Improving Deep Forest by Exploiting High-order Interactions

Yi-He Chen*, Shen-Huan Lyu*, Yuan Jiang†
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
{chenyh, lvsh, jiangy}@lamda.nju.edu.cn

2021 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-2398-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDM51629.2021.00118

***Abstract*—Recent studies on deep forests have shown that deep learning frameworks can be built on non-differentiable modules without a backpropagation training process. However, the feature representations of deep forests only consist of predicted class probabilities. The information these class probabilities deliver is very limited and lacks diversity, especially when the number of output labels is far less than the number of input features. Besides, the prediction-based representations require us to save multiple layers of random forests to use them during testing, which is high-memory and high-time cost. In this paper, we propose a novel deep forest model that utilizes high-order interactions of input features to generate more informative and diverse feature representations. Specifically, we design a generalized version of Random Intersection Trees (gRIT) to discover stable high-order interactions and apply Activated Linear Combination (ALC) to transform them into hierarchical distributed representations. These interaction-based representations obviate the need to store random forests in the front layers, thus greatly improving the computational efficiency. Our experiments show that our method achieves highly competitive predictive performance with significantly reduced time and memory cost.**

I. Introduction

In recent years, deep forests (DFs) have achieved state-of-the-art performance on the categorical and mixed modeling tasks, e.g., financial application [1], medical application [2], [3] and geoscience [4], while deep neural networks (DNNs) dominate the numerical tasks, e.g., computer vision (CV) [5], automatic speech recognition (ASR) [6] and natural language processing (NLP) [7]. Although the recent development of deep neural networks on numerical data is rather attractive, more real-world problems belong to categorical and mixed modeling tasks. In such tasks, there are a large number of attributes from which we can only get qualitative characteristics rather than quantitative characteristics. Therefore, applying and exploring deep forests and other non-neural network deep models to a wide range of tasks is an important direction for the future of deep learning [8].

By summarizing the key ingredients of deep learning may lie in: *layer-by-layer processing*, *sufficient model complexity*, and *in-model feature transformation*, Zhou and Feng [9] proposed gcForest, the first non-NN-style deep models with these characteristics. Essentially, gcForest is a decision tree ensemble that employs a cascade structure to do representation learning. In this cascade structure, each level consists of an ensemble of decision tree forests, i.e. an ensemble of ensembles. Each level receives feature information processed by the preceding level and outputs estimated class probabilities which are then concatenated with the original feature vector as the input to the next level. Lyu et al. [10] provided a detailed theoretical analysis that proved deep forests have sufficient model complexity with enough depth, and the cascaded structure boosts the feature representations layer by layer instead of the predictions.

Although gcForest has shown its great potential both empirically and theoretically, we argue that the prediction-based feature representation of gcForest is a critical deficiency. Firstly, as stated by original authors [9], the predicted class probabilities deliver very limited information. A majority of tasks have much more input features than output labels, which causes the information in predictions to likely be drowned out by original features when concatenating them together. Secondly, since decision tree forests are already pretty stable predictors, an ensemble of different forests may result in similar predictions, which causes feature representations to be redundant and lack diversity. Thirdly, the prediction-based representations rely on the storing of multi-layered forest models to do prediction level-by-level during testing, thus requiring a large amount of memory and time consumption. Therefore, it is necessary to design more informative feature representations with less computational cost for deep forests.

In this paper, we propose a novel deep forest model named high-order interaction Deep Forest (hiDF), which leverages stable high-order interactions of input features to generate informative and diverse feature representations. Specifically, we design a generalized version of Random Intersection Trees (gRIT) to discover stable high-order interactions and apply Activated Linear Combination (ALC) to transform these interactions into new feature representations, which can interact with input features across multiple layers. In such iterations, hiDF can effectively mine high-order interactions between input features and utilize them to improve predictive performance.

Our contributions are twofold. First, the proposed hiDF method firstly provides hierarchical distributed representations from low-order to high-order interactions for deep forests, greatly enhancing the effectiveness and diversity of feature representations. Second, these representations obviate the need to store random forests in the front layers, reducing the time cost and memory requirement by one order of magnitude.

*: These two authors contributed equally.
†: Yuan Jiang is the corresponding author.

