# Dense Adaptive Cascade Forest: A Densely Connected Deep Ensemble for Classification Problems

Haiyang Wang
Southwest Jiaotong University, ChengDu, SiChuan, China
529570509@qq.com
ORCID: 0000-0003-2273-2686

**Abstract.** Recent research has shown that deep ensemble for forest can achieve a huge increase in classification accuracy compared with the general ensemble learning method. Especially when there are only few training data. In this paper, we decide to take full advantage of this observation and introduce the Dense Adaptive Cascade Forest (daForest), which has better performance than the original one named Cascade Forest. And it is particularly noteworthy that daForest has a powerful ability to handle high-dimensional sparse data without any preprocessing on raw data like PCA or any other dimensional reduction methods. Our model is distinguished by three major features: the first feature is the combination of the SAMME.R boosting algorithm in the model, boosting gives the model the ability to continuously improve as the number of layer increases, which is not possible in stacking model or plain cascade forest. The second feature is our model connects each layer to its subsequent layers in a feed-forward fashion, to some extent this structure enhances the ability of the model to resist degeneration. When number of layers goes up, accuracy of model goes up a little in the first few layers then drop down quickly, we call this phenomenon degeneration in training stacking model. The third feature is that we add a hyper-parameter optimization layer before the first classification layer in the proposed deep model, which can search for the optimal hyper-parameter and set up the model in a brief period and nearly halve the training time without having too much impact on the final performance. Experimental results show that daForest performs particularly well on both high-dimensional low-order features and low-dimensional high-order features, and in some cases, even better than neural networks and achieves state-of-the-art results.

**Keywords:** Deep Forest, Ensemble, Deep Learning, Boosting, Dense Con nectivity

## 1 Introduction

As is well known, deep neural networks play an increasingly important role nowadays [1, 2], in many fields, such as image classification and speech recognition, deep neural network even become the dominant approach [4]. With the improvement of computer hardware, researchers have the ability to train increasingly deeper and more complex networks, neural networks gradually surpassed humans in solving certain problems [8, 9, 10].

For deep neural networks mentioned above, we generally refer to convolutional neural networks, which is composed of convolutional layers and fully-connected layers [2]. Convolutional layers assists models to extract high-order features from raw data and feed these features into fully-connected layers to get the final prediction result, many experiments have proved that this mechanism is efficient and useful [5, 6]. However, training a deep neural network is not an easy task, first, researchers need collect huge amount of training data in a particular field, and more importantly, neural network doesn't work well in unsupervised mode, therefore they need to hire a large number of people to label this data, which is costly. Second, it's computationally expensive to train a network with hundreds of layers, and it's not possible to train or contain these deep architectures on a single GPU. Last but not the least, when data is uncorrelated in space and time series, convolutional neural network is not likely to play an important role as always. When we talk about the depth of neural networks, in fact, we are talking about the depth of convolutional layers, as a classifier, fully-connected layers have not changed much in these years since they were invented and used almost 30 years ago [5, 7]. And it is widely recognized that fully-connected network is highly prone to overfitting when training data is small. Imagine that training data is fewer than neural network parameters, network will converge to local optimal and be overfit no doubt. Dropout and regularization is widely use to alleviate this problem [11, 12, 13]. But still, overfitting is a severe problem that is hard to avoid and solve thoroughly for neural network.

As deep neural network gradually become deeper, more powerful and more efficient, even more research problems emerge:

– In era of big data, many enterprises and research institutions have collected huge amount of data, however, a large portion of this data is space or time uncorrelated, which means that most deep neural network structures cannot be applied, convolutional layers cannot assist model to complete spatial feature extraction. Therefore, in order to fully exploit this large training data, researchers have to utilize ordinary learning model, such as linear or generalized linear models, support vector machine, random forest, etc. These ordinary learning models are structurally simple and relatively less inclined to overfit, and ensemble approaches enable these models to achieve a reasonable balance between bias and variance [15].

– Even in the era of big data, in many fields, collecting huge amount of data is impossible or costly, big data makes deep neural network work well, but small data make neural network more prone to overfitting. Numerous studies have shown that transfer learning can make DNN work well on new dataset[16, 17, 18], training model on one topic data set, then fine tune the trained model on another same domain dataset, considering that most of the data or tasks are relevant, we can transfer the model parameters we have learned through transfer learning and share it to the new model in a certain way so as to speed up and optimize the learning efficiency of the model instead of learning from scratch. Employing transfer learning on DNN requires a large amount of same domain data and does not help with spatial and time uncorrelated features, because the widely used transfer learning method is locking the parameters of convolutional layers and fine-tuning parameters of fully connected layers. As for traditional learning model, small scale data can fully train these ordinary models, and ensemble can greatly improve the capacity of such models [15, 20].

– The biggest problem of deep neural network is the non-interpretability, which means applying theoretical analysis on deep neural network is extremely difficult, leading deep neural network to be treated as black box [19]. Compared with deep neural network, traditional learning models have better interpretability and it is more easily to apply theoretical analysis on these models.

In order to alleviate deficiencies of neural network mentioned above in a certain extent, and fully utilize the potential of ensemble, in this paper, we propose a novel decision tree ensemble method named dense adaptive cascade forest (daForest). It is experimentally proved that ensemble model can enhance single learning model's representational ability, achieve better prediction performance and have better generalization ability [15, 20]. Motivated by recent influential research results on deep neural network[21, 22], the depth of the model is one of the key factors that affect its performance, machine learning model can gradually learn and generate some important high-order features from features generated by its preceding layer when the model goes deeper. The deep ensemble model we proposed has the characteristics of both. Besides, we also noticed that degree of discrimination is different among samples, the contribution of different samples in the model training process is not the same, ordinary stacking model or deep model does not make good use of the fact [15, 20, 14]. therefore, we introduce adaptive boosting approach to enhance performance of our model, adaptive boosting can adjust weights of samples and turn the whole cascade model into a deep additive model [25, 26, 27, 28], such transformation endow some particularly good features into the deep cascade model, daForest has the ability to distinguish different samples with different degree of discrimination, the model can adjust its attention on more important sample in each layers, more importantly, the final prediction can make full use of the superior predictions of each layer because of the weighted additive characteristic of adaptive boosting. Deep ensemble and boosting make daForest not just simply stack base classifier to obtain a large nominal depth, but fully release the potential of the depth model. In daForest, we also employ a distinctive structure called dense connectivity, which is first introduced in DenseNet [29].

