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

——An information theory based node pruning method

Author: Kr.Cen 康瑞 岑 Date: 2023.6.12

Background

Interpretation of Deep Neural Networks(DNNs)

- Challenges for development of DNNs

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.

Background(cont'd)

Interpretation of Deep Neural Networks(DNNs)

- Traditional researches 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 or interneuron.

- However— —

This traditional direction cannot 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.

What did I do?

The work done by this paper

- 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:

- Entropy (whose entropy?)
- Mutual Information (w.r.t. which two random variables?)
- Kullback-Leibler Selectivity (what’s the definition?)

The quetions I mentioned above by myself will be discussed later.

What did I do?(cont'd)

The work done by this paper

- 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). I employ the metrics I proposed (i.e. Entropy, MI and so on) to decide which neuron will be pruned.

- 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.

What did I do?(cont’d)

Two ways of pruning neurons

- Whole-Network ablation

Perform ablation on the whole neural network.

- Layer-wise ablation

Perform ablation on a particular layer of the neural network.

Propose information theoretic metrics

Basic Setup

Consider classifification via fully-connected feed-forward NNs.

$\mathscr{C}$:classification set, where $|\mathscr{C}| = C$

$\mathscr{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ denotes the dataset.

x_i : i-th input

y_i : I-th output

$h_j^{(i)}(x_l)$: the output of j-th neuron in i-th hidden layer if given the input x_l

b : bias vector

σ : non-linear activation function

Then we have :

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

$\mathscr{Q} : \mathbb{R} \to \mathscr{H}$ maps outputs to a finite set $\mathscr{H}$

Y : Random variable over set $\mathscr{C}$ of classes

$H_j^{(i)}$: Random variable over set $\mathscr{H}$

Define the joint distribution of Y and $H_j^{(i)}$ via the joint frequences of $\{(y_l, Q(h_j^{(i)}(x_l)))\}$ in the validation set.

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

Where 1 is the indicator.

Propose information theoretic metrics

Basic Setup (cont'd)

Evaluate single neuron's importance ⇒ Consider its output as a random variable

- Entropy： $\mathbb{H}(H_j^{(i)}) = -\sum_{h \in \mathscr{H}} P_{H_j^{(i)}}(h) \log P_{H_j^{(i)}}(h)$, it quantifies the uncertainty of the output of the neuron.
- Mutual Information： Denote the decision result of the model is a random variable Y, then we consider MI w.r.t. Y and $H_j^{(i)}$.$I(H_j^{(i)}; Y) = \mathbb{H}(H_j^{(i)}) - \mathbb{H}(H_j^{(i)} \mid Y)$
- KL-Selectivity is defined as the maximum **specific information** over all classes for a measure of neuron importance : KL-Selectivity = $\max_{y \in \mathscr{C}} D_{KL}(P_{H_j^{(i)}|Y=y} || P_{H_j^{(i)}})$, where $D_{KL}(P_{H_j^{(i)}|Y=y} || P_{H_j^{(i)}})$ is the specific information.

Experiment Setup

Network: (expansion from 2 layers to 3 layers)

A trained NN with 2 hidden layers (200 neurons each)

A trained NN with 3 hidden layers (200 neurons each)

Apply 1-bit quatization, $|T| = 2$, sigmoid threshold = 0.5

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \Rightarrow MNIST$$

The dataset is divided into: 50000 training samples, 10000 validation samples, 10000 testing samples.

