Deep Coupling of Random Ferns

Sangwon Kim, Mira Jeong, Deokwoo Lee, Byoung Chul Ko
Dept. of Computer Engineering
Keimyung University, Daegu, Korea 42601
{eddiesangwonkim, mystroll24}@gmail.com,{dwoolee,niceko}@kmu.ac.kr
http://cvpr.kmu.ac.kr

Abstract

The purpose of this study is to design a new lightweight explainable deep model instead of deep neural networks (DNN) because of its high memory and processing resource requirement as well as black-box training although DNN is a powerful algorithm for classification and regression problems. This study propose a non-neural network style deep model based on combination of deep coupling random ferns (DCRF). In proposed DCRF, each neuron of a layer is replaced with the Fern and each layer consists of several type of Ferns. The proposed method showed a higher uniform performance in terms of the number of parameters and operations without a loss of accuracy compared to a few related studies including a DNN based model compression algorithm.

1. Introduction

Recently, deep neural networks (DNNs) have been very successful in many applications of supervised learning such as visual and speech recognition. DNNs are very powerful recognition algorithms, but it also has many limitations.

In terms of algorithms, DNNs requires a large number of hyper-parameters such as learning rate, cost function, normalization, mini-batch, etc. Moreover, learning performance of DNNs depends seriously on careful parameter tuning to generalize effectively and it requires large amounts of learning data to avoid overfitting. The biggest problem is that DNNs is a black box model, so it is difficult to understand the learning process except for incoming and outgoing data [1][2]. In addition, the structure of the model should be determined before DNNs is learned.

In terms of system resource, DNNs requires extremely expensive computational resource to train multiple layers of nonlinear activation functions based on backpropagation. Therefore, DNNs cannot operate training medium-to-large network on a single CPU because it uses large amounts of memory as well as long training time [2]. In particular, because wide and deep top-performing networks are not well suited for applications with memory or time limitations, some researches [3] [4][5] [6] have been tried to reduce the convolution layer of DNNs for improving the compression rate as well as the overall speedup. However, compressed DNN models still demands high memory for a large amounts of parameters and processing resource for multiplication [6].

To build a lightweight, explainable deep model that does not require backpropagation while maintaining the superior performance of DNN, [1] and [2] proposed a deep forest (DF) [1] and deep random forest (DRF) [2] that combines decision trees. In these researches, each neuron of a layer is composed of the individual decision trees and each layer is considered as a type of random forest. This random forest based networks can be quickly trained layer-by-layer instead of paying the high computational cost of training a DNN all at once [2]. These decision tree ensemble approaches have much fewer hyper-parameters than a DNN, and complexity can be determined automatically in a data-dependent manner. Experimental results show that the performance is quite robust to hyper-parameter setting and gave excellent performance compared to DNN based methods [1].

In general, ensemble of multiple random forests [1] or multiple decision trees [2] are used for individual layer of the deep architecture and the output of random forest or decision trees are passed to subsequent layers. The main advantage of the decision tree ensemble is that it achieves higher tree diversity by choosing features for the split function in a different way and improve generalization by combining randomized decision trees compared to single decision tree.

The decision tree creates a split function based on information gain for each node to grow the tree. However, such a node evaluation scheme has a problem of decreasing the diversity of the tree and increasing the generalization error of the tree ensemble. Therefore, this paper uses the random ferns (RFs) to make individual layer of a deep model instead of random forest because of following reasons:

- One Fern works the same as one decision tree does. The decision tree classifies samples into a top-down form, but Fern classifies samples into a combination of binary tests.
- Ferns do not perform evaluations regarding the binary tests that comprise the Ferns.

- Binary tests used in RFs can improve diversity because they are selected completely randomly.
- RFs is particularly easy to implement, does not overfit, does not require ad hoc parameter tuning, and allows fast and incremental training [7].
- Ferns are just as reliable as the randomized trees but much faster and simpler to implement [7].

Because the performance of the RFs is based on the fact that when the tests are chosen randomly, the power of the approach derives not from the tree structure itself but from the fact that combining groups of binary tests allows improved classification rates.

