Neural Network Custom Loss Function for Flexible and High-Quality Prediction Intervals

By

Haoyuan Wang

A thesis submitted in partial fulfillment

of the requirements for the qualifying exam of

Doctor of Philosophy

in

Computational Science

Middle Tennessee State University

August 2024

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to all those who contributed to this study. First, we express our sincere gratitude to ten public dataset, who provided the data that was the foundation for this project. Their valuable contributions are essential to our understanding and the development of better diagnostic and intervention strategies. We sincerely thank Dr. Donglin Wang and Dr. Qiang Wu for their expertise, guidance, and collaboration throughout the project. Their insights and dedication have greatly enriched our research process and results. Finally, we express our gratitude to all individuals who contributed to the realization of this study, no matter how big or small. Your joint efforts have contributed to the pursuit of knowledge in the field of neurological disorders and the classification and diagnosis of autism. Thank you all for your unwavering support and commitment.

ABSTRACT

In the realm of deep learning, interval prediction plays a crucial role in diverse applications. However, the rise of complex challenges in these domains requires more flexible and high-quality prediction intervals. To address this need, we present a novel custom loss function specifically designed for deep learning models. This function optimizes both prediction interval width and coverage, boasting superior flexibility compared to traditional approaches. Specifically, our loss function utilizes multiple penalty coefficients to precisely control the interval width. Higher coefficients favor wider intervals to guarantee coverage, while lower ones prioritize narrower intervals for improved prediction accuracy. This delicate balance allows the model to adapt to different situations and application demands. Extensive experiments on ten public datasets across various domains reveal the impressive performance of this custom loss function. It consistently generates high-quality prediction intervals with remarkable generalization capabilities, readily adapting to diverse application scenarios.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I.
INTRODUCTION

Prediction interval (PI) is a promising tool for quantifying effects of uncertainties on predicted values [12]. It has been widely used in finance, healthcare, biology and other fields. It is comprised of upper bound and lower bound that can bracket a future unknown value with a desired probability called a confidence level [(1-α)%] [11]. In the traditional context, point estimation obtains an estimate of the population parameter by adding and subtracting the estimation error of the sample statistic. It is simple and easy to understand and can provide estimates of overall parameters, and is also widely used. However, since this method uses sampling indicators to directly replace the overall indicators, errors will inevitably occur. Unlike point estimation, PI can give a probability measure of the closeness of the sample statistic to the population parameter based on the sampling distribution of the sample statistic. Since traditional point estimation cannot handle the uncertainty in the process well, and the PI itself is affected by outliers, which may lead to inaccurate estimated intervals and ignore the possible impact of small probability events, in practical applications in order to obtain numerical values estimation and its reliability place high demands on the construction and methods of PIs. Generally, there are two principles to construct PIs. The first one is the coverage probability. The other one is the interval width. High-quality PIs aim to ensure a high coverage probability while minimizing the interval width. The challenge lies in balancing these principles to construct PIs that are not only statistically robust but also practically meaningful. This involves accounting for uncertainties and outliers, ensuring that the estimated intervals are reliable and informative in real-world applications [10].

The availability of PIs provides decision-makers and operational planners with an efficient means to quantify the level of uncertainty associated with point forecasts. It allows for the consideration of multiple solutions or scenarios, accommodating both optimal and worst-case conditions [11]. With the increasing diversity and complexity of the data, nonlinear relationship also becomes more important. Deep neural networks (DNNs) are introduced into the construction of PIs under this circumstance [1, 5, 8].

In the realm of existing research, the pursuit of the highest possible quality in constructing PIs has been a predominant focus. However, it is crucial to acknowledge that different fields exhibit varying tolerances for errors, specifically E_U (error between upper bound and true value) and E_L (error between true value and lower bound) are different. A poignant example can be found in the financial sector, where institutions are understandably cautious about transactions involving customers with low credit. In the context of predicting customer credit, stringent control over the tolerance for E_L is imperative, while amore flexible approach can be taken about E_U. This paper strategically aligns with the aforementioned idea, incorporating both quantile regression and robust regression methodologies. By doing so, it not only ensures the delivery of high-quality predictions, but it also facilitates the creation of a prediction interval that can be dynamically adjusted to cater to the unique considerations of specific real-world problems. Integrating quantile regression allows for a targeted focus on specific percentiles, addressing asymmetry in error tolerance. Simultaneously, robust regression techniques enhance the model's resilience to outliers, contributing to the reliability and adaptability of the prediction interval in practical applications. Furthermore, the assessment of model performance is enriched by benchmarking comparisons, drawing inspiration from recent work of Pearce et al. [17]. This benchmarking approach provides valuable insights by evaluating our proposed methodology in the context of contemporary advancements, establishing a comprehensive understanding of its efficacy and potential advantages in comparison to existing models.

CHAPTER II.
LITERATURE REVIEW

In this section, we consider methods used to predicted interval. We divide methods into two categories according to whether the distribution of the data is known. For data with normal distribution, and the standard deviation (σ) is known, the prediction interval can be calculated using the following formula

$$\bar{x} \pm z_{\alpha/2} \cdot \sigma \sqrt{1 + \frac{1}{n}}$$

For data with normal distribution, but the standard deviation (σ) is unknown, the following formula is used.

$$\bar{x} \pm t_{\alpha/2,n-1} \cdot s \sqrt{1 + \frac{1}{n}}$$

where $\bar{x}$ is the sample mean, n is the sample size, σ is the known population standard deviation, s is the sample standard deviation, $z_{\alpha/2}$ is the critical value from the standard normal distribution, $t_{\alpha/2,n-1}$ is the critical value from the t-distribution with $n-1$ degrees of freedom, α is the significance level.