The rest of this paper is organized as follows: In section 2, we introduce related work of cascade forest and daForest. In section 3, we present overall structure of daForest in detail. In section 4, we compare daForest with several popular machine learning models, and present experimental results and evaluations. In section 5, we conclude the paper.

## 2 Related Work

daForest is an improved tree stacking model, individual learner of daForest is random forest and extremely randomized trees [23, 24]. Stacking is a widely used ensemble approach, usually take the output of the preceding layer as the input of the next layer, it is widely recognized that the most popular stacking models are deep convolutional neural network [2] and deep belief nets [34], which opens a door on the eve of deep learning sweep across the globe. Experimental and theoretical studies have proved that stacking can indeed improve the performance of individual learners [15, 20]. Unlike deep neural network, composed of thousands of nonlinear differentiable base classifiers, which can be trained by backpropagation algorithm, stacking method of ordinary base learner is trickier than it looks. Without backpropagation, end to end training is impossible to be applied on ordinary stacking model, which seriously restricts the depth of the stacking model and makes the model more prone to overfitting. In order to relieve the problem to the maximum extent, we employ two methods: concatenate output probabilistic features with original input features of first layer in each subsequent layers (we call such stacking model with individual tree learner plain cascade forest) and change weights of each sample through boosting [25, 26, 27, 28]. The new stacking approach proposed is highly competitive with current ensemble methods (such as bagging, boosting and plain stacking) and extremely easy to implement.

Actually there is a pioneer study on deep ensemble/stacking model named gcForest [14], which is a plain cascade forest facilitated with multi-grained scanning (works like convolutional kernel) to extract spatial or temporal correlation of features, performance recorded in the literature is quite promising and inspiring, on minist and cifar-10 image classification dataset, gcForest even outperform deep belief nets and shallow convolutional neural network. And on discrete features dataset, such as imdb, yeast, adult, etc., gcForest achieves state of the art results on most of them, in this paper, we also compared our model with gcForest and other classification methods, and experimental results show that daForest outperforms gcForest and other models on these datasets. Compared with deep neural networks, gcForest has fewer hyper parameters and faster training speed, in addition, structural complexity of the model is dynamically determined in training process. From a certain perspective, cascade forest can be regarded as a kind of boosting method. daForest is a densely connected cascade forest boosted by SAMME.R algorithm [28], which can be regarded as a boosting of boosting method.

Adaboost [25, 26] is a popular boosting algorithm, which can change samples' distribution by assigning different weight to each sample. The weight of samples reflects the difficulty of samples being classified, the higher the weight, the higher the probability of being misclassified. Besides, Adaboost is also an additive boosting method, each model generated through training process has a different weight that reflects the performance of the model. In the whole classification process, some classifiers may have better classification results than others on some indistinguishable samples. In this way, a complementary classifier series is established. SAMME.R is an improvement of original Adaboost, which naturally extends the original Adaboost algorithm to the multi-class case without reducing it to multiple two-class problems [28] and reduces the minimum accuracy requirement for individual classifiers. A study shows that Adaboost can even be used to reduce the dimensionality of image features [31]. Experimental results indicate that this algorithm can greatly enhance our deep model.

Our model employ random forest [23] and extremely random forest [24] as individual learners. By using bagging and randomly splitting features space, random forest can achieve very good classification result and outperform deep neural network on non-spatial or non-temporal dataset. Extremely random forest is more random that ordinary random forest, which randomly chooses subspace division in construction of decision tree while random forest choose the best division, as it is well known that diversity is a pivotal characteristic of ensemble model [15, 20]. Some studies have shown that random forest can even be combined with deep neural network [32, 33] and back propagate gradients for neurons.

## 3 The Proposed Deep Ensemble Approach

Fig. 2 shows overall procedure of daForest. daForest consists of cascade forest, boosting procedure and dense connectivity and comprises $\mathcal{L}$ layers. Given input and output space $\mathcal{X} \subset \mathcal{R}^d$ and $\mathcal{Y}$, each layer can be regarded as an individual ensemble module $\mathrm{E}_l$, where $l$ indicates index of layers, and each of which implements a classification function $E_l(\cdot): \mathcal{X} \rightarrow [0, 1]$. An individual ensemble module has several individual learners $\mathrm{F}_{li}$, which can be random forest [23] or extremely random forest [24] and apply transformation $F_{li}(\cdot): \mathcal{X} \rightarrow [0, 1]$ on each sample $x_i$, finally we get predictive probability of $x_i: F_{li}(x_i)$. We indicate output probabilistic features of $l^{th}$ layer as $x_l$ and original input of first layer as $x_0$. In this section, we will introduce daForest in this order and give details of implementation at the end.