978-1-6654-2398-4/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDM51629.2021.00118

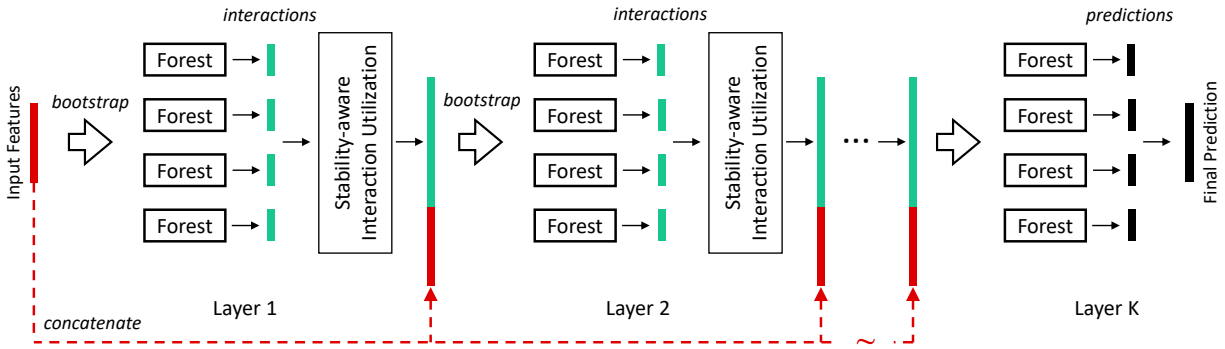


Fig. 1. The framework of High-order Interaction Deep Forest (hiDF).

II. High-order Interaction Deep Forest

In this section, we provide a detailed description of our hiDF algorithm. The framework of hiDF is illustrated in Figure 1. It employs a multi-layered structure to do representation learning, each layer consists of an ensemble of decision tree forests as DF. The most significant difference between DF and hiDF is that DF uses the prediction of each layer as representations while hiDF employs high-order feature interactions as representations. hiDF utilizes both feature interactions and multi-layered structure to construct multiple levels of feature representations and learn a hierarchy of explanatory factors internally. This deep structure promotes the *re-use* of features [11] and the ultimate goal is to form hierarchical distributed representations of data, which is useful to our learning task.

Specifically, hiDF utilizes feature interactions mainly in three steps. First, given an ensemble of random forests fitted on the input features, numerous decision rules are extracted from these decision trees. These decision rules are then processed by generalized Random Intersection Trees (gRIT) to identify prevalent feature interactions. Second, an outer bagging procedure performs the previous step on bootstrap samples. This extra perturbation of the data allows us to do "interaction selection" by assessing the stability associated with the identified interactions. Only those feature interactions with high stability scores are utilized through Activated Linear Combinations (ALC) to generate new feature representations. Finally, a metric-aware mechanism is applied to grow layers adaptively to reduce the risk of overfitting. We will discuss these three steps in more detail in the following subsections.

A. Extract feature interactions through gRIT and ERF

We identify feature interaction as a collection of conditions inside a decision rule, a decision rule is of the simple form:

$$\text{IF: } condition_1 \,\&\ldots\&\, condition_i, \text{ THEN: } response. \quad (1)$$

Accordingly, a feature interaction can be described as $condition_1 \& \ldots \& condition_i$, which is the premise of a decision rule. The conditions are based on input variables. For a continuous variable x_i, it could take the form: $x_i \geq t_i$, where t_i is a determined threshold. For categorical variables, we transform them using one-hot encoding so we can get similar forms. In this work, we only consider the premise of a decision rule regardless of the response. To represent decision rules succinctly, we use signed feature index $\pm i$ with threshold t_i for describing a condition, in which a positive index means $x_i \geq t_i$ and a negative index means $x_i < t_i$. Following this notation, a decision rule $\mathbb{R}$ of i conditions is represented by $\mathbb{I}(\mathbb{R}) = \{\mathbb{I}^{index}, \mathbb{I}^{threshold}\} = \{(x_1, \ldots, x_i), (t_1, \ldots, t_i)\}$, where $\mathbb{I}^{index}$ is a vector of i signed feature indices and $\mathbb{I}^{threshold}$ represents associated thresholds.