For data with unknown distribution, there are two popular method to predict interval:

- The **Quantile regression** (QR) [20], originated from the field of econometrics, is capable of reflecting the impact of the independent variables on the induced variables. The QR models use the least absolute deviations method to minimize the absolute values of the errors[15]. Commonly used loss functions include quantile loss function and pinball loss function.

- The **Bootstrap** method [18, 21] is a statistical technique used for estimating the sampling distribution of a statistic by resampling with replacement from the observed data. The primary goal of the Bootstrap method is to provide a robust and computationally efficient way to estimate the uncertainty associated with a sample statistic, without relying on traditional parametric assumptions.

The construction of PIs traditionally relies on theoretical approaches assuming unbiased forecast models and forecast errors adhering to a determined distribution with zero mean [14]. However, the diversity and complexity of actual data make us know very little about the information of the data. Therefore, methods of constructing PIs based on deep neural networks (DNNs) were proposed. The Delta method is a specific approach proposed to construct PIs utilizing DNNs as forecast models [14]. This method incorporates theories used in building confidence intervals for general non-linear regression models and involves estimating model uncertainty. However, it is computationally demanding, requiring the use of the Hessian matrix [9, 17]. Similar to the Delta method, Bayesian technique is also used to predict interval. In this approach, parameters in a neural network (or any model) are viewed not as single values but as a distribution of values representing various degrees of belief. The objective is to find the posterior probability distribution for the weights after observing the dataset. If the MCMC algorithm is used, the computational cost may be too high [23]. Therefore, another method called Mean-variance estimation (MVE) method with easy calculation [16] was proposed. It directly construct PIs using two DNNs models. The main assumption of this technique is to consider a normal distribution for forecast errors around the true target mean y [7]. For two DNNs models, one represents the mean and the other represents the variance of a normal distribution. This dual-model setup enables the estimation of data noise variance. The loss function used is the Negative Log Likelihood (NLL) of the predicted distribution given the data [17].

Most of the previous methods require certain assumptions or preconditions during their implementation, which makes their application to real-world data highly contingent on the accuracy of these assumptions and preconditions. Recognizing this limitation, recent years have seen the emergence of novel methods that do not rely on such assumptions. One such method is the lower upper bound estimation (LUBE) method, which constructs an DNN with two outputs for estimating the prediction interval bounds of PIs. DNN model training involves minimizing a proposed PI-based objective function, covering both interval width and coverage probability. Crucially, this method does not necessitate any prior information about the upper and lower bounds of PIs for training the DNN [10]. On this basis, taking into account the uncertainty of the model, Tim Pearce et al. [17] proposed another interval prediction method (QD method) based on High-Quality principle and Distribution-Free. The QD method considers the generation of prediction intervals (PIs) by neural networks for quantifying uncertainty in regression tasks. It is axiomatic that high-quality PIs should be as narrow as possible, whilst capturing a specified portion of data. The loss function is directly derived from this axiom, requiring no distributional assumption. Benchmark experiments show the method outperforms current state-of-the-art uncertainty quantification methods, reducing average PI width by over 10%. Both the LUBE and QD methods are grounded in the high-quality principle, thereby avoiding the impact of restrictive assumptions. These methods have found widespread application in various fields, including forecasting of streamflow discharges [19], wind power and speed interval prediction [13], air quality prediction [6], and so on.

However, there is a problem that these two methods use NNs to find the upper and lower bounds with lowest loss which will give us a fixed interval. Actually, we hope to find a method for interval prediction that can be flexible and variable according to actual conditions under High-Quality principle. At the same time, the impact of outliers must also be considered. So robust model can be used, such as the one proposed by the study [22].

For instance, in medical research, different patient groups may have different requirements for upper and lower bounds of interval prediction. Consider acute hypotension, a significant risk factor for in-hospital mortality at intensive care units. Prolonged hypotension can lead to tissue hypoperfusion, leading to cellular dysfunction and severe injuries to multiple organs. Consequently, timely medical interventions are crucial for managing acute hypotensive episodes [3]. In this case, we need to pay more attention on the lower bound. Similarly, within the realm of housing finance, predicting default among low-income families remains a critical area of interest due to its direct and lasting impact on the economy [2]. If the lending conditions of a credit institution are very strict, then the lower bound should be paid attention to. As long as the user is at risk of indebtedness, the credit institution will terminate the loan. If the borrowing conditions are loose, then you should focus on the upper bound. As long as the user has a certain possibility of paying back the loan, then the credit institution can borrow.

Considering these factors, we designed a bespoke loss function that is adept at fine-tuning the upper and lower bounds while adhering to the high-quality principle. This method enables the adaptable calibration of prediction intervals, accommodating various real-world scenarios and domain-specific requirements, showcasing resilience against outliers. Essentially, encapsulated within a custom loss function, our approach harmonizes flexibility, robustness, and high-quality principles. It dynamically adjusts prediction intervals to meet the unique demands of diverse research domains.

