Interpreting Deep Forest through Feature Contribution and MDI Feature Importance

YI-XIAO HE, National Key Laboratory for Novel Software Technology, and School of Artificial Intelligence, Nanjing University, China
SHEN-HUAN LYU, Key Laboratory of Water Big Data Technology of Ministry of Water Resources, and College of Computer Science and Software Engineering, Hohai University, China
YUAN JIANG*, National Key Laboratory for Novel Software Technology, and School of Artificial Intelligence, Nanjing University, China

Deep forest is a non-differentiable deep model that has achieved impressive empirical success across a wide variety of applications, especially on categorical/symbolic or mixed modeling tasks. Many of the application fields prefer explainable models, such as random forests with feature contributions that can provide a local explanation for each prediction, and Mean Decrease Impurity (MDI) that can provide global feature importance. However, deep forest, as a cascade of random forests, possesses interpretability only at the first layer. From the second layer on, many of the tree splits occur on the new features generated by the previous layer, which makes existing explaining tools for random forests inapplicable. To disclose the impact of the original features in the deep layers, we design a calculation method with an estimation step followed by a calibration step for each layer, and propose our feature contribution and MDI feature importance calculation tools for deep forest. Experimental results on both simulated data and real-world data verify the effectiveness of our methods.

CCS Concepts: • **Computing methodologies** → **Supervised learning**; **Ensemble methods**; **Classification and regression trees**.

Additional Key Words and Phrases: deep forest, feature importance, interpretability

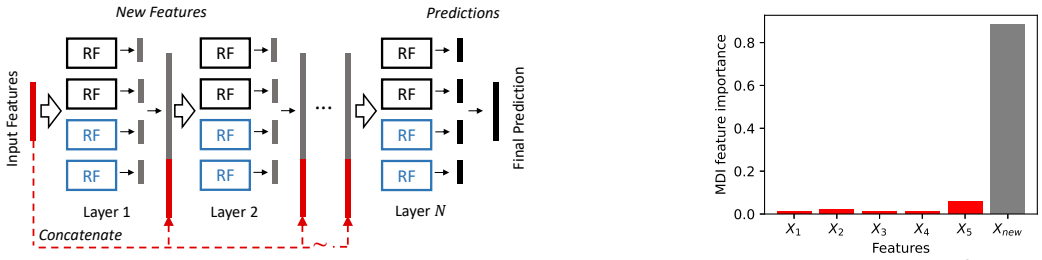
1 INTRODUCTION

By suggesting that the key to deep learning may lie in the *layer-by-layer processing*, *in-model feature transformation* and *sufficient model complexity*, Zhou and Feng [42] propose the first deep forest model and the gcForest algorithm, which are realized by non-differentiable modules without backward-propagation in training. It consists of a cascade forest structure, each layer contains multiple random forests. The predictive probability vectors output by the forests are concatenated with the original features, then serve as the input to the next layer [43]. Benefiting from the feature transformation in the cascade structure, deep forests (DFs) outperform various classical tree-based algorithms, e.g., classification and regression tree [7, CART], AdaBoost [28], random forest [6, RF], gradient

*Corresponding author

Authors' addresses: Yi-Xiao He, heyx@lamda.nju.edu.cn, National Key Laboratory for Novel Software Technology, and School of Artificial Intelligence, Nanjing University, Nanjing, China, 210023; Shen-Huan Lyu, lvsh@hhu.edu.cn, Key Laboratory of Water Big Data Technology of Ministry of Water Resources, and College of Computer Science and Software Engineering, Hohai University, Nanjing, China, 211100; Yuan Jiang, jiangy@lamda.nju.edu.cn, National Key Laboratory for Novel Software Technology, and School of Artificial Intelligence, Nanjing University, Nanjing, China, 210023.

ACM 1556-4681/2024/1-ART
https://doi.org/10.1145/3641108



(a) The cascade forest structure of deep forests. The colors black and blue indicate different types of random forests. The vectors in gray color are the new features generated by individual forests, being input to the next layer.

(b) A typical MDI feature importance result of the second and above layers. X_{new} is the sum of MDI of new features. The new features tend to have dominating importance.

Fig. 1. The new features play an important role in deep forests but are hard to interpret.

boost decision tree [12, GBDT], extremely random forest [13, ERF] and XGBoost [8] in empirical study. In recent years, deep forests have been widely extended to various real-world applications [5, 33, 39] and learning tasks [34–37]. There are also variants aiming at improving performance and reducing computational and memory cost [9, 21, 26, 40].

Deep forest [42] is a non-differentiable deep model built with decision tree ensembles. Its main structure is a cascade forest. As illustrated in Figure 1(a), the model is built of multiple layers of random forests, where different colors indicate different types of forests. For a C-class classification problem, a forest outputs a C-dim probability vector. For regression problems, each forest outputs a scalar prediction. The outputs of all the forests in the same layer are concatenated to serve as the augment features (in gray), which are further concatenated together with the original features (in red) to be input to the next layer. We refer to this operation as *feature concatenation*. Feature concatenation is an important component of deep forest model, and has been proven to lead to a faster convergence rate than random forest [19]. The total number of layers is automatically determined by the validation accuracy. With layer-by-layer processing, the predictive performance can be improved accordingly.

In many applications of prediction models, such as fraud detection [3], disease classification [2, 38], we not only need a high-accuracy model, but also hope to understand how does the model make predictions. For random forests, there already exists explaining tools, such as Mean Decrease Impurity [10, MDI] and feature contribution [16, 24]. However, when it comes to deep forest, the feature concatenation operation causes difficulties in interpreting deep forest. While the new features are used together with the original features, empirical results and theoretical analyses [1, 18, 19] show that the new features dominate the splits in random forests in the second and following layers. Figure 1(b) provides an example. We can observe that X_{new} plays a dominating role according to the MDI feature importance, but we don't know how it is related to the original features.

To tackle this issue, we exploit the fact that the new features are the predictions from the preceding layer, which is the key to our analysis and method design. We develop a two-step estimation-then-calibration process to enable feature contribution and feature importance calculation for deep forests. The estimation step associates the contributions of the new features with the contributions of the original features at the preceding layer through the prediction of specific training samples in the splitting nodes. The calibration step ensures that our proposed method yields proper feature contribution and feature importance. Our main contributions are summarized as follows.

- We propose a feature contribution calculation method for deep forest, enabling us to explain its predictions for each instance, i.e., whether the considered feature enlarges or reduces the predicted value for the considered class, and by how much.
- We propose an MDI feature importance calculation method for deep forest, enabling us to tell the overall impact of each feature in building the whole model.
- We analyze the properties that feature contribution and feature importance should have to ensure that our designed methods are proper feature contribution and feature importance.
- Experimental results show that our proposed tools faithfully reflect the influence of each feature on deep forest prediction for both regression and classification problems. Furthermore, since deep forest is more powerful than random forest, our MDI feature importance also exhibits better estimation quality.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 briefly describes feature contribution and MDI feature importance for random forests, and discusses the properties they should have. Section 4 and Section 5 present our proposed explaining tools. Section 6 reports the experimental results and Section 7 concludes the paper.

2 RELATED WORK

Since Zhou and Feng [42] proposed the first deep forest model, there are mainly three lines of work to study it. A line of work establishes a screening framework to reduce computational cost and memory requirement for deep forest, including confidence screening [25], feature screening [26], hash-based method [21, 40] and stability-based method [9]. The second line of work extends deep forest algorithms to different learning tasks and real applications, such as multi-label problems [37], weak-label learning [36], metric learning tasks [34, 35], and financial anti-fraud system [39]. The third line of work is the theoretical analysis of deep forests. Both its generalization performance [18, 20] and consistency [1, 19] have been analyzed. Recently, explainable Artificial Intelligence has attracted attention [2]. However, very little work has been done to help users understand the predictions made by deep forest models.

Its component, random forest, has been extensively studied for interpretation. In this paper, we care about two types of model interpretation. 1) Characterizing the overall importance of each feature. There are two widely used measures, the Mean Decrease Impurity [10, MDI] and the Mean Decrease Accuracy [6, MDA]. MDI sums up the total reduction of splitting criterion caused by each feature. It is known to favor features with many categories [22, 31] and may lead to systematic bias in feature selection by incorrectly assigning high importance to irrelevant features [17, 23, 32, 41]. On the other hand, MDA measures the importance of a feature by the reduction in the accuracy after randomly permuting the sample values of a given feature. Different permuting choices have been studied by Janitza et al. [14], Strobl et al. [30]. MDA is applicable to any black-box model. However, it relies on the model's accuracy and has to be calculated by repeated prediction, while MDI is an intrinsic measure for random forest and can be directly obtained once the training is finished. 2) Interpreting every single prediction of random forest through each feature's contribution to the prediction. Palczewska et al. [24] and Saabas [27] proposed feature contribution, recording the change in average response from parent to child node as the contribution of a splitting feature. Note that with the help of feature contribution, a debiased MDI estimate can be obtained [17]. It functions well even when there are a large number of irrelevant features and severe noise that MDA fails because of poor accuracy.