Since decision rules can be encoded by the decision paths from root nodes to leaf nodes, we use enriched random forests (ERFs) [12] to extract decision rules from data. ERFs can be seen as RFs with a soft dimension reduction process, i.e. selecting more informative features in the input space.

Given numerous decision rules detected from ERFs, one can naturally treat them as feature interactions. However, they come with two drawbacks. First, each decision rule corresponds to only a small number of instances, which lacks statistical importance. Second, the decision rules are long and intricate, therefore do not generalize very well. To address these two drawbacks, we apply gRIT to process the detected decision rules. The main idea of gRIT is to "prune" those decision rules with respect to statistical meaning. After gRIT, statistically important feature interactions with better generalization ability can be discovered.

gRIT is summarized in Algorithm 1. A gRIT contains L intersection trees. Inside each intersection tree $\ell \in \{1, \ldots, L\}$, J example indices $\{i_1, \ldots, i_J\}$ are uniformly sampled from data and their corresponding decision rules are represented as $\{\mathbb{I}_{i_1} \ldots \mathbb{I}_{i_J}\}$ using the signed feature index notation mentioned earlier. Then, it performs J-fold intersections $\mathbb{I}_{i_1} \cap \cdots \cap \mathbb{I}_{i_J}$ to keep informative features and prune away noisy features. The main intuition behind this is that if a feature x_i of an interaction $\mathbb{I}$ is sufficiently prevalent, it will survive the intersection with high probability. The original RIT algorithm [13] is restricted to only binary categorical features, so it has a limited application range. Here, our generalized version can deal with both categorical and continuous features with threshold information.

Algorithm 1: generalized Random Intersection Trees

Require: Rule set $\mathcal{R} = \left\{\mathbb{I}_i;\ \mathbb{I}_i = \{\mathbb{I}_i^{index}, \mathbb{I}_i^{threshold}\}\right\}_{i=1}^{n}$
Ensure: $\mathcal{T} = \cup_{\ell=1}^{L}\mathcal{T}_\ell$
for tree ℓ in $\{1, 2, \ldots, L\}$ **do**
  Let ℓ be a tree of depth D. Let J be the total number of nodes in the tree, and index the nodes such that every parent-child pair, larger indices are assigned to the child than the parent. For each node $j = 1, \ldots, J$, denote the parent of node j as $pa(j)$, let i_j be a uniform sample index from the training data and corresponds to node j.
  Set $T_1^{index} \leftarrow \mathbb{I}_{i_1}^{index},\ T_1^{threshold} \leftarrow \mathbb{I}_{i_1}^{threshold}$
  for j in $\{2, \ldots, J\}$ **do**
    $T_j^{index} \leftarrow \mathbb{I}_{i_j}^{index} \cap T_{pa(j)}\ ,\ T_j^{threshold} \leftarrow \emptyset$
    for feature index i in T_j^{index} **do**
      $T_j^{threshold} \leftarrow T_j^{threshold} + \left\{\ \max(t_i, t_i') \text{ if } i < 0, \text{ else } \min(t_i, t_i').\ \ t_i \in \mathbb{I}_{i_j}^{threshold},\ t_i' \in T_{pa(j)}^{threshold}\right\}$
    end for
  end for
  $\mathcal{T}_\ell^{index} \leftarrow \left\{T_j^{index} : depth(j) = D\right\},\ \ \mathcal{T}_\ell^{threshold} \leftarrow \left\{T_j^{threshold} : depth(j) = D\right\}$
  $\mathcal{T}_\ell \leftarrow \{\mathcal{T}_\ell^{index}, \mathcal{T}_\ell^{threshold}\}$
end for

B. Stability-aware interaction utilization

Statistical results should at least be reproducible relative to "reasonable" data and model perturbations [14]. Towards this goal, we assess the stability associated with the identified feature interactions using an extra outer bagging step. Specifically, we generate B bootstrap sample data $\mathcal{D}_{(b)}, b = \{1, \ldots, B\}$, fit B enriched random forests ERF on each sample $\mathcal{D}_{(b)}$. Then, we apply gRIT to discover B sets of feature interactions $\mathcal{T}_{(b)}$ from these samples. We use the stability score defined as (2). It reflects the prevalence an interaction $\mathbb{I} \in \cup_{b=1}^{B}\mathcal{T}_{(b)}$ appears out of B bootstrap samples.

