Using information theoretic metrics to study the importance of individual neurons in DNNs

Author: Kr.Cen 康瑞 岑

2023 年 6 月 10 日

目录

1 Abstract

This paper mainly refers to the paper Understanding Neural Networks and Individual Neuron Importance via Information-Ordered Cumulative Ablation, the first of my reference list. On the basis of this paper, I conducted an extended experiment on the MNIST dataset.

In this paper, I use **information theoretic metrics** for node pruning to learn the importance of **individual neurons** at different levels in the whole DNN. Entropy, Mutual information and KL-Selectivity are used to determine the order of **ablation**. I extend the 2-layer model adopted in the original paper to the 3-layer for more insights. When the number of hidden layers increases, more generalized and reliable conclusions can be drawn: 1. The distribution of the proposed metrics changes from layer to layer. 2. Hypotheses about the interactions of neurons based on information metrics can be formulated. 3. Deeper layers may have larger redundancy. 4. It is reasonable to use **mutual information** and **KL-Selectivity** as indicators of node pruning, indicating that they are strongly correlated with the classification results, but not strictly increase with the depth because of overfitting and important features sharing.

2 Introduction

There are several challenges for interpretability of (deep) neural networks nowadays: 1. How to understand NN(Neural Networks) theoretically. 2. How to understand NN functionality. 3. Where to find interpretability of results. Especially when neural networks have high computational complexity, i.e., have large depth, the interpretability of DNN is hard to explain.

In traditional researches for interpretability of DNN, researchers usually try to understand how neural networks underlie the decision making process by visualizing the semantics of interneurons or by inferring the importance scores of the input/interneuron. However, this traditional direction can only understand what neural networks model on the surface, but can not explain and analyze the more essential expressive ability of neural networks. As a result, most of the current interpretability studies cannot be used in the design and training of feedback-guided neural networks.

In this paper, I took a different approach. I investigated the importance of individual neurons at different levels to the prediction accuracy of the entire neural network using three information theoretic metrics:

1. Entropy
2. Mutual Information
3. Kullback-Leibler Selectivity (which will be defined later)

To value the importance of a single neuron, it is obvious that a single point operation of the neural network is required, and the **cumulative ablation** method is used in this experiment (i.e. **node pruning**), the main steps of cumulative ablation:

1. Removing one or more neurons or a layer from the network. The removal can be done by setting the weights or activations of the selected neurons or layer to zero or by excluding them from the network architecture.
2. Evaluate the performance of the ablated network on the same task or dataset used for training. This evaluation can involve measuring metrics such as accuracy, loss, or any other relevant performance measure.

3 Propose Information Theory Metrics

Experiment Basics. We consider classifiication via fully-connected feed-forward NNs. $\mathcal{C}$ denotes classification set, where $|\mathcal{C}| = C$. Dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. x_i denotes the i-th input and y_i denotes i-th output. Denote $h_j^{(i)}(x_l)$ to be the output of the j-th neuron in i-th hidden layer if given the input x_l. b denotes the bias vector. $\sigma : R \to R$ denotes a non-linear activation function (usually ReLU), then we have

$$h_j^{(i)}(x_l) = \sigma(\sum_p w_{p,j}^{(i-1)} h_p^{(i-1)}(x_l) + b_j^{(i)}) \tag{1}$$

Let $\mathcal{Q} : R \to \mathcal{H}$ be a quantizer that maps outputs to a finite set $\mathcal{H}$. Let Y be a random variable over the set $\mathcal{C}$ of classes and $H_j^{(i)}$ a random variable over $\mathcal{H}$, i.e., the output of the j-th neuron of the i-th hidden layer.

Define the joint distribution of Y and $H_j^{(i)}$ via the joint frequencies of $\{(y_l, Q(h_j^{(i)}(x_l)))\}$ in the validation set. $\forall c \in \mathcal{C}, h \in \mathcal{H}$,

$$P_{Y,H_j^{(i)}}(c,h) = \frac{\sum_{l=1}^{N} 1[y_l = c, Q(h_j^{(i)}(x_l)) = h]}{N} \tag{2}$$

where $1[\cdot]$ is the indicator function.