In this paper, we aim to show the impact of input features in the deep forest model through feature contribution and MDI feature importance that are specifically designed for deep forests. Although we can directly apply existing feature contribution and MDI calculation to the forests in each layer, the impact of the transformed features is less helpful in interpreting deep forests. Therefore, a method that can relate the new features to the original features needs to be designed. Recently, Kim et al. [15] simplified a lightweight multilayered random

forest model by selecting the most important paths in the component forests. Although this simplification can be applied to deep forests, this study did not provide an analysis of feature contribution when new features are concatenated with original features.

3 FEATURE CONTRIBUTION AND MDI FEATURE IMPORTANCE

We first briefly describe feature contribution and MDI feature importance for random forests, and their relationship. Then we analyze the properties that feature contribution and feature importance should have, which will serve as guidance for developing our interpretation tools for deep forests. The key symbols and notations used in this paper are listed in Table 1.

Subject	Sign	Description
Setting	$\mathcal{D}$	The training data.
	n	The total number of training instances.
	K	The number of features.
	C	The number of classes.
	$(\boldsymbol{x}, y)$	A sample $\boldsymbol{x}$ and its ground-truth label y.
Tree & Forest	t	A tree node.
	l	The depth of a leaf node.
	$I(T)$	All the internal nodes in tree T.
	$n(t)$	The number of training instances in node t.
	$s(t)$	The splitting feature of an internal node t.
	$\Delta_{\mathcal{I}}(t)$	The decrease of impurity by splitting node t.
	μ_0	The average response of training data.
	$\mu(t)$	The average response of the training data in node t.
	$\Delta\mu(t_i)$	The change in average response of node t_i and its parent node t_{i-1}.
	$f_F(\boldsymbol{x})$	The prediction of forest F on $\boldsymbol{x}$.
	$f_{F,k}(\boldsymbol{x})$	The amount that feature k contributes to F's prediction value on $\boldsymbol{x}$.
	$\mathrm{MDI}(k, F)$	The MDI feature importance of feature k in forest F.
Deep Forest	$\hat{\mu}(t)$	The average predictive value by the preceding layer of the training data in node t, which can be viewed as average response estimated by the preceding layer's prediction.
	$\Delta\hat{\mu}(t_i)$	The change in average response of node t_i and its parent node t_{i-1} estimated by the preceding layer's prediction.
	$\Delta\hat{\mu}(t_i, k)$	Estimated average response change caused by feature k.
	$\Delta\widetilde{\mu}(t_i, k)$	The calibrated average predictive response change caused by feature k.
	k'	A new feature generated by the preceding layer's prediction.
	K'	The number of new features in the second and above layers.
	g	The calibration function that maps the estimated feature contribution $(\Delta\hat{\mu}(t_i, k))_{k=1}^{K}$ to the calibrated feature contribution $(\Delta\widetilde{\mu}(t_i, k))_{k=1}^{K}$.
	$\widetilde{f}_{F',k}(\boldsymbol{x})$	The calculated contribution of the original feature k for the second layer forest F'.
	$f_L(\boldsymbol{x})$	The prediction of $\boldsymbol{x}$ from the last layer of deep forest.
	$\widetilde{f}_{L,k}(\boldsymbol{x})$	The calculated contribution of feature k in the last layer of deep forest.
	$\widetilde{\mathrm{MDI}}(k, DF)$	The estimated MDI feature importance of feature k in DF.

Table 1. Key symbols and notations.

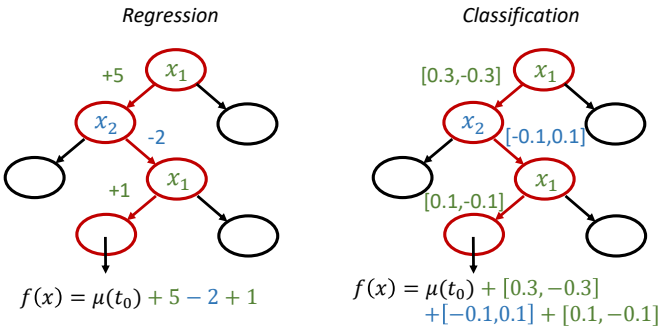


Fig. 2. The contribution of each split to the prediction of an instance $\boldsymbol{x}$. Color blue or green corresponds to the change of average response by splitting on x_1 or x_2. Positive and negative values indicate the split enlarges or decreases the predictive value or the probability for the corresponding class.

3.1 Feature Contribution

Feature contribution is applicable to every single prediction. It enables us to decompose the prediction into the sum of contributions from each feature [24].

For Decision Trees. The prediction of an instance $\boldsymbol{x}$ is the average of the training instances in the leaf node it falls in. Let l denote the depth of the leaf node that $\boldsymbol{x}$ falls in. Along the decision path to the leaf node, as $\boldsymbol{x}$ goes through node $t_0, t_1, \ldots, t_l$, the average response of the training instances changes from node to node. As demonstrated in Figure 2, the final prediction can be represented as

$$f_T(\boldsymbol{x}) = \mu(t_0) + \sum_{0<i\leq l} \Delta\mu(t_i)\,, \tag{1}$$

where $\Delta\mu(t_i) = \mu(t_i) - \mu(t_{i-1})$. $\Delta\mu(t_i)$ is the change in average response caused by splitting node t_i from t_{i-1}. Note that this calculation is applicable for both regression trees and classification trees. For regression, $f_T(\boldsymbol{x})$, $\mu(t_0)$ and $\Delta\mu(t_i)$ are scalars. For classification, they are C-dimensional vectors, where C denotes the number of classes. Let $s(t)$ denote the splitting feature of node t. Let K denote the number of input features. If we sum up the split contributions made by the same feature, the prediction of $\boldsymbol{x}$ can be written as

$$f_T(\boldsymbol{x}) = \mu(t_0) + \sum_{k=1}^{K} f_{T,k}(\boldsymbol{x})\,, \tag{2}$$

where

$$f_{T,k}(\boldsymbol{x}) = \sum_{0<i\leq l: s(t_{i-1})=k} \Delta\mu(t_i) \tag{3}$$

is the contribution of feature k in the prediction of tree T on $\boldsymbol{x}$.

For Random Forests. To extend feature contribution from a tree to a forest, we only need to take the average of the trees in the forest. Assuming each tree has the same distribution of training instances, we use μ_0 to substitute $\mu(t_0)$. Therefore,

$$f_F(\boldsymbol{x}) = \mu_0 + \sum_{k=1}^{K} f_{F,k}(\boldsymbol{x}), \tag{4}$$

where

$$f_{F,k}(\boldsymbol{x}) = \frac{1}{|F|}\sum_{T\in F} f_{T,k}(\boldsymbol{x}) \tag{5}$$

represents the contribution of feature k in forest F's prediction.

3.2 Feature Importance

Mean Decrease Impurity [10, MDI] is an intrinsic feature importance measure for random forests. It measures the overall importance of each feature in building the whole model. For a tree T, its MDI is calculated by summing the impurity decrease in each internal node weighted by the fraction of training data in the node. Let n denote the number of training instances, $I(T)$ denote all the internal nodes in tree T, $n(t)$ denote the number of training instances falling in node t. The feature importance of feature k in tree T is calculated as

$$\mathrm{MDI}(k,T) = \sum_{t\in I(T), s(t)=k} \frac{n(t)}{n}\Delta_{\mathcal{I}}(t)\,, \tag{6}$$

where the decrease impurity of any node t is defined as

$$\Delta_{\mathcal{I}}(t) \triangleq \mathrm{Impurity}(t) - \frac{n\left(t^{\mathrm{left}}\right)}{n(t)}\mathrm{Impurity}\left(t^{\mathrm{left}}\right) - \frac{n\left(t^{\mathrm{right}}\right)}{n(t)}\mathrm{Impurity}\left(t^{\mathrm{right}}\right)\,, \tag{7}$$

and the impurity of any node t is defined as

$$\mathrm{Impurity}(t) \triangleq \frac{1}{n(t)}\sum_{i:\boldsymbol{x}_i\in t}(y_i-\mu(t))^2\,. \tag{8}$$

Here t^{left} and t^{right} are two child nodes of node t, $\mu(t)$ is the average response of training instances in node t. Note that the above expressions assume using the sample variance of labels as the impurity measure for regression problems. For classification problems, Li et al. [17] disclose that using the one-hot encoding of the categorical responses, i.e., substituting y and $\mu(t)$ with vectors, the impurity expression in Eq. (8) is equivalent to Gini index, a popular impurity measure for classification.

