end
```

Essentially, NSS-sum performs weighted sum of different splines. The motivation behind is that c-spline with fewer parameters can be more robust against overfitting, whereas p-spline increases the expressiveness of the splines.

### NSS-chain

Another proposed NSS design is NSS-chain. We focus on the chaining operator due to its expressiveness. This design is inspired by the success of normalizing flow (Rezende and Mohamed 2015), where a sequence of bijector transforms is utilized to transform distributions. Different from normalizing flow which has practical applicability challenges, NSS-chain only requires the forward pass of the transformation, not the inverse as normalizing flow does. This significantly reduces the computational complexity and broadens the feasibility of transformations. As mentioned, quantile function takes input attributes level $(X)$ to predict the target value $(y)$ at quantile level $(\alpha)$.

$$y = q_\theta(X, \alpha), \tag{6}$$

where $X \in R^m$ and $\alpha \in [0, 1]$. We present two designs to chain different transformations (see Fig. 4 (a)). We note that chaining of transformation is not limited to the two designs.

- **$\alpha$-chaining**
  The $\alpha$-chaining is when we consider the condition level $(X)$ unchanged during the chain of transformation, and the output of each transformation is a scaled version of quantile level for the next transformation. In particular, after each transformation, we normalize the output $y$ to be in the range $[0, 1]$, and then the normalized output is re-input as the new $\alpha$ to the next transformation. This is

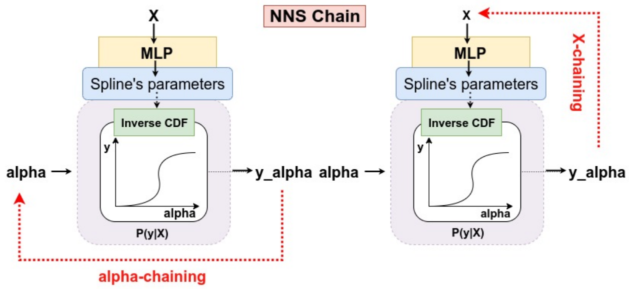

Figure 4: (a) Illustration of NSS-chain methods. The diagram demonstrates chaining for NSS-chain. Left: $\alpha$-chaining. The output $y$ of the spline, after re-scaling to [0, 1], is re-inputted to the quantile spline at quantile level $\alpha$. Right: $X$-chaining. The output $y$ is instead re-inputted to the quantile spline as $X$. Both rely on input attributes $X$.

repeated until the maximum depth is reached. This design is more similar with normalizing flow methods.

$$y = q_{\theta_K}(X, ...f_n(q_{\theta_2}(X, f_n(q_{\theta_1}(X, \alpha))))) \qquad (7)$$

$\theta_k$ for $k$=1,2,..$K$ are parameters for different splines in K-length chain. $f_n$ is the normalization function.

- **$X$-chaining**
  $X$-chaining is when we consider quantile level $\alpha$ level is unchanged during chaining, as each transformation learns a suitable condition level (or feature) for next iteration. Similarly with $\alpha$-chaining in the iterative manner, except that the output $y$ of each transformation is projected to generate $X$ for the next iteration of Eq. 6.

  $$y = q_{\theta_K}(...q_{\theta_2}(q_{\theta_1}(X, \alpha), \alpha), \alpha) \qquad (8)$$

  The advantage of this approach, compared tp $\alpha$-chaining, is that we keep quantile levels $\alpha$ unchanged, and re-normalizing output is not needed.

**Remarks on NSS: . (1) why a simple spline-based algorithm, e.g. C-spline, is not enough?** Although in theory spline-based algorithms can represent any arbitrary distributions with sufficiently high number of knots $K$, in practice, we find a large $K$ often lead to unstable training, as also studied in (Park et al. 2022). In contrast, we find the combination (combined or chained) over a relatively restricted splines are more robust in capturing the overall of the target distribution **(2) Include both spline-based distribution and classic parametric distribution** In addition to spline-based distribution, we also encourage incorporating parametric distribution (e.g. Gaussian) as basis distribution for NSS, especially when prior knowledge (say Gaussian noise) is available. Because, it is challenging for spline based methods to reconstruct Gaussian distribution even with infinite number of knots; and , the benefits of combining the two are the parametric distribution offers advantage of classic statistics and robust to noise, and the non-parametric spline offers flexibility.