CHAPTER III.
LOSS FUNCTION

Derivation

The derivation of the flexible loss function is based on the High-Quality principle mentioned above. The input data is divided into independent variable X and dependent variable y for n data points and with $x_i \in \mathbb{R}^D$ denoting the ith D dimensional input corresponding to y_i. For each y_i, we will create upper bound prediction $\hat{y}_{Ui}$ and lower bound prediction $\hat{y}_{Li}$. To formulate a robust prediction interval, we establish the following conditions::

- Upper bound prediction $\hat{y}_{Ui}$ should be greater than lower bound prediction $\hat{y}_{Li}$.

$$\hat{y}_{Ui} > \hat{y}_{Li}$$

- The true value y_i should be in the prediction interval.

$$\hat{y}_{Ui} > y_i > \hat{y}_{Li}$$

- The true value y_i should be approximately equal to the average of upper bound prediction $\hat{y}_{Ui}$ and lower bound prediction $\hat{y}_{Li}$.

$$y_i = \frac{\hat{y}_{Li} + \hat{y}_{Ui}}{2}$$

- Prediction interval should cover some desired proportion $(1-\alpha)$ of the true value y_i. Commonly, α is 0.05 or 0.01.

$$Pr(\hat{y}_{Ui} > y_i > \hat{y}_{Li}) \geq (1-\alpha)$$

According to the conditions, we firstly introduce quantile regression as the first part of custom loss function. From the regression angle, Quantile regression is a statistical modeling approach that focuses on understanding how changes in an independent variable affect specific percentiles (quantiles) of the distribution of the dependent variable. Instead of estimating the mean relationship (as in ordinary least squares or OLS regression), quantile regression allows us to analyze how the relationships vary across different points in the distribution. In essence, it involves dividing the dataset into segments based on the values of the dependent variable and examining the regression relationships within each segment. The key idea is to assign different weights to the residuals of the observations based on their position relative to a specified quantile. For example, residuals on the left side of the quantile are given a weight of α, while residuals on the right side are given a weight of $(1-\alpha)$.

In this project, we adjust the quantile loss function. And it can be write as follows:

$$L_\tau(y, f(x;\theta)) = \alpha \cdot (\max(0, y-\hat{y}_{Li}) + \max(0, \hat{y}_{Ui} - y))$$

Where $\hat{y}_{Li}$ is the prediction of lower bound and $\hat{y}_{Ui}$ is the prediction of upper bound. They are extracted from the each prediction of y through NNs with two outputs. For each data point, it calculates the positive difference between the true value and the predicted upper bound m and the positive difference between the predicted lower bound k and true value. It ensures that only positive differences contribute to the loss. If the difference is negative, it is set to zero. Then add quantile level α as weight to calculate the quantile loss.

In addressing real-world data, particular attention must be given to outliers. Acknowledging the effective handling of outliers by Huber regression, we incorporate it into our custom loss function. Huber regression minimizes a loss function that combines the squared error loss of OLS with a linear loss for large residuals. Its loss function is as follows:

$$L(y,\hat{y}) = \begin{cases} \frac{1}{2} * (y-\hat{y})^2, & |y-\hat{y}| \leq \delta \\ \delta * |y-\hat{y}| - \frac{1}{2} * \delta^2, & |y-\hat{y}| > \delta \end{cases}$$

where y is the true value; $\hat{y}$ is the predicted value of the data point(It will be middle value mentioned below); δ is a parameter that need to be tuned according to the data at which the loss switches from squared error (like MSE) to linear error (similar to MAE). The incorporation of Huber regression into our loss function enhances its robustness, particularly in scenarios where outliers and unusual points can significantly impact traditional regression models. Huber regression provides a balanced approach, effectively addressing both the influence of outliers and the general fitting of the model, making it a valuable addition to our comprehensive custom loss function

A high-quality prediction interval is characterized by a narrow interval and a high coverage probability. To introduce flexibility into our interval, we incorporate upper and lower bound losses with different weights as the final components of the custom loss function. For the upper bound loss, when the prediction upper bound is bigger than the true value, $\hat{y}_{Ui} - y_i$ should be added into the loss. Otherwise, it will be regarded as 0.

$$max(0, \hat{y}_{Ui} - y_i)$$

For the lower bound loss, when the prediction lower bound is smaller than the true value, $y_i - \hat{y}_{Li}$ should also be added into the loss. Otherwise it will be regarded as 0.

$$max(0, y_i - \hat{y}_{Li})$$

Then, we set upper bound loss weight as γ, and set lower bound loss function weight as λ

$$\gamma \cdot max(0, \hat{y}_{Ui} - y_i)$$

$$\lambda \cdot max(0, y_i - \hat{y}_{Li})$$

Hence, we can tune the γ and λ to achieve our different requirements to upper bound and lower bound. So, the construction of total custom loss function can be written as follows:

