ICT Express 11 (2025) 974–980

Contents lists available at ScienceDirect

ICT Express

journal homepage: www.elsevier.com/locate/icte

Neural-NGBoost: Natural gradient boosting with neural network base learners

Jamshidjon Ganiev[1], Deok-Woong Kim[1], Seung-Hwan Bae*

Department of Electrical and Computer Engineering, Inha University, Incheon, Republic of Korea

ARTICLE INFO

Keywords:
Natural gradient boosting
Neural networks
Probabilistic prediction
Uncertainty estimation

ABSTRACT

NGBoost has shown promising results in probabilistic and point estimation tasks. However, it is vague still whether this method can be scalable to neural architecture system since its base learner is based on decision trees. To resolve this, we design a Neural-NGBoost framework by replacing the base learner with lightweight neural networks and introducing joint gradient estimation for boosting procedure. Based on natural gradient boosting, we iteratively update the neural based learner by inferring natural gradient and update the parameter score with its probabilistic distribution. Experimental results show Neural-NGBoost achieves superior performance across various datasets compared to other boosting methods.

1. Introduction

Regression in supervised learning typically involves predicting a single best guess, known as point estimation. For instance, a financial regression model might predict tomorrow's stock price as $150 or a weather model might forecast the temperature as exactly 25 °C. However, point estimates struggle to capture the uncertainty inherent in such forecasts, which is crucial in real-world scenarios such as finance and healthcare. Probabilistic regression addresses this by modeling predictive distributions rather than point estimates, allowing for explicit uncertainty quantification [1,2]. In particular, gradient boosting methods [3–6] have been extended to probabilistic regression tasks [2,7], iteratively improving model predictions through ensemble learning. Among these, NGBoost [5] combines gradient boosting with natural gradient optimization [8] to estimate multiple parameters of predictive distributions (*e.g.* mean, variance) simultaneously. NGBoost first estimates distribution parameters based on a scoring rule (*e.g.* negative log-likelihood). It then computes the natural gradient, which captures the geometry of the distribution parameter space, to train simple decision tree base learners, which then guide the update of distribution parameters iteratively. Despite NGBoost's success in its modular design, its reliance on decision trees constrains its capability to model complex relationships, particularly in larger datasets, and restricts its scalability and performance. Furthermore, while the original NGBoost trains separate base learners for each parameter of the target distribution gradients (*e.g.* one for the mean μ and another for the log variance) at each boosting step, we presume that this decoupled boosting procedure likely means the base learners do not share any learned feature correlations, which can limit their ability to model complex, interdependent interactions within the predictive distribution, especially in high-dimensional data.

To address these, we propose a novel boosting framework that leverages neural networks and joint gradient estimation to overcome the limitations of simple tree-based base learners. Specifically, Neural networks are well-known for their success in many tasks [9–12] and for modeling complex nonlinear patterns in high-dimensional data. To this end, we replace NGBoost's original boosting procedure with a unified neural network base learner that jointly estimates the gradients for all parameters. Our unified boosting procedure addresses the limitations of decoupled boosting procedure which may constrain the model's ability to exploit meaningful correlations among the gradients of distributional parameters. By jointly leveraging these interdependencies, our framework enables the model to better capture such relationships and correlations in a more discriminative manner, resulting in improved performance and faster convergence during training. To ensure both practicality and efficiency, our proposed method is designed to be computationally efficient, allowing Neural-NGBoost to handle large-scale datasets without introducing excessive overhead. Notably, Neural-NGBoost yields substantial speedups in both training and inference time when handling larger and higher-dimensional datasets (see Fig. 6). Extensive experiments show that Neural-NGBoost outperforms NGBoost and other methods across various datasets.

The main contributions of this work are:

* Corresponding author.
E-mail addresses: jamshid24@inha.edu (J. Ganiev), k5000plus@inha.edu (D.-W. Kim), shbae@inha.ac.kr (S.-H. Bae).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.icte.2025.08.003
Received 18 April 2025; Received in revised form 24 July 2025; Accepted 10 August 2025
Available online 19 August 2025