Entropy. It's a matric that quantifies the uncertainty. Denote $H[\cdot]$ as entropy, then

$$H(H_j^{(i)}) = -\sum_{h \in \mathcal{H}} P_{H_j^{(i)}}(h) \log P_{H_j^{(i)}}(h) \tag{3}$$

Mutual Information. The zero entropy of neuron output indicates that it has little relationship with classification performance, but the reverse cannot be true, and the high entropy of neuron output cannot indicate its importance in classification problem, so we introduce the mutual information between neuron output and classification result as a measure of matric. It measures how the knowledge of $H_j^{(i)}$ helps predicting Y. Its mathematical form is

$$I(H_j^{(i)}; Y) = H(H_j^{(i)}) - H(H_j^{(i)}|Y) \tag{4}$$

Kullback-Leibler Selectivity. For deeper neurons, its output may play a decisive role in the classification result. Mathematically, for such a neuron there exists a class y s.t. conditional distribution $P_{H_j^{(i)}|Y=y}$ differs significantly from the marginal distribution $P_{H_j^{(i)}}$. Namely, $D_{KL}(P_{H_j^{(i)}|Y=y} \| P_{H_j^{(i)}})$ is large (other researchers call it specifific information). We denote KL selectivity as the maximum specific information over all classes for a measure of neuron importance, as

$$\max_{y \in \mathcal{C}} D_{KL}(P_{H_j^{(i)}|Y=y} \| P_{H_j^{(i)}}) \tag{5}$$

By definition, KL selectivity is high when the influence of a single neuron on the classification result of a particular class is large.

4 Node Pruning Experiment Setup

We use a trained NN with 2 or 3 hidden layers with 200 neurons each (totally 400 or 600 neurons), and apply one-bit quatization, i.e., $|T| = 2$, sigmoid threshold $= 0.5$. For the output of the activation function, values less than the threshold are mapped to 0, and values greater than or equal to the threshold are mapped to 1. For dataset, we use MNIST, a set of 28∗28 gray-scale images. The dataset is selected and divided into 50000 training samples, 10000 validation samples and 10000 testing samples. For loss function, we apply cross-entropy loss and L_2-norm regularization. If bias balancing is applied, its mathematical form is: $w^{(i)}_{j,k}\sum_l \frac{h^{(i)}_j(x_l)}{N} + b^{(i+1)}_k$.

The following are the costs and accuracy during training process, only the 3-layer neural network model is shown here, I also generate violin plots of the corresponding metrics distribution(if you need results for 2-layer, check out the github link provided at the end of the paper)

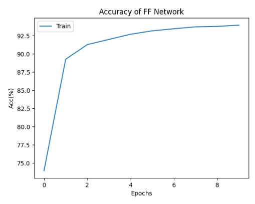

Fig 1. Accuracy

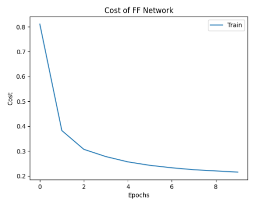

Fig 2. Cost

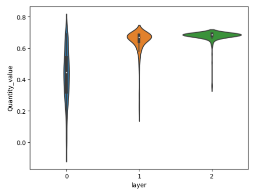

Fig 3. Entropy

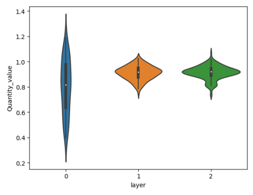

Fig 4. MI

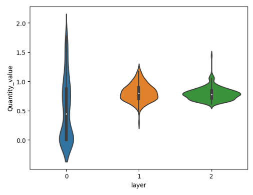

Fig 5. KL-Selectivity

5 Results and Analysis of Node Pruning

First, I test whether applying bias balancing or not has an effect on node pruning on 2-layer model, the results are:

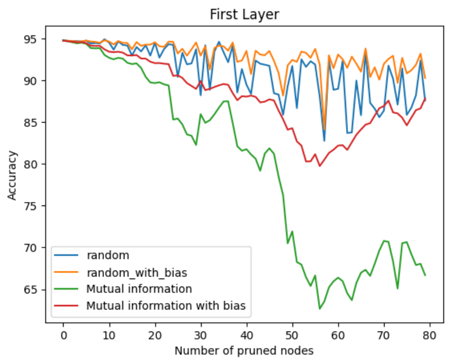

Fig 6. Pruning nodes of 1-th layer

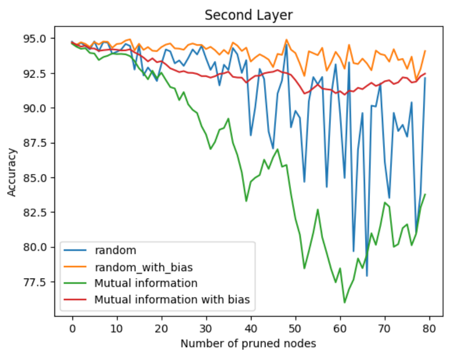

Fig 7. Pruning nodes of 2-th layer

As for 3-layer model(test for bias balancing):

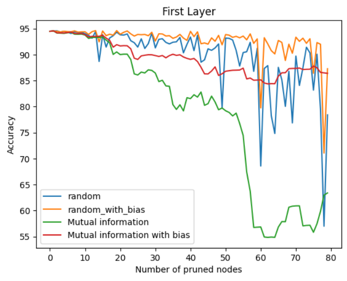

Fig 8. Pruning 1-th layer

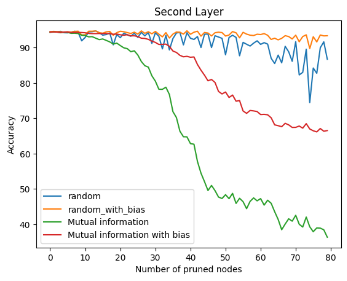

Fig 9. Pruning 2-th layer

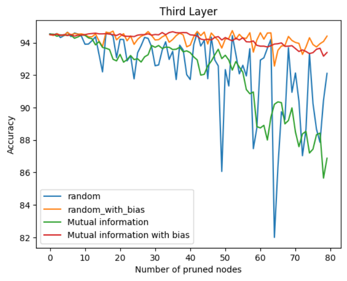

Fig 10. Pruning 3-th layer

From these pruning results, we can conclude that with bias balancing, the overall impact on the model is reduced and the accuracy curve flattens, which also helps to reduce the error caused by different training results of the model (i.e. reduce the occasionality of the results).

Therefore, all our operations in the following apply bias balancing.

Layer-wise ablation. I perform layer-wise ablation to find interactions throughout the layer, the results:

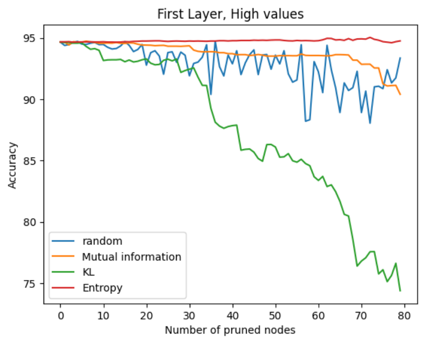

Fig 11. 1-th HVF

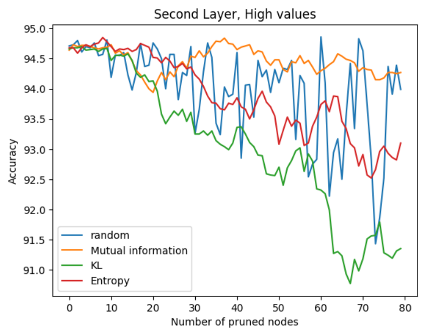

Fig 12. 2-th HVF

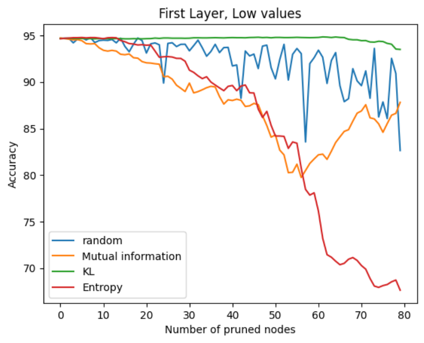

Fig 13. 1-th LVF

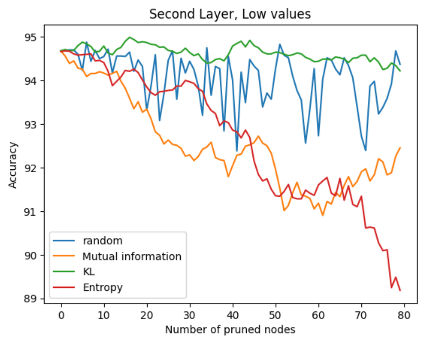

Fig 14. 2-th LVF

Here I denote **HVF** as *high value first* and **LVF** as *low value first*. For example, figure 11 represents that the neurons in the 1-th layer are pruned, and the neurons with high corresponding metric values are strictly pruned first.

As for 3-layer model layer-wise ablation:

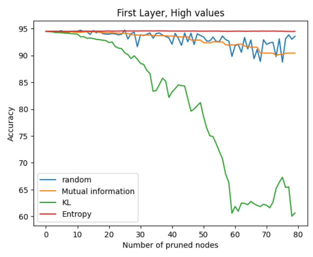

Fig 15. 1-th HVF

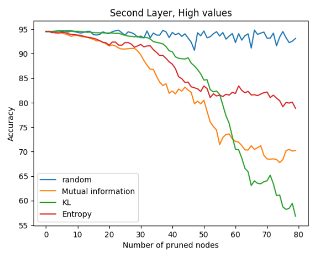

Fig 16. 2-th HVF

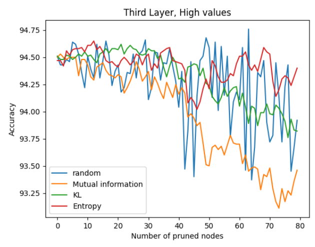

Fig 17. 3-th HVF

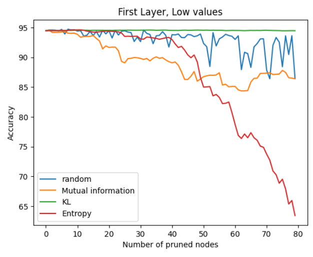

Fig 18. 1-th LVF

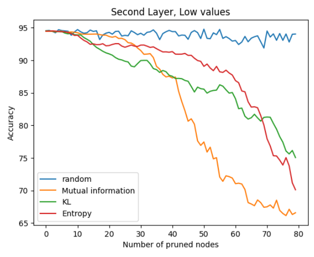

Fig 19. 2-th LVF

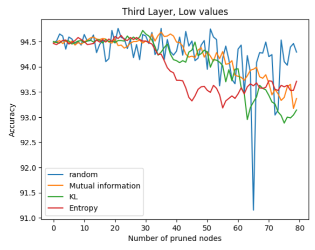

Fig 20. 3-th LVF

Obviously, random pruning is a moderate choice regardless of the level of pruning. In addition, the following insights can be summarized by layer-wise ablation:

Consider Mutual Information and KL-Selectivity. For shallow layers, LVF is a better choice than HVF, which intuitively makes sense because high KL-Selecitivity and high Mutual Information mean that neurons are highly correlated with classification results. But a phenomenon appears for both 2-layer and 3-layer models: when performing pruning for the last layer, KL-Selectivity HVF and MI HVF will be better than LVF, which is somewhat **counterintuitive**. I will give explanations for this phenomenon in next section.