In proposed deep coupling RFs (DCRF), each neuron of a layer is replaced with the Fern and each layer consists of several type of Ferns. Let's assume that the value of features f_n are the outcome of a simple binary test only depends on the intensities of two pixel locations $\mathrm{d}_{n,1}$ and $\mathrm{d}_{n,2}$ of the input image I,

$$f_n = \begin{cases} 1 & if\ I(\mathrm{d}_{n,1}) < I(\mathrm{d}_{n,2}) \\ 0 & otherwise \end{cases} \quad (1)$$

Since features are very simple, many features are needed for accurate classification. However when the number of features is too large, learning the joint likelihood distributions over all features is most likely intractable. Therefore, Naive Bayes makes the simplifying assumption that features are conditionally independent given the class label C_k. However, although it is usually easy to learn the 1-d conditional densities $P(f_i|C_k)$, this independence assumption is usually false and the resulting approximation tends to grossly underestimate the true posterior probabilities. Therefore, Özuysal [7] et al. grouped features consist of L small sets of size S to make the problem tractable while accounting for these dependencies. These groups are called as a Fern, $\mathrm{F}_l = \{f_{l,1}, f_{l,2}, \dots, f_{l,S}\}$ and we can compute the joint probability for features in each Fern by assuming that groups are conditionally independent called Semi-Naïve Bayes:

$$\mathrm{P}(\mathrm{f}_1, \mathrm{f}_2, \dots, \mathrm{f}_N|C_k) = \prod_{l=1}^{L} P(F_l|C_k) \quad (2)$$

A single Fern does not give great classification performance, but we can improve the classification performance by building an ensemble of independent Ferns by randomly choosing different subsets of features. From the Eq. (2), we learn the class-conditional distributions for each Fern and apply Bayes rule to obtain posterior by combining their outputs using Semi-Naive Bayes,

$$\mathrm{Class}(\mathbf{f}) \equiv \arg\max_k P(C_k) \prod_{l=1}^{L} P(F_l|C_k) \quad (3)$$

At test time, each Fern consists of a small amount of binary test returns the probability for the input vector, which belongs to one of the learned classes during learning. These responses of Ferns are combined with the Naïve Bayes and finally classified into the class having the largest probability value.

2. Cascading of Multi-layer Ferns

The proposed DCRF is composed of an encoding model and a cascade forest model inspired by DF [1]. The encoding model transforms an input image into class probability features using multi-grained scanning, and a cascade Fern layers is used to estimate the final class by applying the transformed feature vector to the cascade of a cascade Ferns. DF [1] and DRF [2] apply multi-grained scanning (MGS) to the input image in order to extract feature vectors for the input of the first layer. These methods use the three window sizes, and a feature vector of each size is applied to each corresponding the random forests. The output vector of the first layer is the concatenation the results of each random forests. However, one random forest of DF [1] must consist of 500 trees, and four random forests must form one layer. Therefore a DF [1] is not suitable for a real-time applications because it still has a deep and wide structure, which is difficult to process quickly and is optimized only for simple recognition problems. DRF [2] consists of only two layers consisting of 2,000 decision trees per a single layer without the use of several random forests. However, this method also has a disadvantage in that the operation speed is slow owing to an excessive number of trees.

In this paper, raw features are extracted by using one size of scanning window instead of MGS which requires many computation time and parameters.

As Fig. 1 indicates, the proposed DCRF model consists of a layer-to-layer structure. The role of the first layer is to transform raw features into class probabilities, and these probability outputs are concatenated as a single transformed feature vector for the new input of the next layer. Input feature vector extracted from raw gray pixels are applied to two RFs composed of 16 Ferns which including 20 binary tests, respectively.

From the second layer, each layer is used to generate a new feature vector for the next layer or to predict the final class at the final layer. In detail, one layer consists of five RFs, and one RF consists of 18 Ferns including 11 binary tests.