## Training

Once we select the operators and splines, the parameters of the splines are trained in an end-to-end way by optimizing CRPS (Eq. 4).

**Algorithm 2:** Training with CRPS

```
Data: $N$ data points $\{X_i \in \mathbb{R}^d, y_i \in \mathbb{R}^1\}_{i=1}^N$, $m$
      quantile levels, $T$ transformation, which
      takes selected splines $S_{\text{select}}$ and selected
      operators $O_{\text{select}}$ from NSS. $lr$ is learning
      rate.
Result: Neural network weights $\theta$
$e \leftarrow 1$;
while $e \leq Nepoch$ do
    f = Transform($S_{\text{select}}$, $O_{\text{select}}$) $\ell \leftarrow 0$ ;
    for $\alpha$ in [0, $\frac{1}{m}$, $\frac{2}{m}$, ..1] do
        $y_\alpha^{pred}$ = f$_\theta$($X$, $\alpha$) ;
        $\ell \leftarrow \ell$ + pinball_loss ($y_\alpha^{pred}$, y, $\alpha$)
    end
    CRPS = $\ell/m$ ;
    $\theta \leftarrow \theta - lr \cdot \nabla_\theta$ CRPS ;
    $e \leftarrow e + 1$;
end
```

Algorithm 2 overviews the training of NSS for spline parameter selection. Because of the form of the transformations, the analytical solution of CRPS integration is intractable. Thus, we use a Monte Carlo estimation for the CRPS loss. In particular, we sample $m$ number of $\alpha$ values from the range of $[0, 1]$ and average them for the corresponding pinball loss. Specifically, during training, we fit parameters by optimizing over with the empirical mean of CRPS over $N$ data points:

$$\theta^* = \arg\min_\theta 1/N \sum_{i=1}^{N} \mathbb{E}_y[\text{CRPS}(y, q_\theta(X_i, \alpha))]. \qquad (9)$$

# Experiments

## Comparison Methods

**QD** (Pearce et al. 2018) generates prediction intervals (PIs) for estimating uncertainty for regression tasks with the assumption that high-quality PIs should be as narrow as possible. **Deep Quantile Aggregation** (Kim et al. 2021) proposes weighted ensembling strategies where aggregation weights vary over both individual models and feature values plus (pairs of) quantile levels. The monotonization layer in the network is applied to avoid crossing of quantile estimates. **RQspline** (Durkan et al. 2019) proposes a fully-differentiable module based on monotonic rational-quadratic splines, which enhances the flexibility of coupling and autoregressive transforms while retaining analytic invertibility. **Global-Coarse** (Ratcliff 1979) provides an analysis of distribution statistics of group reaction time distributions. **MLE (NB)** and **Mix. MLE** are Negative Binomial and mixture likelihood based methods (Awasthi et al. 2021). **C-spline** is proposed in (Gasthaus et al. 2019), where C-spline is used as the quantile function in time-series forecasting.

## Metrics

For point predictions, we focus on the following metrics: Mean absolute error (MAE): $\frac{1}{n}\sum_{t=1}^{n}|T_t - P_t|$ where $T_t$ and

| **Methods** | **Boston** | **Concrete** | **kin8nm** | **Power** | **Protein** | **Wine** |
|---|---|---|---|---|---|---|
| **Gaussian** | 0.0754 | 0.0564 | 0.048 | 0.0449 | 0.2116 | 0.0978 |
| **QD** | 0.5003 | 0.4150 | 0.3945 | 0.3688 | 0.6689 | 0.4456 |
| **RQspline** | 0.0917 | 0.0622 | 0.0479 | 0.0485 | 0.2153 | **0.0912** |
| **p-sline** | 0.0778 | 0.0570 | 0.0444 | 0.0453 | — | 0.0966 |
| **c-spline** | 0.0806 | 0.0543 | 0.0430 | 0.0447 | 0.2002 | 0.0947 |
| **NSS-X-chain** | 0.0787 | 0.0588 | 0.0430 | 0.0448 | 0.2052 | 0.0962 |
| **NSS-$\alpha$-chain** | 0.0846 | 0.0568 | 0.0417 | 0.0448 | 0.2067 | 0.0976 |
| **NSS-sum** | **0.0709** | **0.0512** | **0.0414** | **0.0442** | **0.1949** | 0.0957 |
| **Gain percentage** | 12.0% | 17.7% | 3.7% | 1.1% | 2.6% | - |

Table 1: Mean Absolute Error (MAE) on UCI benchmarks. Test performance of the proposed method (NSS) and existing methods on UCI benchmarks. We use the 50th quantile estimator as our estimates. The dash indicates unavailability. The shaded area is the proposed methods. Bold is the top one. Lower is better. Gaussian: Gaussian kernel; QD is quantity-driven methods proposed in (Pearce et al. 2018); RQ spline proposed in (Durkan et al. 2019); c-spline proposed in (Gasthaus et al. 2019). Boston, Concrete, Power is short for Boston Housing, Concrete Strength, Power Plant. Gain percentage is computed as (best nss - best baseline)/best baseline.

