



Conformal deep forest for uncertainty-aware classification

Jing Zhang¹ · Yunfei Qiu¹ · Libo Dong²

Received: 6 April 2025 / Accepted: 10 July 2025 / Published online: 12 August 2025
© The Author(s) 2025

Abstract

Uncertainty in deep learning models significantly impacts their performance, robustness, and reliability, making explicit uncertainty quantification a critical research focus. However, existing methods often fail to incorporate relationships between classes into uncertainty quantification, which are essential for capturing class-dependent uncertainty. To address this challenge, this paper presents two key contributions. First, the conformal dual uncertainty metric (CDUM), derived from conformal prediction, is proposed as a novel uncertainty criterion. CDUM quantifies class-wise uncertainty by capturing class relationships from two perspectives: the influence of other classes on the current class and the dominance of the current class over others. Second, the conformal deep forest (CDForest), an uncertainty-aware deep model, is proposed by incorporating three CDUM-based modules into the deep forest (DF) architecture. Leveraging these CDUM-based modules, CDForest achieves three core capabilities: (1) layer-wise uncertainty filtering to reduce high-uncertainty features, (2) class-wise weighted averaging to refine ensemble predictions, and (3) class-wise aggregation for conformal predictors to highlight class-specific differences and facilitate more precise uncertainty quantification. Extensive experiments demonstrate that CDForest significantly outperforms state-of-the-art models in terms of classification accuracy, robustness against noise, and decision reliability, further confirming its effectiveness as a highly competitive solution for uncertainty-aware classification tasks.

Keywords Conformal prediction · Deep forest · Class-wise uncertainty · Class relationship

1 Introduction

In recent years, deep learning models have made significant advances across a broad spectrum of domains and applications, including (but not limited to) visual speech analysis (Sheng et al. 2024), natural language processing (Otter et al. 2021), autonomous driving (Ma et al. 2024), pollutant prediction (Liu et al. 2024), tabular data analysis (Zheng et al. 2023), and fraud detection (Chen et al. 2024). Most studies

on deep learning are built upon deep neural networks, i.e., multiple layers of parameterized nonlinear differential modules that can be trained by backpropagation (Wright et al. 2022). Numerous studies (Jeon et al. 2023; Castellano and Vessio 2021; Jiang et al. 2023) ascribe the widespread adoption of deep neural networks to their remarkable ability of extracting intricate patterns and representations from large-scale datasets, resulting in substantial enhancements in predictive accuracy and decision-making capabilities. However, despite these impressive achievements, there still exist some challenges for deep neural networks, such as sensitivity to noise corruption (Liu et al. 2021), too many hyperparameters to tune (Yi and Bui 2021), requirement of large-scale training data (Shu et al. 2023), and inefficiency in handling uncertainty (GawLikowski et al. 2023).

Particularly, uncertainty in deep learning models has a significant impact on model performance, robustness, and generalization (GawLikowski et al. 2023). Uncertainty in deep learning models primarily stems from variability in real-world data (e.g., noise, randomness, overlapping feature distribution in different classes, unknown data from different domains), architectural design flaws,

Yunfei Qiu and Libo Dong contributed equally to this work.

✉ Jing Zhang
15804295800@163.com

Yunfei Qiu
7415575@qq.com

Libo Dong
dlblngc@163.com

¹ School of Business Administration, Liaoning Technical University, Huludao 123105, Liaoning, China

² School of Software, Liaoning Technical University, Huludao 123105, Liaoning, China

and training process inconsistencies (He et al. 2024). The potential threats posed by uncertainty are not to be underestimated. For example, uncertainty tends to be propagated and amplified through the layer-wise information transfer in deep neural networks; it induces bias in the training process of model parameters, thereby leading to a deterioration in model performance and robustness; the black-box characteristic of deep neural networks makes it challenging to explicitly analyze uncertainty within the network architecture.

Although numerous techniques have been proposed to mitigate the adverse effects of uncertainty, such as pruning (Cheng et al. 2024), regularization (Tian and Zhang 2022), and adversarial training (Qian et al. 2022), explicitly quantifying and handling uncertainty in deep learning models remains a critical and ongoing research issue (Psaros et al. 2023). The key factors for explicitly addressing the negative effects of uncertainty in deep learning models are as follows:

First, the architecture and learning mechanism of deep learning models should be designed to enhance the interpretability of intermediate layers and lay the foundation for explicit layer-wise uncertainty quantification.

Second, the uncertainty quantification approach should be designed to adequately represent and explore the diverse uncertainties within deep learning models.

For the first factor, decision-tree-based models exhibit advantages compared with neural network-based models (Yang et al. 2023). Zhou and Feng (2017, 2019) have rethought the fundamental reasons for the success of deep neural networks, attributing the success to three essential characteristics: layer-by-layer processing, in-model feature transformation, and sufficient model complexity. Furthermore, they propose an alternative to deep neural networks: a forest-based ensemble deep learning framework referred to as deep forest (DF), also known as gcForest (multi-grained cascade forest) (Zhou et al. 2017, 2019). DF employs a cascade structure where multiple layers of random forests are stacked, allowing features to be progressively augmented and refined (Zhou and Feng 2019). Additionally, DF utilizes multi-grained scanning to extract rich feature representations from raw data, making it suitable for classification and regression tasks. DF and its variants (Huang et al. 2023; Pang et al. 2022; Zhu et al. 2024), leveraging their hierarchical forest architecture, exhibit enhanced robustness to noisy data and improved interpretability compared to deep neural networks, which makes them particularly suitable for scenarios where model transparency and stability under data perturbations are critical. These characteristics of DF exhibit great potential for layer-wise explicit uncertainty quantification, which makes it possible to utilize quantified uncertainty for promoting model performance and robustness.

For the second factor, numerous uncertainty quantization methods have been proposed for deep learning models, which can be categorized into three branches, i.e., model uncertainty, data uncertainty, and the combination of model and data uncertainty (He et al. 2024). Methods addressing model uncertainty, such as Bayesian neural networks (Jospin et al. 2022), ensemble methods (Abdar et al. 2021), and deep Gaussian processes (Cao et al. 2022), focus on uncertainty arising from parameters, architectures, and distribution approximations, respectively. Methods accounting for data uncertainty, e.g., deep discriminative models (Feng et al. 2024), and deep generative models (Teng et al. 2021), seek to represent the uncertainty from the inherent randomness or noise in the data. Methods considering both model and data uncertainty, including hybrid approach (Lakshminarayanan et al. 2017), evidential deep learning model (Bao et al. 2021), and conformal prediction (Angelopoulos and Bates 2023), attempt to jointly quantify two types of uncertainty (model and data) using one model framework. Although most existing uncertainty quantification methods can quantify total uncertainty from different sources, they suffer from many limitations (Fontana et al. 2023), such as strict assumptions, lack of distribution-free validity, unreliability in finite samples, and model-specific dependence. To overcome these limitations, conformal prediction has been proposed, gaining widespread attention in the academic community (Angelopoulos et al. 2024). Conformal prediction is a statistical framework that quantifies uncertainty by constructing prediction sets for new observations, which guarantees finite-sample validity under the assumption of data exchangeability. Conformal prediction does not require strict distributional or model-specific assumptions (Angelopoulos and Bates 2023; Fontana et al. 2023; Angelopoulos et al. 2024). A large number of studies (Khurjekar and Gerstoft 2023; Lei and Bellotti 2023; Olsson et al. 2022) reveal that conformal prediction and its variants hold significant potential in the uncertainty quantification of deep learning models.

Therefore, considering the above discussions, it becomes evident that combining the advantages of both DF and conformal prediction provides a promising approach to mitigating the impact of uncertainty on deep ensemble models, thereby enhancing their performance, robustness, and reliability. While DF draws inspiration from deep architectures, it fundamentally differs from conventional neural network-based deep learning models. Hence, in this work, we regard DF as an independent ensemble-based framework with hierarchical representation capability. However, there still exist some implicit barriers that hinder the in-depth exploration of fine-grained class-wise uncertainty information in deep ensemble models. The uncertainty quantification methods (including conformal prediction and its variants) primarily focus on evaluating uncertainty based on the model's outputs

(He et al. 2024). Nevertheless, there is a lack of effective solutions for quantifying intermediate layer uncertainty and explicitly reducing uncertainty propagation within such models. This requires a more reasonable architecture and learning mechanism to better manage uncertainty within deep ensemble frameworks. Moreover, for most uncertainty quantization methods (He et al. 2024), the uncertainty of each class predicted by the classifier is individually evaluated, and the interactive relationships between different classes have not been effectively explored. In fact, the classifier exhibits a certain level of competitive relationship in its predictions across different classes (Gimeno et al. 2021). The classifier's excessive dominance in one class often implies an increase in uncertainty among its predictions for other classes (Nguyen et al. 2022). It requires an appropriate quantification way to address the impact of uncertainty from other classes to the current class, as well as the opposite impact from the current class to other classes.

This paper aims to overcome the implicit barriers associated with uncertainty quantization methods, thereby enabling the integration of the strengths of conformal prediction methods (hereafter referred to as the conformal predictor) and the DF architecture. This integration can explicitly quantify diverse uncertainty in the DF architecture and leverage the quantified uncertainty to enhance its performance, robustness, and reliability.

Specifically, this paper proposes a novel uncertainty quantization criterion based on conformal prediction, named conformal dual uncertainty metric (CDUM). Based on the outputs of conformal prediction, CDUM formulates two indicators to measure class-wise uncertainty from dual perspectives: (1) the uncertainty influenced by other classes with respect to the current class, and (2) the uncertainty caused by the dominance of the current class over other classes. By combining these two indicators, CDUM provides a comprehensive and explicit quantification of class-wise uncertainty.

Subsequently, this paper proposes the conformal deep forest (CDForest) model, incorporating three CDUM-based modules into the DF architecture. First, during training, CDForest performs the layer-wise feature selection using the normalized CDUM values, which filters out high uncertain features generated by random forests in the previous layer. Second, during inference, CDForest integrates predictions from the random forests in the output layer via the class-wise weighted average, where the weights are determined by CDUM values representing classification uncertainty for each class. Third, during post-processing (output-layer uncertainty quantization), CDForest aggregates the ensemble conformal predictors, corresponding to the random forests in the output layer, using a CDUM-based weighted aggregation. This CDUM-based weighted aggregation enables the output-layer uncertainty quantization to account for

the differences in the effectiveness of conformal predictors across different classes.

The contributions of this paper are summarized as follows:

- This paper proposes the conformal dual uncertainty metric (CDUM), which advances conformal prediction by meticulously incorporating the interactive relationships between different classes into the uncertainty quantification process. CDUM provides a diverse and explicit way to quantify class-wise uncertainty. In addition, it supports the construction of multi-functional embedded modules for more reliable deep learning models;
- This paper proposes the CDForest, which incorporates three CDUM-based modules into the DF architecture. These CDUM-based modules in CDForest are responsible for three vital tasks: layer-wise uncertainty filtering, class-wise weighted average of predictions, and class-wise weighted aggregation of conformal predictors. CDForest not only quantifies its internal uncertainty layer-by-layer, but also makes use of the quantified uncertainty to enhance its performance, robustness, and reliability;
- The proposed CDForest is a highly competitive solution for uncertainty-aware classification tasks. Extensive experiments show that the proposed CDForest outperforms DF and its variant in terms of classification performance, robustness and reliability.

The remainder of this paper is organized as follows. Section 2 reviews related works, including DF, conformal prediction, and their variants. Section 3 presents the preliminaries of the proposed model. Section 4 explains the formulation of CDUM. Section 5 describes the architecture and key modules of CDForest. Section 6 presents comprehensive experiments evaluating CDForest against state-of-the-art classification models with respect to accuracy, robustness, and reliability. Finally, Section 7 concludes the paper by summarizing the key contributions and outlining potential future research directions.

2 Related works

2.1 Deep Forest (DF) and its variants

While deep learning has achieved remarkable success in various fields, its complexity, reliance on large-scale data, black-box property, and difficulty in hyperparameter tuning have limited its applicability (Talaie Khoei et al. 2023). DF serves as an alternative approach that offers comparable performance to deep neural networks with easier training and usage (Zhou and Feng 2017, 2019). DF seeks to combine

the advantages of tree-based models (e.g., ease of training, insensitivity to hyperparameters) with hierarchical representation learning. The initial version of DF, i.e., *gcForest* (Zhou and Feng 2019), uses a cascaded structure of multi-layer forests for hierarchical feature representation learning and model training. Then, numerous DF's variants have been proposed, improving DF's structural design (Utkin and Rya-binin 2018; Utkin 2020), feature representation (Ming et al. 2024), computational efficiency (Pang et al. 2022; Yin et al. 2024), and other relevant characteristics. DF and its variants have been successfully applied in various fields, such as soft sensor modeling (Cui et al. 2023), fault diagnosis (Ming et al. 2024), and time series processing (Cui et al. 2023).

DF still has enormous potential to be explored. The number of layers in the cascade forest structure of DF can be adaptively determined based on the validation set (Wang et al. 2020). This provides an inspiration that designing an appropriate validation metric is an effective way to improve the performance of DF. Moreover, DF exhibits superior inherent interpretability compared to deep neural networks owing to its tree-based structure (Li et al. 2024), which also facilitates uncertainty quantification. Leveraging these advantages of DF facilitates layer-wise uncertainty quantification and feature selection while harnessing the benefits of ensemble learning. These aspects together constitute an attractive research direction.

2.2 Conformal prediction and its variants

The predictions of trained models are susceptible to errors due to various sources of uncertainty, including noisy data, insufficient data, overfitting/underfitting, distribution shifts, and algorithmic limitations. Uncertainty quantification improves decision-making based on model predictions by providing insights into the model's confidence levels and inherent limitations. Conformal prediction, a statistical framework, effectively quantifies uncertainty in model outputs (Park et al. 2024). Conformal prediction enables rigorous uncertainty quantification without imposing assumptions on the model, while requiring only minimal assumptions about the data distribution (i.e., data exchangeability). It circumvents the need for asymptotics, limit theorems, and Gaussian approximations (Gazin 2024). Recently, extensions of conformal prediction have been proposed (Foygel Barber et al. 2023; Gibbs and Candès 2021) to deal with non-exchangeable data, which have expanded the applicability scope of conformal prediction. Given a relatively small calibration set, numerous variants of conformal prediction can be applied in both online and offline settings without the need for model retraining (Bhatnagar et al. 2023).

In conformal prediction, the prediction set (hereafter named conformal prediction set) serves as a key output that quantifies the uncertainty associated with predictions while

maintaining statistical reliability. The construction of a conformal prediction set relies on the calibration of conformity scores, which measure how well a given instance conforms to the distribution of the training data. Using a holdout calibration set, these conformity scores are ranked to determine a threshold corresponding to the desired significance level. The conformal prediction set then includes all potential labels whose conformity scores are above this threshold, ensuring the set's validity under the assumption of exchangeability. In practice, the size of the conformal prediction set is usually used as a metric to offer insights into uncertainty (Vovk et al. 2022).

However, the size of the conformal prediction set reveals only partial information on model performance and uncertainty. Although some variants of conformal prediction are proposed to quantify class-dependent uncertainty with label-conditional validity mechanism (Vovk et al. 2022), they treat classes individually and cannot adequately capture their relationships. The relationships between different classes are associated with the model's epistemic and estimation uncertainty, both of which are primary sources of uncertainty in machine learning (Gruber et al. 2023). For instance, the misclassification of samples from one class as belonging to another indicates confusion in the model's decision boundaries. This confusion exposes deficiencies in the model's structure and parameter training (Gruber et al. 2023).

Moreover, conformal prediction sets of conformal prediction are predominantly employed in post-processing stages, such as model calibration and uncertainty quantification (Angelopoulos and Bates 2023), yet effective mechanisms for integrating them into the model optimization process have not been sufficiently developed. A key challenge is that conformal prediction sets merely reflect the uncertainty contained in the model's final outputs. As a result, they offer limited insight into the internal uncertainty representations, especially for those within the intermediate layers of deep learning models.

3 Preliminaries

The conditional inductive conformal prediction (CICP) (Vovk et al. 2022) and DF framework (Zhou and Feng 2019) form the foundation of the proposed CDForest, and they are summarized in this section.

3.1 Conditional Inductive Conformal Predictor (CICP)

Inductive conformal predictor (ICP) (Angelopoulos and Bates 2023) is a widely used approach to conformal prediction that effectively enhances the computational efficiency over

standard conformal prediction. ICP splits the training set into a proper training set and a calibration set. A classifier (or regressor) is trained using the proper training set, and the conformal prediction set is obtained from the calibration set. The flexibility of ICP and its variants facilitates their integration with existing classifiers, enabling faster, scalable predictions without compromising reliability (Angelopoulos and Bates 2023). For classification tasks, it has been shown by Ding et al. (Ding et al. 2023) that both standard conformal prediction and ICP tend to overpredict the majority class and underpredict the minority class. The main reason is that the p -values of standard conformal prediction (or ICP) are estimated across samples of all classes in the training set (or the calibration set). In contrast, CICP (Vovk et al. 2022) is proposed to achieve class-balanced coverage on every ground truth class. CICP estimates the p -values within each class separately, rather than across all classes. The formulation of CICP is provided as follows.

Given a training set $\mathcal{Z} = \{z_1, z_2, \dots, z_N\}$, where N is the number of instances in $\mathcal{Z}$. Each instance $z_i = (x_i, y_i) \in \mathcal{Z}$ consists of an object $x_i \in \mathbb{R}^d$ and its label $y_i \in \mathcal{Y} = \{1, \dots, M\}$, where d is the feature dimension and M is the number of classes. In CICP, the training set $\mathcal{Z}$ is split into two non-empty parts: the proper training set $\mathcal{Z}_T$ and the calibration set $\mathcal{Z}_C$, where (T, C) forms a partition of the index set $\{1, 2, \dots, N\}$, $T \cup C = \{1, 2, \dots, N\}$, $T \cap C = \emptyset$, $\mathcal{Z}_T = \{z_j = (x_j, y_j) \in \mathcal{Z} | j \in T\}$, and $\mathcal{Z}_C = \{z_i = (x_i, y_i) \in \mathcal{Z} | i \in C\}$.

