Visualizing Decision-making Process in Deep Neural Decision Forest: Supplementary Material

1. Summary

Here we provide more details about the training algorithm and hyper-parameters used in the experiments.

2. Model Training

2.1. Classification

For a splitting node $\mathcal{S}_i$, we denote nodes in its left and right sub-trees as node sets $\mathcal{S}_i^l$ and $\mathcal{S}_i^r$, respectively. We denote the probability of recommending the input $\mathbf{x}$ to a leaf node $\mathcal{L}_i$ as $\mathbb{P}(\mathcal{L}_i|\mathbf{x})$. The optimization target is to minimize the negative log-likelihood loss over the whole training set containing N instances $\mathbb{D} = \{\mathbf{x}_t, y_t\}_{t=1}^N$,

$$L(\mathbb{D}) = -\sum_{t=1}^{N} \log(\mathbb{P}(y_t|\mathbf{x}_t)) \tag{1}$$

where $\mathbb{P}(y_t|\mathbf{x}_t)$ is the predicted probability that the input belongs to class y_t (the y_tth entry in the final prediction vector $\mathbf{P}$). This loss function is differentiable with respect to each recommendation score s_i, and the gradient $\frac{\partial L(\mathbb{D})}{\partial s_i}$ can be computed as [1, 2],

$$\sum_{t=1}^{N}\left(\frac{\sum_{\mathcal{L}_j \in \mathcal{S}_i^r} \mathbf{p}_j(y_t)\mathbb{P}(\mathcal{L}_j|\mathbf{x}_t)}{(1-s_i)\mathbb{P}(y_t|\mathbf{x}_t)} - \frac{\sum_{\mathcal{L}_j \in \mathcal{S}_i^l} \mathbf{p}_j(y_t)\mathbb{P}(\mathcal{L}_j|\mathbf{x}_t)}{s_i\mathbb{P}(y_t|\mathbf{x}_t)}\right) \tag{2}$$

This gradient can be back-propagated into each splitting node to optimize its parameters. We use constant mapping function $\mathcal{M}_i(\mathbf{x}) = \mathbf{p}_i$ at each leaf node, which means we simply store the answers at the leaf nodes. When the splitting nodes are fixed, the answer at each leaf node can be updated iteratively [1],

$$\mathbf{p}_j^{T+1}(y) = \frac{1}{Q_j^T}\sum_{t=1}^{N} \frac{\mathbb{1}(y_t = y)\mathbf{p}_j^T(y_t)\mathbb{P}(\mathcal{L}_j|\mathbf{x}_t)}{\mathbb{P}(y_t|\mathbf{x}_t)} \tag{3}$$

where Q_j^T is a normalization factor to ensure $\sum_{y=1}^{|\mathbf{p}_j|} \mathbf{p}_j^{T+1}(y) = 1$. The gradient-based optimization and the leaf node update are carried out alternately to train the model.

2.2. Regression

For a multi-task regression dataset with N instances $\mathbb{D} = \{\mathbf{x}_t, \mathbf{y}_t\}_{t=1}^N$, we directly use the squared loss function,

$$L(\mathbb{D}) = \frac{1}{2}\sum_{t=1}^{N} ||\mathbf{P}_t - \mathbf{y}_t||^2 \tag{4}$$

Here we also assume simple leaf nodes which have constant mapping functions just as the classification case. Similarly, $\frac{\partial L(\mathbb{D})}{\partial s_i}$ is computed as,

$$\frac{\partial L(\mathbb{D})}{\partial s_i} = \sum_{t=1}^{N} (\mathbf{P}_t - \mathbf{y}_t)^T \left(\frac{\mathbf{A}_l}{s_i} - \frac{\mathbf{A}_r}{(1-s_i)}\right) \tag{5}$$

where $\mathbf{A}_l = \sum_{\mathcal{L}_j \in \mathcal{S}_i^l} \mathbb{P}(\mathcal{L}_j|\mathbf{x}_t)\mathbf{p}_j$ and $\mathbf{A}_r = \sum_{\mathcal{L}_j \in \mathcal{S}_i^r} \mathbb{P}(\mathcal{L}_j|\mathbf{x}_t)\mathbf{p}_j$. Similar to (3), we update the leaf node answer as

$$\mathbf{p}_j^{T+1} = \frac{\sum_{t=1}^{N} \mathbf{y}_t \mathbb{P}(\mathcal{L}_j|\mathbf{x}_t)}{\sum_{t=1}^{N} \mathbb{P}(\mathcal{L}_j|\mathbf{x}_t)} \tag{6}$$

This update rule is inspired from traditional regression trees which compute an average of target vectors that are routed to a leaf node. Here the target vectors are weighted by how likely it is recommended into this leaf node.