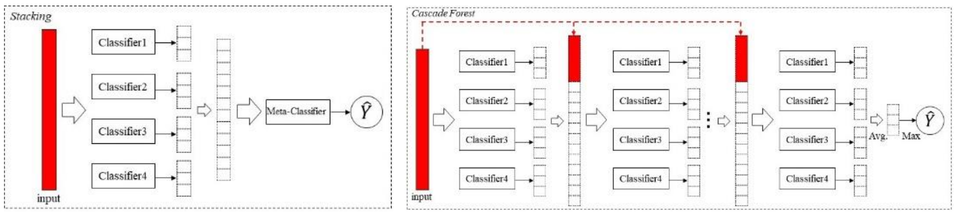

**Fig. 1.** left: traditional stacking model. right: cascade forest structure.

### 3.1 Ordinary Cascade Forest

Ordinary cascade forest is the fundamental framework of the entire model. Fig. 1 illustrates the layout of cascade forest. Cascade forest can be regarded as a deep ensemble of random forest and extremely random forest, which are treated as individual learners and packaged into a single ensemble module in each layer, we can construct cascade forest by stacking several single ensemble modules layer by layer and feeding concatenation of each layer's probabilistic output features and original input features into the next ensemble module. It is noteworthy that concatenating probabilistic features and original input features into a single feature vector is an effective method to prevent overfitting. Traditional stacking model connect the $l^{th}$ layer's output as input to the $(l+1)^{th}$ layer, which can be described as:

$$\mathrm{x}_l = E_l\,(x_{l-1}) = [F_{l1}\,(x_{l-1}), F_{l2}\,(x_{l-1}), \ldots, F_{ln}\,(x_{l-1})], l = 1, \ldots, \mathcal{L} \qquad (1)$$

However, such structure is highly prone to overfitting when number of layers goes up and may impede the diversity of individual learners, because each layer is only related to its preceding layer and individual learners tend to be homogenized, diversity of traditional stacking model is going to be worse when depth is getting deeper. One way to solve the problem is combining original features $x_0$ and probabilistic features $x_l$ together before training $l^{th}$ layer's individual learners (as illustrated in Fig. 1):

$$x_l = [x_0, E_{l-1}(x_{l-1})] = [x_0, F_{l1}(x_{l-1}), \dots, F_{ln}(x_{l-1})]\,, l = 2, \dots, \mathcal{L} \quad (2\text{-}1)$$
$$x_l = E_l(x_0) = [F_{l1}(x_0), \dots, F_{ln}(x_0)], when\ l = 1 \quad (2\text{-}2)$$
$x_l$ are probabilistic features produces by forests and final prediction of cascade forest is the output of the last layer. In order to get probabilistic features of each instance, we must record class distribution at leaf nodes in each decision tree of the random forest, then average the class percentage of instance we concerned across all trees and concatenate probabilistic features generated by each forest in the same layer. We employ stratified k-fold validation in each layer so as to do reduce the impact of overfitting and keep class distribution unchanged in each fold. For example, suppose that we use 3-fold cross validation and employ 2 forests in each layer, we cut training data X into 3 pieces that keep the same class distribution, which is [$X_1$, $X_2$, $X_3$], then feed the data to classifiers in three times and eventually get prediction [$Y_1$, $Y_2$, $Y_3$]:

fold 1: training classifier $C_{11}$ and $C_{12}$ with [$X_2$, $X_3$], Y1=[$C_{11}(X_1)$, $C_{12}(X_1)$]
fold 2: training classifier $C_{21}$ and $C_{22}$ with [$X_1$, $X_3$], Y2=[$C_{21}(X_2)$, $C_{22}(X_2)$]
fold 3: training classifier $C_{31}$ and $C_{32}$ with [$X_1$, $X_2$], Y3=[$C_{31}(X_3)$, $C_{32}(X_3)$]

Final prediction Y of X that can be fed to next layer is concatenation of [$Y_1$, $Y_2$, $Y_3$].

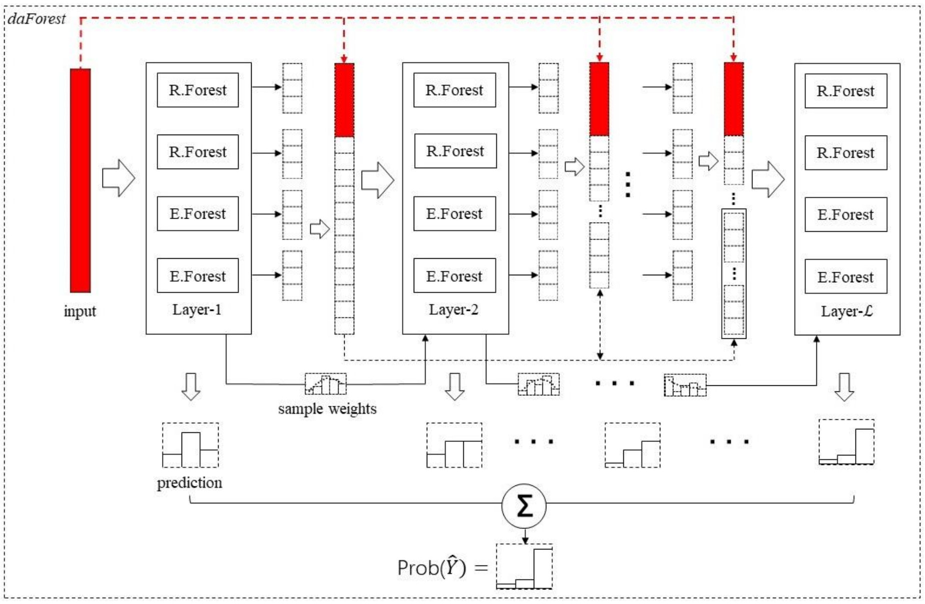

**Fig. 2.** The overall procedure of daForest. Each layer consists of 4 classifiers (2 random forests and 2 extremely random forests), there are 3 class to predict. Hyper-parameter optimization layer has been omitted for brevity.