Algorithm 1 Custom loss function algorithm

```
Require: Target value $y_i$, predictions of each target value's lower and upper bound $\hat{y}_{Ui}$, $\hat{y}_{Li}$,
    desired coverage $(1-\alpha)$, threshold parameter $\delta$, hyperparameters $\lambda$, $\gamma$
1: Initialize $\lambda$, $\gamma$ to some starting values
2: $\hat{y}_i = (\hat{y}_{Li} + \hat{y}_{Ui})/2$
3: Quantile = $\frac{1}{n}\alpha \sum_{i=1}^{n} [max(0, y_i - \hat{y}_{Li}) + max(0, \hat{y}_{Ui} - y_i)]$
4: Huber = $\frac{1}{n} \sum_{i=1}^{n} \begin{cases} \frac{1}{2}(y_i - \hat{y}_i), & |y_i - \hat{y}_i| \le \delta \\ \delta(|y_i - \hat{y}_i| - \frac{1}{2}\delta), & |y_i - \hat{y}_i| > \delta \end{cases}$
5: Upper bound loss = $\gamma \frac{1}{n} \sum_{i=1}^{n} max(0, \hat{y}_{Ui} - y_i)$
6: Lower bound loss = $\lambda \frac{1}{n} \sum_{i=1}^{n} max(0, y_i - \hat{y}_{Li})$
7: Custom loss = $\frac{1}{n} \sum_{i=1}^{n} [(\alpha + \lambda)(max(0, y_i - \hat{y}_{Li})) + (\alpha + \gamma)(max(0, \hat{y}_{Ui} - y_i))]$+Huber
   $(y_i, \hat{y}_i)$
```

<u>Comparison to QD Method</u>

QD method mentioned above serves as a benchmarker, and its loss function plays a pivotal role in evaluating its performance. We derive this custom loss function is to compare it with the QD method.

$$Loss_{QD} = MPIW_{capt.} + \beta \frac{n}{\alpha(1-\alpha)} max(0, (1-\alpha) - PICP)^2$$

where $MPIW_{capt.}$ is the interval width of the points captured by the prediction interval. And $PICP$ is the coverage probability.

$$k_i = \begin{cases} 1, & \hat{y}_{Ui} > y_i > \hat{y}_{Li} \\ 0, & else \end{cases}$$

$$PICP := \frac{\sum_{i=1}^{n} k_i}{n}$$

$$MPIW_{capt.} := \frac{1}{\sum_{i=1}^{n} k_i} \sum_{i=1}^{n} (\hat{y}_{Ui} - \hat{y}_{Li}) \cdot k_i$$

Actually, both of the two loss function are derived from different angle. Here we only discuss the main differences that $Loss_{QD}$ did not pay attention to.

- $Loss_{QD}$ uses $MPIW_{capt.}$ to improve the quality of prediction interval. However, this kind of method will not solve some data set with outliers, especially for small data set. Form statistics angle, outliers need to be ignored when we construct the model.
- In the $Loss_{QD}$, there is only one parameter β need to be tuned. So, the prediction interval will follow the High Quality principle. But, we talked about the importance of flexibility of prediction interval. In the new custom loss function, we give different weight for upper bound loss and lower bound loss. It can help us get different prediction interval according to the real problems by tuning parameter λ and γ.

Training with Gradient Descent

Gradient Descent Optimization [4] is the most commonly used optimization algorithm for neural network model training. For deep learning models, gradient descent algorithms are basically used for optimization training. The principle behind the gradient descent algorithm: the gradient of the objective function $J(\theta)$ with respect to the parameter θ will be the direction in which the loss function (loss function) rises fastest. To minimize the loss, we only need to advance the parameters by one step in the opposite direction of the gradient to achieve the decline of the objective function (loss function). This step size η is also called the learning rate.

Here we will divide the custom loss function into 4 parts:

- Quantile Loss Gradient

$$\frac{\partial \text{Quantile}}{\partial k} = -\alpha \cdot I(y < k)$$

$$\frac{\partial \text{Quantile}}{\partial m} = \alpha \cdot I(y \geq m)$$

- Huber Loss Gradient

$$\frac{\partial \text{Huber}}{\partial k} = \begin{cases} \frac{y - \frac{(k+m)}{2}}{\delta}, & |y - \frac{(k+m)}{2}| > \delta \\ 0, & \text{otherwise} \end{cases}$$

$$\frac{\partial \text{Huber}}{\partial m} = \begin{cases} \frac{y - \frac{(k+m)}{2}}{\delta}, & |y - \frac{(k+m)}{2}| > \delta \\ 0, & \text{otherwise} \end{cases}$$

- Lower Bound Loss Gradient:

$$\frac{\partial \text{LowerBound}}{\partial k} = -I(y < k)$$

- Upper Bound Loss Gradient

$$\frac{\partial \text{UpperBound}}{\partial m} = I(y \geq m)$$

where k and m are prediction lower bound and prediction upper bound outputed from NNs,

I is the indicator function, which is usually expressed as:

$$I = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

If we denote μ as the set of model parameters, μ will be updated as follows:

$$\mu_{\text{new}} = \mu_{\text{old}} - \eta \cdot \frac{\partial L}{\partial \mu}$$

Simulation Data Example

To see how the loss change, data was generated from $y = 2.5x_1 + 1.7x_2 + 0.8x_3 + \varepsilon$, with $\varepsilon \sim N(0,1)$. λ and γ are randomly chose from $[0.0001, 0.001, 0.01]$. With different combination of λ and γ, the loss can be plotted as follows:

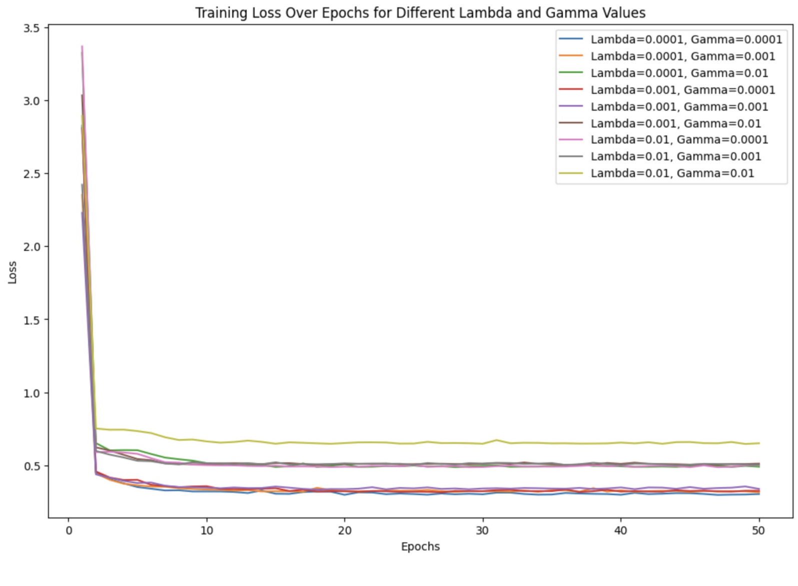

Figure 1: Simulation Data Loss with Different Combinations of Parameters

According to the Figure 1, the gradual convergence of loss indicates that the model is effectively learning the features and patterns of the training data. This can be explained by the model making progress on the training data, gradually adjusting its parameters to minimize the difference between predictions and the actual target.

The flexibility of the loss function, as mentioned earlier, allows us to customize the emphasis on either the lower or upper bounds depending on the specific requirements of various real-world applications. This adaptability extends to scenarios involving simulated data, enhancing the applicability of our approach. As depicted in Figure 2, our emphasis is on the lower bound of the data, from the figure, it can be observed that the lower bound is much closer to the real values; while in Figure 3, our focus shifts to the upper bound of the data, the upper bound is much closer to the real values.

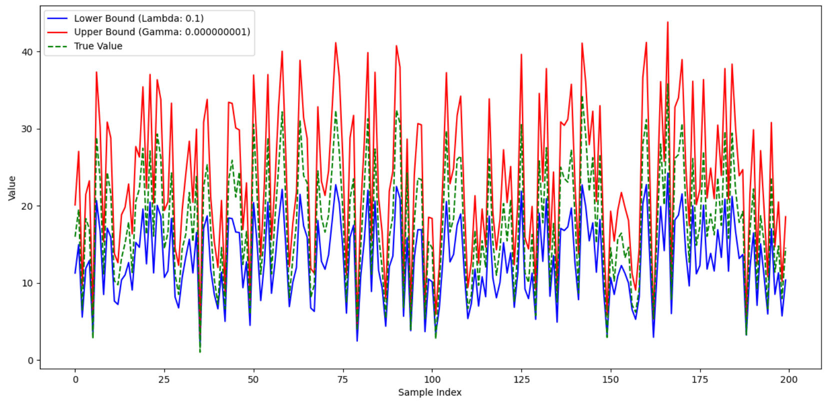

Figure 2: Lower Bound

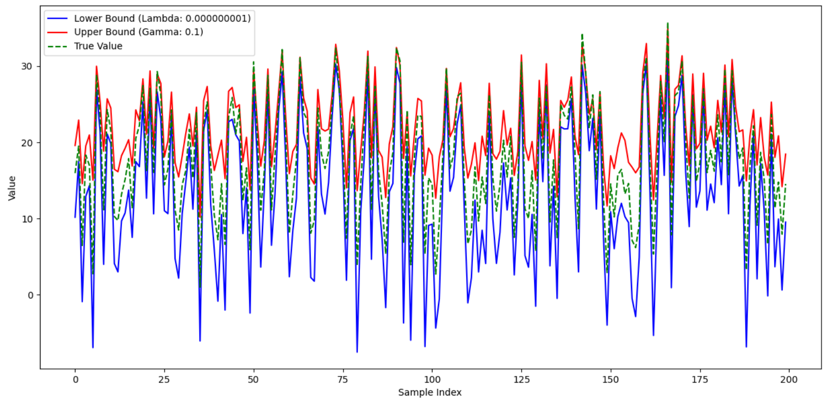

Figure 3: Upper Bound

Avoid Overfitting

Overfitting occurs when a trained model performs well on the training set, but performs poorly on the test set. In the Figure 4, as epoch increases, loss increases on the validation data set. Therefore we introduce two methods to prevent overfitting.

- **Early Stopping** is a technique in neural network training that stops the training process when the model performance on a validation set stops improving, thus preventing overfitting. This helps ensure the model generalizes well without fitting noise in the training data.