Another oddness is that our intuition is that pruning deeper neurons should be better than pruning shallower.This is also **counterintuitive**, this might because of limitations of information transmission, learning difficulty, and training instability.

Whole network ablation. Last of all, I perform whole network ablation to get more insights:

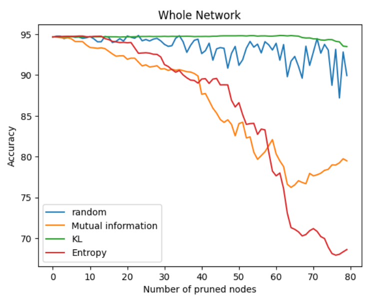

Fig 21. 2-layer: Ablation on whole NN

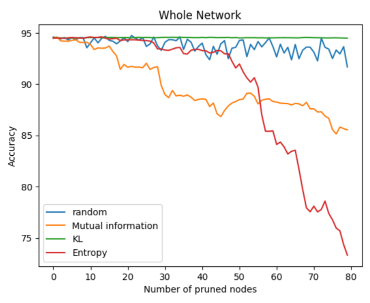

Fig 22. 3-layer: Ablation on whole NN

I find that when node pruning is performed on the whole NN, the curves of Entropy LVF and MI LVF decrease greatly, and the corresponding inference is that entropy and MI of neurons in shallow layers are lower than those in deep layers, while KL-selectivity is higher than that in deep layers. And I find that the effect of random pruning is pretty good, indicating that the whole neural network has a relatively large redundancy.

Explanation: Since shallow neurons receive raw input data or less processed data, it may be easier for them to extract some of the salient features in the data, and thus reduce some of the uncertainty. As a result, shallow neurons may have relatively low entropy.

6 Conclusions and Conjectures

The distribution of the proposed metrics changes from layer to layer. Generally, deeper layers have larger values, especially mutual information and KL-Selectivity. Therefore, it's not reasonable for us to compare neurons at different levels for these metrics.

Formulate hypotheses about the interactions of neurons. Information theory metrics based cumulative neuron ablation with sorting allows us to formulate hypotheses about the interactions of neurons throughout the layer. This is a great merit of the model.

Deeper layers may have larger redundancy. Although the unexpected result of *ablation of neurons in 2-th layer* was the worst in the 3-layer experiment, in general, the effect of ablation in the last layer on the performance of the neural network in both two experiments is small. And is much smaller than the previous layers, which shows deeper layers have larger redundancy, but this redundancy does not strictly increase with the increase of depth.

The correlation between metrics considered and neurons has dependency. The correlation between the considered metrics and neuron importance depends on the depth and structure of the considered layer, and on the NN architecture as a whole.

Explanations of the counterintuitive phenomenon presented in the previous section. Namely, the unnatural results of HVF/LVF for KL-Selectivity and MI.

Possible explanations are: **1. Overfitting problem.** Neurons with high KL-Selectivity often play an important role in the classification result of a particular class, but this does not mean that they are better for generalization. In some cases, these neurons with high KL-Selectivity may focus too much on noise or redundant features, causing the network to overfit. **2. Sharing important features.** In DNN, the abstraction level of features gradually increases as the NN propagates forward. In the last layer, neurons may focus more on the decision contribution to the overall classification result rather than individual classes. Neurons with high KL-Selectivity may have a large impact on the classification of a particular class, but other neurons may have learned similar important features as well.

7 Reference and Attachment

Reference

[1] Rana Ali Amjad, Student Member, IEEE, Kairen Liu, and Bernhard C. Geiger, Senior Member, IEEE. Understanding Neural Networks and Individual Neuron Importance via Information-Ordered Cumulative Ablation.