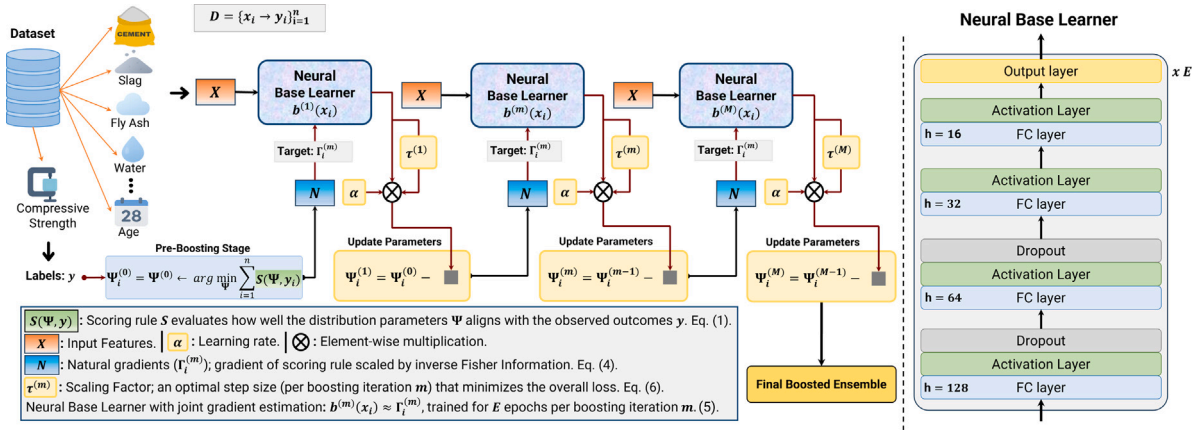

Fig. 1. Neural-NGBoost framework with unified neural network base learners. We first initialize the distribution parameters by minimizing Eq. (2). At each boosting iteration, we compute the natural gradient of the scoring rule Eq. (4) and use it to train the base learner with the loss function defined in Eq. (5). Finally, we update the distribution parameters from Eq. (7). The dimension of hidden state **h** can be tuned based on dataset size and complexity.

- A novel and practical integration of a single, unified lightweight neural network as base learners within the framework of probabilistic gradient boosting.
- A joint gradient estimation for the boosting procedure to more effectively capture interdependencies across predictive distribution parameters and improve learning efficiency.
- Extensive experiments across regression and classification tasks on multiple benchmark datasets demonstrate more enhanced performance, scalability, and uncertainty calibration of Neural-NGBoost compared to NGBoost and most other established methods.

2. Related works

Boosting frameworks [4,13,14] have achieved the widespread success in regression tasks. For instance, XGBoost [4] uses optimized tree-based algorithms and parallel processing for efficient and accurate predictions. LightGBM [13] adopts a histogram-based approach, significantly reducing memory usage and accelerating training. GrowNet [14] leverages shallow neural networks as weak learners and incorporates a corrective step to refine predictions by globally fine-tuning all previously added learners via back-propagation. Despite their effectiveness, these methods rely on the task of only point estimates, not probabilistic ones.

For this, several probabilistic approaches have been proposed. NGBoost [5] combines gradient boosting with natural gradient, showing strong performance in probabilistic prediction tasks. GAMLSS [15] extends generalized linear models by allowing distribution parameters to vary with predictors, enabling nuanced modeling of response distributions. Distributional random forests [16] replace point estimates with conditional distributions, but rely on pre-specified distributional assumptions, which can limit model accuracy if chosen incorrectly. Neural network-based approaches have also emerged as strong alternatives for uncertainty estimation. Dropout sampling [17], a Bayesian approximation method, repeatedly samples the outputs of a neural network during inference by randomly dropping neurons, thus capturing uncertainty with minimal architectural changes. Deep ensembles [18] train multiple neural networks independently and aggregate their outputs, effectively modeling both data and model-related uncertainty [19]. However, neural network-based approaches often require substantial computational resources, limiting their practicality in real-world applications [20,21]. By contrast, our method preserves training efficiency while reducing inference time.

Among these methods, NGBoost stands out for its modular design, allowing flexibility in base learners, distributions, and scoring rules. However, its performance often lags behind traditional machine learning approaches like Gradient Boosting or Random Forests [3,22]. To this end, our proposed Neural-NGBoost method replaces standard decision tree base learners with lightweight but effective neural networks and joint gradient estimation during each boosting procedure. Our method offers computational efficiency in both large-scale training and inference time while building on NGBoost's flexibility and leveraging the representational ability of neural networks.

3. Methodology

3.1. Preliminaries

The overall architecture of Neural-NGBoost is illustrated in Fig. 1. To estimate the probability distribution $X_i(y_i \mid x_i) \sim \mathcal{N}(\mu, \log\sigma)$ for each data point, we first construct the initial parameters $\Psi^0 = (\mu, \log\sigma)$ by minimizing the following proper scoring rule as:

$$S(\Psi, y_i) = -\log X_i(y_i \mid x_i), \tag{1}$$

$$\Psi^{(0)} = \arg\min_{\Psi} \sum_{i=1}^{n} S(\Psi, y_i), \tag{2}$$

Here, the scoring rule S indicates how well the distribution parameterized by Ψ aligns with the observed outcomes y [23]. In practice, minimizing this score is equivalent to maximizing the likelihood estimation (MLE) [24]. As a result, we obtain a globally marginal distribution to set Ψ^0 where its parameter values are uniformly shared across all samples. Then, we compute the ordinary gradient of the scoring rule $g_i^{(m)}$ for each boosting iteration m as follows:

$$g_i^{(m)} = \nabla_{\Psi}\, S\left(\Psi_i^{(m-1)}, y_i\right), \tag{3}$$

However, directly adopting Eq. (3) to update the parameters can be suboptimal, as the gradient direction in parameter space may not align with the optimal direction in distribution space. To this end, the natural gradient of scoring rule $\Gamma_i^{(m)}$ can be defined as:

$$\Gamma_i^{(m)} \leftarrow \mathcal{I}_S\left(\Psi_i^{(m-1)}\right)^{-1} \cdot g_i^{(m)}, \tag{4}$$

where $\mathcal{I}_S\left(\Psi_i^{(m-1)}\right)^{-1}$ denotes the inverse Fisher information matrix, which adjusts the scoring rule's gradient to reflect the local curvature of the parameter space. This enables the gradient to capture the manifold structure of the distribution, ensuring invariance under reparameterization. After that, we leverage these natural gradients as regression targets for the base learner as follows:

$$\mathcal{L}_{base} = \frac{1}{N} \sum_{i=1}^{N} \left| |b^{(m)}(x_i) - \Gamma_i^{(m)}| \right|_2^2, \tag{5}$$

Table 1
Neural base learner hyper-parameter configurations for each dataset.