Loss function : CE loss + L2-norm regularization

Bias(if applied) : $w_{j,k}^{(i)} \sum_l \frac{h_j^{(i)}(x_l)}{N} + b_k^{(i+1)}$

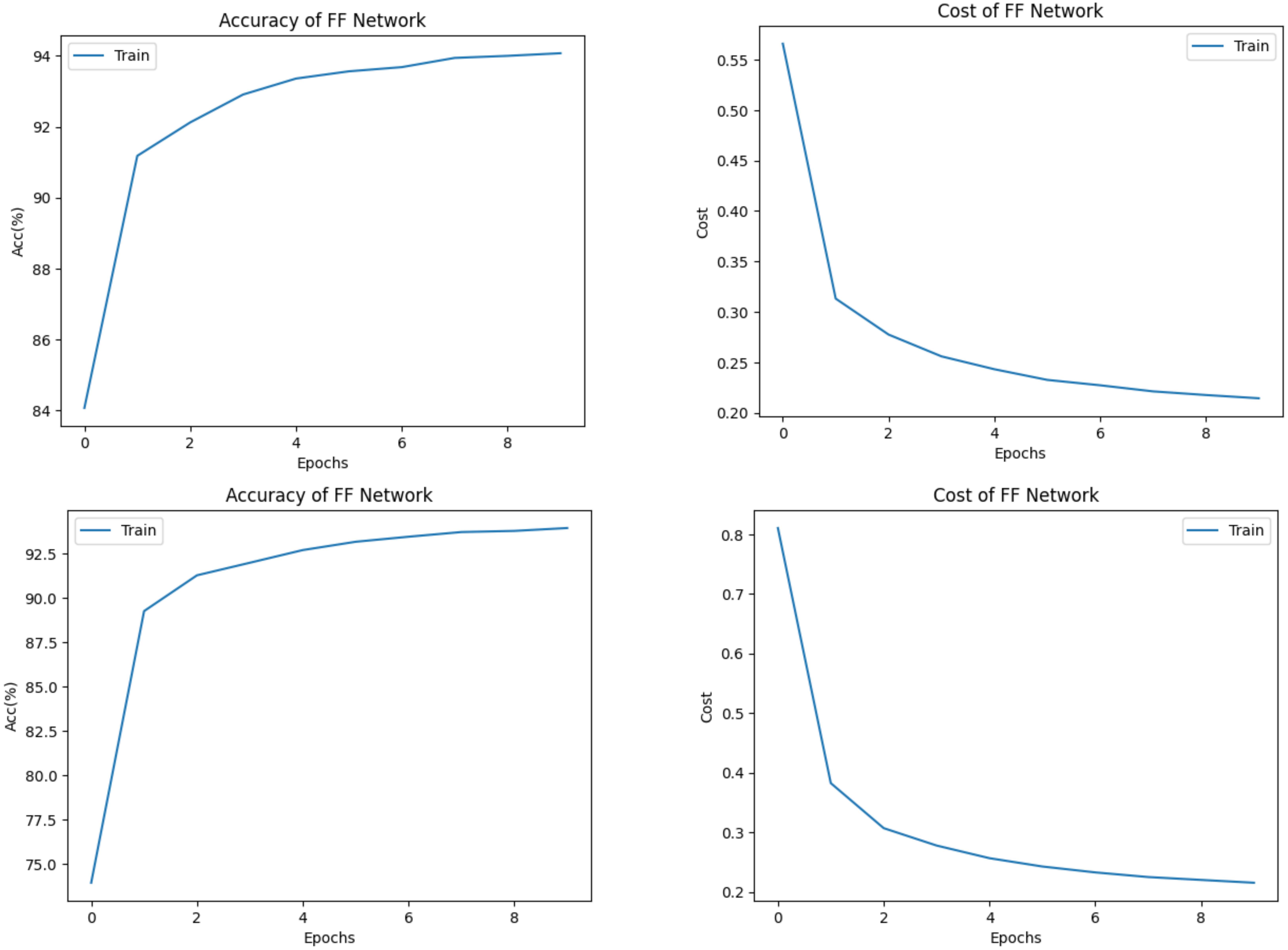
Network

```
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
```