The underlying classifier is trained using the proper training set $\mathcal{Z}_T$, and then it is used to build the conformity score function $\mathcal{A}$. Given a test object $x \in \mathbb{R}^d$, its actual class label is unknown and needs to be predicted. Suppose $y \in \mathcal{Y}$ be a potential class of x , then the conformity score for the instance $z = (x, y)$ can be computed by $\mathcal{A}(\mathcal{Z}_T, (x, y))$ which is used to measure how well the instance z conforms to the proper training set $\mathcal{Z}_T$. In practice, $\mathcal{A}$ is usually defined based on the underlying classifier (for details, refer to Section 4.1).

Let $C^y = \{i \in C | y_i = y\}$ denote the subset of C containing the indices of calibration instances belonging to class y . The class-wise p -value for the test object x and its potential class y is defined by

$$p(x, y) = \left(\left| \{i \in C^y | \alpha_i < \alpha^y\} \right| + \tau \left| \{i \in C^y | \alpha_i = \alpha^y\} \right| \right) / (|C^y| + 1) \quad (1)$$

where $|\cdot|$ denotes the set size operator (recording the number of elements in a set), $p(x, y)$ denotes the p -value of x with respect to class label y , $\alpha^y = \mathcal{A}(\mathcal{Z}_T, (x, y))$, $\alpha_i = \mathcal{A}(\mathcal{Z}_T, (x_i, y_i))$, $(x_i, y_i) \in \mathcal{Z}_C$, $\tau \in [0, 1]$.

Based on the class-wise p -value (see Eq. (1)), CICP is defined as a set predictor for the test object x :

$$\Gamma^\epsilon(x) = \{y \in \mathcal{Y} | p(x, y) > \epsilon\} \quad (2)$$

where $\Gamma^\epsilon(x)$ is the conformal prediction set provided by CICP with respect to the test object x , $\epsilon \in (0, 1)$ is the significance level ($1 - \epsilon$ is known as the confidence level).

3.2 Deep Forest (DF)

DF (Zhou and Feng 2019) mainly consists of two key components: the multi-grained scanning procedure and the cascade forest structure. The multi-grained scanning procedure of DF is a feature augmentation method that transforms the raw features of an input object into multiscale high-dimensional features. For the cascade forest structure of DF, the first layer receives objects with high-dimensional features; each subsequent layer receives two types of features: outputs of random forests in the previous layer and high-dimensional features of objects produced by the multi-grained scanning procedure. The architecture and description of DF can be found in Zhou and Feng (2019). The final prediction of DF is obtained by averaging the outputs (i.e., class membership vectors) of the random forests in its output layer. The assumption behind the layer-wise concatenated information propagation and the final prediction is that the random forests in DF have consistent classification reliability across multiple classes. It is too strict for practical applications. Some validation metric is needed to capture the differences in classification uncertainty of random forests across classes.

4 Conformal Dual Uncertainty Metric (CDUM)

The conformal prediction set refers to a collection of possible outcomes or predicted labels generated by a model for a given input, based on a specified confidence level (Xu and Xie 2023). Unlike traditional methods that provide a single-point prediction, the conformal prediction set encompasses all potential predictions that are consistent with the model's uncertainty, offering a broader, more comprehensive view of the possible outcomes (Angelopoulos and Bates 2023; Dheur et al. 2024; Park and Simeone 2024). There are several criteria derived from conformal prediction sets (Vovk et al. 2022) which primarily address overall classification uncertainty or individual class-wise uncertainty. However, these criteria fail to adequately capture the impact of interactions between different classes on uncertainty quantization. This limitation highlights the challenge of incorporating interactive relationships between different classes into the uncertainty quantization process.

CDUM is proposed as a class-wise uncertainty quantization criterion to enable the in-depth exploration of conformal prediction sets. CDUM quantifies the class-wise uncertainty of a machine learning model from dual perspectives: the passive uncertainty and the active uncertainty. The *passive uncertainty*

indicator quantifies uncertainty based on the influence of other classes on the current class. In contrast, the *active uncertainty indicator* quantifies uncertainty by evaluating the impact of the current class on other classes.

To achieve the purpose of CDUM, the training set $\mathcal{Z} = \{z_1, z_2, \dots, z_N\}$ needs to be further split into three non-empty parts: the proper training set $\mathcal{Z}_T$, the calibration set $\mathcal{Z}_C$ and the validation set $\mathcal{Z}_V$, where (T, C, V) forms a partition of the index set $\{1, 2, \dots, N\}$, T, C, V are mutually disjoint, and $T \cup C \cup V = \{1, 2, \dots, N\}$, $\mathcal{Z}_T = \{z_j = (x_j, y_j) \in \mathcal{Z} | j \in T\}$, $\mathcal{Z}_C = \{z_i = (x_i, y_i) \in \mathcal{Z} | i \in C\}$, $\mathcal{Z}_V = \{z_v = (x_v, y_v) \in \mathcal{Z} | v \in V\}$. The classifier is trained using $\mathcal{Z}_T$. The conformal prediction is performed based on $\mathcal{Z}_C$ and $\mathcal{Z}_V$. Before introducing the computation steps of CDUM, we first propose the random forest-based conformity score.

4.1 Random forest-based conformity score

Let $\mathcal{F}$ denote a random forest trained by the proper training set $\mathcal{Z}_T$. Based on $\mathcal{F}$, the conformity score function $\mathcal{A}$ with respect to an object x and its class label y is given by

$$\mathcal{A}(\mathcal{Z}_T, (x, y)) = (1 + \mathcal{F}'(x) - \max_{y' \in \mathcal{Y}, y' \neq y} \{\mathcal{F}'(x)\})/2 \quad (3)$$

where $y \in \mathcal{Y} = \{1, 2, \dots, M\}$ denotes the class label, $\mathcal{F}'(x) \in (0, 1)$ denotes the membership values predicted by the random forest $\mathcal{F}$ for classifying object x into class y . $\mathcal{A}(\mathcal{Z}_T, (x, y))$ represents the difference between the membership value of class y and the maximum of the remaining classes' membership values with respect to object x in a certain degree.

4.2 Passive uncertainty indicator

Let $z_v = (x_v, y_v)$ denote an instance in the validation set $\mathcal{Z}_V$, where x_v is the object vector and y_v is the actual class label of x_v , $v \in V$. Treat x_v as a test object by temporarily masking its actual class label y_v . Then, substitute x_v into Eq. (1) in place of x to compute the class-wise p -value $p(x_v, y)$, where $y \in \mathcal{Y}$ represents a potential class for x_v .

According to Eq. (2), the conformal prediction set with respect to x_v can be computed as $\Gamma^\epsilon(x_v) = \{y \in \mathcal{Y} | p(x_v, y) > \epsilon\}$, where $\epsilon \in (0, 1)$ is the significance level. Based on $\Gamma^\epsilon(x_v)$, the cardinality of the conformal prediction set associated with the misclassification of the instance (x_v, y_v) is defined as

$$\varrho(x_v, y_v) = \begin{cases} |\Gamma^\epsilon(x_v) \setminus \{y_v\}|, & |\Gamma^\epsilon(x_v)| > 0 \\ 1, & |\Gamma^\epsilon(x_v)| = 0 \end{cases} \quad (4)$$

where $\varrho(x_v, y_v)$ quantifies classification uncertainty for the instance (x_v, y_v) using piecewise computation of set size,

$\Gamma^\epsilon(x_v) \setminus \{y_v\}$ denotes the set difference between $\Gamma^\epsilon(x_v)$ and $\{y_v\}$, $|\cdot|$ denotes the set size operator, $v \in V$, $(x_v, y_v) \in \mathcal{Z}_V$. When $\Gamma^\epsilon(x_v)$ is nonempty, $\varrho(x_v, y_v)$ represents the number of labels in $\Gamma^\epsilon(x_v)$ that are irrelevant to y_v ; when $\Gamma^\epsilon(x_v)$ is empty, it implies the classifier cannot provide a definitive decision for x_v , $\varrho(x_v, y_v)$ quantizes this uncertainty situation as 1; when $\Gamma^\epsilon(x_v)$ only contains label y_v , $\varrho(x_v, y_v) = 0$ which indicates the quantized uncertainty is zero for object x_v .

Based on Eq. (4), the *passive uncertainty indicator* for class $y \in \mathcal{Y}$, $\varphi(y)$, is defined as

$$\varphi(y) = \frac{1}{|V^y|} \sum_{v \in V^y} \varrho(x_v, y_v) \quad (5)$$

where $V^y = \{v \in V | y = y_v\}$ denotes the subset of V that contains the indices of instances belonging to class y . The smaller $\varphi(y)$, the lower the classifier's uncertainty about misclassifying objects of class y as other classes (or leaving them unclassified).

4.3 Active Uncertainty Indicator

The *passive uncertainty indicator* focuses on representing the influence of other classes on the current class. In contrast, the influence of the current class on other classes should also be considered. To address this problem, the *active uncertainty indicator* is proposed, which quantifies the uncertainty by evaluating the impact of the current class on other classes. Let $V^{-y} = V \setminus V^y = \{v \in V | y \neq y_v\}$ denote the set difference between V and V^y , where y_v is the actual class label of x_v , $v \in V$. Namely, V^{-y} contains the indices of instances belonging to all classes in $\mathcal{Y}$ except class y . The *active uncertainty indicator* for class $y \in \mathcal{Y}$, $\psi(y)$, is defined as

$$\psi(y) = \frac{1}{|V^{-y}|} \sum_{v \in V^{-y}} |\Gamma^\epsilon(x_v) \cap \{y\}| \quad (6)$$

where $\Gamma^\epsilon(x_v)$ denotes the conformal prediction set associated with x_v . A smaller value of $\psi(y)$ indicates that the classifier is less likely to treat class y as a dominant class over other classes. In other words, the smaller $\psi(y)$, the lower the classifier's uncertainty about overestimating the dominance of class y .

4.4 Definition of CDUM

CDUM aims to provide a reliable way for the evaluation of class-wise uncertainty by integrating both the *passive uncertainty indicator* and the *active uncertainty indicator*. The CDUM for class $y \in \mathcal{Y}$, $\mathcal{U}(y)$, is defined as

$$\mathcal{U}(y) = \lambda \varphi(y) + (1 - \lambda) \psi(y) \quad (7)$$

where $\lambda \in (0, 1)$ is a coefficient, $\mathcal{U}(y) \in (0, 1)$. The smaller $\mathcal{U}(y)$, the lower the classifier's uncertainty for class y .

CDUM addresses the limitation of conformal prediction in capturing interactions among classes, thus providing a refined metric for characterizing class-wise uncertainty. CDUM can be seamlessly integrated into model stacking frameworks, providing support for feature selection, class-wise weighted averaging of ensemble predictions, and aggregation of conformal predictors. The algorithms for computing CDUM are detailed in Tables 12, 13, 14, and 15 of the Appendix 1.

5 Conformal Deep Forest (CDDForest)

CDDForest is proposed by introducing DCUM into the workflow of DF, where new modules are constructed for the purposes of performance improvement and capability enhancement. Section 5.1 presents an overall architecture of CDDForest and highlights the key differences between CDDForest and DF. Sections 5.2 and 5.3 describe the training and inference processes of CDDForest, respectively.

5.1 Overall architecture of CDDForest

The overall architecture of CDDForest is shown in Fig. 1. CDDForest inherits the multi-grained scanning procedure from DF, which generates multiscale high-dimensional features through multiple sliding windows. In each layer of the cascade forest structure in CDDForest, except for the output layer, the outputs of the random forests are selected based on normalized CDUM values to filter out high-uncertainty membership values. The selected membership values are concatenated with features generated by the multi-grained scanning procedure, forming the input for the next layer in the cascade forest structure. In the output layer of CDDForest, the outputs of the random forests are integrated using the conformal class-wise weighted averaging to generate the final prediction. Similarly, the ensemble conformal predictors derived from the random forests for uncertainty quantization are aggregated via the conformal class-wise weighted averaging to ensure consistent uncertainty assessment. The weights of random forests, used for both final prediction generation and aggregated uncertainty quantization, are derived from CDUM values, which quantify the class-wise classification uncertainty.

The main differences between DF and CDDForest can be summarized as follows:

- In DF, the outputs of random forests in the cascade forest structure are concatenated with the input features generated by the multi-grained scanning procedure to form the input for the next layer. In contrast, CDDForest selects the outputs of random forests based on CDUM values to

exclude predictions with high uncertainty, thereby providing more reliable inputs to the next layer;

- In DF, the final class membership vector is obtained by averaging the membership vectors produced by random forests in the output layer. Each random forest is assigned equal weight in the classification process. In contrast, CDDForest employs the conformal class-wise weighted averaging to compute the final class membership vector. The weights of random forests are derived from their CDUM values, reflecting the class-wise uncertainty of each random forest. This contributes to a more precise final decision by incorporating class-wise uncertainty into the weight computation;
- DF provides classification results for test instances. In contrast, CDDForest not only provides classification results but also quantifies the classification uncertainty using the aggregated conformal prediction, which leverages the strengths of ensemble conformal predictors to improve robustness and mitigate the biases in uncertainty quantification caused by a single conformal predictor.

5.2 CDDForest training

5.2.1 Multi-grained scanning in CDDForest

CDDForest employs the multi-grained scanning procedure on the raw objects of training set $\mathcal{Z}$ to generate the proper training sets $\{\mathcal{Z}_{T\backslash}\}_{\backslash=1}^S$, the calibration sets $\{\mathcal{Z}_{C\backslash}\}_{\backslash=1}^S$ and the validation sets $\{\mathcal{Z}_{V\backslash}\}_{\backslash=1}^S$, where S denotes the number of sliding windows, each with a different size, and $\backslash$ denotes the index of a sliding window. The detailed description of the multi-grained scanning can be found in Zhou and Feng (2019). According to the cascade forest structure of CDDForest (as shown in Fig. 1), the relationship between the layer index ℓ and the sliding window index $\backslash$ can be defined based on function $f(\ell, S)$:

$$\backslash = f(\ell, S) = 1 + ((\ell - 1) \bmod S) \quad (8)$$

where $f(\ell, S)$ denotes the function to derive the sliding window index $\backslash$ from layer index ℓ and sliding window number S , $\bmod$ denotes the modulus operator, ℓ denotes the layer index with $\ell \in \{1, \dots, L\}$, and L denotes the total number of layers in CDDForest.

$f(\ell, S)$ provides an effective way to determine which training set's objects should be concatenated with the selected outputs of the ℓ -th layer to form the extended objects that serve as inputs for the $(\ell + 1)$ -th layer. For example, suppose there are three different sliding windows in the multi-grained scanning procedure, i.e., $S = 3$, then the sliding window index is determined by the Eq. (8), i.e., $f(\ell, 3) = 1 + ((\ell - 1) \bmod 3)$. This formula cycles through the three sliding windows (1, 2, 3) as ℓ increases. For instance, at the second layer (i.e., $\ell = 2$), $f(2, 3) = 2$, which indicates that $\mathcal{Z}_{T_2}$ should be the training set whose objects will be concatenated with the outputs of

the trained random forests in the second layer. When $\ell = 4$, the sliding window index resets to $f(4, 3) = 1$, which indicates that objects in $\mathcal{Z}_{T1}$ will be concatenated with random forests' outputs of the fourth layer, then the concatenated objects will be fed to the fifth layer.

Several specific improved versions of multi-grained scanning procedures, such as confidence screening and feature screening (Pang et al. 2022), can also be applied to CDForest to improve the quality of generated features. This paper focuses on introducing CDUM into DF framework; thus, detailed derivations of the multi-grained scanning procedure and its modifications are not included here due to space limitations, they are extensively covered in Pang et al. (2022).

5.2.2 Layer-wise training with CDUM-based class-wise output selection

As shown in Fig. 1, CDForest adopts the cascade forest structure of DF. In the first layer ($\ell = 1$), random forests are trained using the proper training set $\mathcal{Z}_{Tf(1,S)}$, and their outputs are concatenated with the objects in $\mathcal{Z}_{Tf(1,S)}$, resulting in the extended object set that serves as the input for the random forests in the second layer.

In each subsequent layer $\ell \in \{2, \dots, L\}$ of CDForest, random forests are trained on the extended object set formed in the previous layer, and their outputs are selected and concatenated with objects in the proper training set $\mathcal{Z}_{Tf(\ell,S)}$ to construct the extended object set for the next layer. The class-wise output selection for random forests in each layer is performed based on CDUM values, as detailed below.

Let $\mathcal{F}_{\ell,t}$ denote the t -th trained random forest in the ℓ -th layer of CDForest, the CDUM values based on $\mathcal{F}_{\ell,t}$ can be computed using the algorithm shown in Appendix Table 15. Then, the normalized CDUM values for class y across random forests $\mathcal{F}_{\ell,t}$, $t \in \{1, \dots, T\}$, can be defined as:

$$u_{\ell,t}^y = \mathcal{U}_{\ell,t}(y) / \left(\sum_{t \in \{1, \dots, T\}} \mathcal{U}_{\ell,t}(y) \right) \quad (9)$$

where $\mathcal{U}_{\ell,t}(y)$ denotes the CDUM value for class $y \in \mathcal{Y}$ computed based on random forest $\mathcal{F}_{\ell,t}$ (see Appendix Table 15), T denotes the total number of random forests in the ℓ -th layer. A smaller $u_{\ell,t}^y$ indicates lower uncertainty of random forest $\mathcal{F}_{\ell,t}$ with respect to class y .

Let $\hat{u}_{\ell}^y$ denote the $\lceil (T+1)\vartheta \rceil / T$ -th empirical quantile of $u_{\ell,1}^y, \dots, u_{\ell,T}^y$, where $\lceil \cdot \rceil$ represents the ceiling function, and $\vartheta \in (0, 1)$ is a coefficient. Let $\tilde{\mathbf{x}}_{\ell}$ denote an extended object (feature vector) that is formed by concatenating the selected outputs of the random forests in the $(\ell - 1)$ -th layer and the objects in $\mathcal{Z}_{Tf(\ell)}$. The extended object $\tilde{\mathbf{x}}_{\ell}$ serves as the input to the random forests in the ℓ -th layer. Subsequently, the class-wise output selection for random forests in the ℓ -th layer of CDForest is conducted as follows:

First, the random forests in the ℓ -th layer are selected under the following condition:

$$\sqcup_{\ell}^y = \left\{ t \in \{1, \dots, T\} \mid u_{\ell,t}^y \leq \hat{u}_{\ell}^y \right\} \quad (10)$$

where $\sqcup_{\ell}^y$ is the index set of the random forests in the ℓ -th layer whose normalized CDUM values for class y are smaller than or equal to the quantile $\hat{u}_{\ell}^y$.