Dataset	Depth L	Hidden dimensions $\{h_\ell\}$	M
Concrete	2	$\{32 \to 16\}$	200
Wine	2	$\{32 \to 16\}$	200
Kin8nm	2	$\{32 \to 16\}$	300
Power	2	$\{32 \to 16\}$	300
Protein	3	$\{64 \to 32 \to 16\}$	300
Slice Localz.	3	$\{64 \to 32 \to 16\}$	300
Year MSD	4	$\{128 \to 64 \to 32 \to 16\}$	300

For the base learner, we adopt our proposed neural base learner as described in Section 3.2. At each iteration m, we train a base learner $b^{(m)}$ to predict $\Gamma_i^{(m)}$ from input features x_i, by minimizing Eq. (5) over E epochs (E in the 80 to 150 range is recommended for achieving optimal performance, as training beyond this range may yield only marginal gains at increased computational cost). This process allows $b^{(m)}$ to approximate the natural gradient. Finally, $\Psi^{(m)}$ for each data point can be updated as:

$$\tau^{(m)} \leftarrow \arg\min_{\tau} \sum_{i=1}^{n} S\left(\Psi_i^{(m-1)} - \tau \cdot b^{(m)}(x_i),\ y_i\right), \tag{6}$$

$$\Psi_i^{(m)} \leftarrow \Psi_i^{(m-1)} - \alpha \cdot \tau^{(m)} \cdot b^{(m)}(x_i), \tag{7}$$

where α is the learning rate and $\tau^{(m)}$ is a scaling factor determined via Eq. (6) to control the update step. By iteratively stacking these base learners across $m = 1, \ldots, M$ boosting iterations, the resulting ensemble progressively refines both the per-sample mean and variance estimates of the predictive distribution.

3.2. Overall architecture

Our neural base learner in Fig. 1, implemented using PyTorch and scikit-learn [25,26], can be integrated seamlessly into the NGBoost framework. Building upon the foundational elements of NGBoost, including natural gradient optimization [8] and the core boosting methodology that we modified to utilize a unified neural base learner with joint gradient estimation, our proposed method enhances the capacity and robustness of the overall probabilistic boosting procedure, particularly on large-scale datasets. To this end, we introduce a neural base learner based on a multi-layer perceptron (MLP) [27] architecture composed of fully-connected layers as:

$$\begin{cases} \mathcal{H}_0 = x_i, \\ \mathcal{H}_\ell = \text{Dropout}\left(\text{Act}\left(W_\ell \mathcal{H}_{\ell-1} + b_\ell\right)\right), \\ \hat{\eta}_i = W_L \mathcal{H}_{L-1} + b_L \end{cases} \tag{8}$$

Here, $\hat{\eta}_i \in \mathbb{R}^K$ is the vector output, where K indicates the number of the target distribution parameters, and Act $(\cdot)$ denotes a non-linear activation function. In our experiments, we primarily use ReLU and LeakyReLU activations [10,28], but other non-linear activations can also be utilized. For each layer ℓ, the hidden representation $\mathcal{H}_\ell$ is computed via a linear transformation using learnable weights [29] W_ℓ and biases b_ℓ, followed by a non-linear activation function. For large-scale datasets, Dropout [30] can selectively be applied to deeper hidden layers to mitigate overfitting and improve generalization. The final output layer produces the prediction of base learner $\hat{\eta}_i \in \mathbb{R}^K$ as a K-dimensional vector. With this approach, our neural base learner enables the model to capture complex nonlinear patterns and extract richer feature representations compared to traditional decision trees.

Moreover, to ensure adaptability across various datasets, the number of layers L and hidden dimensions $\mathbf{h}$ are configurable based on the size and complexity of a dataset. Table 1 summarizes the specific neural base learner configurations used in our experiments, which were found to be effective and do not require aggressive tuning in general, offering practitioners flexibility. In practice, overly large architectures (*e.g.* with 512 hidden units) tend to overfit on smaller datasets, whereas overly small architectures may underfit on large-scale datasets (see Fig. 5).

A core architectural and methodological innovation in Neural-NGBoost lies in its unified neural network base learner with joint gradient estimation. In the original NGBoost, each boosting iteration relies on separate decision tree base learners to obtain the predicted parameter Ψ for an input x_i. For instance, for a Normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $K = 2$ parameters (μ and $\log\sigma$), NGBoost involves two independent base learners, $b_\mu^{(m)}$ and $b_{\log\sigma}^{(m)}$, collectively denoted as $b^{(m)} = \left(b_\mu^{(m)},\ b_{\log\sigma}^{(m)}\right)$. This inherent separation in NGBoost may limit the model's ability to capture shared feature learning across parameters. In contrast, our method addresses this by training only a single, unified base learner $b_{\mu,\log\sigma}^{(m)}$ with a shared feature network to predict the $\Gamma_i^{(m)}$ for all distribution parameters from x_i by minimizing Eq. (5). Through this approach, the base learner can jointly approximate the natural gradients more efficiently by leveraging learned feature correlations at each boosting step.