Training results

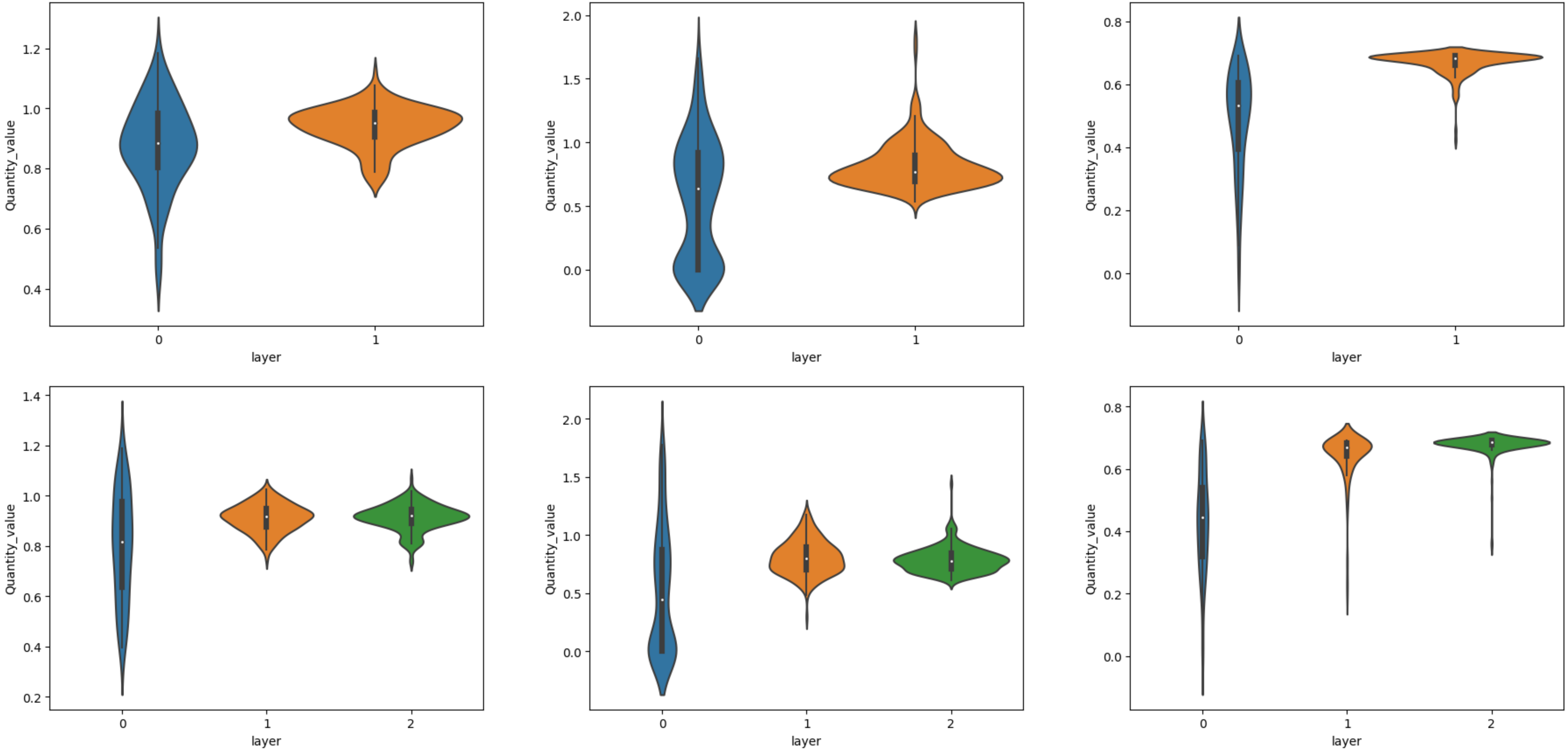
The upper is the 2-layer model.
The bottom is the 3-layer model