Second, the set of selected outputs (class-wise memberships) for the extended object $\tilde{\mathbf{x}}_{\ell}$, $\mathbf{f}_{\ell}^y$, is obtained as:

$$\mathbf{f}_{\ell}^y = \left\{ \mathcal{F}_{\ell,t}^y(\tilde{\mathbf{x}}_{\ell}) \mid t \in \sqcup_{\ell}^y \right\} \quad (11)$$

where $\mathcal{F}_{\ell,t}^y(\tilde{\mathbf{x}}_{\ell})$ denotes the membership value of class y for $\tilde{\mathbf{x}}_{\ell}$, which is predicted by the random forest $\mathcal{F}_{\ell,t}$, $t \in \sqcup_{\ell}^y$ and $y \in \mathcal{Y} = \{1, \dots, M\}$. Third, the selected output vector of the ℓ -th layer, $\mathbf{o}_{\ell}$, is built as:

$$\mathbf{o}_{\ell} = [\mathbf{f}_{\ell}^1, \dots, \mathbf{f}_{\ell}^M] \quad (12)$$

where M denotes the total number of classes.

The extended object input to the $(\ell + 1)$ -th layer, $\tilde{\mathbf{x}}_{\ell+1}$, can be formed by concatenating the object $\mathbf{x} \in \mathcal{Z}_{Tf(\ell)}$ and the selected output $\mathbf{o}_{\ell}$ as follows:

$$\tilde{\mathbf{x}}_{\ell+1} = [\mathbf{x}, \mathbf{o}_{\ell}] \quad (13)$$

Objects in the proper training set $\mathcal{Z}_{Tf(\ell)}$, the calibration set $\mathcal{Z}_{Cf(\ell)}$, and the validation set $\mathcal{Z}_{Vf(\ell)}$ can be extended following above steps, so as to form the extended sets $\tilde{\mathcal{Z}}_T$, $\tilde{\mathcal{Z}}_C$ and $\tilde{\mathcal{Z}}_V$ for the random forests in the $(\ell + 1)$ -th layer, respectively. Then, in the $(\ell + 1)$ -th layer of CDForest, random forests are trained using $\tilde{\mathcal{Z}}_T$. CDUM related computations subsequently performed using the extended calibration set $\tilde{\mathcal{Z}}_C$ and the extended validation set $\tilde{\mathcal{Z}}_V$, based on the trained random forests. Tables 16, 17 and 18 in Appendix 1 illustrate the algorithms related to the training process of CDForest.

For DF (Zhou and Feng 2019), all the outputs (membership values) of the random forests in the current layer are concatenated with the features generated by the multi-grained scanning process to serve as inputs for the next layer. However, the uncertainty in the membership values of each random forest has not been sufficiently addressed. In contrast, for CDForest, the outputs of the random forests in each layer are first selected based on the class-wise classification uncertainty, and then concatenated with the corresponding objects provided by the multi-grained scanning process.

5.3 CDForest inference with conformal class-wise weighted averaging

As described in Zhou and Feng (2019), in the last layer of DF, the outputs of random forests are averaged to generate the

final class membership vector, and the class with the highest membership value is selected as the final decision for the input object. This approach is based on the assumption that the random forests contribute equally to the final decision and exhibit consistent reliability across all classes. However, this assumption often fails in real-world applications, as different random forests exhibit varying levels of reliability across different classes (or even within the same class), particularly in scenarios involving highly complex and diverse datasets.

CDForest addresses the limitation of treating all random forests as equally important in the final decision by employing a CDUM-based weighted average mechanism. Specifically, in the output layer (i.e., the L -th layer) of CDForest, the outputs of the random forests are integrated using class-wise weights derived from CDUM values to reflect class-specific uncertainties. Based on CDUM values, the weight of the t -th random forest in the L -th layer of CDForest for class $y \in \mathcal{Y}$, $w_{L,t}^y$, is defined by

$$w_{L,t}^y = (1 - \mathcal{U}_{L,t}(y)) / (1 + \mathcal{U}_{L,t}(y)) \quad (14)$$

where $\mathcal{U}_{L,t}(y) \in (0, 1)$ denotes the CDUM value of class y for the t -th random forest in the L -layer of CDForest, it can be obtained based on Eqs. (3)–(7) with extended objects in $\tilde{\mathcal{Z}}_C$ and $\tilde{\mathcal{Z}}_V$, $w_{L,t}^y \in (0, 1)$, $t \in \{1, \dots, T\}$. A larger value of $w_{L,t}^y$ indicates that the t -th random forest in the L -th layer of CDForest contributes more to the final decision of objects belonging to class y .

For a test object $\mathbf{x}$, CDForest employs the multi-grained scanning procedure to generate S test objects, $\{\mathbf{x}_{\setminus}\}_{\setminus=1}^S$, where S denotes the number of sliding windows, $\mathbf{x}_{\setminus}$ denotes the object derived from $\mathbf{x}$ by applying the $\setminus$ -th sliding window, $\setminus = 1, \dots, S$. The function $f(\ell)$ defined in Eq. (8) provides a systematic way to determine which test object should be selected for use in the ℓ -th layer of CDForest. For instance, in the inference process of CDForest, the test object $\mathbf{x}_{f(\ell)}$ should be used as part of the extended object in the ℓ -th layer.

Guided by $f(\ell)$, test objects $\{\mathbf{x}_{\setminus}\}_{\setminus=1}^S$ are fed into the corresponding layers of CDForest and they undergo the layer-wise information propagation. Finally, in the output layer (the L -th layer) of CDForest, the extended test object can be constructed by $\tilde{\mathbf{x}}_L = [\mathbf{o}_{L-1}, \mathbf{x}_{f(L-1)}]$, according to Eqs. (9)–(13). This extended object is used as the basis for producing the final output of CDForest.

Let $\mathcal{F}_{L,t}$ denote the t -th random forest in the L -th layer of CDForest, and let $\mathcal{F}_{L,t}^y(\tilde{\mathbf{x}}_L)$ denote the membership value of class y for the extended test object $\tilde{\mathbf{x}}_L$, as predicted by $\mathcal{F}_{L,t}$. Considering all T random forests in the L -th layer of CDForest, the weighted average of the membership values of class y for $\tilde{\mathbf{x}}_L$ is given by

$$\tilde{f}_L^y(\tilde{\mathbf{x}}_L) = \sum_{t \in \{1, \dots, T\}} w_{L,t}^y \mathcal{F}_{L,t}^y(\tilde{\mathbf{x}}_L) \quad (15)$$

where $\tilde{f}_L^y(\tilde{\mathbf{x}}_L)$ denotes the weighted average membership value of class y for $\tilde{\mathbf{x}}_L$, $y \in \mathcal{Y} = \{1, \dots, M\}$, $\mathcal{Y}$ denotes the set of M classes. This weighted average represents the combined contribution of all T random forests in the L -th layer toward determining the membership value of class y .

Then, based on the SoftMax function, the final class membership value of class $y \in \mathcal{Y}$ for the extended object $\tilde{\mathbf{x}}_L$ is defined by

$$\kappa^y(\tilde{\mathbf{x}}_L) = \exp(\tilde{f}_L^y(\tilde{\mathbf{x}}_L)) / \left(\sum_{y'=1}^M \exp(\tilde{f}_L^{y'}(\tilde{\mathbf{x}}_L)) \right) \quad (16)$$

The final class membership vector of CDForest for the extended test object $\tilde{\mathbf{x}}_L$ is constructed by $\boldsymbol{\kappa}(\tilde{\mathbf{x}}_L) = [\kappa^1(\tilde{\mathbf{x}}_L), \dots, \kappa^M(\tilde{\mathbf{x}}_L)]$. Thus, the final decision for $\tilde{\mathbf{x}}_L$ predicted by CDForest is given by

$$\hat{y} = \underset{y \in \{1, \dots, M\}}{\operatorname{argmax}} \kappa^y(\tilde{\mathbf{x}}_L) \quad (17)$$

5.4 Uncertainty quantization with conformal weighted aggregation

The uncertainty quantification of random forests in CDForest can be performed using conformity scores, p -values, and CICP prediction sets (Angelopoulos and Bates 2023). However, such conventional methods focus on individual random forests, assuming independence or ignoring the interactions between multiple forests in ensemble models. Consequently, these methods fail to construct appropriate p -values that adequately capture the uncertainty associated with ensemble decisions in CDForest. This limitation highlights the need for uncertainty quantification methods tailored to ensemble models, since decisions in ensemble models are made based on aggregated outputs.

Many approaches to ensemble conformal predictors have been proposed to improve the performance of conformal predictors (Linusson et al. 2020; Campagner et al. 2024), which provide inspiration for this paper. Specifically, preserving the conformity scores of random forests, rather than relying solely on the final average decision, enables more accurate uncertainty quantification. Designing an effective method for aggregating ensemble conformal predictors is the key problem to be addressed.

The general idea of aggregating ensemble conformal predictors is to combine p -values from various inductive conformal predictors, such as cross conformal predictor (Vovk et al. 2020), bootstrap conformal predictor (Vovk et al. 2022), and aggregated conformal predictor (Angelopoulos and Bates 2023; Yang and Kuchibhotla 2024). The

main aggregation criterion of these studies depends on the arithmetic mean of p -values, which treats each conformal predictor equally for every class of objects. However, the arithmetic mean of p -values cannot reveal the class-specific differences among conformal predictors. CDForest overcomes this limitation by using a CDUM-based weighted aggregation method to combine class-wise p -values of ensemble conformal predictors.

For the t -th trained random forest in the L -th layer of CDForest, its p -value with respect to the extended test object $\tilde{\mathbf{x}}_L$ and its potential class $y \in \mathcal{Y}$, $p_t(\tilde{\mathbf{x}}_L, y)$, can be computed using Eq. (1) as follows:

$$p_t(\tilde{\mathbf{x}}_L, y) = \left(\left| \{i \in C^y | \tilde{\alpha}_i < \tilde{\alpha}^y\} \right| + \tau \left| \{i \in C^y | \tilde{\alpha}_i = \tilde{\alpha}^y\} \right| \right) / (|C^y| + 1) \quad (18)$$

where $t \in \{1, \dots, T\}$, $\tilde{\alpha}_i = \mathcal{A}(\tilde{\mathcal{Z}}_T, (\tilde{\mathbf{x}}_L, y_i))$, $(\tilde{\mathbf{x}}_L, y_i) \in \tilde{\mathcal{Z}}_C$, $\tilde{\alpha}^y = \mathcal{A}(\tilde{\mathcal{Z}}_T, (\tilde{\mathbf{x}}_L, y))$, $\tau \in [0, 1]$, and $\mathcal{A}(\tilde{\mathcal{Z}}_T, \cdot)$ is defined by Eq. (3).

In the output layer (i.e., the L -th layer) of CDForest, the weighted aggregation of p -values for the extended test object $\tilde{\mathbf{x}}_L$ and its potential class y is defined by

$$\tilde{p}_L(\tilde{\mathbf{x}}_L, y) = \left(\sum_{t \in \{1, \dots, T\}} w_{L,t}^y p_t(\tilde{\mathbf{x}}_L, y) \right) / \left(\sum_{t \in \{1, \dots, T\}} w_{L,t}^y \right) \quad (19)$$

where $\tilde{p}_L(\tilde{\mathbf{x}}_L, y)$ denotes the aggregated p -value for the pair $(\tilde{\mathbf{x}}_L, y)$ obtained through the weighted aggregation of individual p -values $p_t(\tilde{\mathbf{x}}_L, y)$, $t \in \{1, \dots, T\}$ and $y \in \mathcal{Y}$.

Then, based on Eq. (2), the aggregated conformal predictor for $\tilde{\mathbf{x}}_L$ is given by

$$\tilde{\Gamma}^\epsilon(\tilde{\mathbf{x}}_L) = \left\{ y \in \mathcal{Y} \mid \tilde{p}_L(\tilde{\mathbf{x}}_L, y) > \epsilon \right\} \quad (20)$$

where $\epsilon \in (0, 1)$ is the significance level ($1 - \epsilon$ is known as the confidence level). Tables 19 and 20 in Appendix 1 illustrate the algorithms of inference and aggregated conformal prediction of CDForest, respectively. It is worth noting that conformity scores of instances in calibration set and validation set can be cached for further utilization instead of recalculating them.

5.5 Discussion

The proposed CDUM, which represents class-wise uncertainty using dual indicators, quantifies not only the uncertainty contributed by other classes to the current class but also the uncertainty propagated from the current class to other classes. CDUM can be integrated as multiple plugins into deep models to enable uncertainty quantification and enhance model performance. The proposed CDForest inherits the advantages of DF (Zhou and Feng 2019), such as reduced reliance on extensive hyperparameter tuning and the

integration of ensemble learning techniques, while leveraging CDUM to enhance uncertainty quantification for classifiers.

Three CDUM-based modules are incorporated into CDForest. The first module, employed in each layer of CDForest, filters out high-uncertainty membership values (random forests' outputs) using normalized CDUM values. The second module, applied in the last layer of CDForest, facilitates the final decision-making for the ensemble of random forests using conformal class-wise weighted averaging, with CDUM-derived weights reflecting reliability differences among random forests for each class. The third module, adopted in the uncertainty quantization process of CDForest, combines class-wise p -values from ensemble conformal predictors using a CDUM-based weighted aggregator, highlighting class-wise differences among these conformal predictors.

The training-time complexity of a random forest is $\mathcal{O}(\|d\| \mathcal{Z}_T \log(|\mathcal{Z}_T|))$ where $\|$ is the number of decision trees in the random forest, d is the number of features selected for each node, and $|\mathcal{Z}_T|$ is the size of the proper training set $\mathcal{Z}_T$. The testing-time complexity of a trained random forest for each test instance is $\mathcal{O}(\|\cdot\|)$ where $\langle \cdot \rangle$ is the maximum depth of each decision tree. Precompute and cache the conformity scores of objects in the calibration set $\mathcal{Z}_C$ and the validation set $\mathcal{Z}_V$, and then use them to compute the CDUM values. This approach can effectively prevent redundant computations. Thus, the computational complexity of CDUM is $\mathcal{O}((|\mathcal{Z}_C| + |\mathcal{Z}_V|) \|\cdot\|)$, where $|\mathcal{Z}_C|$ and $|\mathcal{Z}_V|$ denote the sizes of the calibration set $\mathcal{Z}_C$ and the validation set $\mathcal{Z}_V$, respectively. Both training and testing complexities of CDForest grow linearly with the number of layers in the cascade forest structure. This paper focuses on introducing CDUM into CDForest to improve classification performance, robustness, and uncertainty quantization capability. Regarding the reduction of time complexity, some modified screening mechanisms (Pang et al. 2022) can be considered in the future.

6 Experiments

This paper proposes the class-wise uncertainty quantization criterion, i.e., CDUM, and integrates it into DF architecture to construct the CDForest. The experiments aim to evaluate the effectiveness of CDUM-based modules in CDForest for enhancing classification performance, robustness, and uncertainty quantization capability.

6.1 Experimental Setup

6.1.1 Datasets and Preparation

In the experiments, ten benchmark datasets are used to validate classification methods for binary and multiclass

classification. The datasets used in the experiments can be categorized into tabular data and image data. Table 1 summarizes the key characteristics of these datasets, such as the sizes of training set and test set, feature number, and class number. For tabular data, the datasets HIGGS, Adult, and Covertype are obtained from the UCI Machine Learning Repository (Kelly et al. n.d.), and their detailed descriptions are provided in Borisov et al. (2024). The Jannis dataset, formatted by OpenML (Vanschoren et al. 2014), is designed for multiclass classification with numerical features. The datasets Homesite Insurance, Ecom Offers, and Home Credit Default, sourced from Kaggle competitions, are employed in the experiments for binary classification tasks. The sEMG dataset referenced in Zhou and Feng (2019) comprises 3000 records, each corresponding to one of six distinct hand movements: spherical, tip, palmar, lateral, cylindrical, and hook. For image data, two widely used datasets are employed: Fashion-MNIST (Xiao et al. 2017) and CIFAR-10 (Krizhevsky 2009). Fashion-MNIST contains 60,000 training and 10,000 testing images, each sized 28×28 pixels and categorized into ten classes. Similarly, CIFAR-10 includes 50,000 training images and 10,000 testing images, each sized 32×32 pixels and distributed among ten classes.

The sizes of the training and test sets for each dataset are shown in Table 1. For each dataset, the validation set is split from the training set using tenfold cross-validation and used for tuning the hyperparameters of the classification models. The test set of each dataset is strictly reserved, never participating in model training or hyperparameter tuning. For each conformal predictor, the training set is further split into the proper training set, the calibration set, and the validation set using tenfold cross-validation. Specifically, within the training set, one fold serves as the calibration set, another constitutes the validation set, and the remaining eight folds are combined to form the proper training set.

sEMG is a time-series dataset, where EMG sensors capture 500 features per second and each record is associated

with 3000 features, the preprocessing of sEMG is based on Cornwell's solution (Zhou and Feng 2019). The preprocessing steps of Kaggle data sets Homesite Insurance, Ecom Offers, and HomeCredit Default, are based on the solutions used in Rubachev et al. (2024). For other tabular datasets, their numerical features are normalized, and their categorical features are converted into numeric features via the one-hot encoding representations. The image datasets Fashion-MNIST and CIFAR-10 are preprocessed using data augmentation (e.g., random cropping and flipping) and normalization techniques.

To evaluate the robustness of classification models, symmetric label noise (Song et al. 2023) is introduced into the training set. Specifically, the labels of training objects are randomly flipped from their ground truth classes to other classes with probability $\eta \in [0, 1)$. In experiments, two levels of symmetric label noise ($\eta = 10\%$ and $\eta = 20\%$) are used to generate the corresponding corrupted training sets. In contrast, the validation set and the test set both remain clean, i.e., free from symmetric label noise, to preserve the ground truth for performance evaluation. This experimental setup follows the methodology described in Song et al. (2023), which is widely used to evaluate the robustness of machine learning models against label noise.

6.1.2 Overview of Competing Methods

To conduct a comprehensive evaluation, the proposed CDForest is compared with nine state-of-the-art classification models, which are widely employed to deal with tabular and image data. A brief summary of these models is provided below.