$P_t$ are true and predicted value; Mean Absolute Percentage Error (MAPE): $\frac{1}{n}\sum_{t=1}^{n}|\frac{T_t-P_t}{T_t}|$. Weighted Average Percentage Error (WAPE): $\frac{\sum_{t=1}^{n}|T_t-P_t|}{\sum_{t=1}^{n}|T_t|}$; and Root Mean Square Error (RMSE): $\sqrt{\frac{\sum_{t}^{N}(T_t-P_t)^2}{n}}$. For quantile predictions, we use the Pinball Loss (Eq. 2), with $50\%$-th, **Q50**; $90\%$-th, **Q90**; and $10\%$-th **Q10** quantiles.

### Training

For simplicity, the proposed NSS methods use depth-2 splines, which contain {(c-spline, p-spline), (c-spline, p-spline), (c-spline, c-spline), (p-spline, p-spline)}. **NSS-sum** is tuned with $\lambda$ in the range of $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$. **NSS-chain** normalizing of $y$ in $\alpha$ chaining can be achieved by applying sigmoid layer or scaling by max value. As splines are monotonically-increasing functions, the spline value $y$ with $\alpha = 0$ is the minimum value of $y$ and $\alpha = 1$ yields the maximum value of $y$. *Scale* is $y_{scale} = \frac{y-y_{min}}{y_{max}-y_{min}}$. We use a batch size=128 and a learning rate of $0.005$ for $100$ epochs.

## Results

To demonstrate the effectiveness of proposed methods, we conduct experiments on synthetic, real-world tabular regression, and time series forecasting datasets.

### Synthetic Data

**Dataset**. We generate 2000 data points ($X \in \mathbb{R}^1$ and $y \in \mathbb{R}^1$), where $X$ is in the range of $[-2, 2]$ and $y$ has Gaussian distribution $y \sim \mathcal{N}(0.3\sin(3x), 0.2x^2)$, where sin is the sinusodial function. We construct the validation and test sets to come from the same distribution. Unlike real-world data, the synthetic data would have known quantile levels, that can be used for evaluating the accuracy of quantile estimates. We make the task more challenging by setting a data-dependent variance for the Gaussian noise to evaluate the ability of learning condition-specific quantile values. Fig. 5 shows that the proposed NSS-chain and NSS-sum can capture the true underlying quantiles, whereas QD (Pearce et al. 2018) struggles on the varying variance locations (e.g. around $x = 0$). The upper and lower black lines are the predicted 2.5%-th and 97.5%-th quantiles for the observed data (e.g. red dots), shown along with the ground truth quantiles (e.g. shaded red area). The results indicate that more expressive NSS transformations are superior in more challenging scenarios, where true data points are distributed differently (e.g., distributions depend on the value of the inputs"). Fig. 6 shows the calibration plot of the predicted vs. true distributions at different quantile levels. Here, we show the true percentile $p$ as the fraction of data in the dataset such that the $p$ percentile of the predictive distribution is larger than the ground truth data. The perfect prediction would be the diagonal line. Fig. 6 indicates that the proposed methods NSS-sum and NSS-chain can capture the proposed true distribution at various levels by close to the red line, whereas QD does not fit as well.