Since the forest is an average of all the individual trees, the feature importance of feature k in forest F is also the average

$$\mathrm{MDI}(k,F) = \frac{1}{|F|}\sum_{T\in F}\mathrm{MDI}(k,T)\,. \tag{9}$$

3.3 Relationship Between Feature Contribution and Feature Importance

Recently, Li et al. [17] disclosed that the original expression of MDI as Eq. (6) can be written as

$$\mathrm{MDI}(k,T) = \frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}} f_{T,k}\left(\boldsymbol{x}_i\right)\cdot y_i = \mathrm{Cov}\left(f_{T,k}\left(\boldsymbol{x}_i\right), y_i\right). \tag{10}$$

Here $\mathcal{D}$ is the dataset we use for computing MDI feature importance. If $\mathcal{D}$ is the same as the training set, then Eq. (10) yields the same result as the original MDI expression. Li et al. [17] claim that using out-of-bag data to calculate Eq. (10) can achieve an unbiased result.

Equation (10) offers a new way of computing MDI feature importance in a decision tree T, i.e., the covariance of feature contribution $f_{T,k}(\boldsymbol{x})$ and label y. Before, we could only compute MDI by summing the decrease of impurity at each node. Equation (10) enables us to compute feature importance by summing over instances, decoupling the computation of feature contribution from the tree training process.

For a forest F, suppose we use the same dataset $\mathcal{D}$ to calculate MDI for every tree in it, its MDI can be calculated as

$$\mathrm{MDI}(k, F) = \frac{1}{|F|} \sum_{T \in F} \mathrm{MDI}(k, T) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} y_i f_{F,k}(\boldsymbol{x}_i) . \tag{11}$$

In Section 4, we will extend the calculation of feature contribution to deep forest, then in Section 5 we derive the calculation of MDI feature importance of deep forest using their relationships.

3.4 Properties of Feature Importance and Feature Contribution

3.4.1 Properties of Feature Contribution. The term *feature contribution* means decomposing the prediction on a single instance into a sum of contributions from each feature. A proper feature contribution should satisfy the following properties,

- Feature contribution is defined for each single prediction.
- In regression problems, feature contribution is a K-dimensional vector, whose element $f_k(\boldsymbol{x})$ denotes the contribution of feature k in the final prediction. In classification problems, feature contribution is a matrix of shape (K, C), its element $f_{k,c}(\boldsymbol{x})$ denoting the contribution of feature k in the predictive probability of class c.
- The elements of feature contribution can be positive, negative, or zero.
- Given an instance $\boldsymbol{x}$, the sum of bias and feature contribution equals the prediction. That is, for regression,

$$f(\boldsymbol{x}) = \mu_0 + \sum_{k=1}^{K} f_k(\boldsymbol{x}) . \tag{12}$$

  For classification,

$$f(\boldsymbol{x})_c = \mu_{0,c} + \sum_{k=1}^{K} f_{k,c}(\boldsymbol{x}), 1 \le c \le C . \tag{13}$$

3.4.2 Properties of Feature Importance. The term *feature importance* means the overall impact of each feature in building the whole model. A proper feature importance measure should have the following properties,

- Feature importance is defined over a whole dataset, with labels available.
- Feature importance is a K-dimensional vector, for both regression and classification.
- The elements of feature importance are non-negative.
- The sum of feature importance is no larger than the total response variance of the dataset. For regression, the total response variance is $\frac{1}{n}\sum_i (y_i - \mu_0)^2$. For classification, the total response variance is defined to be $\frac{1}{n}\sum_i \sum_{1 \le c \le C} (y_{i,c} - \mu_{0,c})^2$.

Note that here we use variance-based definitions for feature importance. We can easily scale the feature importance to sum to 1 by normalizing with the total response variance.

4 FEATURE CONTRIBUTION FOR DEEP FOREST

We propose a feature contribution calculation method for deep forests, which breaks down the predictions of the cascade forest structure into the contributions of the original features. With the feature contribution result, utilizing the relationship introduced in Section 3.3, we will easily acquire the MDI feature importance in Section 5. Our proposed feature contribution for deep forest has a layer-by-layer calculation procedure. The output of the first layer of forest can be directly disassembled to the contributions of the original features. From the second layer, we trace the contribution of the new feature back to the contribution of the original feature through the output of the previous layer. This process can be repeated until the final layer.

For forests in the second layer of a deep forest (so do forests in the following layers), splits often occur on new features. Let $\Delta\mu(t_i)$ denote the change in average response from node t_{i-1} to t_i,

$$\Delta\mu(t_i) = \mu(t_i) - \mu(t_{i-1}) = \frac{1}{n(t_i)} \sum_{j:\boldsymbol{x}_j \in t_i} y_j - \frac{1}{n(t_{i-1})} \sum_{j:\boldsymbol{x}_j \in t_{i-1}} y_j \ . \tag{14}$$

If node t_{i-1} splits on a new feature, then $\Delta\mu(t_i)$ represents the contribution of the new feature. But what we want to know is the contribution of the original features. To uncover their connection, we propose a feature contribution calculation method for deep forests with an estimation step followed by a calibration step.

4.1 The Estimation Step

The key to analyzing the contribution of a new feature is the fact that the new feature happens to be the predicted value output by a forest in the preceding layer. We propose to use the predicted value to estimate the average response. This estimation is applicable because generally, we observe that as n increases, the predicted value of a forest approximates the ground truth label, as consistency of random forests [29] and completely random forests [4] have been proved under different mild assumptions.

Since the predicted value by a first-layer forest can be represented as the contribution of the original features (Section 3.1), once the average response is replaced by the predicted value, we can build the connection between the contributions of the new feature and the original features.

Suppose the node t_i in a second-layer tree T' uses a new feature k' to split, where k' is the output by the first-layer forest F. Then

$$\begin{aligned}
\hat{\mu}(t_i) &= \frac{1}{n(t_i)} \sum_{j:\boldsymbol{x}_j \in t_i} f_F(\boldsymbol{x}_j) \\
&= \frac{1}{n(t_i)} \sum_{j:\boldsymbol{x}_j \in t_i} \left(\mu_0 + \sum_{k=1}^{K} f_{F,k}(\boldsymbol{x}_j) \right) \\
&= \mu_0 + \frac{1}{n(t_i)} \sum_{j:\boldsymbol{x}_j \in t_i} \left(\sum_{k=1}^{K} f_{F,k}(\boldsymbol{x}_j) \right).
\end{aligned} \tag{15}$$

According to Eq. (15), the estimation of average response using $f_F(\boldsymbol{x})$ is decomposed to the contribution of original features $f_{F,k}(\boldsymbol{x}),\ k = 1, \ldots, K$. Then the estimated response change from splitting cell t_{i-1} into t_i is

$$\Delta\hat{\mu}(t_i) = \frac{1}{n(t_i)} \sum_{j:\boldsymbol{x}_j \in t_i} \left(\sum_{k=1}^{K} f_{F,k}(\boldsymbol{x}_j) \right) - \frac{1}{n(t_{i-1})} \sum_{j:\boldsymbol{x}_j \in t_{i-1}} \left(\sum_{k=1}^{K} f_{F,k}(\boldsymbol{x}_j) \right) \ . \tag{16}$$

Let

$$\Delta\hat{\mu}(t_i, k) = \frac{1}{n(t_i)} \sum_{j:\boldsymbol{x}_j \in t_i} f_{F,k}(\boldsymbol{x}_j) - \frac{1}{n(t_{i-1})} \sum_{j:\boldsymbol{x}_j \in t_{i-1}} f_{F,k}(\boldsymbol{x}_j) \tag{17}$$

denote the contribution of feature k in the estimated change of average response, then

$$\Delta\hat{\mu}(t_i) = \sum_{k=1}^{K} \Delta\hat{\mu}(t_i, k) \ . \tag{18}$$

In this way, we attribute the estimated response change caused by splitting node t_i using feature k' to the contribution of original features.

4.2 The Calibration Step

Although the prediction on $\boldsymbol{x}$ by a first-layer forest $f_F(\boldsymbol{x})$ is close to its label y, there are inevitably errors, otherwise there is no need for a multi-layer cascade forest structure to improve performance. Therefore, the estimated response change $\Delta\hat{\mu}(t_i)$ and the true response change $\Delta\mu(t_i)$ are usually unequal. To avoid the error propagating layer by layer, we propose a calibration step after the estimation step to ensure the computed feature contribution satisfies the property that the sum of bias and feature contribution equals the prediction as stated in Section 3.4.