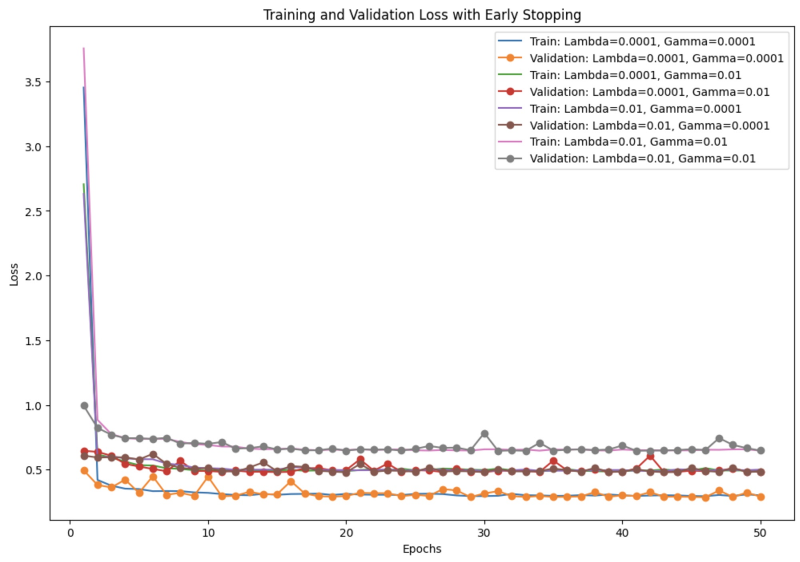

Figure 4: Loss on Train and Test Data for Early Stopping

- **Regularization** The main goal of regularization is to balance the fitting ability and generalization ability of the model. In this project, $L2$ norm is chosen to do the regularization. For each layer with L_2 regularization, the weights are updated not only based on the loss gradient:

$$\mu_{\text{new}} = \mu_{\text{old}} - \eta \nabla L(\mu_t)$$

but also by adding a penalty proportional to the square of the weights:

$$\mu_{\text{new}} = \mu_{\text{old}} - \eta \left(\nabla L(\mu_t) + \lambda \mu_t\right)$$

CHAPTER IV.

CASE STUDY

Data Description

To assess the performance of the custom loss function, we performed evaluations using 10 publicly available datasets. The datasets are summarized in the table 1:

Dataset	**Number**	**Dimension**
Boston Housing	506	13
Concrete	1030	8
Energy Efficiency	768	8
Kin8nm	8192	8
Naval Propulsion	11934	16
Power Plant	9568	4
Protein Structure	45730	9
Wine Quality Red	1599	11
Yacht	308	6
Year MSD	515345	90

Table 1: Data Description

Result

Model is asked to output 95% prediction interval. For different datasets there are three parameters being tuned, which are λ, γ, δ. And five layers NNs with two outputs (upper bound and lower bound) in the last layer is used. In our model, we assume the predicted value need to be in the middle of upper bound and lower bound. Table 2 shows the parameters used in the best-performing models across various datasets.

In this project, we are doing interval prediction. So the main metrics for evaluating interval predictions include two parts: Mean Prediction Interval Width (MPIW) and Prediction Interval Coverage Probability (PICP).

MPIW quantifies the average width of prediction intervals, providing insights into the overall spread of predicted values. Smaller MPIW values indicate more precise predictions with a narrower range for the prediction intervals. PICP assesses the percentage of true values that fall within the prediction intervals. Higher PICP values indicate a more reliable

Dataset	$\boldsymbol{\lambda}$	$\boldsymbol{\gamma}$	$\boldsymbol{\delta}$	L_2 **weight**	**Patience**	**Epoch**
Boston Housing	0.04	0.0404	0.5	0.001	10	200
Concrete	0.03902	0.040502	0.5	0.001	10	200
Energy Efficiency	0.0629	0.0789	0.5	0.00001	10	200
Kin8nm	0.048989	0.048989	0.4	0.0001	10	200
Naval Propulsion	0.082	0.061	0.4	0.0001	10	200
Power Plant	0.052	0.055	0.4	0.0001	10	200
Protein Structure	0.0399	0.000001	0.9	0.00001	10	200
Wine Quality Red	0.025	0.017	0.65	0.001	10	200
Yacht	0.035	0.0851	1.0	0.0001	10	200
Year MSD	0.000001	0.019	0.8	0.00001	10	200

Table 2: Parameters of Ten Benchmarking Datasets

and coverage quality of the predictions. The following forms show how to calculate the two metrics.

$$PICP = \frac{t}{n}$$

$$MPIW = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_{Ui} - \hat{y}_{Li})$$

where t is the total number of true value in the prediction interval. n is the number of observations. Achieving a high coverage probability while minimizing the prediction interval width is a primary objective when models are considered for prediction.

In the experimental design, we emphasized the principle of ensuring a fair comparison. Each dataset was randomly divided, with 90% used as training data and 10% as the test data. This process was repeated 20 times to ensure a comprehensive evaluation across different combinations of training and test data. Additionally, for consistency in data processing, we normalized both input and target variables, setting their mean to 0 and standard deviation to 1. This normalization approach aligns with the method employed by the benchmarker QD, ensuring comparability in experimental results. The consistent processing method helps eliminate potential evaluation biases arising from differences in the scale of variables, thus enhancing model performance and bolstering the reliability of experimental outcomes. We summarize the analysis results for the ten datasets in Table 3 compared to the QD benchmark method.