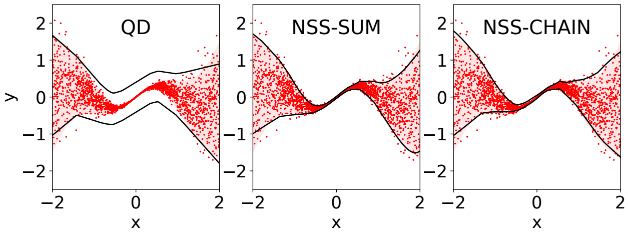

Figure 5: NSS on Synthetic data. We compare the performance of proposed NSS against existing methods $QD$ (Pearce et al. 2018). The red dots are observed data points, shaded red area is the ground truth $2.5\%$ and $97.5\%$ quantile levels, and the dark black lines are the predicted $2.5\%$ and $97.5\%$ quantile levels.

### Real-world Tabular Regression

We use UCI benchmarks (Dua and Graff 2017)

We evaluate the accuracy for both point predictions and quantiles. As the point predictions, we use the 50th quantile estimator as our estimates. Table 1 shows that the proposed NSS methods outperform the other existing methods on most datasets in mean absolute error (MAE). We observe that the NSS-sum performs better than NSS-chain. For quantile

| Methods | Boston | Concrete | kin8nm | Power | Protein | Wine |
|---|---|---|---|---|---|---|
| **Gaussian** | 0.0276 | 0.0203 | 0.0171 | 0.0158 | 0.0725 | 0.0357 |
| **Global-Coarse**$^*$ | 0.0745 | 0.0596 | 0.0681 | 0.0473 | 0.1321 | — |
| **Deep Quantile Aggregation**$^*$ | 0.0754 | 0.0541 | 0.0684 | 0.0441 | 0.1253 | — |
| **QD** | 0.1212 | 0.1076 | 0.1004 | 0.0972 | 0.1547 | 0.1164 |
| **RQspline** | 0.0458 | 0.0418 | 0.0203 | 0.0189 | 0.0863 | 0.0424 |
| **p-sline** | 0.0308 | 0.0211 | 0.016 | 0.0160 | — | 0.0358 |
| **c-spline** | 0.0312 | 0.0198 | 0.0157 | 0.0159 | 0.0688 | **0.0351** |
| **NSS-X-chain** | 0.0311 | 0.0216 | 0.0165 | 0.0162 | 0.0707 | 0.0358 |
| **NSS-$\alpha$-chain** | 0.0322 | 0.0208 | **0.0151** | 0.0159 | 0.0726 | 0.0363 |
| **NSS-sum** | **0.0265** | **0.0191** | 0.0152 | **0.0157** | **0.0674** | 0.0357 |
| **Gain percentage** | 4.0% | 3.5% | 3.8% | 0.6% | 7.0% | - |

Table 2: Average pinball loss on UCI benchmarks. The test pinball loss (the lower, the better) is over 99 quantile levels, $\alpha = \{0.01, 0.02, ...0.99\}$. The compared methods are Global-Coarse proposed in (Ratcliff 1979); QD (Pearce et al. 2018); Deep Quantile Aggregation (DQA) (Kim et al. 2021); RQspline (Durkan et al. 2019); $*$ indicates entries are from (Kim et al. 2021) (under the same experiment setup).

| Methods | MAPE | WAPE | RMSE | Q50 | Q90 | Q10 |
|---|---|---|---|---|---|---|
| **MLE (NB)** | **0.44434** | 0.27240 | 7.70958 | 0.27240 | 0.10907 | 0.15275 |
| **Mix MLE** | 0.44839 | 0.26838 | 7.22556 | 0.26838 | 0.10293 | 0.14508 |
| **c-spline** | 0.44672 | 0.26635 | 7.06332 | 0.26635 | **0.10238** | 0.14241 |
| **p-spline** | 0.44912 | 0.26834 | 7.14643 | 0.26834 | 0.10343 | 0.14333 |
| **NSS-sum** | 0.44501 | 0.26545 | 6.96697 | 0.26545 | **0.10238** | 0.14266 |
| **NSS-chain** | 0.44883 | **0.26420** | **6.91726** | **0.26420** | 0.10243 | **0.14149** |

Table 3: Performance comparisons for time series forecasting on M5. Different evaluation metrics are included in this table for M5. Detailed descriptions of the metrics are in Sec . $Q_k$ indicates the pinball loss of $k$-th quantile. e.g. $Q50$ is the pinball loss of 50th quantile. Lower is better.