XGBoost (Chen et al. 2016) is an optimized gradient boosting algorithm that builds decision trees sequentially, with each tree addressing the errors of the previous one. It incorporates regularization and advanced optimization techniques to enhance model performance and mitigate overfitting in classification tasks.

gcForest (Zhou and Feng 2019) is the standard deep forest model as an alternative to deep neural networks for classification tasks. It is realized by non-differentiable modules without backpropagation training. It employs a cascade structure of random forests and completely-random trees, automatically determining its depth and improving classification performance layer by layer. gcForest (Zhou and Feng 2019) is particularly effective for small datasets and requires less hyperparameter tuning than deep learning models.

The weighted cascade deep forest model (WCDForest) (Huang et al. 2023) is a variant of gcForest that modifies the multi-grained scanning procedure. It adopts class vector weighting and feature enhancement to optimize the cascade structure, thereby improving classification performance and reducing information loss.

Table 1 Characteristics of the 10 datasets used in experiments

Name	# Train	# Test	# Feature	# Label
HIGGS	8,800,000	2,200,000	27	2
Adult	32,561	16,281	14	2
Homesite Insurance	208,603	52,150	298	2
Ecom Offers	130,046	32,011	119	2
Home Credit Default	305,332	76,332	696	2
Jannis	66,986	16,747	54	4
sEMG	2100	900	2500	6
Covertype	464,809	116,203	54	7
Fashion-MNIST	60,000	10,000	784	10
CIFAR-10	50,000	10,000	1024	10

Multilayer perceptron (MLP) is a type of fully connected neural network that consists of an input layer, one or more hidden layers, and an output layer. An effective variant of MLP, MLP-PLR (Gorishniy et al. [n.d.](#)), is employed in the classification experiments of tabular data. MLP-PLR inherits the architecture of MLP and enhances classification performance by leveraging a nonlinear periodic feature embedding.

ResNet (He et al. [2016](#)) is a deep learning architecture that introduces residual connections into the network, facilitating more effective gradient flow during training. This structure mitigates the vanishing gradient problem and enables the construction of very deep networks, thereby maintaining high performance in image classification tasks. Gorishniy et al. ([2021](#)) constructs a variation of ResNet tailored for tabular data classification. To differentiate it from the standard ResNet, this ResNet-like tabular classifier is referred to as ResNetTab in this paper.

FT-Transformer (Gorishniy et al. [2021](#)) is a straightforward adaptation of the Transformer architecture (Vaswani et al. [2017](#)) specifically designed for tabular data. It transforms all features, both categorical and numerical, into embeddings. Categorical features are embedded using an embedding layer, while numerical features are embedded by applying element-wise multiplication with learnable weight vectors and adding bias terms. These embeddings are then processed through a stack of Transformer layers, which effectively capture complex feature interactions and relationships within the data.

RDNet-L (Kim et al. [2024](#)) is an enhanced densely connected convolutional network that revitalizes the classic DenseNet architecture by integrating concatenation-based feature mixing within a modular block structure. It comprises four hierarchical stages, each built from stacked LayerNorm-based mixing blocks followed by transition layers, using a growth-rate pattern to regulate channel concatenation.

ViT-L/16 (Vision Transformer Large with 16×16 patches) (Dosovitskiy et al. [2021](#)) is a large-scale transformer-based image classification model that divides input images into 16×16 pixel patches and processes them using pure transformer blocks. It contains approximately 307 million parameters and achieves strong performance on various vision benchmarks without relying on convolutional operations.

XGBoost, MLP-PLR, ResNetTab, and FT-Transformer are employed exclusively for tabular classification tasks due to their design or optimization for tabular data, while ResNet, RDNet-L, and ViT-L/16 serve as the baselines exclusively for image classification tasks. gcForest, WCDForest, and the proposed CDForest are applied to both tabular and image classification tasks, leveraging their flexibility to handle heterogeneous data types without task-specific modifications.

The proposed CDForest quantifies uncertainty via conformal weighted aggregation (see Section [5.4](#)), whereas other competitive classification models lack the capability to quantify uncertainty. This feature is crucial for applications requiring reliable decision confidence. Nevertheless, these competitive models can serve as the underlying models for conformal predictors, and their performance significantly influences the reliability of the conformal predictors. In the experiments, conformal prediction is employed for these competitive models to post-process their predictions, thereby generating uncertainty quantification to reflect model reliability. For a fair comparison, all methods in the experiments use the same conformity score function defined by Eq. (3), ensuring that uncertainty quantification results are comparable across models without introducing biases.

6.1.3 Evaluation Metrics

This paper employs six metrics to evaluate classification performance, robustness against perturbations, and reliability. These metrics are designed to independently assess these aspects, where reliability refers to the model's ability to provide consistent and confident predictions. The evaluation metrics are described as follows.

- The Area Under Receiver Operating Characteristic Curve (AUROC) (Yang et al. [2022](#)) is employed to evaluate the classification performance of each competitive method. AUROC measures how well a classification model distinguishes between positive and negative classes. It is widely used in binary classification tasks to compare the discriminative power of different models. For multiclass classification tasks, the one-vs-one AUROC is employed instead of AUROC (Yang et al. [2022](#)). The detailed formula definitions and theoretical analyses of AUROC and one-vs-one AUROC can be found in Yang et al. ([2022](#)). The larger AUROC (or one-vs-one AUROC), the better the model's performance.
- Precision measures the proportion of true positive predictions among all positive predictions made by the model. In multi-class settings, macro-Precision is computed by averaging the per-class Precision scores, treating each class equally regardless of its frequency.
- Recall quantifies the proportion of true positive predictions among all actual positive instances. In multi-class classification, macro-Recall is computed as the unweighted average of Recall values across all classes, giving equal importance to each class.
- F1-Score is the harmonic mean of Precision and Recall, providing a balanced measure that accounts for both false positives and false negatives. For binary classification, the F1-Score is defined as:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

For multi-class classification, the Macro-F1 Score is computed by first calculating the class-wise F1-Score independently and then averaging them across all K classes:

$$\text{Macro-F1} = \frac{1}{K} \sum_{k=1}^K F1_k = \frac{1}{K} \sum_{k=1}^K \frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k}$$

- The gap in AUROC (or one-vs-one AUROC) under different levels of symmetric label noise is used as the metric for the robustness of classification models. Let AUROC_0 and AUROC_η denote the AUROC values of the classification model trained on the clean training set and

the η -level label corrupted training set, respectively. The gap in AUROC can be defined by $|\text{AUROC}_0 - \text{AUROC}_\eta|$. In the experiments, $\eta = 10\%$ and 20% . A smaller gap in AUROC indicates a more robust classification model.

- The reliability of classification models, particularly for conformal predictors, is often evaluated using a calibration plot (Vovk et al. 2022). The calibration plot is a visual tool that compares the expected error rate, determined by the significance level $\epsilon \in (0, 1)$, with the empirical error rate observed in the model's predictions. The x -axis of the calibration plot represents the significance level ϵ , which indicates the expected proportion of predictions falling outside the conformal prediction set. The y -axis represents the empirical error rate, defined as the actual proportion of test objects for which the true labels are not contained within the conformal prediction set:

$$\text{EmpiricalErrorRate} = \frac{\#\text{TestObjectswhose true label outside conformal prediction set}}{\#\text{Total number of test objects}}$$

For a well-calibrated conformal predictor, the calibration plot should align with the diagonal line ($y = x$), indicating that the model's empirical error rates closely match the expected error rates across different significance levels. Significant deviations from this diagonal line indicate miscalibration, where the model may either be overconfident (empirical error rate higher than expected) or underconfident (empirical error rate lower than expected).

6.1.4 Parameter setting and implementation details

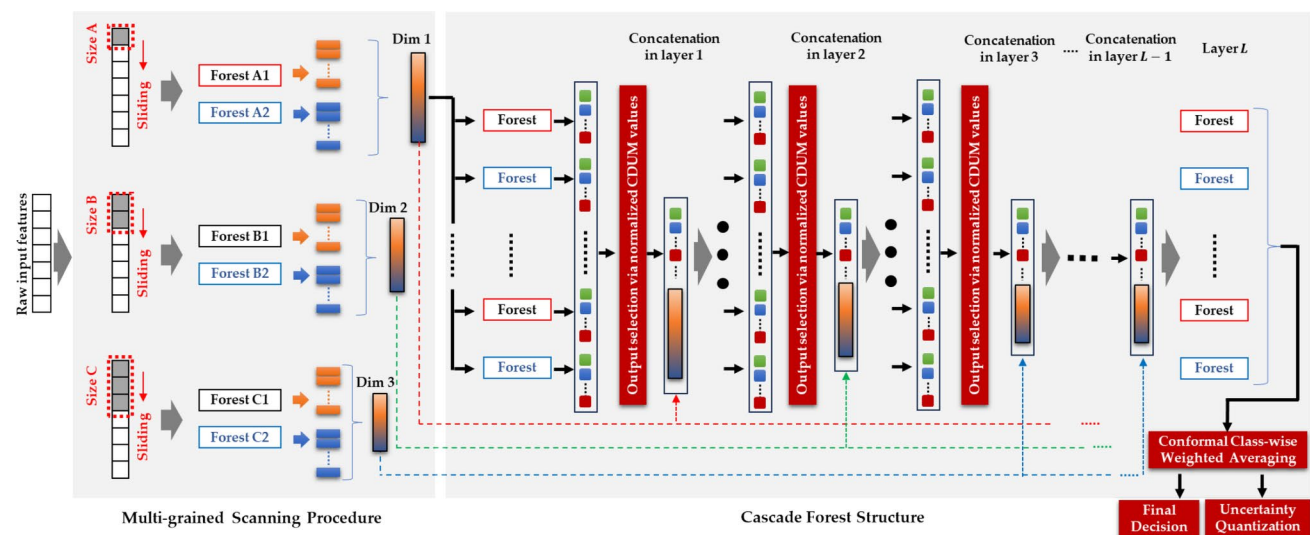
Based on the experimental findings reported in Gorishniy et al. 2021, the hyperparameter tuning of the employed classification models, including XGBoost, MLP-PLR, ResNet-Tab, and FT-Transformer, is primarily conducted using the Optuna framework (Akiba et al. 2019). The search ranges and fixed values of the hyperparameters associated with these models are summarized in Tables 21, 22, 23 and 24 of Appendix 2, respectively. The detailed descriptions on these hyperparameters can be found in the corresponding studies (Chen et al. 2016; Gorishniy et al. 2021, n.d.), which are beyond the scope of this paper. The architecture of ResNet for image classification is chosen to be ResNet34 (He et al. 2016), due to its proven performance in prior studies. The optimization of ResNet depends on Stochastic Gradient Descent (SGD) with a weight decay 5×10^{-4} and a momentum of 0.9. The maximum training epoch for ResNet is 200. The learning rate is initially set to 0.1 and decreased by the factor of 10 after 150 epochs. RDNet-L is trained for up to 100 epochs using SGD optimizer with an initial learning

rate of 0.05, a cosine decay schedule, and weight decay of 5×10^{-4} . For ViT-L/16, the official implementation in the Timm library is employed. Fine-tuning is conducted for up to 50 epochs using the AdamW optimizer with a learning rate of 3×10^{-4} , weight decay of 0.05, and a cosine annealing schedule.

In the experiments, DF models—gcForest, WCDForest, and the proposed CDForest—employ the same multi-grained scanning procedure. For raw features of objects with dimension d , the multi-grained scanning procedure uses sliding windows of three sizes: $\lfloor d/16 \rfloor$, $\lfloor d/8 \rfloor$ and $\lfloor d/4 \rfloor$, which are proportional to d . The detailed implementation of the multi-grained scanning procedure applied to image data is comprehensively described in Zhou and Feng (2019). For gcForest, WCDForest, and CDForest, the number of layers in the cascade forest structure is automatically determined in a similar way. Specifically, the original training set is split into a training set and a validation set (for CDForest, it is further split into a proper training set, a calibration set, and a validation set) using tenfold cross-validation. The training set is used to grow the cascade layer, while the validation set is used to estimate performance. If adding a new layer does not improve performance, the growth process terminates, and the total number of layers is determined. To ensure fair performance comparison, each layer in the cascade forest structure of these DF models consists of four completely random forests and four random forests. In the completely random forests, trees are built by randomly selecting features and split points, whereas in the random forests, trees are constructed using bootstrap samples and data-driven optimal

Table 2 Recall values of seven classification models on eight tabular datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	Recall						
	XGBoost	gcForest	WCDForest	MLP-PLR	ResNetTab	FT-Transformer	CDForest
HIGGS	0.760	0.764	0.767	0.758	0.759	0.783	0.795
Adult	0.835	0.820	0.834	0.815	0.817	0.824	0.842
Homesite Insurance	0.864	0.872	0.877	0.866	0.852	0.866	0.883
Ecom Offers	0.507	0.531	0.539	0.530	0.527	0.508	0.554
Home Credit Default	0.726	0.763	0.785	0.758	0.751	0.758	0.787
Jannis	0.767	0.627	0.672	0.631	0.624	0.788	0.689
sEMG	0.630	0.650	0.655	0.474	0.512	0.608	0.665
Covertypes	0.883	0.766	0.834	0.880	0.730	0.880	0.867
Average	0.747	0.724	0.745	0.714	0.697	0.752	0.760

**Fig. 1** The overall architecture of Conformal Deep Forest (CDForest). The multi-grained scanning procedure in CDForest employs three sliding window sizes. In each layer of the cascade forest structure of CDForest, the outputs of random forests are selected based on normalized CDUM values. The selected outputs are concatenated

with the corresponding features generated by the multi-grained scanning procedure. In the output layer (the L -th layer), the outputs of random forests are passed to the conformal class-wise weighted averaging module, which generates the final decision and performs uncertainty quantization

splits. Each forest contains 500 trees. The layer numbers of gcForest, WCDForest, and CDForest typically fall within the range of Jiang et al. (2023); He et al. (2024). Furthermore, gcForest and WCDForest use tenfold cross validation for class vector generation, and CDForest uses it for conformal prediction related computations, e.g., CDUM.

In conformal prediction, the significance level ϵ (see Eq. (2)) determines the maximum tolerated error rate, i.e., the probability that the true label falls outside the prediction set. A smaller ϵ yields more conservative (wider) prediction sets with higher coverage, while a larger ϵ allows for narrower sets but increases the risk of miscoverage. Thus, ϵ governs the trade-off between prediction set size and confidence. CDForest selects the optimal range of ϵ value using

an AUROC- ϵ plot, where the significance level ϵ is considered as the x -axis and AUROC as the y -axis. In Appendix 2, Fig. 6 presents the AUROC- ϵ plot generated by CDForest on the HIGGS dataset. The parameter λ in the CDUM computation (see Eq. (7)) regulates the relative contribution between the passive uncertainty indicator and the active uncertainty indicator. As λ approaches 0, CDUM primarily considers the active uncertainty, focusing on the uncertainty from the current class to all other classes. Conversely, as λ approaches 1, CDUM emphasizes the passive uncertainty, which reflects the uncertainty imposed by other classes on the current class. CDForest selects the optimal range of λ value using the similar way as that of ϵ , i.e., AUROC- λ plot. Fig. 7 in Appendix 2 shows the AUROC- λ plot generated

by CDForest on the HIGGS dataset. The trends of ϵ and λ in CDForest on other datasets are similar to those shown in Figs. 6 and 7, respectively. To reduce the complexity of parameter selection in CDForest, $\epsilon=0.05$ and $\lambda=0.9$ are fixed for all experiments.

The hyperparameter settings of gcForest, WCDForest, and CDForest are much easier than those of deep learning models (e.g., MLP-PLR, ResNetTab, FT-Transformer, and ResNet) and XGBoost. The experiments are conducted in the environment of Baidu Cloud Computing Service, using two Xeon Platinum 8350C CPUs (32 cores), two NVIDIA A100 Tensor Core GPUs (80 GB), and 400 GB of main memory. This paper focuses on the classification performance, robustness, and reliability of classification models. Improving the computational efficiency of the models will be the primary emphasis of future work.

6.2 Results and analyses

6.2.1 Classification results

There are two types of benchmark datasets used in the experiments to evaluate performance: tabular data and image data. Seven classification models—namely XGBoost, MLP-PLR, ResNetTab, FT-Transformer, gcForest, WCDForest, and the proposed CDForest—are employed for classification tasks on eight tabular datasets. For classification tasks on two image datasets, six models are used: ResNet, gcForest, WCDForest, RDNet-L, ViT-L/16, and the proposed CDForest. Tables 2, 3, 4 and 5 present the AUROC, Precision, Recall, and F1-Score results of seven models for eight tabular datasets, respectively. Table 6, 7, 8, and 9 present the AUROC, Precision, Recall, and F1-Score results of six models for two image datasets, respectively. All these results are averaged over ten independent trials, with the highest value for each dataset highlighted in bold.

Table 3 Precision values of six classification models on two image datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	Precision (%)					
	ResNet	gcForest	WCDForest	ViT-L/16	RDNet-L	CDForest
Fashion-MNIST	95.0	94.4	94.7	98.9	95.8	95.2
CIFAR-10	92.6	92.1	92.3	99.4	99.3	92.6
Average	93.8	93.3	93.5	99.2	97.6	93.9

Table 4 Recall values of six classification models on two image datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	Recall (%)					
	ResNet	gcForest	WCDForest	ViT-L/16	RDNet-L	CDForest
Fashion-MNIST	94.4	94.1	94.5	98.7	95.7	95.2
CIFAR-10	95.2	95.4	96.1	99.3	99.2	97.5
Average	94.8	94.8	95.3	99.0	97.5	96.4

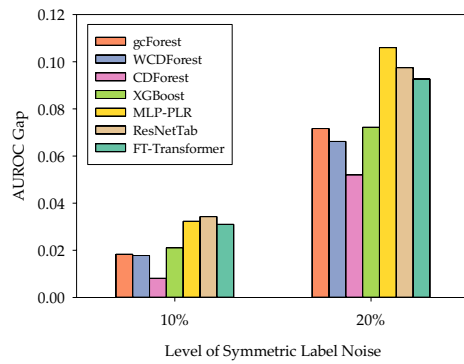
As presented in Tables 2, 3, 4, and 5, the proposed CDForest achieves the best classification performance on six out of eight tabular datasets. For the two tabular datasets, Jannis and Covtype, where CDForest does not achieve the best performance, it still produces results that are highly competitive with the top-performing method, XGBoost. CDForest demonstrates superior performance over other models in terms of the average performance across tabular datasets.