The calibration operation can be expressed as a function

$$g : (\Delta\hat{\mu}(t_i, 1), \Delta\hat{\mu}(t_i, 2), \ldots, \Delta\hat{\mu}(t_i, K)) \rightarrow (\Delta\widetilde{\mu}(t_i, 1), \Delta\widetilde{\mu}(t_i, 2), \ldots, \Delta\widetilde{\mu}(t_i, K)) \ , \tag{19}$$

such that

$$\sum_{k=1}^{K} \Delta\widetilde{\mu}(t_i, k) = \Delta\mu(t_i) \ . \tag{20}$$

Thus the prediction of $f_{T'}(\boldsymbol{x})$ can be interpreted as the following decomposition,

$$f_{T'}(\boldsymbol{x}) = \mu_0 + \sum_{k=1}^{K} \widetilde{f}_{T',k}(\boldsymbol{x}) \ , \tag{21}$$

where

$$\widetilde{f}_{T',k}(\boldsymbol{x}) = \sum_{s(t_{i-1})=k} \Delta\mu(t_i) + \sum_{s(t_{i-1})=k'} \Delta\widetilde{\mu}(t_i, k) \tag{22}$$

is the calibrated contribution of the k-th original feature.

Naive multiplicative calibration. A naive instantiation of Eq. (19) is to re-scale the decomposition using a common scale factor,

$$\Delta\widetilde{\mu}(t_i, k) = \Delta\hat{\mu}(t_i, k) \frac{\Delta\mu(t_i)}{\Delta\hat{\mu}(t_i)} \ . \tag{23}$$

Obviously, Eq. (20) holds. However, since the feature contributions can be both positive and negative, the naive re-scaling might lead to numerical problems.

Naive additive calibration. Using additive calibration can better avoid numerical problems than multiplicative scaling. The naive modification is proportional to the absolute value of its estimated contribution,

$$\Delta\widetilde{\mu}(t_i, k) = \Delta\hat{\mu}(t_i, k) + \frac{|\Delta\hat{\mu}(t_i, k))|}{\sum_k |\Delta\hat{\mu}(t_i, k))|} (\Delta\mu(t_i) - \Delta\hat{\mu}(t_i)) \ . \tag{24}$$

However, if the value of $\Delta\mu(t_i) - \Delta\hat{\mu}(t_i)$ is large, the features whose estimated contribution was negative will be modified to be positive, and the more negative it was, the more positive it will be, which is obviously not sensible.

Partial additive calibration. Here we advocate a partial calibration method. We choose a subset of features according to their signs

$$S(t_i) = \{k : \text{sign}(\Delta\hat{\mu}(t_i, k)) = \text{sign}(\Delta\mu(t_i))\} \ , \tag{25}$$

and we only calibrate this subset of features

$$\Delta\widetilde{\mu}(t_i, k) = \Delta\hat{\mu}(t_i, k) \left(1 + \frac{\Delta\mu(t_i) - \Delta\hat{\mu}(t_i)}{\sum_{k \in S(t_i)} \Delta\hat{\mu}(t_i, k))}\right) \ , \ k \in S(t_i) \ . \tag{26}$$

It is easy to check that Eq. (20) holds. Therefore, the response change brought by splitting on k' has been decomposed into contributions of the original K features.

4.3 The Calculation Procedure

Algorithm 1 illustrates the procedure of calculating feature contribution for a tree in deep forest. When traversing the tree, if the node chooses an original feature to split, we simply add the contribution to this feature. If the node uses a new feature to split, we trace back the contribution to each of the corresponding original features as in Eq. (17) and calibrate as in Eq. (19). The calculation is done by recursively calling *CalculateContribution* and getting the contributions at all the leaf nodes. To compute feature contribution on a given instance, we simply find the leaf node t that $\boldsymbol{x}$ falls in and get $\widetilde{f}_{T',k}(\boldsymbol{x}) = f_{t,k}$.

Algorithm 1 Calculate feature contribution for a tree

Require: Tree node t
Ensure: CalculateContribution(t)
if t is a root node **then**
  $f_{t,k} \leftarrow 0, k = 1, \ldots, K$
end if
for each child node t^* of t **do**
  if $s(t) \in \{1, \ldots, K\}$ **then**
    Calculate $\Delta\mu(t^*, s(t))$ as in Eq. (14)
    $f_{t^*,s(t)} \leftarrow f_{t,s(t)} + \Delta\mu(t^*, s(t))$
  else
    Calculate $(\Delta\widetilde{\mu}(t^*, k))_{k=1}^{K}$ using Eq. (17) and Eq. (19)
    $f_{t^*,k} \leftarrow f_{t,k} + \Delta\widetilde{\mu}(t^*, k),\ k = 1, \ldots, K$
  end if
  if node t^* is a leaf node **then**
    return $\left(f_{t^*,k}\right)_{k=1}^{K}$
  else
    CalculateContribution(t^*)
  end if
end for

Based on the analysis of a tree, it is easy to interpret the prediction of a second-layer forest

$$f_{F'}(\boldsymbol{x}) = \mu_0 + \sum_{k=1}^{K} \widetilde{f}_{F',k}(\boldsymbol{x}) \,, \tag{27}$$

where

$$\widetilde{f}_{F',k}(\boldsymbol{x}) = \frac{1}{|F'|} \sum_{T' \in F'} \widetilde{f}_{T',k}(\boldsymbol{x}) \,. \tag{28}$$

For the above analysis, we assume T' is the individual tree in the second-layer forest F'. But it is easy to see that, after the above calculation, we will get the contributions of the K original features $f_{F',k}(\boldsymbol{x})$. Therefore, from the third layer and beyond, this calculation process can be repeated layer by layer, all the way down to the final layer, so that we obtain feature contribution for deep forest predictions.

Usually, each layer in a deep forest contains several forests. The final prediction is made by taking the average of all the outputs of the last layer of forests. Let L denote the set containing forests in the last layer, the final

prediction $f(\boldsymbol{x})$ made by deep forest can be interpreted as

$$f_L(\boldsymbol{x}) = \mu_0 + \sum_{k=1}^{K} \widetilde{f}_{L,k}(\boldsymbol{x}), \tag{29}$$

where

$$\widetilde{f}_{L,k}(\boldsymbol{x}) = \frac{1}{|L|} \sum_{F \in L} \widetilde{f}_{F,k}(\boldsymbol{x}) \tag{30}$$

is the contribution of feature k in the last layer of forests, which is also the feature contribution for the whole deep forest.

Time complexity. As shown in Algorithm 1, we only need to traverse each tree once and do simple computation concerning the feature contributions of the training instances falling in each node. The depth of a tree is approximately $O(\log n)$. For nodes that have the same depth, all the training instances are processed once. Therefore, the time complexity of the computation for a tree is $O(n \log n)$. Let M_1 denote the number of trees in each layer, M_2 denote the maximum number of layers, and the time complexity for computing feature contribution for deep forests is $O(M_1 M_2 n \log n)$.

Space complexity. When computing feature contribution for a tree, we need to maintain $\left[f_{t,k}\right]_{k=1}^{K}$ through the traversal of a tree and store $\left[f_{t,k}\right]_{k=1}^{K}$ for every leaf node. $f_{t,k}$ is a C-dimensional vector for classification problems (set $C = 1$ for regression problems), and the number of leaf nodes is at most n. Therefore, the space complexity is $O(nKC)$.

While computing feature contribution for a forest, we only need to keep the average of tree contributions on the training instances instead of the nodes of all the trees. Therefore the space complexity is still $O(nKC)$.

While computing feature contribution layer by layer through the cascade forest structure, we only need to keep the feature contribution of the previous layer. Let $|L|$ denote the number of forests in each layer, the space complexity is $O(nKC|L|)$.

5 FEATURE IMPORTANCE FOR DEEP FORESTS

After obtaining the feature contribution for a deep forest, according to Eq. (11), we can compute the MDI feature importance for the deep forest (denoted by DF) as

$$\widehat{\mathrm{MDI}}(k, DF) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \widetilde{f}_{L,k}\left(\boldsymbol{x}_i\right) \cdot y_i, \tag{31}$$

where $\widetilde{f}_{L,k}$ is defined in Eq. (30). We then demonstrate that it is a proper feature importance measure by showing that the sum of our feature importance measure remains unchanged as the classic single-layer forest MDI calculated for the last layer.