4. Experiments

4.1. Experimental setup

To evaluate our method, we conduct experiments on multiple publicly available benchmark datasets across regression and classification tasks. Our primary objective is to achieve lower NLL in probabilistic estimation, lower RMSE in point estimation, and higher Top-1 accuracy for classification. For regression experiments, we randomly select 10% of each dataset for test set and 90% for training set. The selected 90% is then split into an 80/20 training/validation set to determine the number of optimal boosting iterations that results in the lowest validation score (see Table 1). After validation, the full 90% training set is used to retrain the model with the determined boosting iterations M, consistent with [5]. To ensure fairness, we report the mean and standard deviation over 20 cross-validation folds with different random seeds for all the regression datasets, except the Protein and Year MSD datasets, which are trained 5 and 1 times, respectively, due to their large size. For classification task, we report the mean and standard deviation over 3 runs with different random seeds. All experiments are conducted on Intel Xeon Gold 6330 CPU and a single NVIDIA RTX A6000 GPU.

Hyperparameters. To avoid extensive hyperparameter searches, we use fixed learning rates for each task. Specifically, for updating the distribution parameters in the boosting stage, we set the learning rate to 0.01. For training the neural base learners, we use learning rates of 0.01, 0.03, and 0.05 for both regression and classification tasks. While this setup may not be optimal for every dataset, it is sufficient to demonstrate the effectiveness and scalability of Neural-NGBoost.

4.2. Probabilistic estimation

To evaluate probabilistic estimation performance, we use the average negative log-likelihood (NLL) as a metric on the test set. In this case, the lower values indicate higher likelihoods assigned to the true observations, thus better calibrated probabilistic predictions. Note that NLL can be negative when the value of a probability density function (PDF) $X_i(y_i \mid x_i)$ exceeds 1, which is valid for continuous distributions with low variance. For a Normal distribution, when the observed data point x_i is equal to the mean value of the PDF $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, it attains the maximum value of $\frac{1}{\sigma\sqrt{2\pi}}$. Therefore, if $\sigma < 1/\sqrt{2\pi}$, the peak density exceeds 1, leading to a negative NLL in Eq. (1). We compare our results primarily against NGBoost and other well-established probabilistic prediction methods [5,15,16,18]. For consistency, we report the results from the original publications. As shown in Table 2, our method achieves the best or second-best NLL performance across all seven benchmark datasets. Notably, we outperform all other methods on Concrete, Wine, Kin8 nm, Slice Localz, and Year MSD datasets, while remaining highly competitive on the others. These results demonstrate that our approach provides strong probabilistic predictions.

Table 2
Comparison of probabilistic estimation performance evaluated by NLL.

Dataset	N	Features	Ours	NGBoost	Deep ensembles	DistForest	GAMLSS
Concrete	1030	8	**2.79 ± 0.16**	3.04 ± 0.17	3.06 ± 0.18	3.38 ± 0.05	6.72 ± 0.59
Wine	1588	11	**0.83 ± 0.08**	0.91 ± 0.06	0.94 ± 0.12	1.05 ± 0.15	0.97 ± 0.09
Kin8nm	8192	8	**−1.24 ± 0.02**	−0.49 ± 0.02	−1.20 ± 0.02	−0.40 ± 0.01	0.20 ± 0.01
Power	9568	4	2.69 ± 0.03	2.79 ± 0.11	2.79 ± 0.04	**2.68 ± 0.05**	4.25 ± 0.19
Protein	45,730	9	2.64 ± 0.01	2.81 ± 0.03	2.83 ± 0.02	**2.59 ± 0.04**	5.04 ± 0.04
Slice Localz	53,500	385	**1.39 ± 0.01**	2.14 ± 0.02	–	–	–
Year MSD	515,345	90	**3.31 ± 0.00**	3.43 ± 0.00	3.35 ± 0.00	–	–

The best method for each dataset is highlighted in **bold** and the second-best is underlined.

Table 3
Comparison of point estimation performance evaluated by RMSE. Marking is as in Table 2.

Dataset	N	Features	Ours	NGBoost	Gradient boosting	Elastic net	DistForest	GrowNet
Concrete	1030	8	**3.98 ± 0.61**	5.06 ± 0.61	4.46 ± 0.29	12.1 ± 0.05	6.61 ± 0.83	4.48 ± 0.59
Wine	1588	11	0.55 ± 0.04	0.63 ± 0.04	**0.53 ± 0.02**	0.58 ± 0.00	0.67 ± 0.05	0.62 ± 0.03
Kin8nm	8192	8	**0.07 ± 0.00**	0.16 ± 0.00	0.14 ± 0.00	0.20 ± 0.00	0.16 ± 0.00	0.08 ± 0.00
Power	9568	4	3.72 ± 0.10	3.79 ± 0.18	**3.01 ± 0.10**	4.42 ± 0.00	3.64 ± 0.24	4.01 ± 0.18
Protein	45,730	9	3.92 ± 0.05	4.33 ± 0.03	3.95 ± 0.00	5.20 ± 0.00	**3.89 ± 0.04**	4.12 ± 0.18
Slice Localz	53,500	385	**1.13 ± 0.05**	2.11 ± 0.13	2.24 ± 0.04	–	–	5.31 ± 0.35
Year MSD	515,345	90	**8.22 ± 0.00**	8.94 ± 0.00	8.73 ± 0.00	9.49 ± 0.00	–	8.81 ± 0.00

Table 4
Accuracy and training time comparison between NGBoost and Neural-NGBoost on MNIST.