3. Detailed settings and hyper-parameters

3.1. Classification on MNIST and CIFAR-10

MNIST dataset contains 60k training images and 10k testing images of size 28 by 28. CIFAR-10 dataset contains 50k training images and 10k testing images of size 32 by 32. CNN is used to extract features from the input, and every splitting node is randomly assigned to one neuron at the last layer of the CNN. The recommendation score s_i is computed as $sigmoid(f_i(\mathbf{x}))$ where f_i is the function represented by the neuron assigned to the splitting node i. The model architecture used for MNIST and CIFAR-10 dataset are shown in Fig. 2 and Fig. 1, where a more complex feature extractor is used for CIFAR-10. We use one tree of depth 9 for both experiments. Adam optimizer is used with

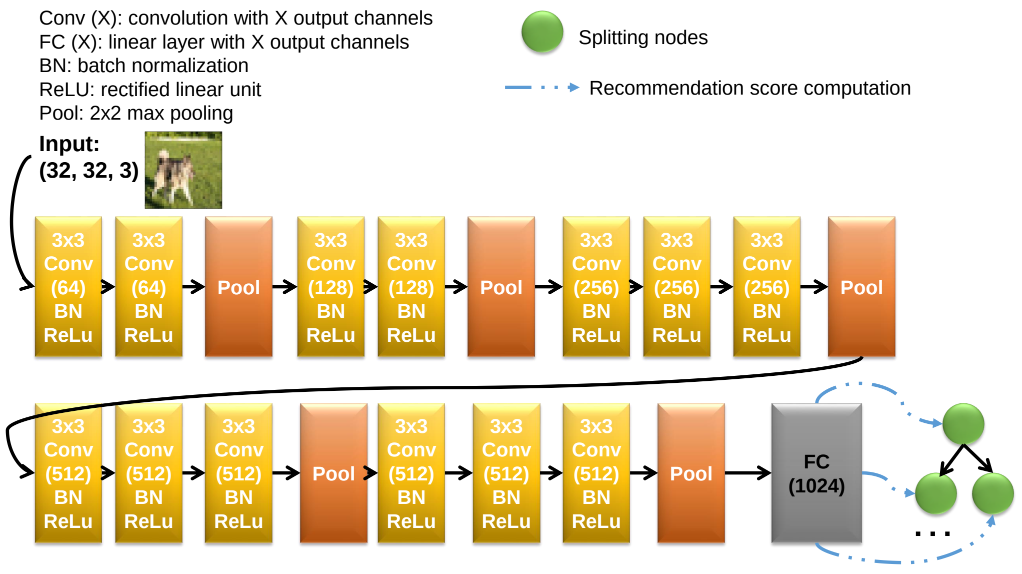

Figure 1. Network architecture (VGG16) for the classification experiment on CIFAR-10. Only 3 splitting nodes are drawn for simplicity.

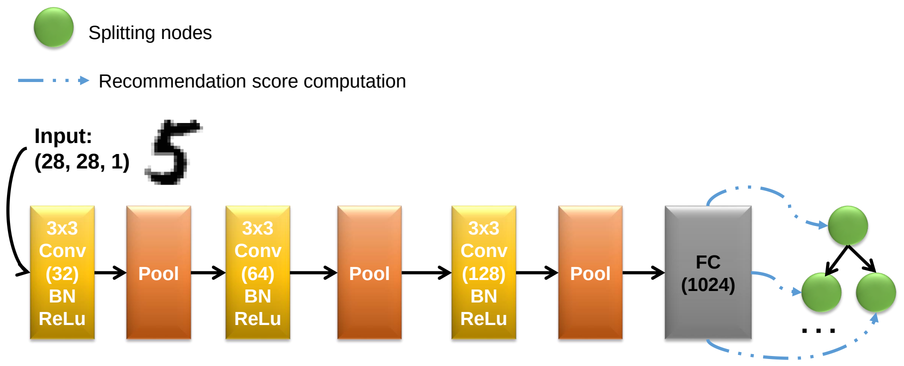

Figure 2. Network architecture for the classification experiment on MNIST.

initial learning rate set as 0.001 and 1e-5 weight decay. For the MNIST experiment, the model is trained 100 epochs and learning rate is halved after 30, 60 and 90 epochs. For the CIFAR-10 experiment, the model is trained 350 epochs and learning rate is multiplied by 0.3 after 150 and 250 epochs.

3.2. Cascaded regression on 3DFAW

3DFAW dataset contains 13670 training images and 4725 validation images. The images are cropped around the facial regions and resized to 256 by 256 pixel. We use a cascade of 10 stages for this regression problem. At each stage, the input is the concatenated patches cropped around the current estimated facial landmarks. The regression target is the difference between the ground truth and current estimated landmark coordinates. During training, we augment each image with 10 random facial shape initialization. During testing we use the mean shape computed from the dataset annotations as the initial shape. For each stage we train an ensemble of 3 trees and each is of depth 5. Adam optimizer is used with initial learning rate set at 0.01. Each stage is trained 6 epochs and the learning rate is halved after 2 and 4 epochs. When plotting the distribution of recommendation scores, we sample from all splitting nodes and all validation images.

References

[1] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bul. Deep neural decision forests. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1467–1475, Dec 2015. 1

[2] W. Shen, Y. Guo, Y. Wang, K. Zhao, B. Wang, and A. L. Yuille. Deep regression forests for age estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1