Proposition 5.1. *The sum of the estimated MDI feature importance (as in Eq.* (31)*) over all the original features equals the sum of the final-layer MDI directly over the $K + K'$ features (K' denoting the number of new features), i.e.,*

$$\sum_{k=1}^{K} \widehat{\mathrm{MDI}}(k, DF) = \frac{1}{|L|} \sum_{F \in L} \sum_{k=1}^{K+K'} \mathrm{MDI}(k, F). \tag{32}$$

Proof. For a forest F in the final layer,

$$\sum_{k=1}^{K} \widehat{\mathrm{MDI}}(k, F) = \sum_{k=1}^{K} \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \widetilde{f}_{F,k}\left(\boldsymbol{x}_i\right) \cdot y_i$$

$$
\begin{aligned}
&= \sum_{k=1}^{K} \frac{1}{|\mathcal{D}|} \sum_{i\in\mathcal{D}} \left(\frac{1}{|F|} \sum_{T\in F} \widetilde{f}_{T,k}(\boldsymbol{x}_i) \right) \cdot y_i \\
&= \sum_{k=1}^{K} \frac{1}{|\mathcal{D}|} \sum_{i\in\mathcal{D}} \left(\frac{1}{|F|} \sum_{T\in F} \left(\sum_{j\in T: s(t_{j-1})=k} \Delta\mu(t_j) + \sum_{k'=K+1}^{K+K'} \sum_{j\in T: s(t_{j-1})=k'} \Delta\widetilde{\mu}(t_j,k) \right) \right) \cdot y_i \\
&= \sum_{k=1}^{K} \frac{1}{|\mathcal{D}|} \sum_{i\in\mathcal{D}} \left(\frac{1}{|F|} \sum_{T\in F} \sum_{j\in T: s(t_{j-1})=k} \Delta\mu(t_j) \right) \cdot y_i + \frac{1}{|\mathcal{D}|} \sum_{i\in\mathcal{D}} \left(\frac{1}{|F|} \sum_{T\in F} \sum_{k'=K+1}^{K+K'} \sum_{j\in T: s(t_{j-1})=k'} \Delta\mu(t_j) \right) \cdot y_i \\
&= \sum_{k=1}^{K} \frac{1}{|\mathcal{D}|} \sum_{i\in\mathcal{D}} \left(\frac{1}{|F|} \sum_{T\in F} f_{T,k}(\boldsymbol{x}_i) \right) \cdot y_i + \sum_{k'=K+1}^{K+K'} \frac{1}{|\mathcal{D}|} \sum_{i\in\mathcal{D}} \left(\frac{1}{|F|} \sum_{T\in F} f_{T,k'}(\boldsymbol{x}_i) \right) \cdot y_i \\
&= \sum_{k=1}^{K} \mathrm{MDI}(k,F) + \sum_{k'=K+1}^{K+K'} \mathrm{MDI}(k',F) \\
&= \sum_{k=1}^{K+K'} \mathrm{MDI}(k,F) \ . \qquad (33)
\end{aligned}
$$

Taking the averaging over all forests in the last layer,

$$
\sum_{k=1}^{K} \widehat{\mathrm{MDI}}(k,DF) = \frac{1}{|L|} \sum_{F\in L} \sum_{k=1}^{K} \widehat{\mathrm{MDI}}(k,F) = \frac{1}{|L|} \sum_{F\in L} \sum_{k=1}^{K+K'} \mathrm{MDI}(k,F) \ .
$$

□

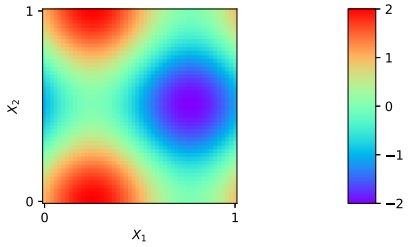
Proposition 5.1 shows that the proposed method is a way of disassembling the importance of the new feature to the original features, thus meeting the properties suggested in Section 3.4 that a proper MDI feature importance should have.

6 EXPERIMENTS

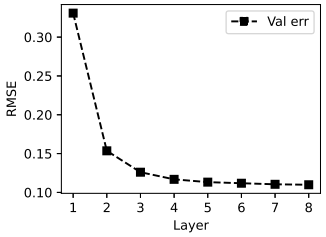
We first show that our computation methods yield reasonable results for feature contribution and feature importance of deep forest in Section 6.1 and Section 6.2. We generate simulated datasets with clear mechanisms behind them, and we conduct experiments on both regression and classification cases. Then in Section 6.3, we compare the quality of our deep forest MDI to other feature importance measures based on its ability to identify relevant features, showing that as deep forest is more powerful than random forest, it also has better estimation of feature importance. In Section 6.4, we further compare the quality of our deep forest MDI method equipped with different calibration methods, showing that partial additive calibration is the most suitable choice. In order to facilitate the understanding of possible applications in real-world tasks, in Section 6.5, a real-world bike-sharing dataset is taken as an example to show the calculation results of feature contribution and feature importance.

6.1 Illustration of Feature Contribution for deep forest

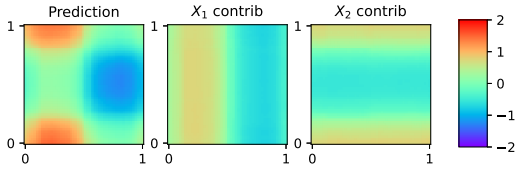
In this subsection, we illustrate our computation of feature contribution for deep forest. With the synthetic datasets, we are able to check whether the calculated feature contributions match the underlying data-generating process. We provide examples for regression and classification respectively. Although technically classification and regression differ only in terms of output, in terms of visualization, the regression example can show richer textures on the feature space while the classification example shows more output dimensions. We first report the



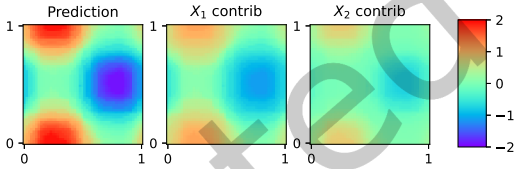
(a) Data generating function.



(b) Decreasing validation error in the training process.



(c) Prediction and feature contributions in the first layer.



(d) Prediction and feature contributions in the last layer.

Fig. 3. Feature contributions for the regression problem. As shown in (b) and the prediction figures in (c) (d), the predictive performance improves from the first layer to the last layer. The feature contributions show that the model captures more details in both input features.

changes in the deep forest's test error layer by layer. Then visualize the feature contributions in the first layer and last layer respectively.

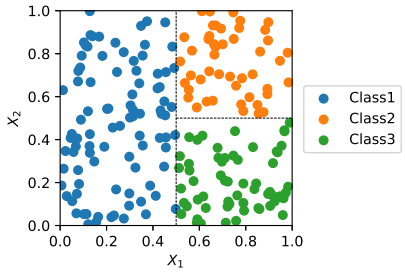
In this experiment, each layer of deep forest has 4 forests, and each forest contains 50 trees, with their maximum depth fixed to 5. The number of cascade layers of deep forest is determined automatically by the performance on the validation set.

Regression. 2000 training samples and 2500 test samples are generated according to the function

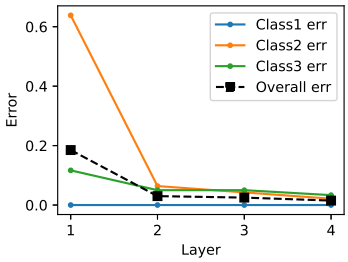
$$f(\boldsymbol{x}) = \sin 2\pi x_1 + \cos 2\pi x_2 \tag{34}$$

and Figure 3(a) is a heatmap visualization of it. The decreasing validation error in Figure 3(b) shows an improvement in performance layer by layer. Figure 3(c) and Figure 3(d) decompose the predictions in the first layer and the last layer to the contribution plots of two input features respectively. It is easy to observe that the first layer's prediction lacks details in the peaks and valleys while the final layer's prediction is much closer to the data generating function. We can also observe corresponding refinements of feature contributions. Note that we can see feature interaction in deep layers. In Figure 3(c) the changes in feature contribution occur only along the considered feature. But in Figure 3(d), we can observe the influence of the other feature. Though this may not honestly recover the data-generating process, it helps deep forest boost performance.

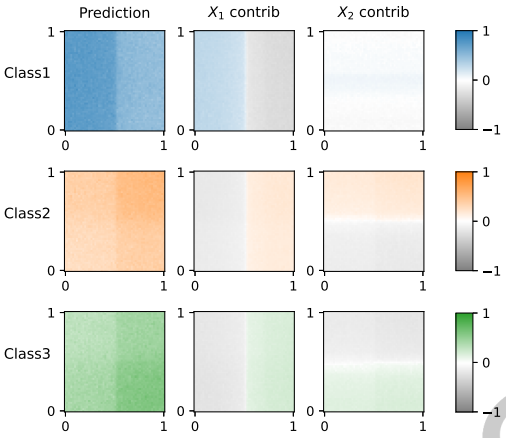
Classification. We generate a 3-class classification problem. Figure 4(a) shows the training data in the first two dimensions. We add another 100 dimensions of irrelevant features uniformly valued between 0 and 1 to simulate background noise. There are 200 training samples and 2500 test samples. With Figure 4(b) confirming that the cascade forest structure improves performance layer by layer, Figure 4(c) and Figure 4(d) visualize the feature contribution in the first layer and the last layer to unveil what extra information the model has learned in the cascade layers. We plot for the three classes separately, using shades of the corresponding colors to indicate



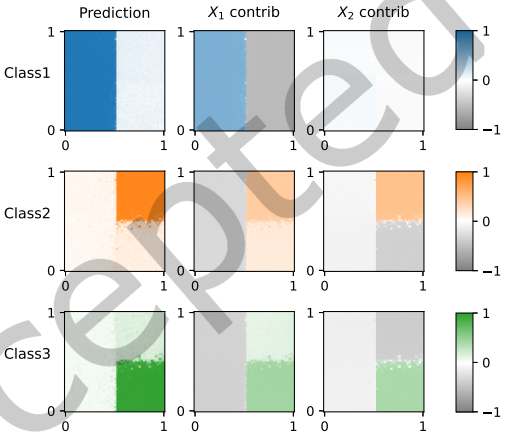
(a) Training data shown in first two dimensions.