Method	Accuracy (%)	Time (s)
NGBoost	85.74 ± 0.29	9104 ± 57.7
Ours	**92.07 ± 0.76**	**304 ± 17.4**

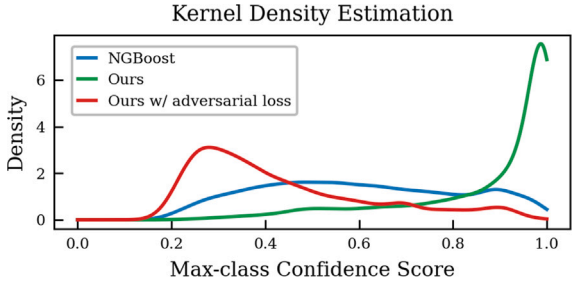

Fig. 2. Uncertainty estimation on domain shift. The x-axis indicates maximum predicted class probability for each sample and y-axis shows the probability density of those scores across NotMNIST dataset.

4.3. Point estimation

To evaluate point estimation performance, the prediction accuracy is evaluated using Root Mean Squared Error (RMSE) as a metric on the test set. Although our method is primarily trained to optimize probabilistic estimation, the point estimates can be easily obtained from the predicted mean values without any hyperparameter tuning. We compare our method against [3,5,16,31], as well as the recent neural network-based boosting approach [14]. For the datasets not covered in their original paper, we evaluate their method using the official implementations with the learning rate over [0.01, 0.05] and the number of MLP layers from 1 to 4 with 32 hidden units per layer and report the best results. Notably, as summarized in Table 3, our approach significantly outperforms other methods in RMSE, confirming the robustness of our method.

4.4. Classification

Classification on MNIST [32]**.** We conduct the performance comparison on the classification task of MNIST, a 10-class benchmark of handwritten digits with 60,000 training and 10,000 test samples. Both NGBoost and Neural-NGBoost are trained on the MNIST training set and evaluated on its test set using the same hyperparameter settings. As summarized in Table 4, it demonstrates that Neural-NGBoost consistently outperforms NGBoost not only for the regression task, but also for the classification task.

Uncertainty evaluation on domain shift. In practice, overconfident predictions on out-of-distribution (OOD) samples are a critical challenge for the safe deployment of machine learning models [18]. Ideally, the model predictions should express higher uncertainty when the data domain is completely different from the training data. To this end, we train the models on the MNIST dataset and test it on NotMNIST [33]. The NotMNIST dataset contains images of alphabetic characters, sharing the same resolution as MNIST digits but representing entirely different class semantics. This allows us to assess whether the model appropriately expresses higher uncertainty when encountering inputs from a different domain. We present a comparative evaluation of uncertainty estimation in Fig. 2. In particular, we compare the maximum confidence scores from the predicted distribution for each test sample in the NotMNIST dataset. Initially, our method exhibits a tendency to produce overconfident predictions on certain test samples compared to NGBoost. To mitigate this issue, we can integrate an adversarial training loss into our proposed method to improve the robustness of the base learner. The adversarial loss is defined as:

$$\mathcal{L}_{\text{adv}} = \frac{1}{N}\sum_{i=1}^{N}\left\| b^{(m)}\left(x_i + \epsilon \cdot \text{sign}\left(\nabla_{x_i}\mathcal{L}_{\text{base}}\right)\right) - \Gamma_i^{(m)}\right\|_2^2, \tag{9}$$

Here, ϵ is a small perturbation magnitude used to generate adversarial samples following the Fast Gradient Sign Method (FGSM) [34]. This encourages the model to moderate its predictive confidence on the inputs during training, thereby strengthening classification robustness, and aligns with the uncertainty estimation strategy for neural networks proposed in [18]. For this experiment, ϵ in the range of 0.1 to 0.3 was found to be effective. Finally, the base learner loss for the uncertainty estimation is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{base}} + \mathcal{L}_{\text{adv}}. \tag{10}$$

By incorporating the adversarial training, our method avoids making overly confident predictions when encountering unseen data and provides more calibrated uncertainty estimates compared to NGBoost. In particular, our approach generates the confidence distribution that is more concentrated at lower confidence values, indicating a more cautious and robust estimation of uncertainty. These results demonstrate that integrating adversarial training into our method can enhance the reliability of uncertainty estimations.