As shown in Tables 6, 7, 8, and 9, although CDForest exhibits slightly inferior performance compared to ViT-L/16 and RDNet-L, the results remain highly comparable. Moreover, CDForest outperforms the neural network-based model (ResNet) as well as tree-based deep architectures (gcForest and WCDForest) on image classification benchmarks.

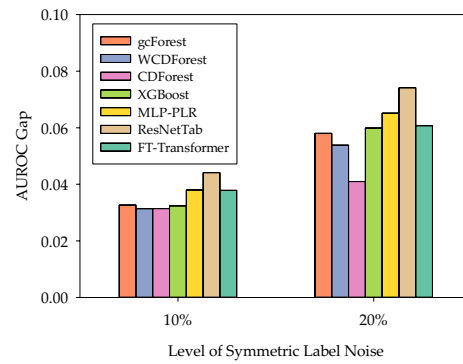
To quantify the impact of the CDUM modules on the computational effort of CDForest, Table 10 presents a comparison of the training time (second) and inference time (second) between CDForest and gcForest on the CIFAR-10 and Fashion-MNIST datasets. The results demonstrated in Table 10 regarding training and inference time confirm that the computational burden imposed by the CDUM modules on CDForest is relatively low.

The superior classification performance of CDForest is attributed to three factors:

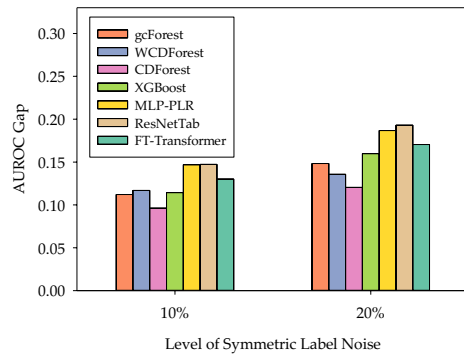
- Inheriting the strengths of tree-based models on tabular data. CDForest retains the ability of tree-based models to handle complex feature interactions and non-linear relationships effectively. Additionally, it leverages these advantages with minimal preprocessing, making it well-suited for tabular data analysis;
- Selecting more reliable features using normalized CDUM. By leveraging CDUM, CDForest identifies and selects more reliable features from the outputs of the previous layer to train the random forests in the subsequent layer. This process mitigates the adverse effects



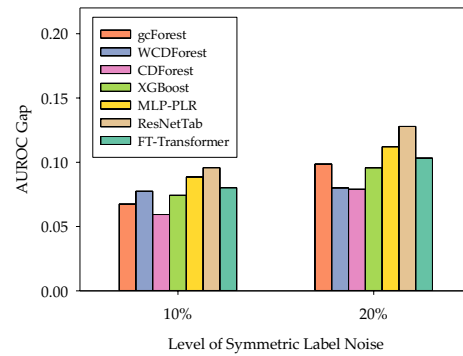
(a) AUROC Gap on HIGGS Dataset



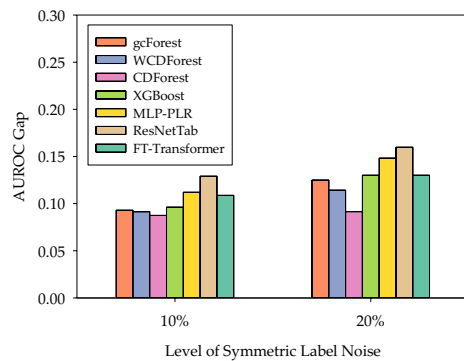
(b) AUROC Gap on Adult Dataset



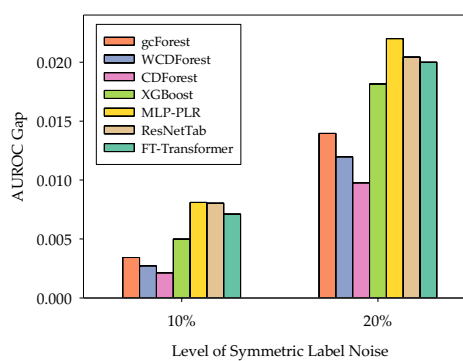
(c) AUROC Gap on Homesite Insurance Dataset



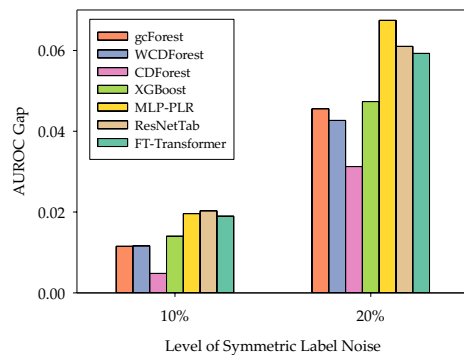
(d) AUROC Gap on Ecom Offers Dataset



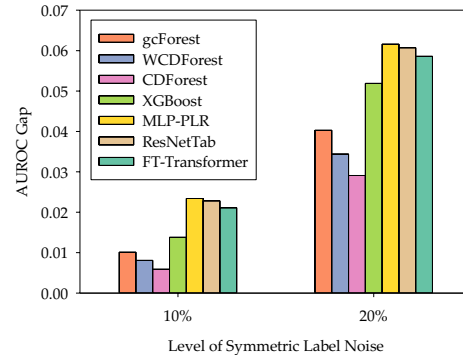
(e) AUROC Gap on Home Credit Default Dataset



(f) AUROC Gap on Jannis Dataset



(g) AUROC Gap on sEMG Dataset



(h) AUROC Gap on Coverttype Dataset

Fig. 2 Histograms of AUROC gaps for seven classification models on eight tabular datasets under two levels of symmetric label noise (10% and 20%). **a** HIGGS dataset; **b** Adult dataset; **c** Homesite Insurance dataset; **d** Ecom Offers dataset; **e** Home Credit Default dataset; **f** Janis dataset; **g** sEMG; **h** Covertype

of uncertainty on feature selection and model training, thereby enhancing the accuracy and robustness of the classification performance;

- Incorporating class-specific reliability for tailored decision aggregation. In the output layer, CDForest utilizes conformal class-wise weighted averaging to effectively account for class-specific reliability differences among random forests. Conformal class-wise weighted averaging generates tailored decision aggregations that align with the distinct characteristics of each class, thereby improving classification accuracy and robustness.

To enhance the transparency and reproducibility of the reported results, we additionally summarize the training epoch at which the best-performing model is obtained on each dataset. Specifically, during training, we monitor the Top-1 AUROC on the validation set after each epoch, and select the model checkpoint that achieves the highest validation accuracy as the final model. The test accuracy reported throughout the paper corresponds to this selected checkpoint. Table 25 in Appendix 2 presents the best epoch for each method and dataset, along with the corresponding Top-1 accuracy on the validation and test sets.

Remark: Deep neural networks typically require a large number of training epochs due to their parametric and differentiable nature. These models contain numerous learnable parameters that are updated iteratively via gradient-based optimization methods such as stochastic gradient descent (SGD). Achieving convergence to an optimal or near-optimal solution often necessitates hundreds of training epochs, particularly when using large datasets or deep architectures. Each epoch incrementally reduces the loss by fine-tuning model weights based on backpropagation. In contrast, deep forest models, such as gcForest, WCDForest and CDForest, are non-parametric, tree-based ensembles that do not rely on backpropagation or weight updates. Instead, they build predictive representations through a cascade of decision tree/forest ensembles trained layer-by-layer. Deep forest models do not require too many epochs of repetitive re-training, because the tree structures are already strong learners and benefit less from repeated exposure to the training data.

6.2.2 Robustness

As mentioned in Section 6.1.3, the AUROC gap under different levels of symmetric label noise serves as a metric

for model robustness. Figure 2a-h show the AUROC gaps in histogram form for classification models across various levels of symmetric label noise on eight tabular datasets. Similarly, Fig. 3a and b show the AUROC gaps for the Flash-MNIST and CIFAR-10 datasets, respectively.

As shown in Figs. 2 and 3, for various tabular and image datasets corrupted by different levels of symmetric label noise, CDForest consistently exhibits smaller AUROC gaps than other competitive classification models. This observation indicates that CDForest is more robust against noise corruption than other competitive classification models. The main reasons for this robustness are twofold. First, by leveraging the CDUM, CDForest effectively filters out the information disturbance caused by noise during layer-wise information propagation. Second, the final decision of CDForest is achieved through the CDUM-based conformal class-wise weighted averaging, which suppresses prediction variance and enhances robustness.

WCDForest also demonstrates strong robustness, its AUROC gaps are always smaller than other models except CDForest (illustrated in Figs. 2 and 3). This is because that WCDForest uses the validation accuracies of random forests as their integration weights in the ensemble, thereby mitigating performance instability induced by noise corruption. CDForest exhibits superior robustness over WCDForest. The fundamental cause is that CDForest utilizes CDUM-based techniques, including the conformal class-wise output selection (see Section 5.2.2) and the conformal class-wise weighted averaging (see Section 5.3), while WCDForest depends solely on validation accuracy for its weighted ensemble approach.

Figures 2 and 3 indicate that neural network-based classification models, such as MLP-PLR, ResNetTab, and FT-Transformer, are more sensitive to noise corruption than forest-based classification models, such as XGBoost, gsForest, CDForest, and WCDForest. The reduced robustness of neural network-based models is primarily attributed to their high structural complexity and the propagation of uncertainty during the layer-by-layer learning process of network weights. Although regularization techniques, such as dropout (Baldi and Sadowski 2012), weight decay (Leung et al. 2010), and batch normalization (Chen et al. 2021), can be employed to improve robustness, their effectiveness is highly dependent on the specific problem and dataset. In contrast, forest-based classification models benefit from the principles of ensemble learning, such as mitigating overfitting to noisy data, reducing errors through model diversity, and controlling variance, which contribute to their superior robustness.

6.2.3 Reliability

The reliability of classification models is closely related to the uncertainty in their predictions. To enable the

Table 5 AUROC values of seven classification models on eight tabular datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	AUROC						
	XGBoost	gcForest	WCDForest	MLP-PLR	ResNetTab	FT-Transformer	CDForest
HIGGS	0.859	0.863	0.867	0.857	0.858	0.885	0.898
Adult	0.928	0.911	0.927	0.905	0.908	0.916	0.935
Homesite Insurance	0.96	0.969	0.975	0.962	0.947	0.962	0.981
Ecom Offers	0.576	0.603	0.612	0.602	0.599	0.577	0.630
Home Credit Default	0.820	0.862	0.887	0.857	0.849	0.857	0.889
Jannis	0.867	0.709	0.759	0.713	0.705	0.890	0.778
sEMG	0.712	0.735	0.740	0.539	0.582	0.691	0.751
Coverttype	0.981	0.866	0.927	0.762	0.825	0.978	0.963
Average	0.838	0.815	0.837	0.775	0.784	0.845	0.853

Table 6 Precision values of seven classification models on eight tabular datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	Precision						
	XGBoost	gcForest	WCDForest	MLP-PLR	ResNetTab	FT-Transformer	CDForest
HIGGS	0.765	0.768	0.772	0.763	0.764	0.788	0.799
Adult	0.854	0.838	0.853	0.833	0.835	0.843	0.860
Homesite Insurance	0.883	0.891	0.897	0.885	0.871	0.885	0.903
Ecom Offers	0.490	0.513	0.520	0.512	0.509	0.490	0.535
Home Credit Default	0.730	0.767	0.789	0.763	0.756	0.763	0.791
Jannis	0.772	0.631	0.676	0.635	0.627	0.792	0.692
sEMG	0.634	0.654	0.659	0.458	0.495	0.587	0.668
Coverttype	0.903	0.771	0.853	0.902	0.734	0.900	0.886
Average	0.754	0.729	0.752	0.719	0.699	0.756	0.767

Table 7 F1-Score values of seven classification models on eight tabular datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	F1-Score						
	XGBoost	gcForest	WCDForest	MLP-PLR	ResNetTab	FT-Transformer	CDForest
HIGGS	0.762	0.766	0.769	0.760	0.761	0.785	0.797
Adult	0.844	0.829	0.843	0.824	0.826	0.833	0.851
Homesite Insurance	0.873	0.881	0.887	0.875	0.861	0.875	0.893
Ecom Offers	0.498	0.522	0.529	0.521	0.518	0.499	0.544
Home Credit Default	0.728	0.765	0.787	0.760	0.753	0.760	0.789
Jannis	0.769	0.629	0.674	0.633	0.625	0.790	0.690
sEMG	0.632	0.652	0.657	0.466	0.503	0.597	0.666
Coverttype	0.893	0.768	0.843	0.890	0.732	0.890	0.876
Average	0.750	0.727	0.749	0.716	0.697	0.754	0.763

classification models used in the experiments (except for the proposed CDForest) to quantify uncertainty, conformal prediction with the conformity score defined by Eq. (3) is applied to post-process the predictions, providing probabilistic guarantees for the prediction sets. Then,

the calibration plot (see Section 6.1.3) is used to evaluate the reliability of classification models across tabular and image datasets. The closer the calibration curve approximates the diagonal line, the more reliable the model. Conversely, deviations from the diagonal line (e.g.,

Table 8 AUROC values of six classification models on two image datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	AUROC					
	ResNet	gcForest	WCDForest	ViT-L/16	RDNet-L	CDForest
Fashion-MNIST	0.969	0.950	0.965	0.998	0.996	0.980
CIFAR-10	0.930	0.926	0.928	0.996	0.995	0.970
Average	0.949	0.938	0.947	0.997	0.996	0.975

Table 9 F1-Score values of six classification models on two image datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	F1-Score (%)					
	ResNet	gcForest	WCDForest	ViT-L/16	RDNet-L	CDForest
Fashion-MNIST	94.0	93.4	94.2	98.6	95.6	95.1
CIFAR-10	95.3	94.2	94.3	99.4	99.3	97.7
Average	94.67	93.8	94.3	98.9	97.5	96.4

Table 10 Comparison of training and inference time (s) between CDForest and gcForest on CIFAR-10 and Fashion-MNIST datasets

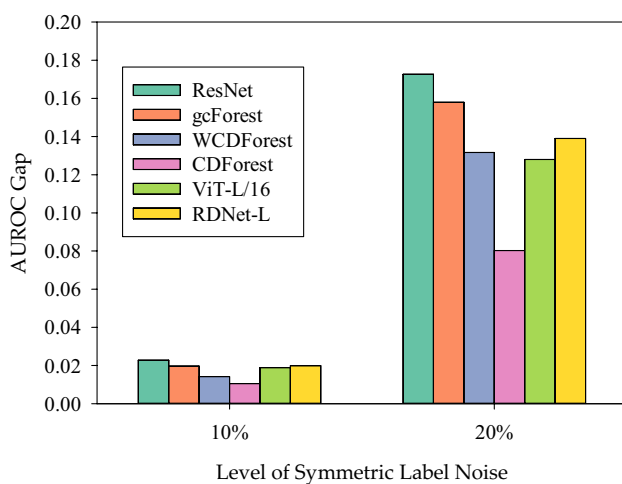
Models	CIFAR-10		Fashion-MNIST	
	Training time (s)	Inference time (s)	Training time (s)	Inference time (s)
gcForest	2005.4	74.1	1240.2	60.9
CDForest	2075.1	77.3	1296.7	62.6

overestimation or underestimation) indicate miscalibration, reflecting inconsistency between model predictions and empirical evidence (Linusson et al. 2017).

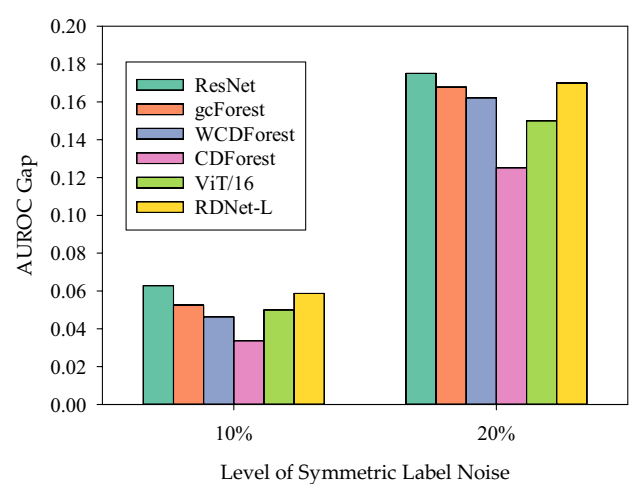
Figure 4a presents the calibration plots for nine classification models evaluated on eight tabular datasets, while Fig. 4b shows those for six classification models on two image datasets. In both figures, the x -axis represents the significance level ϵ , which corresponds to the proportion

of predictions allowed to fall outside the conformal prediction set; the y -axis denotes the average of empirical error rates for classification models, computed across the eight tabular datasets (Fig. 4a) or the two image datasets (Fig. 4b), providing an overall summary of model performance.