## 3.2 Boosting Procedure

Boosting has been a very successful model-guided method for solving two-class or multi-class classification problem [28, 26, 27]. By assigning weights to each classifier and changing weights according to error rate in iteration, boosting can combine the produced weak learners into a strong classifier [20]. The most important characteristic of boosting is that the technique can take advantage of the depth of the proposed deep ensemble model, that is to say, as the number of iterations increases, the distribution of samples is gradually changing, each layer can concentrate more attention on indistinguishable samples. Therefore, daForest is also an attention based model. Actually, attention based models are widely used in various fields, especially in the study of neural networks [35, 36, 37]. Whereas the attention mechanism used in natural language processing or computer vision is feature-wise attention, in daForest, we apply sample-wise attention on each layer. There are numerous different implementations of adaptive boosting, such as Adaboost M1/ M2 [25, 26], Adaboost MH [27], SAMME and SAMME.R [28], as described by Hastie et al. [28], traditional boosting method is much harder to achieve minimum requirement for individual learner in multi-class classification problem, which is predictive accuracy must be greater than random guessing accuracy rate 1 / K, where K refers to K classes. Accordingly, we choose SAMME.R as boosting implementation in the proposed model.

As illustrated in Fig. 2, adaptive boosting is used to change distribution of samples in each layer and drawing attention mechanism into the proposed deep model. Suppose each layer is composed of N individual learners $[F_{l1}, F_{l2}, \dots, F_{ln}]$, and we have $\mathcal{L}$ layers in total, each layer can also be regarded as an individual ensemble module $\mathrm{E}_l$. The whole boosting procedure can be described as:

**Algorithm 1** Boosting Procedure

```
**Require:** original training data $\mathrm{X} = [X_0], X_0 \subset \mathbb{R}^d$, class label $\mathrm{Y}$.
1.  Initialize the weight of each sample: $w_i^1 = 1/m, \mathrm{i} = 1, 2, \dots, \mathrm{m}$.
2.  For $\mathrm{j} = 1$ to $\mathcal{L}$:
3.      Fit ensemble module $\mathrm{E}_j$to the training data $\mathrm{X}$:
4.          Fit individual learners $[F_{j1}, F_{j2}, \dots, F_{jn}]$using sample weights.
5.      Obtain the weighted class probability estimates:
6.          $\mathrm{P}_\mathrm{k}^\mathrm{j}(\mathrm{x}) = \mathrm{Prob}_\mathrm{w}\ (\mathrm{c} = \mathrm{k}|\mathrm{x}) = \mathrm{E}_{\mathrm{jk}}(\mathrm{x}), \mathrm{k} = 1, \dots, \mathrm{K}$.
7.      Set:
8.          $h_k^j(x) = (\mathrm{K} - 1)\left(log\mathrm{P}_\mathrm{k}^\mathrm{j}(\mathrm{x}) - \frac{1}{K}\sum_{k'} log\mathrm{P}_{\mathrm{k'}}^\mathrm{j}(\mathrm{x})\right), \mathrm{k} = 1, \dots, \mathrm{K}$.
9.      Update training data:
10.         $\mathrm{X} = [X_0, h^j(x)], \mathrm{X} \subset \mathbb{R}^{d+k*n}$
11.     Set:
12.         $w_i^{j+1} = w_i^j * \exp\left(-\frac{K-1}{K} y_i^T logp^{(j)}(x_i)\right), i = 1, \dots, m$
13.     Renormalize $w_i$:
14.         $w_i^{j+1} = w_i^{j+1}/sum(w^{j+1})$
15. Output:
16.     Final-Prediction = $\arg\max_k \sum_j h_k^j(x)$
```

Cascade forest is transformed to an additive model when adaptive boosting is applied.

## 3.3 Dense Connectivity

In ordinary cascade forest and gcForest [14], the $l^{th}$ layer is connected to the $(l+1)^{th}$ layer and fed with concatenation of preceding layer's output and original training features as described in Sec. 3.1, we call such structure sparse connectivity. However, diversity is a crucial metric in ensemble method, each layer only has access to the information of its preceding layer in sparse connectivity architecture, much information has been discarded, which may impede diversity and the information flow in the deep model. We noticed that training accuracy curve of ordinary cascade forest is very unstable after a few rounds of iteration, which may have a great negative effect on the final performance of the model. Enlightened by the structure of DenseNet [29], we introduce dense connectivity to ordinary cascade forest, turn it to dense cascade forest, In our experiments, we find that this structure can effectively suppress the violent concussion of training accuracy curve, make the training process more stable and improve the prediction accuracy to a certain extent.

In dense connectivity, each layer is directly connected to all subsequent layers, Fig. 2 illustrates the layout of dense connectivity. Consequently, the $l^{th}$ layer receives the probabilistic features of all preceding layers. Information received by the $l^{th}$ layer:

$$x_l = [x_0, E_{l-1}(x_{l-1}), E_{l-2}(x_{l-2}), \dots, E_1(x_0)]\,, l = 2, \dots, \mathcal{L} \tag{3-1}$$

$$x_l = E_l(x_0) = [F_{l1}(x_0), \dots, F_{ln}(x_0)], when\ l = 1 \tag{3-2}$$

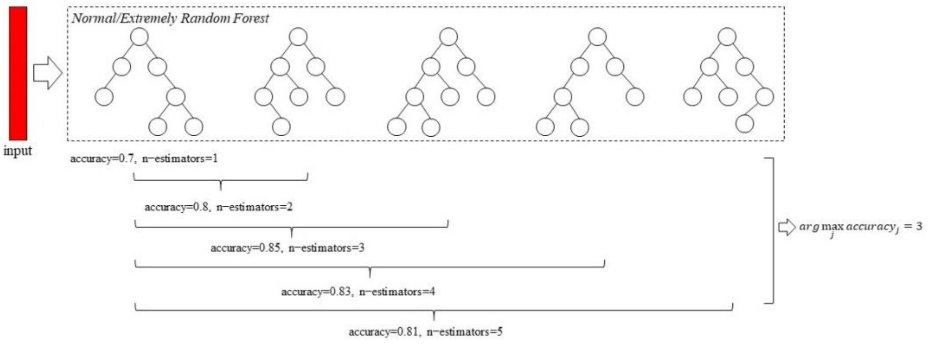
Normal/Extremely Random Forest
input
accuracy=0.7, n−estimators=1
accuracy=0.8, n−estimators=2
accuracy=0.85, n−estimators=3
accuracy=0.83, n−estimators=4
accuracy=0.81, n−estimators=5
$arg\max_j accuracy_j = 3$

**Fig. 3.** Optimization layer, which is a normal random forest or extremely random forest, searching for optimal number of estimators in a cumulative prediction fashion in all trees. Suppose there are five trees in the optimization layer, therefore, the search range is from 1 to 5 with step size 1.