[2] Naftali Tishby, Noga Zaslavsky. Deep Learning and the Information Bottleneck Principle.

[3] Shujian Yu, Student Member, IEEE, Kristoffer Wickstrøm, Robert Jenssen, Member, IEEE, and Jose´ C. Pr´ıncipe, Life Fellow, IEEE. Understanding Convolutional Neural Networks with Information Theory: An Initial Exploration

Here I attach some important codes for better understanding. As for the experiments details, neural network setting, pruning method and etc, you can reach the codes directly on my github repository https://github.com/Kr-Panghu/UNN-on-MNIST.

DNN Definition. The following is the definition of my 3-layer neural network.

```
# 3 hidden layer DNN
class DNN(nn.Module):
    def __init__(self):
        super(DNN, self).__init__()
        # defining fully connected layers
        self.fc1 = nn.Linear(784, 200)
        self.fc2 = nn.Linear(200, 200)
        self.fc3 = nn.Linear(200, 200) # 3 hidden layer
        self.fc4 = nn.Linear(200,10)

    def forward(self, x):
        # flatten the input to (batch_size, 28 * 28)
        x = x.view(x.size(0), −1)
        x = torch.sigmoid(self.fc1(x))
        x = torch.sigmoid(self.fc2(x))
        x = torch.sigmoid(self.fc3(x))
        output = self.fc4(x)
        return output
```

Layer-wise node pruning. The following is the codes for LVF node pruning of 3-th layer of 3-layer model.

```
# Low Value First

# Random Pruning
random_bias_performances = []
for n_abl in tqdm(range(80)):
    random_idx = np.random.choice(np.arange(200), size= n_abl+1, replace = False) + 400
    random_idx = torch.from_numpy(random_idx).cuda()
    mask2 = torch.ones((600,1), device='cuda')
    p_cuda = p.cuda()
    mask2[random_idx,0] = torch.index_select(p_cuda, 0, random_idx)[:,1]
    random_bias_performances.append(cal_performance(Net, test_loader, mask2))

# Mutual information as the metric
MI_bias_performances = []
for n_abl in tqdm(range(80)):
    idx = torch.topk(MI_3st, n_abl+1 , dim= 0, largest = False)[1].squeeze(-1) + 400
    mask2 = torch.ones((600,1), device='cuda')
    p_cuda = p.cuda()
    mask2[idx,0] = torch.index_select(p_cuda, 0, idx)[:,1]
    MI_bias_performances.append(cal_performance(Net, test_loader, mask2))

# KL-Selectivity as the metric
KL_bias_performances = []
for n_abl in tqdm(range(80)):
    idx = torch.topk(KL_sel_3st, n_abl+1 , dim= 0, largest = False )[1].squeeze(-1)+ 400
    mask2 = torch.ones((600,1), device='cuda')
    p_cuda = p.cuda()
    mask2[idx,0] = torch.index_select(p_cuda, 0, idx)[:,1]
    KL_bias_performances.append(cal_performance(Net, test_loader, mask2))

# Entropy as the metric
Entropy_bias_performances = []
for n_abl in tqdm(range(80)):
    idx = torch.topk(Entropy_3st, n_abl+1 , dim= 0, largest = False)[1].squeeze(-1) + 400
    mask2 = torch.ones((600,1), device='cuda')
    p_cuda = p.cuda()
    mask2[idx,0] = torch.index_select(p_cuda, 0, idx)[:,1]
    Entropy_bias_performances.append(cal_performance(Net, test_loader, mask2))

# Visualization
plt.plot(random_bias_performances)
plt.plot(MI_bias_performances)
plt.plot(KL_bias_performances)
plt.plot(Entropy_bias_performances)
plt.title("Third Layer, Low values")
plt.ylabel('Accuracy')
plt.xlabel('Number of pruned nodes')
plt.legend(labels = ["random", "Mutual information", "KL", "Entropy"])
plt.show()
```