Interpretable Neural Network Construction: From Neural Network to Interpretable Neural Tree

Xuming Ouyang¹, Cunguang Feng²

¹School of Software, Shandong University, Jinan, Shandong, 250101, China
²School of Information Science and Technology, Peking University, Beijing, 100871, China

Corresponding author’s e-mail: oyxm@mail.sdu.edu.cn

Abstract. The neural network has made outstanding achievements in many fields, while comparing with traditional machine learning models, the neural network has poor interpretability, which brings great limitation to its practical application. Therefore, many researchers try to combine neural networks with traditional models to improve the interpretability of the neural network. Their methods either result in performance depreciation or lead to computation-intensive. In this paper, we propose to transform a neural network into the interpretable neural tree. In the interpretable neural tree, each node contains transformation function and routing function. Each transformation function corresponds to a layer in the neural network, using for controlling data transformation. The routing function is utilized to control the direction of data flow in the tree structure. Our experiments have indicated that the interpretable neural tree makes the neural network interpretable to some extent while maintaining the performance.

1. Introduction
With the development of deep learning, a neural network with various architecture has shown its capabilities in different fields such as computer vision, natural language processing, and speech recognition and processing. Therefore, it is further applied to products and bringing practical convenience to people’s lives.

The neural network is treated as a black-box since it was hard to interpret, and this becomes an obstacle to the application of neural networks in certain fields. Take the medical field as an example, from the task of computer vision, it is possible to identify lesions in medical images with the help of neural networks. From the task of natural language processing, it can be utilized to perform preliminary screening for diseases based on patient symptoms. Applications in the medical field require not only high accuracy but also interpretability. Interpretability can help people find problems in the model and design better architectures to achieve better performance. Good interpretability and interactive ability can both help people integrate expert knowledge into the model and reveal the patterns learned by the model. Although the neural network has achieved remarkable results in the above tasks, it cannot be applied to the actual diagnosis and treatment process. Another case is the applications of driverless. In fact, such examples are endless. Based on the above description, the research on the interpretability of neural networks is of great significance for the promotion of neural networks to real life.

In this paper, we propose to make the neural network be interpretable by means of transforming a trained neural network to a complete binary tree, the node of which is transformation function and each edge is a routing function. The former determines the data transformation during the flow of data in the
tree, which is designed to maintain the performance of the binary decision tree, while the latter determines the direction of data flow, which can explain the output by revealing the routing paths of data. The experimental result demonstrated that the method we proposed can make the corresponding neural network interpretable to some extent while maintaining the performance.

2. Related Work

Many traditional machine learning algorithms are inherently well interpretable [1], while deep neural networks are not interpretable in terms of linearity, monotonicity, and interactivity. To be well interpretable, it is often combined with the traditional machine learning method. Among them, many works attempt to introduce the idea of decision trees into deep neural networks to improve interpretability.

In 2017 Hinton et al. [2] proposed a binary soft decision tree, the core idea of this soft decision tree is to rely on hierarchical decisions to make it interpretable. Each routing decision uses a linear transformation to extract information from the original data space. However, the learning ability of this soft decision tree is too weak to be suitable for the situation where the data is complex. Other work adds transformation functions to improve performance. In 2015, Kontschieder et al. [3] used GoogLeNet [4] with the last layer removed as a transformation function and combined it with a soft decision tree to achieve amazing results on ImageNet [5]. However, this method uses the entire deep convolutional network as a feature extractor and is not fully interpretable. Besides, Tanno et al. proposed an adaptive neural tree [6] in 2018. The experimental results show that the performance has been significantly improved, but the amount of architecture search calculations is too prohibitive.

3. Interpretable Neural Tree

To make the neural network interpretable to some extent while maintaining the performance, in this paper, we propose to transform the trained neural network into a binary tree. Each node in the INT refers to a transformation function and each edge refers to a routing function. In this part, we will introduce the transformation function and routing function in detail, respectively.

3.1. The transformation from a neural network into a binary tree

We write a neural network with D parts as Equation (1). \( x \) refers to the input and \( l_i \) refers to the \( i \)-th layer of the model. It can be several convolutional layers in a convolutional neural network, it can also be a residual block in ResNet.

\[
NN = l_D(\ldots l_2(l_1(x)) \ldots) \tag{1}
\]

We transfer the given neural network into a tree with \( 2^{D-1} \) leaves, which is a complete binary tree. For convenience, we call it the interpretable neural tree (IPNT) in the following of the paper. In IPNT, \( t_{d,i} \) is the transformation function for \( i \)-th nodes in layer \( d \) and the corresponding input is \( x_{d,i} \). For the \( i \)-th node in layer \( d \), the transformation process can be written as follows,

\[
x_{d+1,i+2} = t_{d,i}(x_{d,i}) \tag{2}
\]