As shown in Fig. 4a and b, the calibration plot of the proposed CDForest aligns more closely with the diagonal line than those of other competitive models, demonstrating superior calibration and reliability. In contrast, the calibration plots of other models generally lie above the diagonal line, indicating higher empirical error rates compared to their expected rates. This reflects that other competitive models are typically overconfident. The superior reliability of CDForest stems from its use of ensemble conformal predictors with CDUM-based weighted aggregation, which reduces variance and enhances calibration. By contrast,

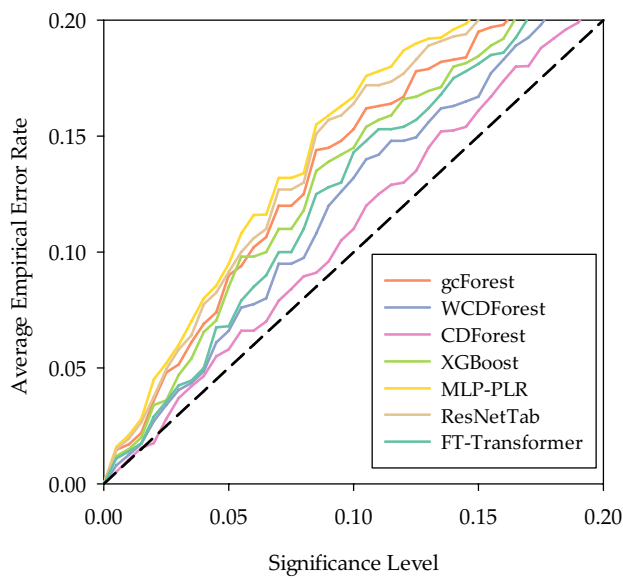


(a) AUROC Gap on Fashion-MNIST Dataset

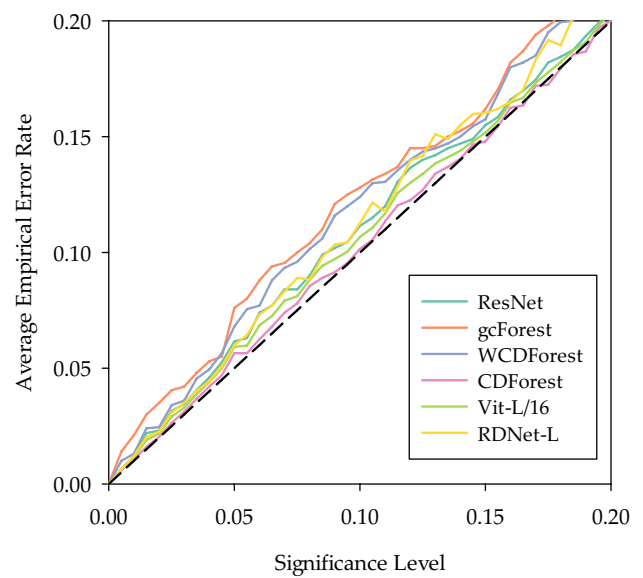


(b) AUROC Gap on CIFAR-10 dataset

Fig.3 Histograms of AUROC gaps for six classification models on two image datasets under two levels of symmetric label noise (10% and 20%). **a** Fashion-MNIST dataset; **b** CIFAR-10 dataset



(a) Calibration plots on tabular datasets



(b) Calibration plots on image datasets

Fig.4 Calibration plots for classification models on tabular and image datasets, showing the relationship between significance level and empirical error rates. **a** Calibration plots for seven classification models (gcForest, WCDForest, CDForest, XGBoost, MLP-PLR, ResNetTab, and FT-Transformer) on tabular datasets; **b** Calibration plots for six classification models (ResNet, gcForest, WCDForest, CDForest,

RDNet-L, and ViT-L/16) on image datasets. The x-axis represents the significance level, which corresponds to the proportion of predictions allowed to fall outside the conformal prediction set, and the y-axis indicates the average empirical error rates across tabular datasets (**a**) or image datasets (**b**)

other models rely on a single conformal predictor for uncertainty quantization, limiting their ability to address variance and resulting in less reliable predictions.

6.2.4 Ablation Study

Three types of CDUM-based modules are incorporated into different stages of the proposed CDForest model:

- The first type of CDUM-based module in each layer of CDForest uses normalized CDUM values to filter out high-uncertainty membership values.
- The second type of CDUM-based module in the final layer of CDForest performs conformal class-wise weighted averaging on the outputs of random forests to refine the ensemble predictions, utilizing the class-wise reliability.
- The third type of CDUM-based module is responsible for class-wise aggregation for ensemble conformal predictors, which is performed in parallel with the classification task.

Let DF-Filter denote the DF model with the first type of CDUM-based modules (i.e., the layer-wise uncertainty filtering module), and let DF-Filter-Average denote DF model with the first and the second types of CDUM-based modules (i.e., the layer-wise uncertainty filtering module

& conformal class-wise weighted averaging module). Thus, CDForest denotes DF model contains the three types of CDUM-based modules. Table 11 illustrates the comparison of classification performance among DF-Filter, DF-Filter-Average, and CDForest.

As shown in Table 11, the DF-Filter model, which applies only the first type of CDUM-based module to the DF, achieves a 2.86% increase in average AUROC (calculated as $((0.863 - 0.839)/0.839) \times 100\%$). When both the first and second types of CDUM-based modules are applied to DF, which yields the DF-Filter-Average model, the average AUROC improves by 4.65%. Therefore, the improvement attributable to the second CDUM-based module is approximately 1.79%. This indicates that both the normalized CDUM-based filtering module and the conformal class-wise weighted aggregation module contribute to classification performance improvement, the former demonstrates a more pronounced impact.

Utilizing normalized CDUM values to filter out high-uncertainty membership values yields a greater performance gain than the conformal class-wise weighted aggregation. This superiority primarily stems from the fact that uncertainty tends to propagate and amplify across DF layers. By performing layer-wise uncertainty filtering, the reliability of predictions at the output layer is better preserved. On the other hand, the contribution of the conformal class-wise weighted aggregation module is mainly through its calculated weights, which allow

Table 11 AUROC values of four classification models on 10 datasets (averaged over ten independent trials, highest values highlighted in bold)

Dataset	AUROC			
	gcForest	DF-Filter	DF-Filter-Average	CDForest
HIGGS	0.863	0.885	0.898	0.898
Adult	0.911	0.925	0.935	0.935
Homesite Insurance	0.969	0.976	0.981	0.981
Ecom Offers	0.603	0.619	0.630	0.630
Home Credit Default	0.862	0.878	0.889	0.889
Jannis	0.709	0.751	0.778	0.778
sEMG	0.735	0.745	0.751	0.751
Coverttype	0.866	0.927	0.963	0.963
Fashion-MNIST	0.950	0.969	0.980	0.980
CIFAR-10	0.926	0.953	0.970	0.970
Average	0.839	0.863	0.878	0.878

for the selection of predictions with the lowest uncertainty from each class, thereby enabling more precise ensemble integration.

Moreover, as shown in Table 11, DF-Filter-Average and CDForest achieve identical classification performance. This is because the third type of CDUM-based module is specifically designed to integrate ensemble uncertainty quantification results, and it operates in parallel with the classification task. For the classification task alone, both DF-Filter-Average and CDForest share the same structure and module configuration.

As discussed earlier (see Sections 6.1.3 and 6.2.3), this paper adopts calibration plots as the evaluation metric for assessing uncertainty quantification performance. Figure 5 shows the calibration plots of gcForest, DF-Filter, DF-Filter-Average, and CDForest across all datasets, including both tabular and image datasets. As shown in Fig. 5, the calibration plot of CDForest aligns more closely with the diagonal reference line compared to the other three models, which substantiates the effectiveness of the third type of CDUM-based module—the class-wise conformal prediction aggregation—in reducing uncertainty quantification bias.

7 Conclusion and future work

This paper presents a novel uncertainty quantization criterion based on conformal prediction, called the conformal dual uncertainty metric (CDUM), which leverages interactive relationships between different classes to enhance the exploration of conformal prediction outputs. CDUM provides a precise and explicit quantification of class-wise uncertainty. Subsequently, this paper proposes an uncertainty-aware deep forest model, named conformal deep

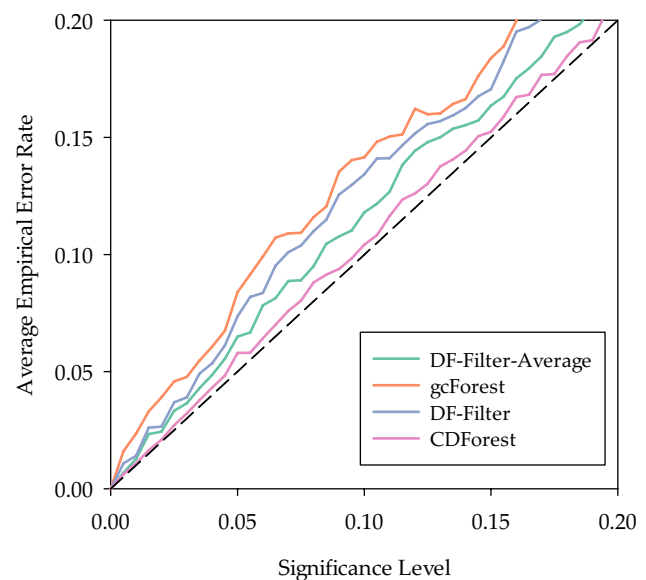


Fig. 5 Calibration plots for gcForest, DF-Filter, DF-Filter-Average, and CDForest models on 10 datasets (tabular and image datasets), showing the relationship between significance level and empirical error rates. The *x*-axis represents the significance level, which corresponds to the proportion of predictions allowed to fall outside the conformal prediction set, and the *y*-axis indicates the average empirical error rates across 10 datasets

forest (CDForest), which incorporates three CDUM-based modules into the deep forest (DF) architecture. The three CDUM-based modules provide the following capabilities: (i) layer-wise filtering for highly uncertain outputs with normalized CDUM values, which reduces noise corruption and suppresses uncertainty propagation; (ii) conformal class-wise weighted averaging for classification predictions, which enhances the robustness of the model; (iii) conformal class-wise aggregation tailored for conformal predictors, which improves prediction reliability. These CDUM-based modules collectively address uncertainty, robustness, and reliability, contributing to the overall performance of the model. CDForest combines the strengths of both CDUM-based modules and the DF architecture, the latter of which provides a hierarchical structure for efficient feature extraction and decision-making. This combination enables CDForest to explicitly quantify class-wise uncertainty layer-by-layer and leverage the quantified uncertainty to improve performance, robustness, and reliability. Extensive experiments on tabular and image datasets show that CDForest is a highly competitive and robust solution for classification tasks requiring effective uncertainty quantification and reliable decision-making, with superior performance in scenarios where precisely handling and quantifying class-wise uncertainty are critical.

In the future, efficient techniques will be investigated to accelerate the training process of CDForest. Furthermore,

the layer-wise uncertainty quantization based on CDUM in CDForest provides a foundation for systematically analyzing the propagation and evolution of uncertainty within DF and its variants. This approach not only offers theoretical

insights into understanding uncertainty dynamics but also contributes to the development of more robust and interpretable models.

Appendix 1

In Appendix 1, algorithms for the proposed conformal deep forest (CDForest) are presented in Tables 12, 13, 14, 15, 16, 17, 18, 19 and 20, which summarize the key algorithmic steps.

Table 12 The algorithm for computing conformity scores of instances in the calibration set

ConformityScoresCali ($Z_C, \mathcal{Y}, \mathcal{F}$)
Input:
Z_C – calibration instance set
$\mathcal{Y}$ – class label set $\{1, \dots, M\}$
$\mathcal{F}$ – a trained random forest
Output:
α – array of conformity scores for instances in Z
// Compute conformity scores based on Eq.(3)
1: $\alpha \leftarrow []$ // initialized as an empty array
2: for instance $(x_i, y_i) \in Z_C$ do
3: $\alpha_i \leftarrow (1 + \mathcal{F}^{y_i}(x_i) - \max_{y' \in \mathcal{Y}, y' \neq y_i} \{\mathcal{F}^{y'}(x_i)\})/2$ // Eq.(3)
4: $\alpha.append(\alpha_i)$
5: end for
6: return α // α size of $ Z $

Table 13 The algorithm for computing conformity scores of an object corresponding to all potential class labels

ConformityScoresObj ($x, \mathcal{Y}, \mathcal{F}$)
Input:
x – an object whose class label is unknown and needs to be predicted
$\mathcal{Y}$ – class label set $\{1, \dots, M\}$
$\mathcal{F}$ – a trained random forest
Output:
α – array of conformity scores for x and its potential classes in $\mathcal{Y}$
// Compute conformity scores based on Eq.(3)
1: $\alpha \leftarrow []$
2: for $y \in \mathcal{Y}$ do
3: $\alpha^y \leftarrow (1 + \mathcal{F}^y(x) - \max_{y' \in \mathcal{Y}, y' \neq y} \{\mathcal{F}^{y'}(x)\})/2$
4: $\alpha.append(\alpha^y)$
5: end for
6: return α // α size of $ \mathcal{Y} $

Table 14 The algorithm for computing p -values and CICIP prediction set**PvaluesCICIPset**($\mathbf{x}$, Z_C , C , $\mathcal{Y}$, ϵ , τ , $\mathcal{F}$)**Input:**

- $\mathbf{x}$ – an object whose class label is unknown and needs to be predicted
- Z_C – calibration set
- C – instance index set of Z_C
- $\mathcal{Y}$ – class label set $\{1, \dots, M\}$
- ϵ – the significance level $\epsilon \in (0, 1)$
- τ – coefficient, $\tau \in [0, 1]$
- $\mathcal{F}$ – a trained random forest

Output:

- $\mathcal{P}$ – array of p -values
- Γ – prediction set

```

01:  $\mathcal{P} \leftarrow [], \Gamma \leftarrow \{\}$ 
02:  $\alpha \leftarrow \text{ConformityScoresCali}(Z_C, \mathcal{Y}, \mathcal{F})$  //  $\alpha$  size of  $|Z_C|$ 
03:  $\alpha' \leftarrow \text{ConformityScoresObj}(\mathbf{x}, \mathcal{Y}, \mathcal{F})$  //  $\alpha'$  size of  $|\mathcal{Y}|$ 
04: for  $y \in \mathcal{Y}$  do
05:    $C^y \leftarrow \{i \in C | y_i = y\}$ 
06:    $\alpha^y \leftarrow$  the  $y$ -th element of  $\alpha'$ 
07:    $\Omega \leftarrow [], \Omega' \leftarrow []$ 
08:   for  $i \in C^y$  do
09:      $\alpha_i \leftarrow$  the  $i$ -th element of  $\alpha$ 
10:     if  $\alpha_i < \alpha^y$  then
11:        $\Omega.append(i)$ 
12:     end if
13:     if  $\alpha_i = \alpha^y$  then
14:        $\Omega'.append(i)$ 
15:     end if
16:   end for
17:    $p(\mathbf{x}, y) \leftarrow (|\Omega| + \tau|\Omega'|) / (|C^y| + 1)$ 
18:    $\mathcal{P}.append(p(\mathbf{x}, y))$ 
19:   if  $p(\mathbf{x}, y) > \epsilon$  then
20:      $\Gamma.append(y)$ 
21:   end if
22: end for
23: return  $\mathcal{P}$  and  $\Gamma$ 

```

Table 15 The algorithm for computing CDUM

CDUM ($\mathcal{Z}_C, \mathcal{Z}_V, C, V, \mathcal{Y}, \lambda, \epsilon, \tau, \mathcal{F}$)
Input:
$\mathcal{Z}_C$ – calibration set
$\mathcal{Z}_V$ – validation set
C – instance index set of $\mathcal{Z}_C$
V – instance index set of $\mathcal{Z}_V$
$\mathcal{Y}$ – class label set $\{1, \dots, M\}$
λ – coefficient $\lambda \in (0,1)$
ϵ – the significance level $\epsilon \in (0,1)$
τ – coefficient, $\tau \in [0,1]$
$\mathcal{F}$ – a trained random forest
Output:
$\mathbf{u}$ – array of CDUM values for all classes in $\mathcal{Y}$ achieved based on $\mathcal{F}$
01: $\mathbf{u} \leftarrow []$, $\Gamma\text{array} \leftarrow []$, $\text{Rarray} \leftarrow []$
02: for $v \in V$ do
03: $(\mathbf{x}_v, y_v) \leftarrow$ the v -th instance in $\mathcal{Z}_C$
04: $\mathcal{P}, \Gamma \leftarrow \text{PvaluesCICPset}(\mathbf{x}_v, \mathcal{Z}_C, C, \mathcal{Y}, \epsilon, \tau, \mathcal{F})$
05: $\Gamma\text{array.append}(\Gamma)$
06: if $ \Gamma > 0$ then
07: $\varrho(\mathbf{x}_v, y_v) \leftarrow \Gamma \setminus \{y_v\} $
08: else // $ \Gamma = 0$
09: $\varrho(\mathbf{x}_v, y_v) \leftarrow 1$
10: end if
11: $\text{Rarray.append}(\varrho(\mathbf{x}_v, y_v))$
12: end for
13: for every $y \in \mathcal{Y}$ do
14: $V^y \leftarrow \{v \in V y_v = y\}$
15: $V^{-y} \leftarrow \{v \in V y_v \neq y\}$
16: $\varphi(y) \leftarrow \frac{1}{ V^y } \sum_{v \in V^y} \text{Rarray}(v)$ // passive uncertainty indicator
17: $\psi(y) \leftarrow \frac{1}{ V^{-y} } \sum_{v \in V^{-y}} \Gamma\text{array}(v) \cap \{y\} $ // active uncertainty indicator
18: $\mathcal{U}(y) \leftarrow \lambda\varphi(y) + (1 - \lambda)\psi(y)$
19: $\mathbf{u.append}(\mathcal{U}(y))$
20: end for
21: return $\mathbf{u}$ // size of $ \mathcal{Y} $