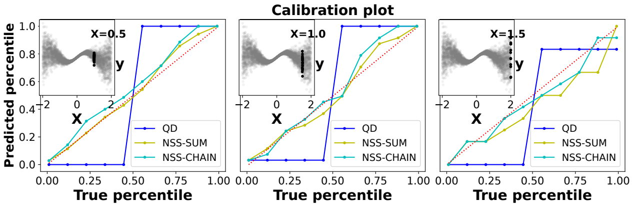

Figure 6: Calibration plots. Predicted vs. ground truth percentiles at condition levels: $X$=0.5, 1.0 and 1.5. The perfect calibration would correspond to the diagonal (red dotted) line.

metrics, we use the pinball loss (Eq. 2) over 100 quantile levels $\alpha = \{0.01, 0.02, ...0.99\}$ in Table 2. The results indicate that NSS consistently outperforms other alternatives across different UCI benchmarks. In pinball loss, NSS-sum performs better than NSS-chain. We attribute the superiority of NSS-sum for regression to make balance between different transformation, which is helpful in explaining the variance in the data.

### Retail Demand Forecasting

For time series forecasting, we focus on the M5 dataset, which contains time-varying sales data for retail goods, along with other relevant covariates like price, promotions, day of the week, special events etc. It represents an important real-world scenario, where the accurate estimation of the output distribution is crucial, as retailers use them to optimize prices or promotions.

The time series forecasting experiments are conducted by performing one-step ahead prediction, yielding predictions in an autoregressive way. Table 3 shows the results of our method compared to other alternatives. We observe consistent outperformance of NSS in various forecasting evaluation metrics. Different from regression tasks, we observe that NSS-chain is better than NSS-sum, indicating its benefit in capturing time-dependent relationship.

**Remarks** on NSS-sum vs NSS-chain. The results show that NSS-sum is superior on regression, while NSS-chain has advantages on time series forecasting. The observations may indicate NSS-sum is suitable for more constrained tasks (e.g. regression, one time step time series-forecasting), where being moderately expressive would suffice. NSS-sum is also more robust and easier to train. On the other hand, NSS-chain may be more expressive, which is beneficial to fit tasks requires more complex distributions at different time steps of the time series, but for individual step NSS-chain is not as accurate as NSS-sum in fitting the distribution.

## Conclusion

We propose a novel approach for modeling uncertainty. The proposed *Neural Spline Search (NSS)* method employs a series of monotonic spline regression transformations, guided by symbolic operators. We demonstrate the effectiveness of NSS for superior modeling of output distributions, on both synthetic and real-world datasets. We leave the extensions to different operators and splines, including parametric distribution transformations to future work.

# References

Awasthi, P.; Das, A.; Sen, R.; and Suresh, A. T. 2021. On the benefits of maximum likelihood estimation for Regression and Forecasting. *arXiv preprint arXiv:2106.10370*.

Begoli, E.; Bhattacharya, T.; and Kusnezov, D. 2019. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1): 20–23.

Buckman, J.; Hafner, D.; Tucker, G.; Brevdo, E.; and Lee, H. 2018. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *arXiv preprint arXiv:1807.01675*.

Cranmer, M.; Sanchez-Gonzalez, A.; Battaglia, P.; Xu, R.; Cranmer, K.; Spergel, D.; and Ho, S. 2020. Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv:2006.11287*.

de Bézenac, E.; Rangapuram, S. S.; Benidis, K.; Bohlke-Schneider, M.; Kurle, R.; Stella, L.; Hasson, H.; Gallinari, P.; and Januschowski, T. 2020. Normalizing kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems*, 33: 2995–3007.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Durkan, C.; Bekasov, A.; Murray, I.; and Papamakarios, G. 2019. Neural spline flows. *NeurIPS*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1): 1997–2017.

Engle, R. F. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the econometric society*, 987–1007.