With the proposed DCRF, each neuron of a DNN layer is replaced with the RF, and each layer consists of several types of four RFs. Each layer consists of randomly generated heterogeneous RFs to encourage diversity and maintain the generality, which is a similar method as a DF [1]. In a DRF method, the transformed feature vectors, augmented using the class vector generated by the previous layer, will then be applied to train the second and third layers of the cascade forests, respectively. However, because the correlation between the newly generated output vector and the augmented transform vector is weak, it may result in a slowing convergence and degrade the performance during the test. Therefore, in this study, we designed a model that uses only the output features of the previous layer as the new input features of the next layer

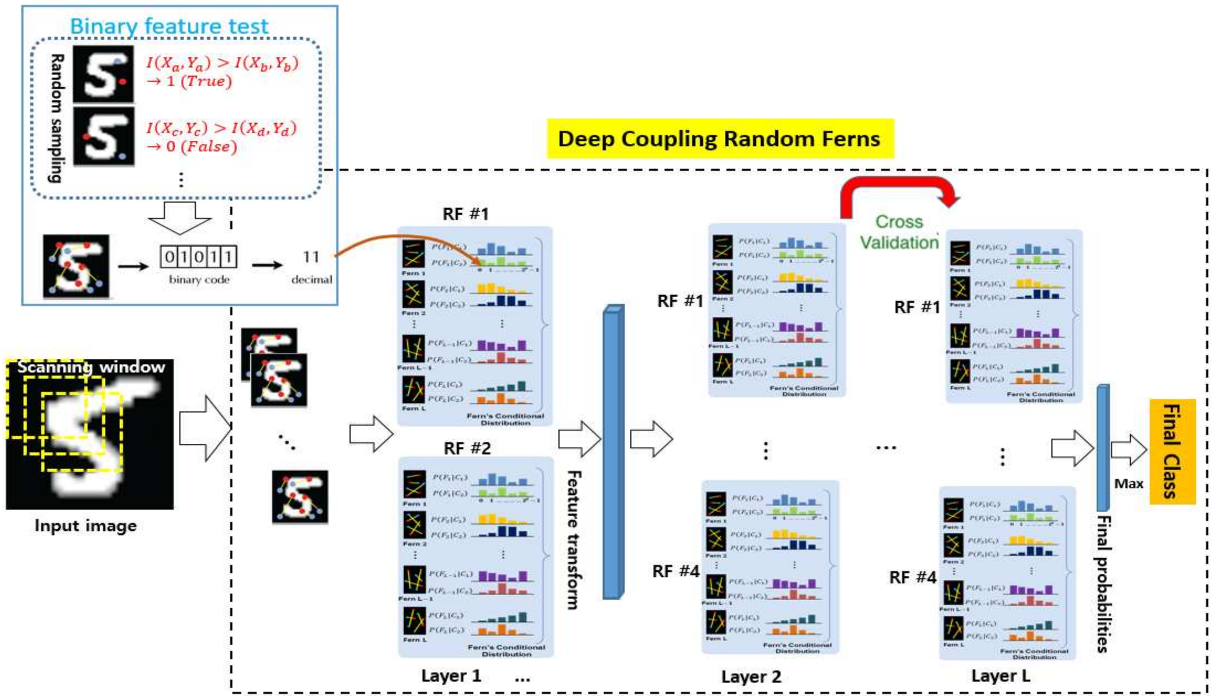

Fig. 1. Deep coupling random ferns model using one scanning window for features extraction and layer-by-layer structure consisting of several random ferns. The outputs of each layer are concatenated and input to the next layer until the data are mapped to the final layer. The final layer averages the probability values of each class and determines the final class with the highest class probability value.

without combining the transformed feature vector.

During the learning process, the output vector of one layer is successively the input vector of the next layer. The decision to add a new layer to the DCRF depends on whether the validation performance converges. To automatically determine the number of layers and parameters while reducing the risk of an overfitting, we use five-fold validation.

After finishing the training of the DCRF, given a test image, we first extract raw feature vectors from a given window size and then input individual feature vector into the first layer one-by-one. The outputs of the first layer are concatenated, and these transformed feature vectors, augmented with the class vector generated by the first layer, are input into the next layer until the data are mapped to the final layer. The final layer averages the probability values of each class and determines the class with the highest probability value as the final class.