## 3.4 Hyper-Parameter Optimization Layer

**Fig. 3** shows the search process for the optimal parameters. A forest consists of many estimators (a.k.a. classification and regression trees), the number of estimators is a key hyper-parameter, which has a significant effect on the performance and training time of the deep model, however, it is a time-consuming task to determine the number of estimators by grid-search. In implementation of gcForest [14], the number of estimators is fixed in 500, which is not a wise choice, because the complexity of different classification task is variant and 500 estimators may be too few or too many for some tasks, which will lead to underfitting or overfitting. The depth of the model (cascade forest, gcForest and daForest) can be dynamically determined by using early-stopping mechanism, terminating training process if the model dose not achieve a continued improvement on validation accuracy. Whereas, the quasi-optimal number of estimators in different types of forests must be found in advance. Therefore, we employ a linear search method. Suppose each layer of deep model is composed of several random forests, and we are finding the quasi-optimal number of estimators in a range of values from 20 to 600 with step size 20, fit a random forest with 600 estimators and record predictions of each decision tree in the forest on validation dataset before training our deep model, then calculating accuracy on these prediction series, the number corresponding to the highest value is what we are looking for. The whole process to find n-estimators can be described as:

$$\mathrm{nestimators} = \arg\max_{j} accuracy\left(\frac{\sum_{i=1}^{j} predictions \cdot of \cdot i^{th} \cdot estimator}{j}\right) \tag{4}$$

where $\mathrm{j} = 20 + (\mathrm{n} - 1) * 20, \mathrm{n} = 1, \ldots, 30$ . The summation term in (4) can be accelerated by utilizing a cache.

### 3.5 Implementation Details

The daForest used in our experiments has one hyper-parameter optimization layer and eight forests in each layer, by default, each forest in the individual ensemble module has 500 decision trees. If hyper-parameter optimization is activated, optimization layer will change number of decision trees in each forest according to the result of linear search as described in Sec.3.4, for some small and low-dimensional dataset (e.g. Yeast, B.C.W.), we will change the search scope to (5, 200) with step size 5. Random forests construct each tree by bagging and randomly choosing $\sqrt{d}$ features in each inner node split. And we turn extremely random forest to completely random forest by fixing maximum number of candidate features in each split to 1, which will bring greater diversity to our model, as suggested in [Zhi-Hua Zhou et al. 2017]. In order to abate the impact of abnormal prediction results of each layer on the overall results, we introduce learning rate for boosting procedure and fix its value to 0.3, learning rate can also prevent the sample weights changing from being too fast.

## 4 Experiments

We demonstrate our model on several benchmark datasets, including both high-dimensional sparse datasets and low-dimensional datasets. We also compare our model with other state-of-the-art architectures on each dataset and find that our model has a competitive performance, in some cases, our model even outperforms state-of-the-art architectures. As described in Sec. 3.1, 3-fold cross validation is applied to generate probabilistic features in each layer, by which overfitting can be effectively suppressed. We use original testing set to validate our model when there exists a testing set in the original data set, instead, we will take 30% of the data set for testing set and 70% for growing daForest.

## 4.1 Classification Results on Amazon Commerce Reviews

Amazon Commerce Reviews are derived from customers' reviews in Amazon Commerce Website [38]. This data set consists of 10000 samples, each sample has 10000 attributes, as described in Table 1. The feature set of A.C.R is composed of four types of writing habits of users: lexical, syntactic, content-specific, and idiosyncratic, each of them has about 1~15 sub-category feature types. There are 19 different feature types in total. The dataset is used in many literatures, but they are all limited to 2~10 authors. Few classification algorithms extend to large number of target classes. In order to validate that daForest can achieve highly competitive and inspiring robustness to ordinary classifiers and deep neural networks, we used all 50 classes in the dataset.

**Table 1.** Description of Amazon Commerce Reviews

| Language | No. of authors | Reviews per author | Average length of reviews per author | No. of Instance | No. of Attributes |
|---|---|---|---|---|---|
| English | 50 | 30 | 856 characters | 1500 | 10000 |

Features of A.C.R. are not spatial correlated, thus convolutional neural networks cannot be used directly. Therefore, we employed a fully connected deep neural network with 1024-1024-512-256 neurons in each hidden layer. We also include experimental result achieved by synergetic neural network (SNN) [38], which associates synergetics with artificial neural network. Fig. 4 represents comparisons of testing accuracy between the proposed model and gcForest [14].