The routing function is utilized to determine the probability of going to the left node or right node. In this paper, the routing function is a linear combination of \( x_{d,i} \). Similar to transformation function, \( r_{d,i} \) is the routing function for \( i \)-th nodes in layer \( d \) as described in Equation (3). All of the \( W \), \( b \) and \( \beta \) are parameters and \( \sigma \) refers to sigmoid. Then probability to left sub-tree \( p_{l_{d,i}} \) and right sub-tree \( p_{r_{d,i}} \). Furthermore, \( p_{r_{d,i}} \) can be calculated as \( 1 - p_{l_{d,i}} \).

\[
p_{l_{d,i}}(x_{d,i}) = \sigma(\beta_{d,i}(W_{d,i}x_{d,i} + b_{d,i})) \tag{3}
\]

Then the probability from the root the to \( i \)-th leaf node can be written as \( p_i \) and the \( q_i \) refers to the corresponding output of the \( i \)-th leaf node. It should be noticed that the form of \( p_i \) can be different with different tasks. For a regression task, \( p_i \) is a value, while for a classification task, \( p_i \) is a vector. Then
the output of the IPNT can be calculated by $p_1, p_2 \ldots p_n$ together with $q_1, q_2 \ldots q_n$, where $n$ is the number of leaf nodes in IPNT.

3.2. Loss function and imbalance penalty
In IPNT, the loss function depends on the loss reflected by all the leaves. Take the cross-entropy loss function in the classification task as an example. The loss can be calculated as Equation 4 and Equation 5. If data of all samples flows along only one path in the tree, then IPNT degenerates into a neural network. To avoid this, we set imbalance penalties $w$ as shown in Equation 5.

$$L_i(x) = -\sum_{k=1}^{m} r^k \log q_i^k$$ (4)
$$L(x) = \sum_{i=1}^{n} p_i L_i(x) + \omega$$ (5)

where $n$ refers to the number of leaves and $m$ refers to the number of categories. Given input $x$, $L_i(x)$ calculate the cross entropy loss in the $i$-th leaf. The imbalance penalty $w$ used to prevent the imbalance in each node. Imbalance means excessive gap in sample numbers that flow to the left sub-tree and right sub-tree. Specifically, the imbalance penalty is calculated as Equation 6.

$$w = -\sum_{d_i} 2^{-d} \left( \frac{1}{2} \log a^{d,i} + \frac{1}{2} \log (1 - a^{d,i}) \right)$$ (6)

$$a^{d,i} = \frac{\sum_{x \in \text{batch}} P^{d,i}_x}{\sum_{x \in \text{batch}} P^{d,i}_x}$$ (7)

3.3. Hard decision and soft decision for interpretable neural tree
There are two methods for IPNT to forward propagation. One is soft decision. Given input, find the probability of reaching each leaf node $p_i$, and then weighted $q_i$ according to $p_i$ calculated by $\sum_{i=1}^{n} p_i q_i$ to get the final prediction. The other is hard decision. Comparing with the soft decision, a greedy strategy is adopted, the larger probability is selected to enter the left and right sub-trees. In the end, only one leaf node is reached. It’s obviously that the hard decision will save calculations comparing with the soft decision.

3.4. Training process
It takes 4 steps to obtain an IPNT. Firstly, we need to pre-train a neural network as the base model. Then the parameters of the transformation function in IPNT will be initialized according to the corresponding layers in the trained neural network. It should be noticed that the transformation function in the same layer of IPNT will be initialized exactly the same. The parameters of the routing function will be initialized randomly. Furthermore, we will fine-tune the parameters of routing function with transformation function fixed by end-to-end training of IPNT. Finally, both the parameters in transformation function and routing function will be fine-tuned to get the final IPNT.

4. Experimental Result and Analysis
In this paper, we try to use two groups of experiments to evaluate the IPNT. Both of the trained models will be transformed into an IPNT. In this part, we will firstly introduce the learning setup. Then the result of the experiment on both hard decision and soft decision will be demonstrated in Section 4.2. Finally, the interpretability analysis of the IPNT will be introduced in Section 4.3.

4.1. Learning setup
In the first experiment, we use a convolutional neural network (CNN) with three convolutional layers to classify the MNIST dataset. The other task is to use ResNet to classify the CIFAR10 dataset. The architecture of the corresponding IPNT is demonstrated in Figure 1. Besides, the data flow in IPNT is shown in Figure2.
Figure 1: The left architecture refers to the transformation from CNN to IPNT, the right architecture refers to the transformation from ResNet18 to IPNT.

Figure 2: The architecture of general IPNT. It should be noticed that the input $x_{1,1}$ can also be written as $x_1$.

For both of the two group experiments, the hard decision and the soft decision will be used in the transformation process to compare the performance. Besides, the parameters in transformation functions on the nodes of the same layer can choose to be shared or not. If it is not shared, the parameters of the transformation function of each node in the same layer are independent between each other. If the parameters are shared, the transformation function of each node in the same layer is the same.

In this paper, we will use classification accuracy to measure the performance of models. Besides, we will list the number of parameters on different datasets with different models to compare the calculation complexity. Particularly, cross-entropy with an imbalance penalty is used to quantify the prediction error. We select the optimal hyper-parameters on the validation sets for all of the models.