$$stability(\mathbb{I}) = \frac{1}{B}\sum_{b=1}^{B}\mathbb{1}\{\mathbb{I} \in \mathcal{T}_{(b)}\}. \tag{2}$$

Given the discovered feature interactions and their stability scores, we want to retain those interactions with top k scores or with scores bigger than a pre-specified threshold. They represent the most stable feature interactions, therefore lead to better generalization performance. This "interaction selection" procedure retains only a small number of stable interactions and discards a large number of unstable ones, which will ease the memory cost burden significantly.

To utilize the discovered feature interactions, we generate one new feature for each interaction, using Activated Linear Combination (ALC) including weighted sum and nonlinear activation functions, and denote these new features by $\boldsymbol{r}_{new}$. Take the feature interaction $\mathbb{I} = \{\mathbb{I}^{index}, \mathbb{I}^{threshold}\} = \{(-2, 3), (t_2, t_3)\}$ as an example, along with each feature's gini importance $\boldsymbol{w} = (w_2, w_3)$. $\boldsymbol{r}_{new}(\boldsymbol{w},\ \mathbb{I})$ is calculated as:

$$\boldsymbol{r}_{new}(\boldsymbol{w},\ \mathbb{I}) = \sigma(-w_2(x_2 - t_2) + w_3(x_3 - t_3)), \tag{3}$$

where $\sigma(x) = x \cdot \mathbf{1}[x \geq 0]$ is a non-linear activation function. We concatenate the new features with the current input features to form the input feature in the next layer. We will discuss the multi-layered structure in the next subsection.

C. Adaptive layer growth

It is widely acknowledged that depth plays a crucial role in the success of deep learning. We believe that deep structures bring two crucial benefits compared to shallow structures. First, deep structures promote the *re-use* of features, with the ultimate goal is to form hierarchical distributed representations of data, which are beneficial to better generalization performance. Second, shallow layers tend to discover low-order feature interactions while deeper layers tend to discover high-order interactions, high-order interactions reflect on greater importance, thus have more impact on the learning task.

To get these two benefits, hiDF employs a multi-layered structure. Specifically, each layer is an ensemble of random forests, based on which we apply ERFs and gRIT to discover a rich family of feature interactions. Then we generate a new feature vector via the stability-aware interaction utilization mechanism described in the previous subsection. The newly generated feature vector is then concatenated with the input feature vector to form the new feature representations, which serve as the input to the next layer.

After expanding each new layer, hiDF estimates the performance of the current whole structure using a separate validation set. If there is no obvious performance boost, hiDF will terminate the training process. This adaptive layer-wise growing strategy can reduce the risk of overfitting, it also facilitates the model complexity of hiDF to be determined automatically. The overall framework of hiDF is illustrated in Figure 1 and summarized in Algorithm 2.

It is worth mentioning that the meaning of high-order in our hiDF is twofold: (1) hiDF can discover high-order interactions from the original features by gRIT, and (2) through our multi-layered approach, hiDF can detect the interactions between original features and the newly generated features, even among only newly generated features, causing the order of interactions to increase layer by layer.

Algorithm 2: High-order Interaction Deep Forest

Require: Training set S, the maximum number of layers K and number of bootstrap samples B.

Ensure: K-layer deep forest model $\left\{RF, \{S_r^{(k)}\}_{k=1}^{K}\right\}$.