**Table 2.** Identification accuracy on Amazon Commerce Reviews

| | | |
|---|---|---|
| daForest | | **85.11%** |
| gcForest | | 83.78% |
| SNN | Balanced attention parameter | 68.31% [S. Liu, 2011] |
| | Self-adaptive attention parameter | 80.49% [S. Liu, 2011] |
| SVM (linear kernel) | | 60.67% |
| MLP | | 61.11% |
| Logistic Regression | | 72.89% |
| Random Forest | | 78.00% |

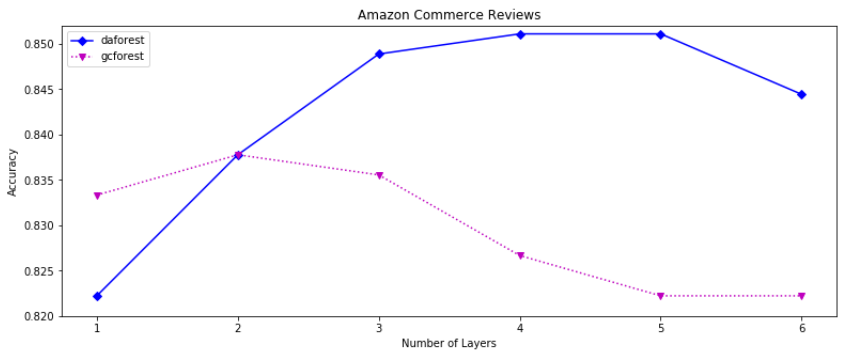

**Fig. 4.** Comparison of testing accuracy between daForest and gcForest

### 4.2 Classification Results on CNAE-9 Data Set

CNAE-9 is a dataset containing 1080 documents of free text business description of Brazilian companies categories into a subset of 9 categories cataloged in a table called National Classification of Economic Activities. The original texts were pre-processed to obtain the current data set: initially, it was kept only letters and then it was removed prepositions of the texts. Next, the words were transformed to their canonical form. Finally, each document was represented as a vector, where the weight of each word is its frequency in the document. This data set is highly sparse (99.22% of the matrix is filled with zeros) [41]. The detailed attributes of CNAE-9 are shown in Table 3. We split the dataset into 756 documents for training and 324 documents for testing.

**Table 3.** Description of CNAE-9

| Language | No. of categories | No. of Instance | No. of Attributes |
|---|---|---|---|
| English | 9 | 1080 | 857 |

Because of its spatial irrelevance, we compare daForest with an MLP with structure input-1024-1024-512-256-output. We also include the results of other classification methods and dimensional reduction methods, such as probabilistic neural network based on evolving system [40], dimensional reduction based on agglomeration and elimination of terms [39] and SIMACA method coupled with variables selection [41]. As shown in Table 4, daForest shows superior performance over other approaches. Fig. 5 represents comparison of testing accuracy between daForest and gcForest.

**Table 4.** Identification accuracy on CNAE-9

| | | |
|---|---|---|
| daForest | | **95.99%** |
| gcForest | | 95.06% |
| ePNN | ePNN1 | 88.71% [P. Marques, 2010] |
| | ePNN2 | 84.45% [P. Marques, 2010] |

| | | |
|---|---|---|
| Dim. Reduction | MI_1 | 92.78% [P. Marques, 2009] |
| | IAE | 91.11% [P. Marques, 2009] |
| SIMCA | VSC-SIMCA | 95.34% [Saleh, 2015] |
| | SIMCA | 94.62% [Saleh, 2015] |
| SVM (linear kernel) | | 93.21% |
| MLP | | 95.01% |
| Logistic Regression | | 94.45% |
| Random Forest | | 89.51% |

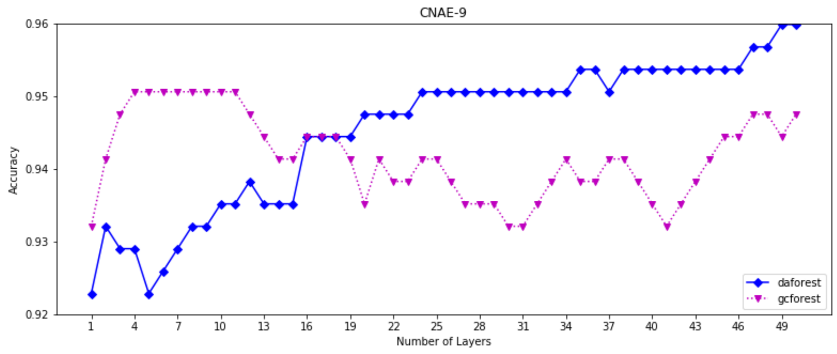

**Fig. 5.** Comparison of testing accuracy between daForest and gcForest

## 4.3 Classification Results on IMDB Movie Reviews

The IMDB movie reviews dataset [43] contains 50000 reviews in all, 25000 for training and 25000 for testing, labeled by sentiment (positive or negative). Reviews have been processed and encoded as a sequence of indexes, each feature in the dataset is a word index, which can be seen as the overall frequency of one word in IMDB dataset. For example, Integer 1 indicates the most frequent word in the dataset. We select the 5000 most frequent words as training and testing attributes. To adjust the data to our model, we apply tf-idf transformation on the reviews, so attributes of samples are represented by the product of term-frequency and inverse document-frequency of corresponding words. Table 5 represents details of the dataset.

**Table 5.** Description of IMDB

| Language | No. of categories | No. of Instance | No. of Attributes |
|---|---|---|---|
| English | 2 | 50000 | 5000 |

We compare daForest with CNNs trained on word vectors [44, 14] and an MLP with hidden layer shape 1024-1024-512-256. We also include results of other traditional classification models, as shown in Table 6. Fig. 6 represents comparison of testing accuracy between daForest and gcForest. gcForest is automatically terminated after few iterations when accuracy stops going up.