Table 16 The algorithm for training CDForest

TrainCDForest ($\{\mathcal{Z}_{Tn}\}_{n=1}^S, \{\mathcal{Z}_{Cn}\}_{n=1}^S, \{\mathcal{Z}_{Vn}\}_{n=1}^S, C, V, L, \mathcal{Y}, \tau, \lambda, \epsilon$)	
Input:	
$\{\mathcal{Z}_{Tn}\}_{n=1}^S$ – S instance sets generated from proper training set $\mathcal{Z}_T$ by multi-grained scanning procedure $\{\mathcal{Z}_{Cn}\}_{n=1}^S$ – S instance sets generated from calibration set $\mathcal{Z}_C$ by multi-grained scanning procedure $\{\mathcal{Z}_{Vn}\}_{n=1}^S$ – S instance sets generated from validation set $\mathcal{Z}_V$ by multi-grained scanning procedure C – instance index set of calibration set V – instance index set of validation set L – the number of layers in CDForest $\mathcal{Y}$ – class label set $\{1, \dots, M\}$ τ – coefficient, $\tau \in [0, 1]$ λ – coefficient $\lambda \in (0, 1)$ ϵ – the significance level $\epsilon \in (0, 1)$	
Output:	
$\mathcal{F}_{all}$ – array of trained random forests in all layers of CDForest $\mathbf{t}_{all}$ – array of selected indices of trained random forests for every class in $\mathcal{Y}$ in all layers of CDForest $\mathbf{u}_L$ – array of CDUM values of trained random forests for every class in $\mathcal{Y}$ in the L -th layer $\tilde{\mathcal{Z}}_{L,C}$ – the extended calibration set for the L -th layer of CDForest	
01:	$\mathcal{F}_{all} \leftarrow [], \mathbf{t}_{all} \leftarrow [], \mathbf{u}_L \leftarrow [], \tilde{\mathcal{Z}}_T \leftarrow [], \tilde{\mathcal{Z}}_C \leftarrow [], \tilde{\mathcal{Z}}_V \leftarrow []$
02:	for $\ell \in \{1, \dots, L\}$ do
03:	$\mathcal{F}_\ell \leftarrow [], \mathbf{t}_\ell \leftarrow [], s(\ell) \leftarrow 1 + ((\ell - 1) \bmod S)$
04:	if $\ell = 1$ then
05:	$\tilde{\mathcal{Z}}_T \leftarrow \mathcal{Z}_{Ts(\ell)}, \tilde{\mathcal{Z}}_C \leftarrow \mathcal{Z}_{Cs(\ell)}, \tilde{\mathcal{Z}}_V \leftarrow \mathcal{Z}_{Vs(\ell)}$
06:	end if
07:	for $t \in \{1, \dots, T\}$ do // Training random forests in the ℓ -th layer
08:	Train the t -th random forest in the ℓ -th layer of CDForest, $\mathcal{F}_{\ell,t}$, with $\tilde{\mathcal{Z}}_T$
09:	$\mathcal{F}_\ell.append(\mathcal{F}_{\ell,t})$
10:	$\mathbf{u}_{\ell,t} = [\mathbf{u}_{\ell,t}(1), \dots, \mathbf{u}_{\ell,t}(M)] \leftarrow \text{CDUM}(\tilde{\mathcal{Z}}_C, \tilde{\mathcal{Z}}_V, C, V, \mathcal{Y}, \lambda, \epsilon, \tau, \mathcal{F}_{\ell,t})$
11:	if $\ell = L$ then
12:	$\mathbf{u}_L.append(\mathbf{u}_{\ell,t})$
13:	end if
14:	end for
15:	for $y \in \mathcal{Y}$ do
16:	for $t \in \{1, \dots, T\}$ do
17:	$u_{\ell,t}^y \leftarrow \frac{u_{\ell,t}(y)}{\sum_{t' \in \{1, \dots, T\}} u_{\ell,t'}(y)}$ // $u_{\ell,t}(y) \in \mathbf{u}_{\ell,t}, t \in \{1, \dots, T\}$, see Eq.(9)
18:	end for
19:	$\hat{u}_\ell^y \leftarrow$ the $[(T + 1)\vartheta]/T$ -th empirical quantile of $u_{\ell,1}^y, \dots, u_{\ell,T}^y$
20:	$\mathbf{t}_\ell^y \leftarrow \{t \in \{1, \dots, T\} u_{\ell,t}^y \leq \hat{u}_\ell^y\}$
21:	$\mathbf{t}_\ell.append(\mathbf{t}_\ell^y)$
22:	end for
23:	$\mathbf{t}_{all}.append(\mathbf{t}_\ell)$
24:	$\mathcal{F}_{all}.append(\mathcal{F}_\ell)$
25:	$\tilde{\mathcal{Z}}_T \leftarrow \text{ExtendSet}(\tilde{\mathcal{Z}}_T, \mathcal{Z}_T, \mathcal{F}_\ell, \mathbf{t}_\ell)$ // extend the proper training set for the next layer
26:	$\tilde{\mathcal{Z}}_C \leftarrow \text{ExtendSet}(\tilde{\mathcal{Z}}_C, \mathcal{F}_\ell, \mathbf{t}_\ell)$ // extend the calibration set for the next layer
27:	$\tilde{\mathcal{Z}}_V \leftarrow \text{ExtendSet}(\tilde{\mathcal{Z}}_V, \mathcal{F}_\ell, \mathbf{t}_\ell)$ // extend the validation set for the next layer
28:	end for
29:	$\tilde{\mathcal{Z}}_{L,C} \leftarrow \tilde{\mathcal{Z}}_C$
30:	return $\mathcal{F}_{all}, \mathbf{t}_{all}, \mathbf{u}_L$ and $\tilde{\mathcal{Z}}_{L,C}$

Table 17 The algorithm for extending instance sets

ExtendSet ($\tilde{Z}, Z, \mathcal{Y}, \mathcal{F}_{layer}, \mathbf{t}_{layer}$)
Input:
$\tilde{Z}_{pre}$ – extended instance set generated in the previous layer
Z – instance set input to the current layer
$\mathcal{Y}$ – class label set $\{1, \dots, M\}$
$\mathcal{F}_{layer}$ – array of trained random forests in the current layer
$\mathbf{t}_{layer}$ – array of index sets of selected random forests in the current layer for every class in $\mathcal{Y}$
Output:
$\tilde{Z}_{next}$ – extended instance set for the next layer
01: $\tilde{Z}_{next} \leftarrow \{\}$
01: for $j \in \{1, \dots, Z \}$ do
02: $(x_j, y_j) \leftarrow$ the object and the class label of the j -th instance of Z
03: $(\tilde{x}_j, y_j) \leftarrow$ the object and the class label of the j -th instance of $\tilde{Z}$
04: $\tilde{x}'_j \leftarrow \text{ExtendObject}(\tilde{x}_j, x_j, \mathcal{Y}, \mathcal{F}_{layer}, \mathbf{t}_{layer})$ // extended object for the next layer
05: $\tilde{Z}_{nextlayer}.append((\tilde{x}'_j, y_j))$
06: end for
07: return $\tilde{Z}_{nextlayer}$

Table 18 The algorithm for extending object

ExtendObject ($\tilde{x}, x, \mathcal{Y}, \mathcal{F}_{layer}, \mathbf{t}_{layer}$)
Input:
$\tilde{x}$ – the extended object generated in the previous layer
x – the object input to the current layer
$\mathcal{Y}$ – class label set $\{1, \dots, M\}$
$\mathcal{F}_{layer}$ – array of trained random forests in the current layer
$\mathbf{t}_{layer}$ – array of index sets of selected random forests in the current layer for every class in $\mathcal{Y}$
Output:
$\tilde{x}_{next}$ – extended object for the next layer
01: $\tilde{x}' \leftarrow [\], \mathbf{o} \leftarrow [\]$
02: for $y \in \mathcal{Y}$ do
03: $\mathbf{t}^y \leftarrow \mathbf{t}_{layer}[y]$
04: $\mathbf{f}^y \leftarrow [\]$
05: for $t \in \mathbf{t}^y$ do
06: $\mathcal{F}_t \leftarrow \mathcal{F}_{layer}[t]$
07: $\mathbf{f}^y \leftarrow [\mathbf{f}^y, \mathcal{F}_t^y(\tilde{x})]$
08: end for
09: $\mathbf{o} \leftarrow [\mathbf{o}, \mathbf{f}^y]$
09: end for
10: $\tilde{x}_{next} \leftarrow [x, \mathbf{o}]$
11: return $\tilde{x}_{next}$

Table 19 The algorithm of inference process of CDForest

Inference ($\mathbf{x}$, C , $\mathcal{Y}$, τ , ϵ , $\mathcal{F}_{all}$, $\mathbf{t}_{all}$, $\mathbf{u}_L$)	
Input:	
$\mathbf{x}$	test object
C	instance index set of calibration set
$\mathcal{Y}$	class label set $\{1, \dots, M\}$
τ	coefficient, $\tau \in [0, 1]$
ϵ	the significance level $\epsilon \in (0, 1)$
$\mathcal{F}_{all}$	array of trained random forests in all layers of CDForest
$\mathbf{t}_{all}$	array of selected indices of trained random forests for every class in $\mathcal{Y}$ in all layers of CDForest
$\mathbf{u}_L$	array of CDUM values of trained random forests for every class in $\mathcal{Y}$ in the L -th layer
Output:	
$\hat{y}$	class label for test object $\mathbf{x}$ predicted by CDForest
$\mathbf{w}_L$	the set of class-wise weights for the random forests in the L -th layer of CDForest
$\tilde{\mathbf{x}}_L$	the extended object generated in the L -th layer of CDForest for test $\mathbf{x}$
// inference	
00:	$\{\mathbf{x}'_1, \dots, \mathbf{x}'_S\} \leftarrow$ implement the multi-grained scanning procedure with S sliding windows on $\mathbf{x}$
01:	$\mathbf{w}_L \leftarrow []$
02:	for $\ell \in \{1, \dots, L-1\}$ do // obtain $\tilde{\mathbf{x}}_L$ for $\mathbf{x}$ through the layer-wise information propagation
03:	$s(\ell) \leftarrow 1 + ((\ell - 1) \bmod S)$
04:	if $\ell = 1$ then
05:	$\tilde{\mathbf{x}}_\ell \leftarrow \mathbf{x}'_1$
06:	end if
07:	$\mathcal{F}_\ell \leftarrow \mathcal{F}_{all}[\ell]$ // trained random forests in the ℓ -th layer
08:	$\mathbf{t}_\ell \leftarrow \mathbf{t}_{all}[\ell]$
09:	$\tilde{\mathbf{x}}_{\ell+1} \leftarrow \text{ExtendObject}(\mathbf{x}'_{s(\ell)}, \tilde{\mathbf{x}}_\ell, \mathcal{Y}, \mathcal{F}_\ell, \mathbf{t}_\ell)$
10:	end for
11:	for $y \in \mathcal{Y}$ do
12:	$\mathbf{w}_L^y \leftarrow []$
13:	for $t \in \{1, \dots, T\}$ do
14:	$w_{L,t}^y \leftarrow \frac{1 - u_{L,t}(y)}{1 + u_{L,t}(y)}$ // $u_{L,t}(y) \in \mathbf{u}_L$
15:	$\mathbf{w}_L^y.append(w_{L,t}^y)$
16:	end for
17:	$\mathbf{w}_L.append(\mathbf{w}_L^y)$
18:	end for
19:	for $y \in \mathcal{Y}$ do
20:	$\tilde{f}_L^y(\tilde{\mathbf{x}}_L) \leftarrow \sum_{t \in \{1, \dots, T\}} w_{L,t}^y \mathcal{F}_{L,t}^y(\tilde{\mathbf{x}}_L)$ // $w_{L,t}^y \in \mathbf{w}_L$
21:	end for
22:	for $y \in \mathcal{Y}$ do
23:	$\kappa^y(\tilde{\mathbf{x}}_L) \leftarrow \frac{\exp(\tilde{f}_L^y(\tilde{\mathbf{x}}_L))}{\sum_{y'=1}^M \exp(\tilde{f}_L^{y'}(\tilde{\mathbf{x}}_L))}$
24:	end for
25:	$\hat{y} \leftarrow \text{argmax}_{y \in \{1, \dots, M\}} \kappa^y(\tilde{\mathbf{x}}_L)$
26:	return $\hat{y}$, $\mathbf{w}_L$ and $\tilde{\mathbf{x}}_L$

Table 20 The algorithm of aggregated conformal prediction for CDForest

AggCP ($\tilde{\mathbf{x}}_L, \tilde{\mathcal{Z}}_{L,C}, C, \mathcal{Y}, \tau, \epsilon, \mathcal{F}_L, \mathbf{t}_L, \mathbf{w}_L$)	
Input:	
$\tilde{\mathbf{x}}_L$	— an extended object input to the L -th layer
$\tilde{\mathcal{Z}}_{L,C}$	— the extended calibration set for the L -th layer of CDForest
C	— instance index set of calibration set
$\mathcal{Y}$	— class label set $\{1, \dots, M\}$
τ	— coefficient, $\tau \in [0,1]$
ϵ	— the significance level $\epsilon \in (0,1)$
$\mathcal{F}_L$	— array of trained random forests in all layers of CDForest
$\mathbf{t}_{all}$	— array of selected indices of trained random forests for every class in $\mathcal{Y}$ in all layers of CDForest
$\mathbf{w}_L$	— array of CDUM values of trained random forests for every class in $\mathcal{Y}$ in the L -th layer
Output:	
$\hat{\Gamma}$	— the aggregated conformal prediction set for $\tilde{\mathbf{x}}_L$ achieved by CDForest
<pre> 01: $\hat{\Gamma} \leftarrow \{\}, \{w_{L,t}^y\}_{t \in \{1, \dots, T\}, y \in \mathcal{Y}} \leftarrow \mathbf{w}_L, \{\mathcal{F}_{L,t}\}_{t \in \{1, \dots, T\}} \leftarrow \mathcal{F}_L$ 02: for $t \in \{1, \dots, T\}$ do // $T = \mathcal{F}_L$ 03: $\mathcal{P}_t, \Gamma_t \leftarrow \text{PvaluesCICPset}(\tilde{\mathbf{x}}_L, \tilde{\mathcal{Z}}_{L,C}, C, \mathcal{Y}, \epsilon, \tau, \mathcal{F}_{L,t})$ // $\mathcal{P}_t = [p_t(\tilde{\mathbf{x}}_L, 1), \dots, p_t(\tilde{\mathbf{x}}_L, M)]$ 04: end for 05: for $y \in \mathcal{Y}$ do 06: $\tilde{p}_L(\tilde{\mathbf{x}}_L, y) \leftarrow \frac{1}{\sum_{t \in \{1, \dots, T\}} w_{L,t}^y} \sum_{t \in \{1, \dots, T\}} w_{L,t}^y p_t(\tilde{\mathbf{x}}_L, y)$ // $p_t(\tilde{\mathbf{x}}_L, y) \in \mathcal{P}_t, w_{L,t}^y \in \mathbf{w}_L$ 07: if $\tilde{p}_L(\tilde{\mathbf{x}}_L, y) > \epsilon$ then 08: $\hat{\Gamma}.\text{append}(y)$ 09: end if 10: end for 11: return $\hat{\Gamma}$ </pre>	

Appendix 2

The hyperparameter ranges or fixed values for XGBoost, MLP-PLR, ResNetTab, and FT-Transformer are provided in Appendix Tables 21, 22, 23, 24, and 25, detailing the configurations for these baseline models.

Table 21 The searching ranges of hyperparameters of XGBoost for Optuna-driven tuning

Hyperparameter	Value & Distribution
booster	'gbtree'
early_stopping_rounds	30
n-estimators	Uniform [500, 2000]
subsample	Uniform [0.5, 1]
eta	LogUniform [1e-5, 1]
colsample_bylevel	Uniform [0.5, 1]
min_child_weight	LogUniform [1e-8, 1e-5]
max_depth	LogUniform [3, 10]
colsample_bytree	Uniform [0.5, 1]
lambda	LogUniform [1e-8, 1e2]
gamma	LogUniform [1e-8, 1e2]
alpha	LogUniform [1e-8, 1e2]

Table 22 The searching ranges of hyperparameters of MLP-PLR for Optuna-driven tuning

Hyperparameter	Value & Distribution
batch size	Uniform {256, 512, 1024, 2048}
number of hidden layers	Uniform {1, ..., 10}
dim of each hidden layer	Uniform {20, ..., 1000}
loss function	cross entropy
optimizer	{'adamw', 'SGD'}
weight_decay	Uniform [0, 1E-3]
dropout	Uniform [0, 0.5]
lr (learning rate)	Uniform [1E-5, 1E-3]
activation	'relu'
patience	16

Table 23 The searching ranges of hyperparameters of ResNetTab for Optuna-driven tuning

Hyperparameter	Value & Distribution
batch size	Uniform{256, 512, 1024, 2048}
d (input dimension of residual block)	[200,1000]
d_hidden_factor	Uniform [1, 5]
hidden_dropout	Uniform[0.1, 0.5]
residual_dropout	Uniform[0, 0.5]
n_layers (number of hidden layers)	Uniform {1, ..., 20}
loss function	cross entropy
optimizer	{'adamw','SGD'}
lr (learning rate)	Uniform[1E-5, 5E-3]
weight_decay	Uniform[0,1E-5]
activation	'relu'
patience	16

Table 24 The searching ranges of hyperparameters of FT-Transformer for Optuna-driven tuning

Hyperparameter	Value & Distribution
Batch_size	Uniform{256, 512, 1024, 2048}
activation	'reglu'
attention_dropout	Uniform[0, 0.5]
d_ffn_factor	Uniform(Sheng et al. 2024; Ma et al. 2024)
d_token	Uniform{100, ..., 800}
ffn_dropout	Uniform[0, 0.5]
n_heads	8
n_layers (number of hidden layers)	Uniform{1, ..., 10}
loss function	cross entropy
optimizer	{'adamw','SGD'}
lr (learning rate)	Uniform[1E-5, 1E-3]
weight_decay	Uniform[1E-6, 1E-3]
patience	16

Table 25 Best Epoch and AUROC of the Top-1 Model on Each Benchmark Dataset

Dataset	Top-1 Model	Best Epoch (or stopping rounds)	Validation AUROC	Test AUROC
HIGGS	CDForest	12	0.911	0.898
Adult	CDForest	6	0.946	0.935
Homesite Insurance	CDForest	8	0.985	0.981
Ecom Offers	CDForest	8	0.643	0.630
Home Credit Default	CDForest	6	0.891	0.889
Jannis	XGBoost	22	0.874	0.867
sEMG	CDForest	6	0.766	0.751
Covertime	XGBoost	36	0.983	0.981
Fashion-MNIST	ViT-L/16	20	0.978	0.977
CIFAR-10	ViT-L/16	8	0.998	0.998

Boosting Rounds: the parameter in XGBoost model. It should clarify that XGBoost does not follow the epoch-based training scheme of deep neural networks. Instead, it builds an ensemble of decision trees incrementally, with each tree added sequentially. The number of stopping rounds is controlled by the parameter `early_stopping_rounds`

The AUROC- ϵ plot and the AUROC- λ plot by CDForest on HIGGS dataset are shown in Appendix Figs. 6 and 7, respectively