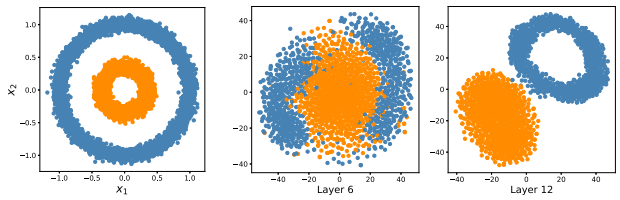
Initialize representation features $\boldsymbol{r}^{(1)} \leftarrow \emptyset$, layer 0 data $S_r^{(0)} = S$, validation error $l_0 = 1$.
for k in $\{1, \ldots, K\}$ **do**
  $S_r^{(k)} \leftarrow concatenate(S_r^{(k-1)}, \boldsymbol{r}^{(k)})$
  Fit RF on $S_r^{(k)}$
  $\boldsymbol{w}^{(k)} \leftarrow$ Gini importance of RF
  for b in $\{1, 2, \ldots, B\}$ **do**
    Generate bootstrap samples $S_{(b)}$ from $S_r^{(k)}$
    Fit an Enriched-$RF(\boldsymbol{w}^{(k)})$ on $S_{(b)}$
    $\mathcal{R}_{(b)} \leftarrow \{\ \mathbb{I}_{i_t} :\ \boldsymbol{x}_i \in S_{(b)},$ falls in leaf node i_t of tree $t\ \}$
    $\mathcal{T}_{(b)}^{(k+1)} \leftarrow \text{gRIT}(\mathcal{R}_{(b)})$
  end for
  Initialize $\boldsymbol{r}^{(k+1)} \leftarrow \emptyset$.
  for $\mathbb{I}^{(k+1)} \in \cup_{b=1}^{B} \mathcal{T}_{(b)}^{(k+1)}$ **do**
    $stability(\mathbb{I}^{(k+1)}) \leftarrow \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\{\mathbb{I}^{(k+1)} \in \mathcal{T}_{(b)}^{(k+1)}\}$
    if $stability(\mathbb{I}^{(k+1)}) > 0.5$ **then**
      $\boldsymbol{r}^{(k+1)} \leftarrow \boldsymbol{r}^{(k+1)} \cup \boldsymbol{r}_{new}\left(\boldsymbol{w}^{(k)}, \mathbb{I}^{(k+1)}\right)$
    end if
  end for
  Compute the cross validation error l_k on $\boldsymbol{r}^{(k+1)}$.
  if $l_k > l_{k-1}$ **then**
    Terminate training and return current model.
  end if
end for
Fit RF on $S_r^{(K+1)}$

III. Experiments

In this section, we conduct several experiments with hiDF on both synthetic data and several widely used benchmark datasets. We first demonstrate that hiDF can obtain informative feature representations using a hierarchical processing structure. We then evaluate the effectiveness of hiDF across several real datasets. The results show that hiDF can achieve the highest classification accuracy on all of these benchmark datasets and it exhibits better performance than deep forest consistently. Meanwhile, hiDF uses significantly less memory space and test time and has a moderate amount of training time compared to deep forest.

A. Synthetic data

First, we use a synthetic dataset to demonstrate that hiDF can learn informative feature representations through its multi-layered feature interaction utilization mechanism. Our synthetic dataset is a binary classification task generated by Gaussian data on $\mathbb{R}^2$ with a spherical decision boundary as illustrated in Figure 2(a). It contains 10,000 examples (80% for training and 20% for testing). We also introduce additional 500 uniform random noise features in the dataset. These uninformative random features make the classification task more complex, thus allowing us to assess the quality of representation features generated by hiDF.



(a) Input features. (b) The t-SNE visualization of features in different layers.

Fig. 2. The quality of feature representations generated by hiDF across different layers on the synthetic data.

The output feature representations from layer 6 and layer 12 of hiDF are illustrated via t-SNE in Figure 2(b). Clearly, hiDF can get informative feature representations from the original input space, and as layers go deeper, data points become linearly separable while preserving some structure information inside one class. This demonstrates that hiDF is equipped with similar representation learning abilities as deep neural networks, which uses hidden layers to distort the input in a non-linear way so that classes become easily separable by the last layer [15]. We believe that leveraging feature interactions on multiple levels facilitates hiDF to learn hierarchical distributed representations of data.

We also examine the importance of new features generated across varying feature interaction orders via the marginal importance metric [16]. Figure 3 depicts the result by a boxplot. We can see that the leftmost bar shows the input features' marginal importance, with all but two informative features having nearly zero importance values. Figure 3 also illustrates that new features generated from higher-order interactions tend to have greater feature importance, thus have more impact on the learning task. Since higher-order feature interactions are usually discovered in deeper layers, it is reasonable to believe that hiDF can benefit from its hierarchical structure.