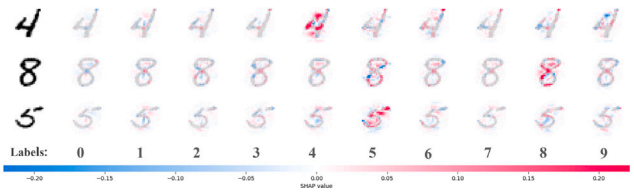

Fig. 3. SHAP-based pixel attributions from Neural-NGBoost on MNIST. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

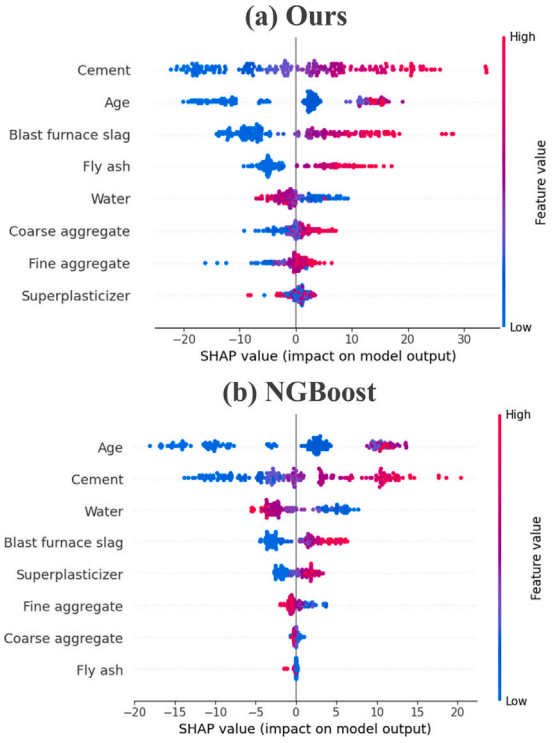

Fig. 4. SHAP summary plots for the Concrete dataset features (y-axis) and their importance (x-axis).

4.5. Interpretability

To analyze the interpretability of our neural network-based learner, we adopt a Shapley value-based feature attribution approach (SHAP) [35] for classification and regression tasks. This approach enables practitioners to interpret the model's predictions regardless of whether the underlying base learners are trees or neural networks, thereby addressing concerns about interpretability.

Interpretability for classification. One limitation of tree-based learner is the reduced interpretability on high-dimensional data such as images, where decision trees cannot easily capture spatial patterns or local feature contributions. In contrast, our Neural-NGBoost leverages differentiable neural networks that enable gradient-based SHAP methods to generate interpretable feature importance score and provide a detailed interpretability analysis. As shown in Fig. 3, we visualize the pixel-level contributions from Neural-NGBoost trained on MNIST. Each row displays a test image alongside SHAP-based attribution maps over the predicted class probabilities. Specifically, red regions indicate pixels that increase the model's confidence for a specific class, while blue regions present areas that suppress its confidence. For instance, in the first row, our method highlights the upper-left stroke contributes positively to the '4' class, while in the second row, circular patterns reinforce the prediction of '8'. It indicates that probabilistic

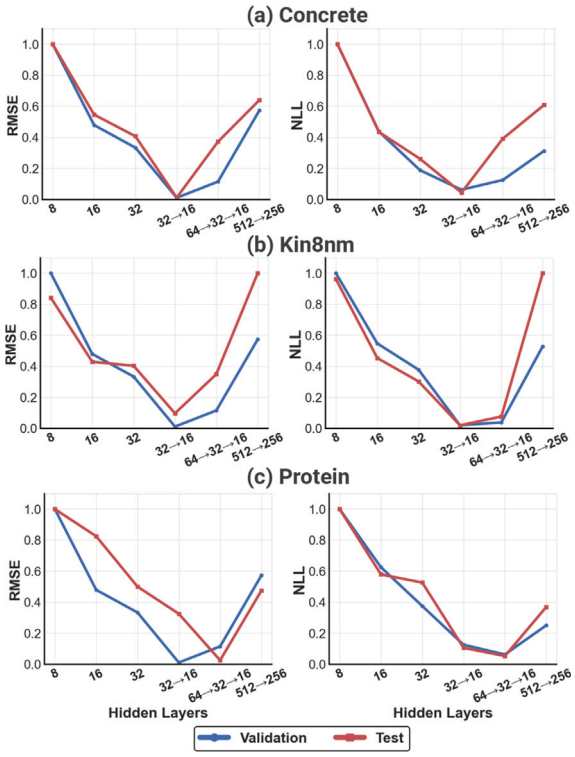

Fig. 5. Validation and test performance (RMSE on the left, NLL on the right) across different hidden layer configurations for the neural base learner on (a) Concrete, (b) Kin8 nm, and (c) Protein datasets.

boosting models with neural networks can provide more detailed class-specific interpretability in classification task than traditional tree-based learners.

Interpretability for regression. We demonstrate the quantitative results of interpretability with our method and NGBoost for tabular data. For instance, SHAP values can also be used to visualize the contribution of each feature to the final model decision, offering comparable or even better interpretability to tree-based learners. As shown in Fig. 4, our method makes more effective use of a diverse set of features compared to NGBoost. This means that our method utilizes a broader range of informative signals to provide a more accurate prediction.