	Loss		PICP		MPIW		
	QD-Ens	FD-Ens	QD-Ens	FD-Ens	QD-Ens	FD-Ens	IMPROVEMENT
Boston Housing	1.33 ± 0.05	0.23 ± 0.03	0.92 ± 0.01	0.98 ± 0.02	1.16 ± 0.02	1.15 ± 0.02	6%
Concrete	1.16 ± 0.02	0.28 ± 0.05	0.94 ± 0.01	0.96 ± 0.01	1.09 ± 0.01	1.49 ± 0.02	NA
Energy Efficiency	0.47 ± 0.01	0.25 ± 0.02	0.99 ± 0.00	0.97 ± 0.01	0.47 ± 0.01	0.41 ± 0.02	13%
Kin8nm	1.24 ± 0.01	0.26 ± 0.02	0.96 ± 0.00	0.95 ± 0.01	1.25 ± 0.01	1.24 ± 0.02	1%
Naval Propulsion	0.27 ± 0.01	0.50 ± 0.01	0.98 ± 0.00	0.97 ± 0.02	0.28 ± 0.01	0.46 ± 0.02	NA
Power Plant	0.86 ± 0.00	0.21 ± 0.01	0.95 ± 0.00	0.96 ± 0.01	0.86 ± 0.00	0.97 ± 0.02	NA
Protein Structure	2.28 ± 0.01	0.37 ± 0.01	0.95 ± 0.00	0.95 ± 0.01	2.27 ± 0.01	2.25 ± 0.01	1%
Wine Quality Red	3.13 ± 0.19	0.34 ± 0.04	0.92 ± 0.01	0.90 ± 0.02	2.33 ± 0.02	2.31 ± 0.02	NA
Yacht	0.23 ± 0.02	0.23 ± 0.05	0.96 ± 0.01	0.97 ± 0.01	0.17 ± 0.00	0.30 ± 0.01	NA
Year MSD	$2.47 \pm$ NA	0.29 ± 0.01	$0.96 \pm$ NA	0.95 ± 0.02	$2.48 \pm$ NA	2.40 ± 0.01	3%

Table 3: PI quality metrics on ten benchmarking regression datasets; mean $\pm$ one standard error, best result in bold. Best was assessed according to the following criteria. If PICP ≥ 0.95 for both, both were best for PICP, and best MPIW was given to smallest MPIW. If PICP ≥ 0.95 for neither or for only one, largest PICP was best and MPIW only assessed if the one with larger PICP also had smallest. (Note: If PICP is significantly improved , but the width is also narrower. Like Boston Housing, as long as the width change does not exceed 0.01, the improvement will be calculated by the PICP)

Additionally, we employed two other metrics to assess the performance of the regression model, including Mean Squared Error(MSE), Root Mean Squared Error(RMSE) and Mean Absolute Percentage Error(MAPE). These metrics offer a comprehensive evaluation of various aspects, encompassing the accuracy, error magnitude, and percentage error of the model during the prediction process. By considering these metrics collectively, we gain a more thorough understanding of the model's performance across different tasks and datasets, providing deeper insights for model selection and performance comparisons. The formula is as follows and the result is shown in table 4.

- MSE (Mean Squared Error):

$$\frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

- RMSE (Root Mean Squared Error):

$$\sqrt{\frac{1}{n}\sum(y_i-\hat{y}_i)^2}$$

- MAPE (Mean Absolute Percentage Error)

$$\frac{1}{n}\sum|\frac{y_i-\hat{y}_i}{y_i}|\cdot 100\%$$

	MSE	MAPE	RMSE	
	FD-Ens	FD-Ens	FD-Ens	QD-Ens
Boston	0.200	1.633	0.445	3.38
Concrete	0.234	1.482	0.482	5.76
Energy	0.086	0.371	0.293	2.30
Kin8nm	0.173	1.576	0.416	0.09
Naval	1.000	NA	1.000	0.00
Power Plant	0.073	1.240	0.271	4.10
Protein	0.611	1.402	0.782	4.98
Wine	0.631	0.795	0.793	0.65
Yacht	0.034	0.306	0.178	1.00
Year MSD	0.718	1.351	0.847	9.30

Table 4: Regression Metrics

CHAPTER V.
DISCUSSION

According to the regression metrics in the table 4, MSE gives a measure of the average squared difference between expected and actual values. RMSE is more widely used than MSE to evaluate the performance of regression models versus other stochastic models because it has the same units as the dependent variable (Y). MAPE calculates the error as a percentage, which facilitates comparisons across datasets and modes, but when the real data is equal to 0, such as the Naval Propulsion dataset, MAPE does not exist. All the regression metrics show that our assumption (The true value y_i should be approximately equal to the median of upper bound prediction $\hat{y}_{Ui}$ and lower bound prediction $\hat{y}_{Li}$) is appropriate. For ten benchmark data sets, the model gives good regression fitting results. In the QD method, they also showed the RMSE. And our results have improved significantly. The RMSE of 70% of the ten benchmaring datasets is significantly smaller than the QD method. As one of the metrics for assessing model performance, the loss value re flects the difference between predicted and actual values. Our proposed method indicates that across multiple datasets, it outperforms the benchmark model, achieving lower loss values. This suggests its su periority in accurately predicting target variables. It is noteworthy that, in the Naval Propulsion and Yacht datasets, the proposed model's loss values are slightly higher compared to the benchmark method, especially in the Yacht dataset, where the two are nearly identical. Compared with QD method, FD method is 60% better than QD method. We mainly discuss it from the following aspects:

- Algorithm. The QD method has achieved very good results in ten benchmarking data sets. It takes interval width and coverage probability as the main of the derive loss function. After neural network training, these two PI metrics should give us great results. But our FD method is derived based on regression, so when we compare the RMSE of the two methods, the FD method is significantly better than the QD method.