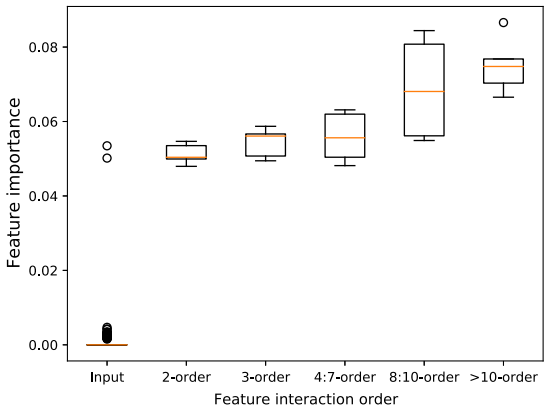


Fig. 3. The importance of features with varying interaction orders

TABLE I
STATISTICS OF THE DATASETS IN TERMS OF NUMBER OF EXAMPLES, NUMBER OF FEATURES AND NUMBER OF CLASSES

Datasets	# of examples	# of features	# of classes
CoverType	581,012	54	7
Adult	48,842	14	2
Bank	41,188	20	2
Credit Card	30,000	23	2
Arrhythmia	452	279	16
YearPredictionMSD	50,000	90	2
Diabetes Readmission	100,000	55	2
Satimage	6,435	36	6
Crowdsourced Mapping	10,845	28	6
Congestive Heart Failure	9809	88	2

B. Real dataset

We select several widely used benchmark datasets of binary and multi-class classification tasks from the UCI Machine Learning Repository [17]. The datasets vary in size: from only 452 examples up to 581,012 examples. They come from different areas, including health and medical research (Arrhythmia and Diabetes Readmission), remote sensing (CoverType and Satimage), financial and business (Bank marketing and Default of Credit card clients), social study (Adult), etc. We select these datasets mainly for two reasons. First, they are representative tabular data of categorical and mixed variable types and tend to contain multiple underlying explanatory factors that can be utilized by our hiDF to discover feature interactions. Second, these datasets have more input features than output labels, which could verify that our hiDF can generate more informative and diverse feature representations than gcForest.

In addition to these 9 UCI datasets, we release a new dataset called Congestive Heart Failure Dataset. This dataset includes 9809 patients with a diagnosis of "Congestive Heart Failure" from the MIMIC-III (Medical Information Mart for Intensive Care) platform [18]. MIMIC is a relational and public database containing tables of data relating to patients who stayed within the intensive care units at Beth Israel Deaconess Medical Center. In this Congestive Heart Failure Dataset, 88 numerical and categorical variables including patients' demographics, vitals' measurements, laboratory results, prescriptions, imaging results, etc are collected to predict the in-hospital mortality of patients. Table I shows the basic statistics of these datasets.

In these experiments, we used the default hyper-parameters defined in our hiDF model and did not fine-tune them. The default number of decision trees in a random forest is 500; the number of maximum layers in hiDF is 10; the early stopping patience is 2; the number of bootstrap samples $B = 10$; the default number of intersection trees is 20 with each tree's depth $D = 5$ and $n_{child} = 2$. For reproducibility, we make the source code and some datasets used in this work publicly available at https://git.io/DM487

C. Performance comparison

We adopt classification accuracy as our evaluation measure. For comparison, we choose several widely used tree ensemble algorithms including GBDT [19], Random Forests [20], gcForest [9], gcForest$_{CS}$ [21], XGBoost [22], and SVM.

The gcForest method is proposed by Zhou and Feng [9] to show that deep learning framework also can be realized by non-differentiable modules. It achieves state-of-the-art performance on many categorical and mixed modeling tasks. Recently, Pang et al. [21] proposed gcForest$_{CS}$ to address the high computational cost problem of gcForest by introducing a confidence screening mechanism. We set the hyper-parameters of gcForest and gcForest$_{CS}$ as default.

XGBoost is a scalable end-to-end tree boosting system proposed by Chen and Guestrin [22]. We set the number of boosting rounds equals to 500, and the maximum tree depth for base learners equals to 6. As for other hyper-parameters, we set them as the default values. We apply the same configurations of hyper-parameters to GBDT.