Gasthaus, J.; Benidis, K.; Wang, Y.; Rangapuram, S. S.; Salinas, D.; Flunkert, V.; and Januschowski, T. 2019. Probabilistic forecasting with spline quantile function RNNs. In *AISTATS*.

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

Jiang, X.; Osl, M.; Kim, J.; and Ohno-Machado, L. 2012. Calibrating predictive model estimates to support personalized medicine. *Journal of the American Medical Informatics Association*, 19(2): 263–274.

Kim, T.; Fakoor, R.; Mueller, J.; Smola, A. J.; and Tibshirani, R. J. 2021. Deep Quantile Aggregation. *arXiv preprint arXiv:2103.00083*.

Koenker, R.; and Bassett Jr, G. 1978. Regression quantiles. *Econometrica: journal of the Econometric Society*, 33–50.

Koenker, R.; and Regression, Q. 2005. Econometric Society Monographs. *Quantile regression*.

Li, L.; Fan, M.; Singh, R.; and Riley, P. 2019. Neural-guided symbolic regression with asymptotic constraints. *arXiv preprint arXiv:1901.07714*.

Matheson, J. E.; and Winkler, R. L. 1976. Scoring rules for continuous probability distributions. *Management science*, 22(10): 1087–1096.

Mhaskar, H. N.; Pereverzyev, S. V.; and van der Walt, M. D. 2017. A deep learning approach to diabetic blood glucose prediction. *Frontiers in Applied Mathematics and Statistics*, 3: 14.

Moon, S. J.; Jeon, J.-J.; Lee, J. S. H.; and Kim, Y. 2021. Learning multiple quantiles with neural networks. *Journal of Computational and Graphical Statistics*, 30(4): 1238–1248.

NASA. 2015. Pluto: The 'Other' Red Planet. https://www.nasa.gov/nh/pluto-the-other-red-planet. Accessed: 2018-12-06.

Parisotto, E.; Mohamed, A.-r.; Singh, R.; Li, L.; Zhou, D.; and Kohli, P. 2016. Neuro-symbolic program synthesis. *arXiv preprint arXiv:1611.01855*.

Park, Y.; Maddix, D.; Aubet, F.-X.; Kan, K.; Gasthaus, J.; and Wang, Y. 2022. Learning quantile functions without quantile crossing for distribution-free time series forecasting. In *International Conference on Artificial Intelligence and Statistics*, 8127–8150. PMLR.

Pearce, T.; Brintrup, A.; Zaki, M.; and Neely, A. 2018. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *ICML*.

Petersen, B. K.; Larma, M. L.; Mundhenk, T. N.; Santiago, C. P.; Kim, S. K.; and Kim, J. T. 2019. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*.

Ratcliff, R. 1979. Group reaction time distributions and an analysis of distribution statistics. *Psychological bulletin*, 86(3): 446.

Rezende, D.; and Mohamed, S. 2015. Variational inference with normalizing flows. In *ICML*.

Salinas, D.; Flunkert, V.; Gasthaus, J.; and Januschowski, T. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191.

Simchi-Levi, D.; Kaminsky, P.; Simchi-Levi, E.; and Shankar, R. 2008. *Designing and managing the supply chain: concepts, strategies and case studies*. Tata McGraw-Hill Education.

Smith, H. J.; Dinev, T.; and Xu, H. 2011. Information privacy research: an interdisciplinary review. *MIS quarterly*, 989–1015.

Tagasovska, N.; and Lopez-Paz, D. 2019. Single-model uncertainties for deep learning. *Advances in Neural Information Processing Systems*, 32.

Waldmann, E. 2018. Quantile regression: a short story on how and why. *Statistical Modelling*, 18(3-4): 203–218.

Wang, Y.; Smola, A.; Maddix, D.; Gasthaus, J.; Foster, D.; and Januschowski, T. 2019. Deep factors for forecasting. In *International conference on machine learning*, 6607–6617. PMLR.

Wen, R.; Torkkola, K.; Narayanaswamy, B.; and Madeka, D. 2017. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*.

Xu, W.; Pan, J.; Wei, J.; and Dolan, J. M. 2014. Motion planning under uncertainty for on-road autonomous driving. In *ICRA*.