- Parameters. There are three parameters need to be tuned on the FD method. To get the best combination of these three, we firstly use hand to narrow down the range. Then grid search is used to find the best combination. However, even then we cannot find the best combination because these parameters, especially λ and γ, are small. In other words, even a small change can lead to completely different results. And for large data sets like Year MSD, using grid search is very time-consuming. We couldn't try every combination. Therefore, the limitations of the hardware equipment mean that the results we show are not a theoretically optimal solution.

- Data. Different data sets have different distributions. In order to improve the coverage rate, we hope that the prediction interval can include as many points as possible. But outliers should be excluded. FD method introduced huber regression. If there are too many outliers in this data set, then the FD method is likely to eliminate these points during the parameter adjustment process. Like Wine Quality Red data set, when we want to increase the PICP, we try to make the MPIW wider. But when we appropriately reduce λ and γ to weaken the upper and lower boundary loss restrictions on the prediction interval, MPIW will become wider than expected. The reason is that the distribution of data causes outliers to have a greater impact on the model during the regression fitting process.

BIBLIOGRAPHY

[1] Carney, J. G., Cunningham, P., and Bhagwan, U. Confidence and prediction intervals for neural network ensembles. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 2, pages 1215–1218. IEEE, 1999.

[2] de Castro Vieira, J. R., Barboza, F., Sobreiro, V. A., and Kimura, H. Machine learning models for credit analysis improvements: Predicting low-income families' default. *Applied Soft Computing*, 83:105640, 2019.

[3] Ghosh, S., Feng, M., Nguyen, H., and Li, J. Hypotension risk prediction via sequential contrast patterns of icu blood pressure. *IEEE journal of biomedical and health informatics*, 20(5):1416–1426, 2015.

[4] Haji, S. H. and Abdulazeez, A. M. Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 18(4):2715–2743, 2021.

[5] Hwang, J. G. and Ding, A. A. Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, pages 748–757, 1997.

[6] Jiang, F., Zhu, Q., and Tian, T. An ensemble interval prediction model with change point detection and interval perturbation-based adjustment strategy: A case study of air quality. *Expert Systems with Applications*, 222:119823, 2023.

[7] Khosravi, A. and Nahavandi, S. An optimized mean variance estimation method for uncertainty quantification of wind power forecasts. *International Journal of Electrical Power & Energy Systems*, 61:446–454, 2014.

[8] Khosravi, A., Nahavandi, S., and Creighton, D. A prediction interval-based approach

to determine optimal structures of neural network metamodels. *Expert systems with applications*, 37(3):2377–2387, 2010.

[9] Khosravi, A., Nahavandi, S., and Creighton, D. Prediction interval construction and optimization for adaptive neurofuzzy inference systems. *IEEE Transactions on Fuzzy Systems*, 19(5):983–988, 2011.

[10] Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks*, 22(3):337–346, 2010.

[11] Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on neural networks*, 22(9):1341–1356, 2011.

[12] Khosravi, A., Nahavandi, S., Srinivasan, D., and Khosravi, R. Constructing optimal prediction intervals by using neural networks and bootstrap method. *IEEE transactions on neural networks and learning systems*, 26(8):1810–1815, 2014.

[13] Li, H. Scada data based wind power interval prediction using lube-based deep residual networks. *Frontiers in Energy Research*, 10:920837, 2022.

[14] Li, K., Wang, R., Lei, H., Zhang, T., Liu, Y., and Zheng, X. Interval prediction of solar power using an improved bootstrap method. *Solar Energy*, 159:97–112, 2018.

[15] Lv, Z., Zhao, J., Liu, Y., and Wang, W. Use of a quantile regression based echo state network ensemble for construction of prediction intervals of gas flow in a blast furnace. *Control Engineering Practice*, 46:94–104, 2016.

[16] Nix, D. A. and Weigend, A. S. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[17] Pearce, T., Brintrup, A., Zaki, M., and Neely, A. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4075–4084. PMLR, 10–15 Jul 2018.

[18] Stine, R. A. Bootstrap prediction intervals for regression. *Journal of the American statistical Association*, pages 1026–1031, 1985.

[19] Taormina, R. and Chau, K.-W. Ann-based interval forecasting of streamflow discharges using the lube method and mofips. *Engineering Applications of Artificial Intelligence*, 45:429–440, 2015.

[20] Taylor, J. W. and Bunn, D. W. A quantile regression approach to generating prediction intervals. *Management Science*, 45(2):225–237, 1999.

[21] Thombs, L. A. and Schucany, W. R. Bootstrap prediction intervals for autoregression. *Journal of the American Statistical Association*, pages 486–492, 1990.

[22] Valencia, F., Collado, J., Sáez, D., and Marín, L. G. Robust energy management system for a microgrid based on a fuzzy prediction interval model. *IEEE Transactions on Smart Grid*, 7(3):1486–1494, 2015.

[23] Van Hinsbergen, C. I., Van Lint, J., and Van Zuylen, H. Bayesian committee of neural networks to predict travel times with confidence intervals. *Transportation Research Part C: Emerging Technologies*, 17(5):498–509, 2009.