Table II reports the accuracy comparison result among these methods. The win/tie/lose counts of hiDF over the other methods are indicated by the last rows. We can see that hiDF achieves the highest accuracy on all of these datasets. We can also clearly see that hiDF exhibits consistently better performance than gcForest and gcForest$_{CS}$, which suggests hiDF's representation learning ability based on high-order feature interactions can indeed boost the performance of the original deep forest.

TABLE II
COMPARISON OF TEST ACCURACY OF EACH METHODS ON THESE 10 DATASETS. THE BEST ACCURACY IS HIGHLIGHTED IN **BOLD** TYPE. • INDICATES THE SECOND-BEST.

Datasets	hiDF	gcForest	gcForest$_{CS}$	Random Forests	XGBoost	GBDT	SVM
CoverType	**97.62 ± 0.08**	96.23 ± 0.10	96.09 ± 0.07	95.63 ± 0.07	94.48 ± 0.05	96.94 ± 0.11 •	71.47 ± 0.12
Adult	**86.90 ± 0.05**	86.17 ± 0.06	86.17 ± 0.09	85.15 ± 0.08	86.40 ± 0.00 •	86.12 ± 0.10	79.86 ± 0.00
Bank	**89.96 ± 0.21**	89.89 ± 0.26	89.95 ± 0.25 •	89.33 ± 0.17	89.13 ± 0.33	88.60 ± 0.25	89.75 ± 0.21
Credit Card	**82.00 ± 0.23**	81.74 ± 0.38 •	81.73 ± 0.20	81.71 ± 0.33	80.61 ± 0.36	81.24 ± 0.27	77.88 ± 0.01
Arrhythmia	**78.24 ± 1.62**	76.26 ± 1.49 •	74.51 ± 1.62	74.51 ± 2.13	75.16 ± 1.32	74.29 ± 3.15	60.66 ± 1.62
YearPredictionMSD	**75.89 ± 0.46**	75.49 ± 0.40	75.61 ± 0.30	73.41 ± 0.30	74.69 ± 0.21	75.85 ± 0.20 •	68.62 ± 0.62
Diabetes	**62.43 ± 0.23**	62.26 ± 0.36	62.37 ± 0.33 •	62.11 ± 0.24	61.00 ± 0.17	61.04 ± 0.45	54.99 ± 0.02
Satimage	**91.75 ± 0.06**	91.63 ± 0.12 •	91.58 ± 0.11	91.21 ± 0.05	90.65 ± 0.00	90.44 ± 0.09	88.60 ± 0.00
Crowdsourced Mapping	**65.07 ± 0.88**	64.93 ± 0.95	65.03 ± 0.56 •	63.47 ± 0.72	62.00 ± 0.00	62.53 ± 0.50	55.67 ± 0.00
Congestive Heart Failure	**90.16 ± 0.53**	88.61 ± 0.59	88.58 ± 0.27	87.90 ± 0.01	89.34 ± 0.01 •	88.83 ± 0.01	85.09 ± 0.00
win/tie/lose	—	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0

D. Computational complexity comparison with gcForest

To show that hiDF can achieve state-of-the-art predictive result while still being computationally efficient, we compare hiDF with gcForest and gcForest$_{CS}$ on two datasets: Adult and CoverType. We use a hardware of 16×3.70 GHz CPUs with 128 GB memory. All of these methods grow 20 layers. Table III summarizes the results. We can see that hiDF uses significantly less memory and test time than gcForest and gcForest$_{CS}$ (by about one order of magnitude).

TABLE III
COMPARISON RESULTS OF TRAINING TIME, TEST TIME(IN CPU SECONDS) AND MEMORY USAGE (IN MEGABYTES) WITH TWO DATASETS.