(b) Decreasing validation error in the training process.



(c) Prediction and feature contributions in the first layer.



(d) Prediction and feature contributions in the last layer.

Fig. 4. Feature contributions for the 3-class classification problem. As shown in (b) and the prediction figures in (c) (d), the predictive performance improves from the first layer to the last layer. The learned feature contributions are coarse and ambiguous in (c), while clear and precise in (d). The colored areas in the contribution plots indicate an increase in the probabilities of data in these areas belonging to the corresponding class in the areas, and the gray areas indicate a decrease in probability.

the predictive probabilities. Then there are two contribution plots for X_1 and X_2 respectively. The colored areas indicate the corresponding feature enlarges the probabilities of data in these areas belonging to the corresponding class, and the gray areas versa.

We can see that the first layer's feature contribution is vague and of light color, while the last layer's feature contribution is of darker color, leading to clearer peak probabilities in the prediction. The refinement of feature contribution matches the improvement of performance, and also better captures the underlying data generating scheme. We can also observe that deep forest learns feature interaction in deep layers. Different from the simulated regression data, there actually is feature interaction in generating the classification data. However, in the first layer, the change of feature contribution is mainly along the considered feature. But in the last layer, the influence of the other feature emerges in the plots, showing that the cascade structure enables deep forest to learn details on the two relevant features.

Given the last layer's feature contribution, we can easily tell how the deep forest makes predictions. For example, the point $(0, 1)$ has the highest predicted probability of belonging to Class 1, and the contribution to

$\boldsymbol{x}$	μ_0	$\widetilde{f}_{L,1}(\boldsymbol{x})$	$\widetilde{f}_{L,2}(\boldsymbol{x})$	$f_L(\boldsymbol{x})$
$(0, 1)$	$[\mathbf{0.46}, 0.24, 0.30]$	$[\mathbf{0.37}, -0.14, -0.16]$	$[\mathbf{0.00}, 0.00, -0.06]$	$[\mathbf{0.88}, 0.08, 0.04]$
$(1, 1)$	$[0.46, \mathbf{0.24}, 0.30]$	$[-0.36, \mathbf{0.24}, 0.03]$	$[0.00, \mathbf{0.28}, -0.21]$	$[0.05, \mathbf{0.88}, 0.07]$

Table 2. Example of feature contributions of two sample points for the 3-class classification problem. Only values related to the two relevant dimensions are presented. The values associated with the predicted class are shown in bold.

Class 1's probability is mainly due to X_1. The point $(1, 1)$ has the highest predicted probability of belonging to Class 2, and the contribution comes both from X_1 and X_2. Table 2 shows the detailed values of decomposing the predicted probabilities of these two points into feature contributions. Only the contributions of the first two features are reported, and the other 100 irrelevant features also contribute to the prediction, but their contributions are small.

6.2 Illustration of Feature Importance for deep forest

In this section, we illustrate the computation of feature importance for deep forests. We use synthetic regression and classification datasets so that we can tell the true relative importance of each feature. The hyper-parameter settings remain the same as in the previous subsection.

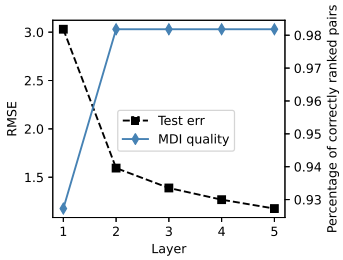
Regression. Because MDI feature importance is a summary of the impact of each feature on the entire dataset, in order to demonstrate the effect of MDI, we need a dataset with more features and different overall importance. Therefore, we generate a regression problem with different coefficients for each feature and no interaction between features so that we can easily check whether the feature importance result is reasonable. We generate 1000 training and test samples each according to the data generation function

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} k x_k \ . \tag{35}$$

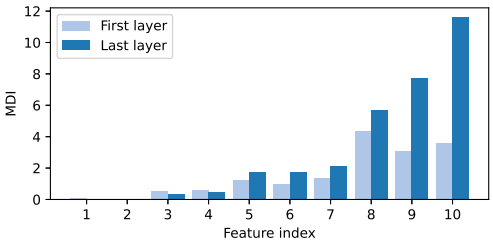
We measure the MDI quality using the percentage of correctly ranked pairs by the estimated importance. Figure 5(a) shows that as the test error decreases layer by layer, the ranking quality of relative MDI feature importance increases. As we would expect, deep forest learns more and improves performance through the cascade structure, so MDI would be more accurate in the deep layers. Figure 5(b) directly compares the estimated MDI in the first layer and the last layer. We can see a clear refinement of the estimated MDI towards the underlying data-generating mechanism.

Classification. We use the same 3-class classification problem as in Figure 4. Figure 6(a) shows the change of global MDI with respect to layers. We can see that

- The global MDI of X_1 and X_2 both increase layer by layer, indicating deep forest gradually mines more information from the relevant features.
- X_1 has a higher MDI than X_2, which matches the data generating process, since using X_1 alone can separate all the instances from Class 1 which amounts to half of the data distribution.
- The average MDI of the 100 irrelevant features is always close to zero, indicating that the model has a good ability to distinguish relevant features from irrelevant features.

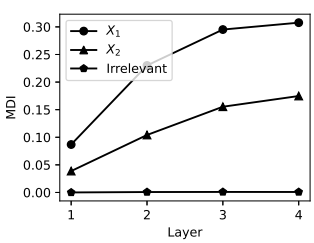


(a) Decreasing test error and increasing MDI quality measured by the percentage of correctly ranked pairs.

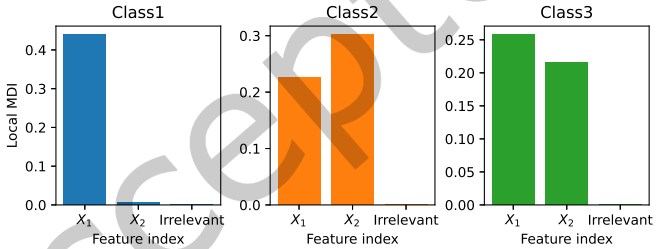


(b) Comparison of MDI of the first layer and the last layer in deep forest. The MDI in the last layer has an obvious refinement compared with the result in the first layer.

Fig. 5. With the growing of layers in deep forest, the estimated MDI captures the underlying importance of features more accurately.



(a) MDI of the two relevant features and the average MDI of the irrelevant features with increasing layers.



(b) Local MDI in deep forest model with respect to each class. X_1 and X_2 play different roles in different classes. The average MDI for irrelevant features is always zero.

Fig. 6. Global MDI and local MDI for the 3-class problem.

Figure 6(b) shows the local MDI calculated for each class. Since Eq. (10) enables us to compute feature importance by summing over instances, we can calculate the local feature importance for each class in this problem,

$$\widehat{\mathrm{MDI}}_c(k, DF) = \frac{1}{|\{i : y_i = c\}|} \sum_{i:y_i=c} \widetilde{f}_{L,k}(\boldsymbol{x}_i) \cdot [\mathbb{I}(y_i = 1), \mathbb{I}(y_i = 2), \mathbb{I}(y_i = 3)], \ c = 1, 2, 3.$$

From Figure 6(b) we can observe that, for Class 1, only feature X_1 matters, and feature X_2 has very low feature importance, while for Class 2 and Class 3, both X_1 and X_2 play very important roles. The above observation clearly matches the data-generating mechanism.

6.3 Comparison of Feature Importance Ranking Quality

In this subsection, we aim to show the competitiveness of our methods. We use a commonly used quantitative measure of feature importance, the ranking quality of relevant features over irrelevant features. Note that there is currently no quantitative performance measure for feature contribution. The effectiveness of our feature contribution is revealed by the superiority of our MDI measure, since our MDI is calculated based on the feature contribution result. Besides, we will show an example of feature contribution in a real-world application task in Section 6.5 to further demonstrate its effectiveness.

Table 3. Dataset Information. We also include the test error (mean±std.) of random forest and deep forest. An entry is marked with a bullet '•' if deep forest is significantly better than random forest based on the Wilcoxon rank-sum test with confidence level 0.1. For each dataset, the entry with the lowest average error is bolded.