4.2. Experimental result

The comparison of the number of parameters and performance of different models is demonstrated in Table 1 and Table 2, respectively. In both of the tables, CNN and ResNet are used to classify MNIST and CIFAR10. In the tables, IPNT-SS and IPNT-SH refer to the soft decision IPNT and hard decision IPNT that node in the same layer shares parameters, respectively. In addition, IPNT-NS and IPNT-NH refer to the soft decision IPNT and hard decision IPNT without shares parameters for the same layer, respectively.

Table 1: The number of parameters in different models.

|            | CNN/ResNet | IPNT (Without Share) | IPNT (Share) |
|------------|------------|-----------------------|--------------|
| MNIST      | 27.5k      | 111k                  | 28.7k        |
| CIFAR10    | 11.2M      | 77.0M                 | 11.3M        |

Table 2: Performance of different models measured by accuracy.

|            | CNN/ResNet | IPNT-SS | IPNT-SH | IPNT-NS  | IPNT-NH  |
|------------|------------|---------|---------|----------|----------|
| MNIST      | 98.0%      | 97.83%  | 97.96%  | 97.66%   | 97.66%   |
| CIFAR10    | 92.24%     | 91.71%  | 91.63%  | 92.77%   | 92.73%   |

From the tables, we can get two conclusions. Firstly, it clearly shows that the performance of IPNT is almost the same as the neural network. Secondly, we can infer that we can reduce the number of parameters in IPNT by sharing the parameters and it will not result in significant performance degradation.
4.3. Interpretability analysis

In this part, we will discuss the interpretability of INPT. Take CNN as an example, this part tries to figure out the decision process of INPT when classifying the MNIST dataset.

We begin by analyzing the change of leaves during the training process of INPT. We can get the following conclusion from Figure3. Each graph in Figure3 corresponding to a leaf node in INPT. In each sub-figure, the abscissa represents the number of training rounds, and the ordinate represents the proportion of different categories in the output of leaf nodes. At the beginning of training, each leaf node will have various types of predictions, and the probability of each category is relatively average, while when training is completed, many leaf nodes only output predictions of one category. This phenomenon shows that IPNT can build connections between leaves and classification categories, which fully illustrated that IPNT can be interpretable to some extent.

By the decision process of INPT shown in Figure4, we can see that the tree is balanced, in the first layer of INPT, the sample predicted as 3, 5, 7, 8, 9 flow into the left sub-tree and the others flow into the right sub-tree. The situation is similar layer by layer. From the IPNT, we have noticed that some nodes are empty, which means no samples will flow into the leaf. Therefore, we can prune it as shown in Figure5. The operation of pruning can help to reduce calculation. Given the decision process of INPT, for a misclassified sample, we can easily find out where it begins to go wrong in the INPT.
The routing function is utilized to calculate the probability to go to the left sub-tree and right sub-tree. The probability can be viewed as confidence, if the probability to left and right are close to 50%, the decision for the direction may be not confidence, while if one of them is close to 100% or 0, the chosen may be of great confidence. By analyzing, we have discovered, for the nodes close to the root, the probability calculated by routing function tend to close to 50%, for the nodes close to the leaf, it tends to close to 100% or 0. This phenomenon is consistent with our intuition that the feature map close to the leaf owns more abstract features than the feature map close to the root. Based on this, we can get confidence information by the probability value calculated by the routing function to account for the misclassified samples.

5. Conclusion
To solve the problem of the interpretability of the neural network, in this paper, we propose to transform a neural network to the interpretable neural tree. In the interpretable neural tree, each node contains transformation function and routing function. The transformation function is used to perform layer-by-layer data transformation just as the neural network. The routing function is utilized to control the direction of data flow in the tree. Our experiments have indicated that the interpretable neural tree makes the neural network interpretable to some extent while maintaining the performance. Finally, we try to show the interpretability by visualizing the change of leaf node during the training process, visualizing the training process and analyzing the direction probability calculated by routing function.

Acknowledgments
The authors are supported by the National Key R&D Program of China 2018YFB1004100.

References
[1] Christoph, M. (2019). Interpretable Machine Learning (pp. 37-38). Retrieved from https://leanpub.com/interpretable-machine-learning
[2] Frosst, N., & Hinton, G. (2017). Distilling a neural network into a soft decision tree.
[3] Kontschieder, P., Fiterau, M., Criminisi, A., and Bul, S. R. Deep neural decision forests. In ICCV, 2015.
[4] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., & Anguelov, D., et al. (2014). Going deeper with convolutions.
[5] Deng, J., Dong, W., Socher, R., Li, L. J., & Li, F. F. (2009). ImageNet: a Large-Scale Hierarchical Image Database. 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA. IEEE.
[6] Tanno, R., Arulkumaran, K., Alexander, D. C., Criminisi, A., & Nori, A. (2018). Adaptive neural trees.