5. Ablation study

We analyze the effects of the various number of layers and hidden units for the neural base learner and evaluate validation and test performance averaged over 5-fold cross-validation on three datasets. Fig. 5 shows that selecting the proper number of layers and hidden units can initially help reduce RMSE and NLL. However, with the excessive number of layers and hidden units, the trained model shows a tendency to overfit, resulting in reduced performance on both validation and test sets, and a significant increase in training time. Specifically, a 2-layer neural network with 32→16 units achieves the optimal balance between model capacity and generalization for the Concrete and Kin8 nm datasets (Fig. 5(a, b)), while a deeper 3-layer network provides better performance for the Protein dataset (Fig. 5(c)). Further increasing the number of hidden units, such as using a 512→256 neural network architecture, leads to degraded performance across both splits, indicating the overfitting trend. The specific base learner configurations reported in Table 1 were selected based on these validation experiments. Note that we apply min–max normalization with $\epsilon = 10^{-2}$ for better visualization for Fig. 5.

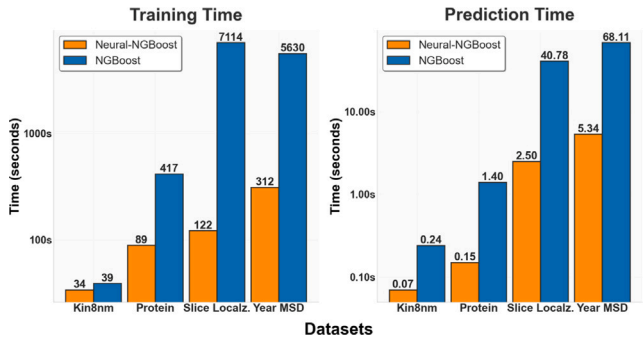

Fig. 6. Training and prediction time (log scale) comparison between NGBoost and Neural-NGBoost.

Fig. 6 presents the comparison of average training and prediction time between our method and the original NGBoost across 5-fold cross-validation. As shown, our method scales substantially better on large datasets with high-dimensional features. In particular, it achieves an impressive 58× speedup in training time on the Slice Localization dataset, and an 18× speedup on the Year MSD dataset, clearly highlighting its scalability. Furthermore, it substantially reduces the inference time across datasets, achieving up to 90% reduction in prediction time.

Algorithm 1 Neural-NGBoost Training and Evaluation

```
from sklearn.preprocessing import StandardScaler
from neural_ngboost import NeuralBaseLearner,
    fit_marginal, compute_natural_gradients,
    find_optimal_scaling

# Data preparation
X, y = ...  # Load your data here (X=features, y=labels)
X = StandardScaler().fit_transform(X)
N_samples = len(X)

# Step 1: Pre-boosting stage
# Compute initial parameters for marginal distribution
Psi = fit_marginal(y) # Perform Eq.(1) and Eq.(2)

# Step 2: Boosting stage
# Refine the distribution parameters for each sample over
    M boosting iterations
for m in range(M):

    # Initialize neural base learner per iteration based
        on dataset size
    base_learner = NeuralBaseLearner()
    base_learner.build_network(N_samples)

    # Store predictions from the previous iteration
    Psi_previous = Psi.copy()

    # 2.1 Compute natural gradients (Eq.(3) and Eq.(4))
    Gamma = [compute_natural_gradients(Psi_previous[i], y
        [i]) for i in range(N_samples)]

    # 2.2 Train the unified base learner (Eq.(5)) with
        joint gradient estimation
    base_learner.fit(X, Gamma, learning_rate=0.03, epochs
        =100)

    # 2.3 Line search to find a scaling factor (Eq.(6))
    tau = find_optimal_scaling(Psi_previous, base_learner
        , X, y)

    # 2.4 Scaled update
    scaled_update = [alpha * tau * base_learner.predict(X
        [i]) for i in range(N_samples)]

    # 2.5 Update parameters to get the new parameters for
        each sample (Eq.(7))
    Psi = [Psi_previous[i] - scaled_update[i] for i in
        range(N_samples)]


# Step 3: Evaluation
# After boosting is complete, evaluate the final boosted
    ensemble model
predicted_distributions = final_boosted_ensemble.
    pred_dist(X)

# Evaluate how well they fit the true target values
NLL = compute_nll(predicted_distributions, y)
RMSE = compute_rmse(predicted_distributions, y)
```

6. Conclusion

This paper presents Neural-NGBoost, an enhanced version of the NGBoost framework that replaces decision tree base learners with a unified neural network architecture. By leveraging joint gradient approximation strategy within each boosting iteration, the proposed method addresses the limitations of tree-based learners by exploiting the correlations among gradients of distributional parameters to learn diverse features, particularly in large-scale and high-dimensional datasets. Experimental results across regression and classification benchmarks demonstrate that Neural-NGBoost improves both probabilistic and point estimation performance, while also providing better uncertainty estimation. Notably, these gains come with the minimal computational overhead, achieving these improvements without a significant increase in training time and offering even faster inference time, making it particularly suitable for large-scale, real-time applications. Specifically, the neural network base learner with joint gradient approximation facilitates precise updates of distribution parameters during boosting. Overall, this work highlights the effectiveness of using neural networks as unified base learners within probabilistic boosting, providing a robust, scalable approach for accurate uncertainty estimation.