Dataset	Number of classes	Number of features (relevant)	Number of training instances	Test error of random forest	Test error of deep forest
abalone	regression	8	417	2.432±0.026	**2.425±0.041**
cpusmall	regression	12	819	3.578±0.046•	**3.420±0.039**
phishing	2	68	829	0.069±0.006	**0.067±0.009**
satimage	6	36	310	0.155±0.005	**0.152±0.007**
pendigits	10	16	749	0.029±0.003•	**0.026±0.001**
usps	10	256	729	0.087±0.006•	**0.074±0.003**

Datasets. We use six benchmark datasets processed to contain irrelevant features. We choose two regression datasets and four classification datasets with the number of classes ranging from 2 to 10. The number of features ranges from 8 to 256. The dataset information is listed in Table 3. The benchmark datasets are processed in two steps. First, we copy a dataset's feature matrix and randomly permute the values in the copied columns, then concatenate the processed copy to the original feature matrix. In this way, the whole feature set becomes half relevant and half irrelevant. Second, since identifying irrelevant features is a relatively easier task compared with prediction, we reduce the number of training samples to avoid a situation where all the methods are equally perfect. We use only 10% of the data as training samples for all the datasets. We also hold a separate validation set for MDA-based methods, whose number of samples is the same as the training set.

Compared methods. We design the first feature contribution and MDI feature importance measure for deep forest. The only quantitative feature importance measure comparable is MDA, since MDA is applicable to any black-box models. We also train a random forest and compare our method with several random forest importance measures. Our aim is to show that as deep forest is more powerful than RF, its feature importance result is also better than RF. If so, this comparison result will also contribute to the validity of our method. In detail, we compare our MDI feature importance for deep forest, named **MDI(DF)**, to the following methods, where 'DF' and 'RF' in parentheses indicate whether this is for a trained deep forest or random forest.

- **MDI(RF)**: the mean decrease of impurity by splitting on the given feature during the training process of random forest [7]. But it tends to overestimate the feature importance of irrelevant features [17].
- **MDI-oob(RF)**: a debiased version of MDI using out-of-bag samples for random forest [17].
- **MDA(RF)**: the mean decrease in accuracy of a trained random forest caused by randomly permuting the values of the given feature.
- **MDA(DF)**: the mean decrease in accuracy of a trained deep forest caused by randomly permuting the values of the given feature.

In each layer of deep forest, there are still 4 forests, each with 50 trees in it. Accordingly, we grow 200 trees in random forest. Since we are dealing with more complicated datasets, the maximum depth of trees in deep forest is fixed to 8, and so are the trees in random forests.

Table 4 reports the performance of the compared methods for identifying relevant features. We can see that MDI(DF) achieves the best average performance on all the datasets, and it is significantly better than MDA(DF) on five out of six datasets. Among the random forest feature importance methods, MDI-oob(RF) achieves the best performance, which coincides with previous studies [17]. However, it still has a lower average AUC than MDI(DF) on five datasets and significantly worse performance on two datasets.

	MDI(RF)	MDI-oob(RF)	MDA(RF)	MDA(DF)	MDI(DF)
abalone	0.911±0.047●	0.984±0.031	0.991±0.013●	**1.000±0.000**	**1.000±0.000**
cpusmall	0.932±0.009●	0.999±0.003	0.971±0.016●	0.961±0.020●	**1.000±0.000**
phishing	0.650±0.026●	0.908±0.027●	0.598±0.090●	0.611±0.071●	**0.926±0.025**
satimage	0.998±0.003●	1.000±0.001	0.689±0.134●	0.593±0.144●	**1.000±0.000**
pendigits	**1.000±0.000**	**1.000±0.000**	0.985±0.028●	0.989±0.019●	**1.000±0.000**
usps	0.998±0.001●	0.995±0.003●	0.586±0.103●	0.581±0.158●	**1.000±0.000**

Table 4. AUC scores for relevant feature identification (mean±std.) of the compared methods of 10 runs. An entry is marked with a bullet '●' if MDI(DF) is significantly better than the corresponding method based on the Wilcoxon rank-sum test with confidence level 0.1. For each dataset, the entry with the highest average AUC is bolded.

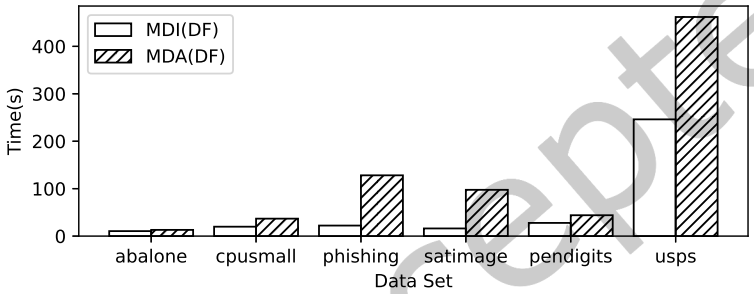


Fig. 7. Running time comparison of our MDI method to that of MDA for deep forest. Each experiment is repeated 10 times and the average running time is reported.

We also report the running time of MDI(DF). In Figure 7, we show that MDI(DF) requires less running time than MDA(DF). The reported running time includes model training and feature importance computation. MDI(DF) can compute feature contribution and feature importance during the training process, which leads to only a mild increase in training time. In contrast, MDA(DF) requires the test process to be executed for each feature separately, and usually multiple times to obtain a reliable estimation of the accuracy decrease.

6.4 Comparison of Different Calibration Methods

To evaluate the effectiveness of using partial additive calibration as Eq. (26) in the step of calculating feature contribution, we compare MDI(DF) to that using naive multiplicative calibration as Eq. (23), denoted by **MDI(DF)***, and that using naive multiplicative calibration as Eq. (24), denoted by **MDI(DF)**$^+$. The datasets and other settings remain the same as in the previous section. And we still use feature importance ranking quality as the evaluation criterion.

Table 5 shows that MDI(DF), which uses partial additive calibration, achieves the best average AUC on all the datasets. The performance of MDI(DF)$^+$, which uses naive additive calibration, follows closely behind. It achieves the best average AUC on all but one dataset. MDI(DF)*, which uses naive multiplicative calibration, achieves the best average AUC on two datasets, while is significantly worse than MDI(DF) on three datasts. This shows that the multiplicative calibration is indeed unstable. The comparison results justify our design of the calibration method.

	MDI(DF)*	MDI(DF)+	MDI(DF)
abalone	0.853±0.181●	**1.000±0.000**	**1.000±0.000**
cpusmall	**1.000±0.000**	**1.000±0.000**	**1.000±0.000**
phishing	**0.926±0.026**	0.917±0.015	**0.926±0.025**
satimage	0.948±0.107●	**1.000±0.000**	**1.000±0.000**
pendigits	0.987±0.029	**1.000±0.000**	**1.000±0.000**
usps	0.977±0.047●	**1.000±0.000**	**1.000±0.000**

Table 5. AUC scores for relevant feature identification (mean±std.) of different calibration methods of 10 runs. An entry is marked with a bullet '●' if MDI(DF) is significantly better than the corresponding method based on the Wilcoxon rank-sum test with confidence level 0.1. For each dataset, the entry with the highest average AUC is bolded.

6.5 Application on Bike Sharing Task

This section does not aim to compare, instead, it provides an example with a real-world bike-sharing task [11], showing how our deep forest interpretation tools can be used in an application.

The bike-sharing dataset contains the records of daily bike rental count and the environmental and seasonal information. For illustration purposes, we process the dataset and only keep six features. The data are randomly partitioned into training and test sets, with 487 days of records used as the training set, and 244 days of records used as the test set.

Table 6 shows the feature contribution of a trained deep forest on two days in the test set. This is a regression task, so the feature contributions are scalars. We can see that the platform was probably doing better in 2012

record		Year	isWorkingDay	isClearDay	Temperature	Humidity	WindSpeed
2011/12/7	feature value	2011	1	0	17°C	97%	18 km/h
	feature contribution	-714	24	-104	-1479	-244	-136
2012/9/21	feature value	2012	1	1	25°C	67%	10 km/h
	feature contribution	1031	44	252	1117	145	74

Table 6. Example of feature contribution of deep forest in the bike sharing task. Positive and negative contributions are based on the training label mean.

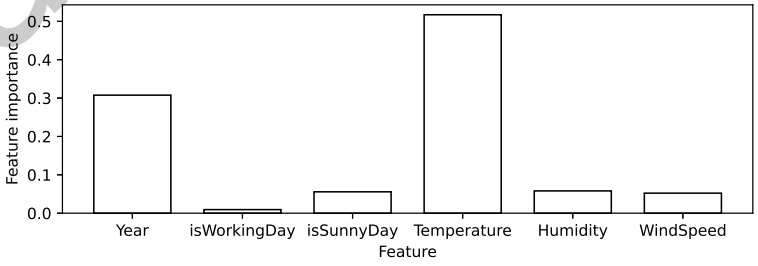


Fig. 8. MDI feature importance of deep forest in the bike sharing task. The sum of feature importance is normalized to 1.

than it did in 2011, at least in terms of the year's impact on these two records. In addition, the low temperature and the rainy and windy weather on 2011/12/7 both contributed to a decrease in the bike rental count. The mild temperature and nice weather have led to an increase in bike rental count on 2012/9/21. Figure 8 shows the MDI feature importance of deep forest. We can see that temperature is the most important feature. In addition, the year is also an important feature, which shows that the platform has improved a lot in two years. Other weather conditions also play a role, but whether it is a working day has little impact on the bike rental count. The explanatory results help the user understand how the deep forest model works, enabling more reliable deployment of the model and helping to find out why a prediction is unexpected.