3. Experiment Results

There are many benchmark databases for evaluating the DNN. In this paper, we use the MNIST handwriting digit recognition dataset, which is commonly used in many DNN related studies, for the fair performance evaluation of the proposed DCRF with other related methods.

In this experiment, we compared accuracy, the number of parameters, and operations with the state-of-the-arts such as DF [1], DRF [2], and MobileNet [4] which are popular model compression method using the MNIST dataset.

As shown in Table I, the proposed method is similar to MobileNet in terms of the number of parameters. However, the accuracy and the number of operation of the proposed method are superior to that of MobileNet. Moreover, unlike MobileNet, the DCRF method has the advantage of a CPU- rather than GPU-based operation.

TABLE I. COMPARISON BETWEEN THE NUMBERS OF PARAMETERS AND OPERATIONS FOR THE PROPOSED METHOD AND FOUR COMPRESSION MODELS USING THE MNIST DATASET.

Methods	**Operat ion**	**Accuracy (%)**	**No. of parameters (M)**	**No. of operations (M)**
MobileNet [4]	GPU	96.5	1.32	76
DF [1]	CPU	98.96	4.15	0.105
DRF [2]	CPU	98.89	3.78	0.061
Proposed DCRF	CPU	**98.0**	**1.8**	**0.016**

Among DF and DRF-based methods, DRF [2] requires smaller number of parameters and operations compared to DF [1] because of its smallest number of trees. However, accuracy of DRF [2] is 0.07% lower than the DF. Although the proposed DCRF has 0.89% and 0.96% lower accuracy than DF and DRF, it requires about 2.1~2.3 times and 10.1~6.6 times less parameters and operations than DF and DRF. In terms of memory and operation, the proposed DCRF method can be optimized for embedded systems. In terms of accuracy, the proposed DCRF is 0.96% lower accuracy than DF because we used only two layers. However, if we add more layers, we can expect better performance. However, when compared with MobileNet, it shows a 1.5% improvement in performance, which means that it can be used efficiently in a CPU-based embedded system.

4. Conclusion

Although DNN is a powerful algorithm for classification and regression problems, it requires too many parameters, a careful parameter tuning, a huge amount of training data, black-box models, and a pre-trained architecture. Therefore this study proposed a DCRF architecture which is a non-neural network style deep model. In the proposed DCRF, we assigned only 90 Ferns to an individual RF to reduce the number of parameters, and we proved that the recognition accuracy is improved based on a performance evaluation conducted using a few state-of-the-art DF, DRF, and DNN-based model compression algorithm. The proposed method showed a higher uniform performance in terms of the number of parameters and operations without a loss of accuracy. Future research will improve the number of layers of the model and reduce the number of Ferns while improving similar performance to existing Deep model and apply it to various application studies.

Acknowledgments
This work (Grants No. S2689892) was supported by project for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Ministry of SMEs and Startups in 2018 and partially supported by the Scholar Research Grant of Keimyung University in 2018.

References

[1] Z.H. Zhou, J. Feng, Deep forest: towards an alternative to deep neural networks, Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 3553-3559, 2017
[2] K. Miller, C. Hettinger, J. Humpherys, T. Jarvis, D. Kartchner, Forward Thinking: Building Deep Random Forests, arXiv:1705.07366, 2017.
[3] G. Hinton, O. Vinyals, J. Dean, Distilling the Knowledge in a Neural Network, NIPS Deep Learning Workshop , pp.1-9, 2014
[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko,W., Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
[5] F. Tung, G. Mori, CLIP-Q: Deep Network Compression Learning by In-Parallel Pruning-Quantization, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7873-7882
[6] S. J. Kim, S. Y. Kwak, B. C. Ko, Fast Pedestrian Detection in Surveillance Video Based on Soft Target Training of Shallow Random Forest, IEEE ACCESS, Vol. 7, pp.12415-12426, Jan. 2019
[7] Mustafa Özuysal, Michael Calonder, Vincent Lepetit and Pascal Fua, Fast Keypoint Recognition using Random Ferns, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(3): 448-461, 2010
[8] Y. Lecun, L. Bottou, Y. Bengio, P. Haffne, Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86(11):2278-2324, 1998.