CRediT authorship contribution statement

Jamshidjon Ganiev: Writing – original draft, Visualization, Validation, Software. **Deok-Woong Kim:** Writing – review & editing, Formal analysis, Data curation. **Seung-Hwan Bae:** Writing – original draft, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Research Foundation of Korea (NRF), South Korea grants funded by the Korea government (MSIT) (No. RS-2022-NR071978) and funded by the Ministry of Education, South Korea (No. RS-2022-NR070869); supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP), South Korea grants funded by the Korea government (MSIT) (No. RS-2022-II220448: Deep Total Recall, 10%, No. RS-2022-00155915: Artificial Intelligence Convergence Innovation Human Resources Development (Inha University)); supported in part by INHA UNIVERSITY, South Korea Research Grant.

References

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U.R. Acharya, et al., A review of uncertainty quantification in deep learning: Techniques, applications and challenges, Inf. Fusion 76 (2021) 243–297.

[2] T. Gneiting, M. Katzfuss, Probabilistic forecasting, Annu. Rev. Stat. Appl. 1 (1) (2014) 125–151.

[3] G. Davis, S. Mallat, M. Avellaneda, Adaptive greedy approximations, Constr. Approx. 13 (1997) 57–98.

[4] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, et al., Xgboost: Extreme gradient boosting, 2015, pp. 1–4, R package version 0.4-2 1 (4).

[5] T. Duan, A. Anand, D.Y. Ding, K.K. Thai, S. Basu, A. Ng, A. Schuler, Ngboost: Natural gradient boosting for probabilistic prediction, in: International Conference on Machine Learning, PMLR, 2020, pp. 2690–2700.
[6] O. Sprangers, S. Schelter, M. de Rijke, Probabilistic gradient boosting machines for large-scale probabilistic regression, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1510–1520.
[7] M.A. Zamee, Y. Lee, D. Won, Self-supervised adaptive learning algorithm for multi-horizon electricity price forecasting, IEEE Access (2024).
[8] S.-I. Amari, Natural gradient works efficiently in learning, Neural Comput. 10 (2) (1998) 251–276.
[9] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nat. 521 (7553) (2015) 436–444.
[10] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning, ICML-10, 2010, pp. 807–814.
[11] S.-H. Bae, Deformable part region learning and feature aggregation tree representation for object detection, IEEE Trans. Pattern Anal. Mach. Intell. 45 (9) (2023) 10817–10834.
[12] V. Chinbat, S.-H. Bae, Ga3n: Generative adversarial autoaugment network, Pattern Recognit. 127 (2022) 108637.
[13] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Adv. Neural Inf. Process. Syst. 30 (2017).
[14] S. Badirli, X. Liu, Z. Xing, A. Bhowmik, K. Doan, S.S. Keerthi, Gradient boosting neural networks: Grownet, 2020, arXiv preprint arXiv:2002.07971.
[15] D.M. Stasinopoulos, R.A. Rigby, Generalized additive models for location scale and shape (GAMLSS) in R, J. Stat. Softw. 23 (2008) 1–46.
[16] L. Schlosser, T. Hothorn, R. Stauffer, A. Zeileis, Distributional regression forests for probabilistic precipitation forecasting in complex terrain, 2019.
[17] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1050–1059.
[18] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, Adv. Neural Inf. Process. Syst. 30 (2017).
[19] H. Lee, S. Lee, B.C. Song, Data and model uncertainty aware salient object detection, IEEE Access 12 (2024) 15016–15025.
[20] J.-Y. Baek, Y.-S. Yoo, S.-H. Bae, Generative adversarial ensemble learning for face forensics, IEEE Access 8 (2020) 45421–45431.
[21] Y.-S. Baek, D.-H. Lee, Y. Jo, S.-C. Lee, W. Choi, D.-H. Kim, Artificial intelligence-estimated biological heart age using a 12-lead electrocardiogram predicts mortality and cardiovascular outcomes, Front. Cardiovasc. Med. 10 (2023) 1137892.
[22] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[23] T. Gneiting, A.E. Raftery, Strictly proper scoring rules, prediction, and estimation, J. Amer. Statist. Assoc. 102 (477) (2007) 359–378.
[24] M. Gebetsberger, J.W. Messner, G.J. Mayr, A. Zeileis, Estimation methods for nonhomogeneous regression models: Minimum continuous ranked probability score versus maximum likelihood, Mon. Weather Rev. 146 (12) (2018) 4323–4338.
[25] A. Paszke, Pytorch: An imperative style, high-performance deep learning library, 2019, arXiv preprint arXiv:1912.01703.
[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[27] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nat. 323 (6088) (1986) 533–536.
[28] A.L. Maas, A.Y. Hannun, A.Y. Ng, et al., Rectifier nonlinearities improve neural network acoustic models, in: Proc. Icml, Vol. 30, Atlanta, GA, 2013, p. 3, 1.
[29] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[31] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc. Ser. B Stat. Methodol. 67 (2) (2005) 301–320.
[32] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (2002) 2278–2324.
[33] Y. Bulatov, NotMNIST dataset, 2011, http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html.
[34] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.
[35] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).