7 CONCLUSION

In this paper, we propose two interpretation tools for deep forests, namely feature contribution and MDI feature importance. The former decomposes every single prediction of deep forest into the contribution of each original feature. The latter characterizes the overall impact of a feature in the whole model. Our proposed methods yield proper feature contribution and MDI feature importance measures. Experiments also validate the effectiveness of our methods. We believe that the interpreting tools we provide can further promote the application of deep forests, as well as deepen our understanding of the data and model.

ACKNOWLEDGMENTS

This research was supported by the National Natural Science Foundation of China (62176117,62306104), Jiangsu Science Foundation (BK20230949), China Postdoctoral Science Foundation (2023TQ0104), Jiangsu Excellent Postdoctoral Program (2023ZB140) and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

[1] Ludovic Arnould, Claire Boyer, and Erwan Scornet. 2021. Analyzing the tree-layer structure of deep forests. In *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139. 342–350.
[2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion* 58 (2020), 82–115.
[3] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J Christopher Westland. 2011. Data mining for credit card fraud: A comparative study. *Decision support systems* 50, 3 (2011), 602–613.
[4] Gérard Biau, Luc Devroye, and Gäbor Lugosi. 2008. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research* 9, 9 (2008).
[5] Yaakoub Boualleg, Mohamed Farah, and Imed Riadh Farah. 2019. Remote sensing scene classification using convolutional features and deep forest classifier. *IEEE Geoscience and Remote Sensing Letters* 16, 12 (2019), 1944–1948.
[6] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[7] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 1984. *Classification and Regression Trees*. Boca Raton, FL: Chapman and Hall/CRC.
[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 785–794.
[9] Yi-He Chen, Shen-Huan Lyu, and Yuan Jiang. 2021. Improving deep forest by exploiting high-order interactions. In *Proceedings of the 21st IEEE International Conference on Data Mining*. 1036–1041.
[10] Adele Cutler, D. Richard Cutler, and John R. Stevens. 2012. Random forests. In *Ensemble Machine Learning: Methods and Applications*. 157–175.
[11] Hadi Fanaee-T and Joao Gama. 2013. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence* (2013), 1–15.
[12] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* 29, 5 (2001), 1189–1232.
[13] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning* 63, 1 (2006), 3–42.
[14] Silke Janitza, Ender Celik, and Anne-Laure Boulesteix. 2018. A computationally fast variable importance test for random forests for high-dimensional data. *Advances in Data Analysis and Classification* 12, 4 (2018), 885–915.

[15] Sangwon Kim, Mira Jeong, and Byoung Chul Ko. 2020. Interpretation and Simplification of Deep Forest. *arXiv preprint arXiv:2001.04721* (2020).
[16] Victor E Kuz'min, Pavel G Polishchuk, Anatoly G Artemenko, and Sergey A Andronati. 2011. Interpretation of QSAR models based on random forest methods. *Molecular informatics* 30, 6-7 (2011), 593–603.
[17] Xiao Li, Yu Wang, Sumanta Basu, Karl Kumbier, and Bin Yu. 2019. A debiased MDI feature importance measure for random forests. *Advances in Neural Information Processing Systems 32* (2019).
[18] Shen-Huan Lyu, Yi-He Chen, and Zhi-Hua Zhou. 2022. A region-based analysis for the feature concatenation in deep forests. *Chinese Journal of Electronics* 31, 6 (2022), 1072–1080.
[19] Shen-Huan Lyu, Yi-Xiao He, and Zhi-Hua Zhou. 2022. Depth is more powerful than width with prediction concatenation in deep forest. In *Advances in Neural Information Processing Systems 35*. 29719–29732.
[20] Shen-Huan Lyu, Liang Yang, and Zhi-Hua Zhou. 2019. A refined margin distribution analysis for forest representation learning. In *Advances in Neural Information Processing Systems 32*. 5530–5540.
[21] Pengfei Ma, Youxi Wu, Yan Li, Lei Guo, He Jiang, Xingquan Zhu, and Xindong Wu. 2022. HW-Forest: Deep forest with hashing screening and window screening. *ACM Transactions on Knowledge Discovery from Data* (2022).
[22] Kristin K Nicodemus. 2011. On the stability and ranking of predictors from random forest variable importance measures. *Briefings in Bioinformatics* 12, 4 (2011), 369–373.
[23] Kristin K Nicodemus and James D Malley. 2009. Predictor correlation impacts machine learning algorithms: Implications for genomic studies. *Bioinformatics* 25, 15 (2009), 1884–1890.
[24] Anna Palczewska, Jan Palczewski, Richard Marchese Robinson, and Daniel Neagu. 2013. Interpreting random forest classification models using a feature contribution method. In *Integration of Reusable Systems*. 193–218.
[25] Ming Pang, Kai-Ming Ting, Peng Zhao, and Zhi-Hua Zhou. 2018. Improving deep forest by confidence screening. In *Proceeding of the 18th IEEE International Conference on Data Mining*. 1194–1199.
[26] Ming Pang, Kai Ming Ting, Peng Zhao, and Zhi-Hua Zhou. 2022. Improving deep forest by screening. *IEEE Transactions on Knowledge and Data Engineering* 34, 9 (2022), 4298–4312.
[27] Ando Saabas. 2014. *Interpreting random forests*. Retrieved May 25, 2022 from https://blog.datadive.net/interpreting-random-forests/
[28] Robert E Schapire and Yoav Freund. 2012. *Boosting: Foundations and Algorithms*. MIT Press, London.
[29] Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. 2015. Consistency of random forests. *Annals of Statistics* 43, 4 (2015), 1716–1741.
[30] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. Conditional variable importance for random forests. *BMC Bioinformatics* 9, 1 (2008), 1–11.
[31] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. 2007. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics* 8, 1 (2007), 1–21.
[32] Carolin Strobl and Achim Zeileis. 2008. Danger: High power exploring the statistical properties of a test for random forest variable importance.
[33] Ran Su, Xinyi Liu, Leyi Wei, and Quan Zou. 2019. Deep-Resp-Forest: A deep forest model to predict anti-cancer drug response. *Methods* 166 (2019), 91–102.
[34] Lev V. Utkin and Mikhail A. Ryabinin. 2018. A Siamese deep forest. *Knowledge-Based Systems* 139 (2018), 13–22.
[35] Lev V. Utkin and Mikhail A. Ryabinin. 2019. Discriminative metric learning with deep forest. *International Journal on Artificial Intelligence Tools* 28, 2 (2019), 1950007:1–1950007:19.
[36] Qian-Wei Wang, Liang Yang, and Yu-Feng Li. 2020. Learning from weak-label data: A deep forest expedition. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, Vol. 34. 6251–6258.
[37] Liang Yang, Xi-Zhu Wu, Yuan Jiang, and Zhi-Hua Zhou. 2020. Multi-label learning with deep forest. In *Proceedings of the 24th European Conference on Artificial Intelligence*, Vol. 325. 1634–1641.
[38] Yiming Zhang, Ying Weng, and Jonathan Lund. 2022. Applications of explainable artificial intelligence in diagnosis and surgery. *Diagnostics* 12, 2 (2022), 237.
[39] Ya-Lin Zhang, Jun Zhou, Wenhao Zheng, Ji Feng, Longfei Li, Ziqi Liu, Ming Li, Zhiqiang Zhang, Chaochao Chen, Xiaolong Li, Yuan (Alan) Qi, and Zhi-Hua Zhou. 2019. Distributed deep forest and its application to automatic detection of cash-out fraud. *ACM Transactions on Intelligent Systems and Technology* 10, 5 (2019), 1–19.
[40] Meng Zhou, Xianhua Zeng, and Aozhu Chen. 2019. Deep forest hashing for image retrieval. *Pattern Recognition* 95 (2019), 114–127.
[41] Zhengze Zhou and Giles Hooker. 2021. Unbiased measurement of feature importance in tree-based methods. *ACM Transactions on Knowledge Discovery from Data* 15, 2 (2021), 1–21.
[42] Zhi-Hua Zhou and Ji Feng. 2017. Deep forest: Towards an alternative to deep neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 3553–3559.
[43] Zhi-Hua Zhou and Ji Feng. 2019. Deep forest. *National Science Review* 6, 1 (2019), 74–86.