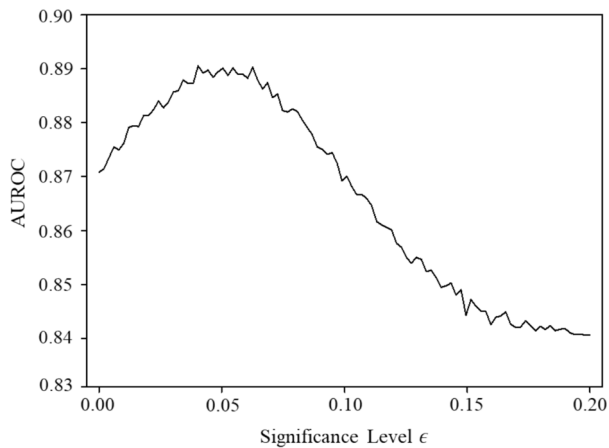


Fig. 6 AUROC- ϵ plot by CDForest on HIGGS dataset

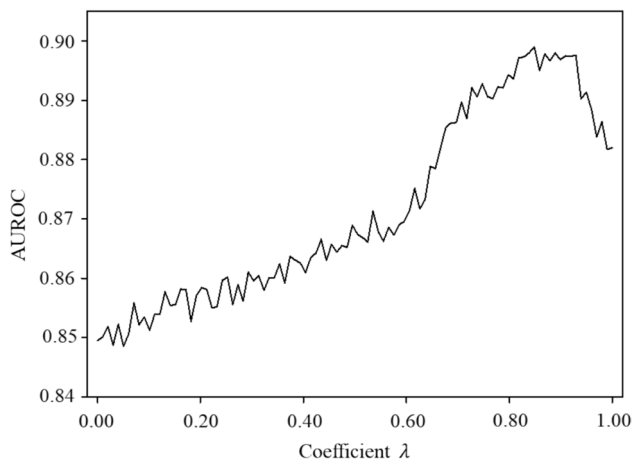


Fig. 7 AUROC- λ plot by CDForest on HIGGS dataset

Appendix 3

The abbreviations used in this paper are summarized as follows:

Abbreviation	Full Name
CDUM	Conformal dual uncertainty metric
DF	Deep forest
CDForest	Conformal Deep Forest
gcForest	Multi-grained cascade forest
WCDForest	Weighted cascade deep forest
ICP	Inductive conformal predictor
CICP	Conditional inductive conformal prediction
XGBoost	Extreme Gradient Boosting
MLP	Multilayer perceptron

Abbreviation	Full Name
MLP-PLR	Multilayer perceptron with embedding function ReLU $\circ$ Linear
ResNet	Residual Network
ResNetTab	ResNet-like tabular classifier
FT-Transformer	Feature Tokenizer Transformer
AUROC	Area Under Receiver Operating Characteristic Curve
SGD	Stochastic Gradient Descent

The notations used in equations of this paper are summarized as follows:

Abbreviation	Full Name
$\mathcal{Z}$	Training set
$\mathbf{z}_i = (\mathbf{x}_i, y_i)$	The i -th instance in $\mathcal{Z}$, $\mathbf{x}_i$ object (vector), y_i class label
$\mathcal{Z}_T$	The proper training set
$\mathcal{Z}_C$	The calibration set
$\mathcal{Z}_V$	The validation set
$\mathcal{A}$	conformity score function
$p(\mathbf{x}, y)$	The class-wise p -value for the test object $\mathbf{x}$ and its potential class y
C^y	The subset of C containing the indices of calibration instances belonging to class y
V^y	The subset of V that contains the indices of instances belonging to class y
ϵ	The significance level
$\Gamma^\epsilon(\mathbf{x})$	The conformal prediction set provided by CICP with respect to the test object $\mathbf{x}, \epsilon \in (0, 1)$
$\mathcal{F}$	Random forest
$\mathcal{F}_{\ell,t}$	The t -th trained random forest in the ℓ -th layer of CDForest
$\varphi(y)$	The <i>passive uncertainty indicator</i> for class y
$\psi(y)$	The <i>active uncertainty indicator</i> for class y
$\mathcal{U}(y)$	The CDUM for class y
$u_{\ell,t}^y$	The normalized CDUM values for class y across random forests $\mathcal{F}_{\ell,t}, t \in \{1, \dots, T\}$, where T denotes the number of random forests in the ℓ -th layer of CDForest
$f(\ell)$	The relationship between the layer index ℓ and the sliding window index $\setminus$
$w_{L,t}^y$	The weight of the t -th random forest in the L -th layer of CDForest for class $y \in \mathcal{Y}$

Funding This work was funded by the General Program of Education Department of Liaoning Province, P.R.China (Grant No. 524053218).

Data availability All data used in this paper are publicly available.

Code availability After the publication of this paper, the source code of CDForest can be requested from the authors via email, and the authors would appreciate it if the intended use could be briefly described.

Declarations

Conflict of interest There is no conflict of interest among the authors.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

References

- Abdar M, Pourpanah F, Hussain S et al (2021) A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* 76:243–297
- Angelopoulos AN, Bates S (2023) Conformal prediction: a gentle introduction. *Foundations and Trends® in Machine Learning* 16(4):494–591
- Angelopoulos AN, Barber RF, Bates S (2024) Theoretical foundations of conformal prediction. *arXiv*. Available online: <https://arxiv.org/abs/2411.11824>. Accessed 20 Nov 2024
- Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: a next-generation hyperparameter optimization framework. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage, AK, USA, 4–8 August 2019, pp 2623–2631
- Baldi P, Sadowski P (2012) Understanding dropout. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Lake Tahoe, Nevada, USA, 3–8 December 2012, pp 2814–2822
- Bao W, Yu Q, Kong Y (2021) Evidential deep learning for open set action recognition. In: *Proceedings of 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 11–17 October 2021, pp 13329–13338
- Bhatnagar A, Wang H, Xiong C, Bai Y (2023) Improved online conformal prediction via strongly adaptive online learning. In: *Proceedings of the 40th International Conference on Machine Learning*, Honolulu, Hawaii, USA, 23–29 July 2023, pp 1–27
- Borisov V, Leemann T, Seßler K, Haug J, Pawelczyk M, Kasneci G (2024) Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 35(6):7499–7519
- Campagner A, Barandas M, Folgado D, Gamboa H, Cabitza F (2024) Ensemble predictors: possibilistic combination of conformal predictors for multivariate time series classification. *IEEE Trans Pattern Anal Mach Intell* 46(11):7205–7216
- Cao D, Zhao J, Hu W et al (2022) Robust deep Gaussian process-based probabilistic electrical load forecasting against anomalous events. *IEEE Trans Industr Inf* 18(2):1142–1153
- Castellano G, Vessio G (2021) Deep learning approaches to pattern extraction and recognition in paintings and drawings: an overview. *Neural Comput Appl* 33:12263–12282
- Chen T, Guestrin C (2016) XGBoost: a scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 13–17 August 2016, pp 785–794
- Chen Z, Deng L, Li G, Sun J, Hu X, Liang L, Ding Y, Xie Y (2021) Effective and efficient batch normalization using a few uncorrelated data for statistics estimation. *IEEE Transactions on Neural Networks and Learning Systems* 32(1):348–362
- Chen H, Shi D, Zhou X, Zhang M, Liu L (2024) Application research of credit fraud detection based on distributed rotation deep forest. *Intelligent Data Analysis* 28(4):1067–1091
- Cheng H, Zhang M, Shi JQ (2024) A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Trans Pattern Anal Mach Intell* 46(12):10558–10578
- Cui C, Tang J, Xia H, Yu W, Qiao J (2023) Multiscale modeling using GAN and deep forest regression with application to dioxin emission soft sensor. *IEEE Trans Instrum Meas* 72:1–13
- Dheur V, Bossert T, Izbicki R, Ben TS (2024) Distribution-free conformal joint prediction regions for neural marked temporal point processes. *Mach Learn* 113(9):7055–7102
- Ding T, Angelopoulos A, Bates S, Jordan M, Tibshirani RJ (2023) Class-conditional conformal prediction with many classes. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023)*, New Orleans, Louisiana, USA, 10–16 December 2023, pp 64555–64576
- Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N (2021) An image is worth 16×16 words: transformers for image recognition at scale. In: *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, 3–7 May 2021. Available online: <https://openreview.net/forum?id=YicbFdNTTy>. Accessed 10 Mar
- Feng Y, Ma L, Meng X et al (2024) Advancing real-world image dehazing: Perspective, modules, and training. *IEEE Trans Pattern Anal Mach Intell* 46(12):9303–9320
- Fontana M, Zeni G, Vantini S (2023) Conformal prediction: A unified review of theory and new challenges. *Bernoulli* 29(1):1–23
- Foygel Barber R, Candès EJ, Ramdas A, Tibshirani RJ (2023) Conformal prediction beyond exchangeability. *Ann Stat* 51(2):816–845
- Gawlikowski J, Tassi CRN, Ali M et al (2023) A survey of uncertainty in deep neural networks. *Artif Intell Rev* 56(Suppl 1):1513–1589
- Gazin U (2024) Asymptotics for conformal inference. *arXiv*. Available online: <https://arxiv.org/abs/2409.12019>. Accessed 20 Oct 2024
- Gibbs I, Candès EJ (2021) Adaptive conformal inference under distribution shift. In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 6–14 December 2021, pp 1–13
- Gimeno P, Mingote V, Ortega A, Miguel A, Lleida E (2021) Generalizing AUC optimization to multiclass classification for audio segmentation with limited training data. *IEEE Signal Process Lett* 28:1135–1139
- Gorishniy Y, Rubachev I, Babenko A (n.d.) On embeddings for numerical features in tabular deep learning. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*, New Orleans, Louisiana, USA, 28 November - 9 December, pp 1–14
- Gorishniy Y, Rubachev I, Khrulkov V, Babenko A (2021) Revisiting deep learning models for tabular data. In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 6–14 December 2021, pp 18932–18943

- Gruber C, Schenk PO, Schierholz M, Kreuter F, Kauermann G (2023) Sources of uncertainty in machine learning—A statisticians' view. *arXiv*. Available online: <https://arxiv.org/abs/2305.16703>. Accessed 10 Oct 2024
- He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, USA, 27–30 June 2016, pp 770–778
- He W, Jiang Z, Xiao T, Xu Z, Li Y (2024) A survey on uncertainty quantification methods for deep learning. *arXiv*. Available online: <https://arxiv.org/abs/2302.13425>. Accessed 1 Oct 2024
- Huang J, Chen P, Lu L et al (2023) WCDForest: A weighted cascade deep forest model toward the classification tasks. *Appl Intell* 53:29169–29182
- Jeon E, Ko W, Yoon JS, Suk H-I (2023) Mutual information-driven subject-invariant and class-relevant deep representation learning in BCI. *IEEE Transactions on Neural Networks and Learning Systems* 34(2):739–749
- Jiang C, Lu Y, Zhong W, Huang B, Tan D, Song W, Qian F (2023) Deep Bayesian slow feature extraction with application to industrial inferential modeling. *IEEE Trans Industr Inf* 19(1):40–51
- Jospin LV, Laga H, Boussaid F, Buntine W, Bennamoun M (2022) Hands-on Bayesian neural networks—A tutorial for deep learning users. *IEEE Comput Intell Mag* 17(2):29–48
- Kelly M, Longjohn R, Nottingham K (n.d.) The UCI machine learning repository. Available online: <https://archive.ics.uci.edu> Accessed 3 Oct 2024
- Khurjekar I, Gerstoft P (2023) DOA uncertainty quantification with conformal prediction. In: *Proceedings of the 2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, Rome, Italy, 17–20 September 2023, pp 1–6
- Kim D, Heo B, Han D (2024) DenseNets reloaded: Paradigm shift beyond ResNets and ViTs. In *Proceedings of the 18th European Conference on Computer Vision (ECCV)*, Milan, Italy, 29 September–4 October 2024, pp 395–415. Springer. https://doi.org/10.1007/978-3-031-72646-0_23
- Krizhevsky A (2009) Learning multiple layers of features from tiny images. Technical Report, University of Toronto. Available online: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>. Accessed 8 Oct 2024
- Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, California, USA, 4–9 December 2017, pp 6405–6416
- Lei H, Bellotti A (2023) Reliable prediction intervals with directly optimized inductive conformal regression for deep learning. *Neural Netw* 168:194–205
- Leung CS, Wang H-J, Sum J (2010) On the selection of weight decay parameter for faulty networks. *IEEE Trans Neural Networks* 21(8):1232–1244
- Li J, Jiang X, Liu X, Jia F, Dai C (2024) Optimizing the feature set and electrode configuration of high-density electromyogram via interpretable deep forest. *Biomed Signal Process Control* 87:105445
- Linusson H, Johansson U, Boström H (2020) Efficient conformal predictor ensembles. *Neurocomputing* 397:266–278
- Liu Q, Li X, Wang K (2024) Atmospheric SO₂ pollutant prediction using mutual information based TCNN-GRU model for flue gas desulfurization process. *Intelligent Data Analysis* 28(6):1723–1740
- Liu A, Liu X, Yu H, Zhang C, Liu Q, Tao D (2021) Training robust deep neural networks via adversarial noise propagation. *IEEE Trans Image Process* 30:5769–5781
- Linusson H, Norinder U, Boström H, Johansson U, Löfström T (2017) On the calibration of aggregated conformal predictors. In: *Gammerman A, Vovk V, Luo Z, Papadopoulos H (eds) Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications*, PMLR 2017, 60, pp 154–173
- Ma X, Ouyang W, Simonelli A, Ricci E (2024) 3D object detection from images for autonomous driving: A survey. *IEEE Trans Pattern Anal Mach Intell* 46(5):3537–3556
- Ming Y, Shao H, Cai B, Liu B (2024) rgfc-Forest: An enhanced deep forest method towards small-sample fault diagnosis of electromechanical system. *Expert Syst Appl* 238:122178
- Nguyen VL, Shaker MH, Hüllermeier E (2022) How to measure uncertainty in uncertainty sampling for active learning. *Mach Learn* 111:89–122
- Olsson H, Kartasalo K, Mulliqi N et al (2022) Estimating diagnostic uncertainty in artificial intelligence-assisted pathology using conformal prediction. *Nat Commun* 13:7761
- Otter DW, Medina JR, Kalita JK (2021) A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems* 32(2):604–624
- Pang M, Ting KM, Zhao P, Zhou ZH (2022) Improving deep forest by screening. *IEEE Trans Knowl Data Eng* 34(9):4298–4312
- Park S, Cohen KM, Simeone O (2024) Few-shot calibration of set predictors via meta-learned cross-validation-based conformal prediction. *IEEE Trans Pattern Anal Mach Intell* 46(1):280–291
- Park S, Simeone O (2024) Quantum conformal prediction for reliable uncertainty quantification in quantum machine learning. *IEEE Transactions on Quantum Engineering* 5:1–24
- Psaros AF, Meng X, Zou Z, Guo L, Karniadakis GE (2023) Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *J Comput Phys* 477:111902
- Qian Z, Huang K, Wang QF, Zhang XY (2022) A survey of robust adversarial training in pattern recognition: Fundamental, theory, and methodologies. *Pattern Recogn* 131:108889
- Rubachev I, Kartashev N, Gorishniy Y, Babenko A (2024) TabReD: Analyzing pitfalls and filling the gaps in tabular deep learning benchmarks. *arXiv*. Available online: <https://arxiv.org/abs/2406.19380>. Accessed 20 Oct 2024
- Sheng C, Kuang G, Bai L, Hou C, Guo Y, Xu X, Pietikäinen M, Liu L (2024) Deep learning for visual speech analysis: A survey. *IEEE Trans Pattern Anal Mach Intell* 46(9):6001–6022
- Shu Y, Cao Z, Gao J, Wang J, Yu PS, Long M (2023) Omni-training: Bridging pre-training and meta-training for few-shot learning. *IEEE Trans Pattern Anal Mach Intell* 45(12):15275–15291
- Song H, Kim M, Park D, Shin Y, Lee JG (2023) Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 34(11):8135–8153
- Talaei Khoei T, Ould Slimane H, Kaabouch N (2023) Deep learning: Systematic review, models, challenges, and research directions. *Neural Comput Appl* 35:23103–23124
- Teng X, Pei S, Lin Y-R (2021) StoCast: Stochastic disease forecasting with progression uncertainty. *IEEE J Biomed Health Inform* 25(3):850–861
- Tian Y, Zhang Y (2022) A comprehensive survey on regularization strategies in machine learning. *Information Fusion* 80:146–166
- Utkin LV, Ryabinin MA (2018) A Siamese Deep Forest Knowledge-Based Systems 139:13–22
- Utkin LV (2020) An imprecise deep forest for classification. *Expert Syst Appl* 141:1–11
- Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014) OpenML: Networked science in machine learning. *ACM SIGKDD Explorations Newsl* 15(2):49–60
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, California, USA, 4–9 December 2017, pp 6000–6010

- Vovk V, Gammerman A, Shafer G (2022) *Algorithmic Learning in a Random World*, 2nd edn. Springer, Cham, Switzerland
- Vovk V, Petej I, Nouretdinov I, Manokhin V, Gammerman A (2020) Computationally efficient versions of conformal predictive distributions. *Neurocomputing* 397:292–308
- Wang H, Tang Y, Jia Z et al (2020) Dense adaptive cascade forest: a self-adaptive deep ensemble for classification problems. *Soft Comput* 24:2955–2968
- Wright LG, Onodera T, Stein MM et al (2022) Deep physical neural networks trained with backpropagation. *Nature* 601:549–555
- Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv*. Available online: <https://arxiv.org/abs/1708.07747>. Accessed 20 Oct 2024
- Xu C, Xie Y (2023) Conformal prediction for time series. *IEEE Trans Pattern Anal Mach Intell* 45(10):11575–11587
- Yang Y, Lv H, Chen N (2023) A survey on ensemble learning under the era of deep learning. *Artif Intell Rev* 56:5545–5589
- Yang Y, Kuchibhotla AK (2024) Selection and aggregation of conformal prediction sets. *J Am Stat Assoc* 00:1–13
- Yang Z, Xu Q, Bao S, Cao X, Huang Q (2022) Learning with multiclass AUC: Theory and algorithms. *IEEE Trans Pattern Anal Mach Intell* 44(11):7747–7763
- Yi H, Bui K-HN (2021) An automated hyperparameter search-based deep learning model for highway traffic prediction. *IEEE Trans Intell Transp Syst* 22(9):5486–5495
- Yin X, Müller F, Laguna AF et al (2024) Deep random forest with ferroelectric analog content addressable memory. *Sci Adv* 10(23):eadk8471
- Zheng Q, Peng Z, Dang Z, Zhu L, Liu Z, Zhang Z, Zhou J (2023) Deep tabular data modeling with dual-route structure-adaptive graph networks. *IEEE Trans Knowl Data Eng* 35(9):9715–9727
- Zhou Z-H, Feng J (2017) Deep forest: towards an alternative to deep neural networks. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, 19–25 August 2017, pp 3553
- Zhou ZH, Feng J (2019) Deep forest. *Natl Sci Rev* 6(1):74–86
- Zhu H, Liang S, Hu W, Li F-Q, Yuan Y, Wang S-L, Cheng G (2024) Improve deep forest with learnable layerwise augmentation policy schedules. In: *Proceedings 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seoul, Korea, 14–19 April 2024, pp 6660–6664

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.