Datasets	Method	Training time	Test time	Memory
Adult	hiDF	3208.7	13.1	2793.2
	gcForest	2763.8	541.3	29243.9
	gcForest$_{CS}$	1063.1	290.2	23272.7
Covertype	hiDF	4505.9	90.9	5562.5
	gcForest	2861.3	1519.4	61981.2
	gcForest$_{CS}$	1582.3	730.1	57658.8

The main reason behind the efficiency of hiDF is: since the feature representations generated by gcForest and gcForest$_{CS}$ are based on RF predictions. As a result, they have to save the trained multilayer RFs to generate feature representations for test instances, which is essentially saving tens of thousands of unpruned decision rules, thus requiring a large memory cost. Besides, gcForest and gcForest$_{CS}$ consumes a lot of time to do layer-wise prediction for test instances. On the other hand, hiDF does not rely on prediction-based representations. Therefore, only a small number (tens) of feature interactions have to be saved in each layer, and test instances can generate feature representations easily based solely on these interactions.

Note that in our current implementation, the B times bootstrap are done sequentially, so the gRIT processing takes the majority amount of time. The training time of our hiDF could be significantly less if using parallel processing.

IV. CONCLUSION

In this paper, we focus on the problem of limited representation learning ability and large computational complexity caused by prediction-based representations in traditional Deep Forest algorithms. To address these shortcomings, we propose a novel method called hiDF to attain interaction-based hierarchical distributed representations through in-model feature transformation. Experiments show the potential of hierarchical distributed representations for reducing the complexity of classification tasks by enhancing the separability of data, and further demonstrate the performance improvement and computational effectiveness of our method.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China No.2020AAA0109400.

The authors would like to thank Doctor Yao Wang from Sir Run-Run Shaw Hospital, Zhejiang University for generously providing the Congestive Heart Failure Dataset, as well as the anonymous reviewers and Yi-Xiao He from Nanjing University and Kewen Zheng from National University of Singapore for constructive suggestions.

REFERENCES

[1] Y.-L. Zhang, J. Zhou, W. Zheng, J. Feng, L. Li, Z. Liu, M. Li, Z. Zhang, C. Chen, X. Li *et al.*, "Distributed deep forest and its application to automatic detection of cash-out fraud," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 5, pp. 1–19, 2019.

[2] L. Sun, Z. Mo, F. Yan, L. Xia, F. Shan, Z. Ding, W. Shao, F. Shi, H. Yuan, H. Jiang, D. Wu, Y. Wei, Y. Gao, W. Gao, H. Sui, D. Zhang, and D. Shen, "Adaptive feature selection guided deep forest for COVID-19 classification with chest CT," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2798–2805, 2020.

[3] Y. Guo, S. Liu, Z. Li, and X. Shang, "BCDForest: A boosting cascade deep forest model towards the classification of cancer subtypes based on gene expression data," *BMC Bioinformatics*, vol. 19, no. 5, p. 118, 2018.

[4] F. Yang, Q. Xu, B. Li, and Y. Ji, "Ship detection from thermal remote sensing imagery through region-based deep forest," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 3, pp. 449–453, 2018.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[6] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "Quartznet: Deep automatic speech recognition with 1D time-channel separable convolutions," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 6124–6128.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[8] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[9] Z.-H. Zhou and J. Feng, "Deep forest," *National Science Review*, vol. 6, no. 1, pp. 74–86, 2019.

[10] S.-H. Lyu, L. Yang, and Z.-H. Zhou, "A refined margin distribution analysis for forest representation learning," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 5530–5540.

[11] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[12] D. Amaratunga, J. Cabrera, and Y.-S. Lee, "Enriched random forests," *Bioinformatics*, vol. 24, no. 18, pp. 2010–2014, 2008.

[13] R. D. Shah and N. Meinshausen, "Random intersection trees," *Journal of Machine Learning Research*, vol. 15, no. 20, pp. 629–654, 2014.

[14] B. Yu, "Stability," *Bernoulli*, vol. 19, no. 4, pp. 1484–1500, 2013.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] S. Basu, K. Kumbier, J. B. Brown, and B. Yu, "Iterative random forests to discover predictive and stable high-order interactions," *Proceedings of the National Academy of Sciences*, vol. 115, no. 8, pp. 1943–1948, 2018.

[17] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[18] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[19] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] M. Pang, K.-M. Ting, P. Zhao, and Z.-H. Zhou, "Improving deep forest by confidence screening," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[22] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.