**Table 6.** Identification accuracy on IMDB

| | |
|---|---|
| daForest | **89.34%** |
| gcForest | 89.15% |
| CNN | 89.02% [Zhou, 2017] |
| SVM (linear kernel) | 88.24% |
| MLP | 85.92% |
| Logistic Regression | 88.62% |
| Random Forest | 85.16% |

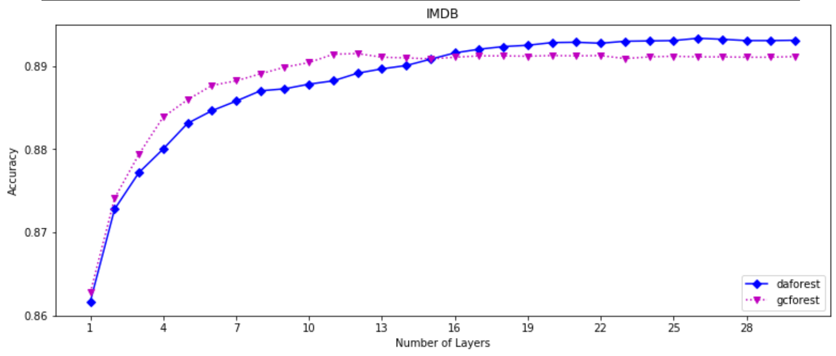

**Fig. 6.** Comparison of testing accuracy between daForest and gcForest

## 4.4 Classification Results on Low-Dimensional Datasets

In this section, we validate our model on 5 UCI datasets [42]: Letter Dataset, Adult Dataset, Yeast Dataset, Breast Cancer Wisconsin (Original) Dataset and Parkinsons Dataset. Details of datasets are shown in Table 7. Letter with 20000 instances, 16000 for training and 4000 for testing. Adult with 48842 instances, 34190 for training and 14652 for testing. Yeast with 1484 instances, 1039 for training and 445 for testing. B.C.W with 699 instances, 490 for training and 209 for testing. Parkinsons with 197 instances, 138 for training and 59 for testing. We employ different MLP configurations on each dataset in our experiment as [Zhou, 2014] suggested. CNNs are not applicable on these low-dimensional datasets, especially some of these datasets have instances less than 2000. Classification results are described as Table 8. It is noteworthy that SVM on Adult is unable to converge automatically, so we limit max iterations of SVM to 1000.

**Table 7.** Details of uci datasets

| Dataset name | No. of categories | No. of Instance | No. of Attributes |
|---|---|---|---|
| Letter | 26 | 20000 | 16 |
| Adult | 2 | 48842 | 14 |
| Yeast | 10 | 1484 | 8 |
| B.C.W | 2 | 699 | 10 |
| Parkinsons | 2 | 197 | 23 |

**Table 8.** Identification accuracy on uci datasets

| Method | LETTER | ADULT | YEAST | B.C.W. | Parkinsons |
|---|---|---|---|---|---|
| daForest | **98.10%** | **86.13%** | **65.02%** | **97.62%** | **86.44%** |
| gcForest | 97.47% | **86.13%** | 63.45% | **97.62%** | 84.75% |
| SVM (linear kernel) | 86.75% | 76.38% | 56.95% | 96.67% | 77.97% |
| SVM (rbf kernel) | **97.90%** | 76.38% | 56.50% | 96.67% | 79.67% |
| MLP | 95.70% | 85.25% | 55.60% | 96.19% | 74.58% |
| Logistic Regression | 72.30% | 79.96% | 52.47% | 96.67% | 83.05% |
| Random Forest | 96.80% | 85.06% | 61.66% | 97.04% | 83.05% |

### 4.5 Running Time

All experiments were carried out on a personal computer with an AMD Ryzen1400 CPU (4 cores) and 16G RAM. In experiments, we disable early-stopping mechanism to take a closer look at the convergence process of the proposed model and force the model to predict probabilistic features for both training and testing samples after each epoch, so, the time cost is actually much higher than pure training process. Even so, time expenditures of daForest are acceptable, on Amazon Commerce Reviews, daForest converges after 4 epochs (layers) and about 93 seconds per epoch. On CNAE-9 dataset, daForest converges after 50 epochs and about 19.2 seconds per epoch. It takes 25 epochs to converge on IMDB dataset, each layer takes about 15 minutes to complete the training and prediction process, it is noteworthy that training and prediction time cost of gcForest can be reduced to 4 minutes per epoch on PC with 2 Intel E5 2695 v4 CPUs as reported in [Zhou, 2017] on IMDB, and time expenditures of daForest and gcForest are quite the same. When the optimization layer is activated as described in Sec. 3.4, training and prediction time cost can be nearly reduced to half the original. It takes 10 minutes to converge on Adult dataset compared with 30 minutes when the optimization layer is activated, 2.5 minutes on Yeast compared with 5 minutes and 2 minutes compared with 4.5 minutes on B.C.W. dataset. As is known, random forest is an inherently parallel learning model, daForest, which is made up of random forest and extremely random forest, also has very good level of parallel computation, further improvements on daForest can greatly reduce the training time.

## 5 Conclusion

It is widely recognized that deep model is good at exploiting high-order features. The most popular deep models are still deep neural networks, which are the killer architectures in image and speed recognition domains. However, in many fields, the most frequently used datasets are still small scale or space/time uncorrelated datasets, where deep neural networks cannot be fully applied. Therefore, it is very interesting and necessary to explore other deep architectures. gcForest is such a pioneer work in this direction, which achieve highly competitive performance compared with traditional learning model and deep neural networks. But gcForest is not a perfect model, because degeneration limits the expression ability and depth of the model, accuracy drops in few epochs, it is very hard to push the model to go deeper. In this paper, we propose a new deep ensemble model, which we refer to as Dense Adaptive Cascade Forest (daForest), by employing boosting procedure and dense connectivity, the proposed model alleviates degeneration to a large extent and tent to yield consistent accuracy improvement with growing number of layers. daForest achieved state-of-the-art results across several datasets. In addition, daForest has less training and prediction time when the optimization layer is activated, which is very crucial when training set is large and running time is sensitive.

There are several directions for future work. First, we can modify the proposed model and apply it to image recognition, for example, add several convolutional layers to extract high-order features and replace fully connected layer with daForest. Second, ordinary decision trees use hard splits in each internal node, which is undifferentiable, thus, we need to train the whole model layer by layer and push it to go deeper by employing boosting procedure and dense connectivity, however, layer by layer training is not an effective method to construct a deep model, we plan to use soft splits in each decision tree to achieve end-to-end training with daForest in future work.

## References

1. G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke,P. Nguyen, T. Sainath, and B. Kingbury (2012) Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine 29(6):82–97.
2. A. Krizhenvsky, I. Sutskever, and G. Hinton (2012) ImageNet classification with deep convolutional neural networks. In NIPS, pages 1097–1105.
3. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989) Backpropagation applied to handwritten zip code recognition. Neural computation 1(4):541–551.
4. I. Goodfellow, Y. Bengio, and A. Courville (2016) Deep Learning. MIT Press, Cambridge, MA.
5. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner (1989) Gradient based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324
6. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh,S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al (2014) Imagenet large scale visual recognition challenge. IJCV.
7. David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams (1986) Learning representations by back-propagating errors. Nature 323: 533–536.
8. Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642.
9. D. Silver, A. Huang, C.J. Maddison, A. Guez et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529: 484–489.
10. D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou et al (2017) Mastering the game of go without human knowledge. Nature 550: 354–359.
11. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014): 1929-1958

12. Federico Girosi , Michael Jones, Tomaso Poggio (1995) Regularization Theory and Neural Networks Architectures. Neural Computation 7(2): 219-269.
13. Dong Yu, Kaisheng Yao, Hang Su, Gang Li, Frank Seide (2013) KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. Acoustics, Speech and Signal Processing (ICASSP).
14. Zhi-Hua Zhou, Ji Feng (2017) Deep Forest: Towards An Alternative to Deep Neural Networks. International Joint Conference on Artificial Intelligence(IJCAI).
15. Z.-H. Zhou (2012) Ensemble Methods: Foundations and Algorithms. CRC, Boca Raton, FL.
16. Sinno Jialin Pan, Qiang Yang (2010) A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22(10): 1345-1359.
17. Y. Ganin, V. Lempitsky (2015) Unsupervised Domain Adaptation by Backpropagation. arXiv:1409.7495v2.
18. M Long, Y Cao, J Wang, MI Jordan Learning (2015) transferable features with deep adaptation networks. arXiv:1502.02791.
19. Sussillo, D. Barak, O. Opening (2013) the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. Neural Computation 25(3): 626–649.
20. Lior Rokach (2010) Ensemble-based classifiers. Artificial Intelligence Review 33(1-2): 1-39.
21. Karen Simonyan, Andrew Zisserman (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556v6.
22. K. He, X. Zhang S. Ren, J. Sun (2016) Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition: 770-778.
23. Leo Breiman (2001) Random Forests. Machine Learning 45(1): 5-32.
24. Geurts, Pierre, Damien Ernst, Louis Wehenkel (2006) Extremely randomized trees. Machine learning 63(1): 3-42.
25. Freund, Yoav, and Robert E. Schapire (1997) A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences 55(1): 119-139.
26. Freund, Yoav, and Robert E. Schapire (1996) Experiments with a new boosting algorithm. International Conference on Machine Learning.
27. Schapire, Robert E., and Yoram Singer (1999) Improved boosting algorithms using confidence-rated predictions. Machine learning 37(3): 297-336.
28. Hastie, Trevor, et al (2009) Multi-class adaboost. Statistics and its Interface 2(3): 349-360.
29. H. Gao, Z. Liu, L. van der Maaten (2017) Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition 1(2): 3-12.
30. Criminisi, Antonio, Jamie Shotton (2013) Decision forests for computer vision and medical image analysis. Springer Science & Business Media.
31. Viola, Paul, Michael Jones (2001) Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition.
32. P. Kontschieder, M. Fiterau, A. Criminisi, S.R. Bulo (2015) Deep Neural Decision Forests. IEEE International Conference on Computer Vision.
33. S.R. Bulo, P. Kontschieder (2014) Neural Decision Forests for Semantic Image Labelling. IEEE conference on computer vision and pattern recognition.
34. G.E. Hinton, S.Osindero, Yee-Whye The (2006) A fast learning algorithm for deep belief nets. Neural computation 18(7)：1527-1554.
35. Mnih, Volodymyr, Nicolas Heess, Alex Graves (2014) Recurrent models of visual attention. Advances in neural information processing systems.
36. Bahdanau, Dzmitry, Kyunghyun Cho, Yoshua Bengio (2014) Neural machine translation by jointly learning to align and translate. International Conference on Learning Representations.
37. K. Xu, J.L. Ba, R. Kiros, K Cho, et al (2015) Show, attend and tell: Neural image caption generation with visual attention. International Conference on Machine Learning. 2015.

38. Sanya Liu, Zhi Liu, Jianwen Sun, Lin Liu (2011) Application of Synergetic Neural Network in Online Writeprint Identification. JDCTA: International Journal of Digital Content Technology and its Applications 5(3): 126 ~ 135.
39. Patrick Marques Ciarelli, Elias Oliveira (2009) Agglomeration and Elimination of Terms for Dimensionality Reduction. Ninth International Conference on Intelligent Systems Design and Applications: 547-552.
40. Patrick Marques Ciarelli, Elias Oliveira, Evandro O. T. Salles (2010) An Evolving System Based on Probabilistic Neural Network. Brazilian Symposium on Artificial Neural Network.
41. Saleh, Ahmed Abdelfattah, Li Weigang (2015) A new variables selection and dimensionality reduction technique coupled with simca method for the classification of text documents. Proceedings of the MakeLearn and TIIM Joint International Conference, Make Learn and TIIM: 583-591.
42. K. Bache, M. Lichman (2012) UCI Machine Learning Repository (University of California, School of Information and Computer Science.). http://archive.ics.uci.edu/ml. Accessed 03 August 2012.
43. A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts (2011) Learning word vectors for sentiment analysis. Association for Computational Linguistics (ACL): 142–150.
44. Y. Kim (2014) Convolutional neural networks for sentence classification. arXiv:1408.5882.