Metrics distribution

The upper is the 2-layer model.
The bottom is the 3-layer model

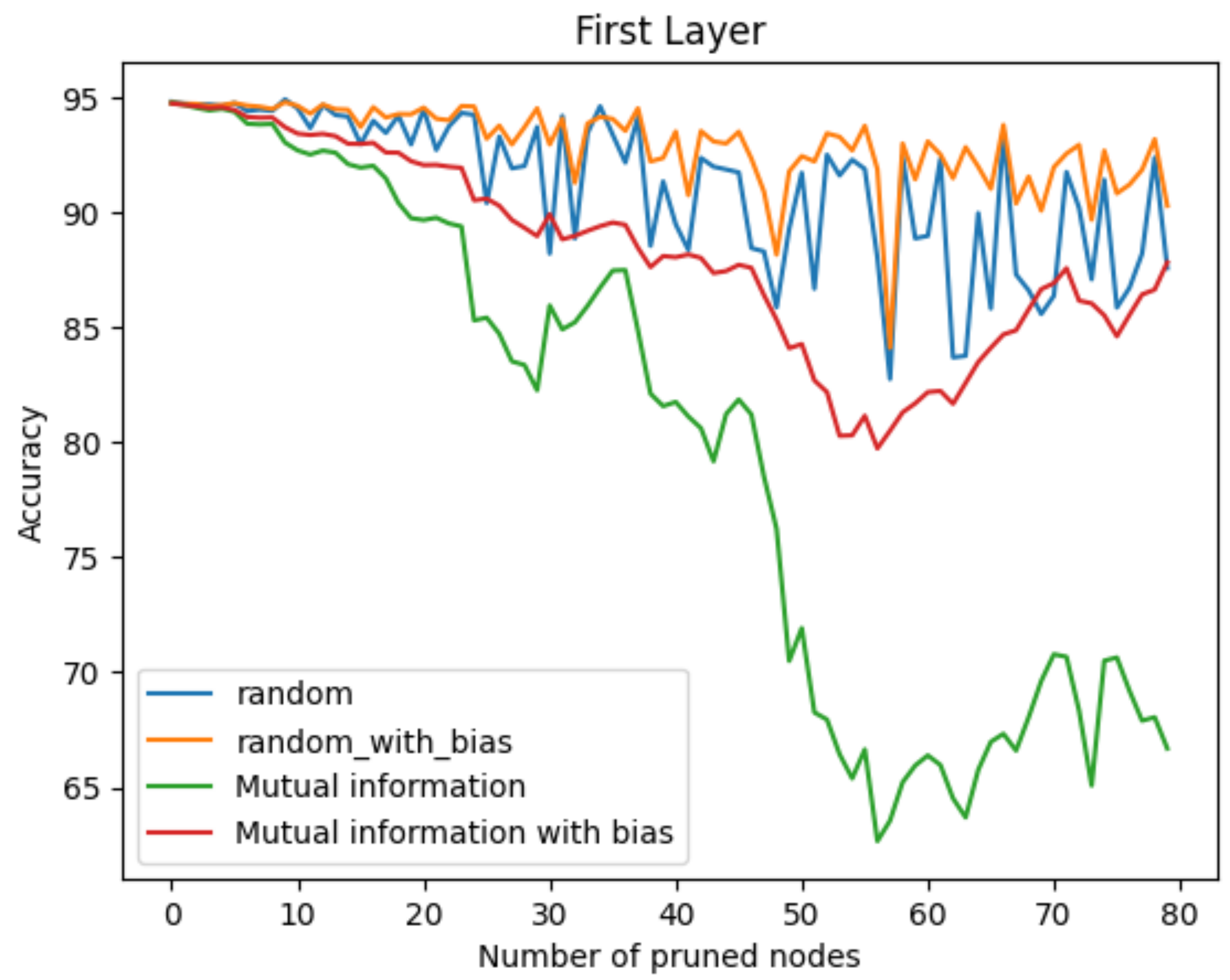
Mutual Information KL-Selection Entropy

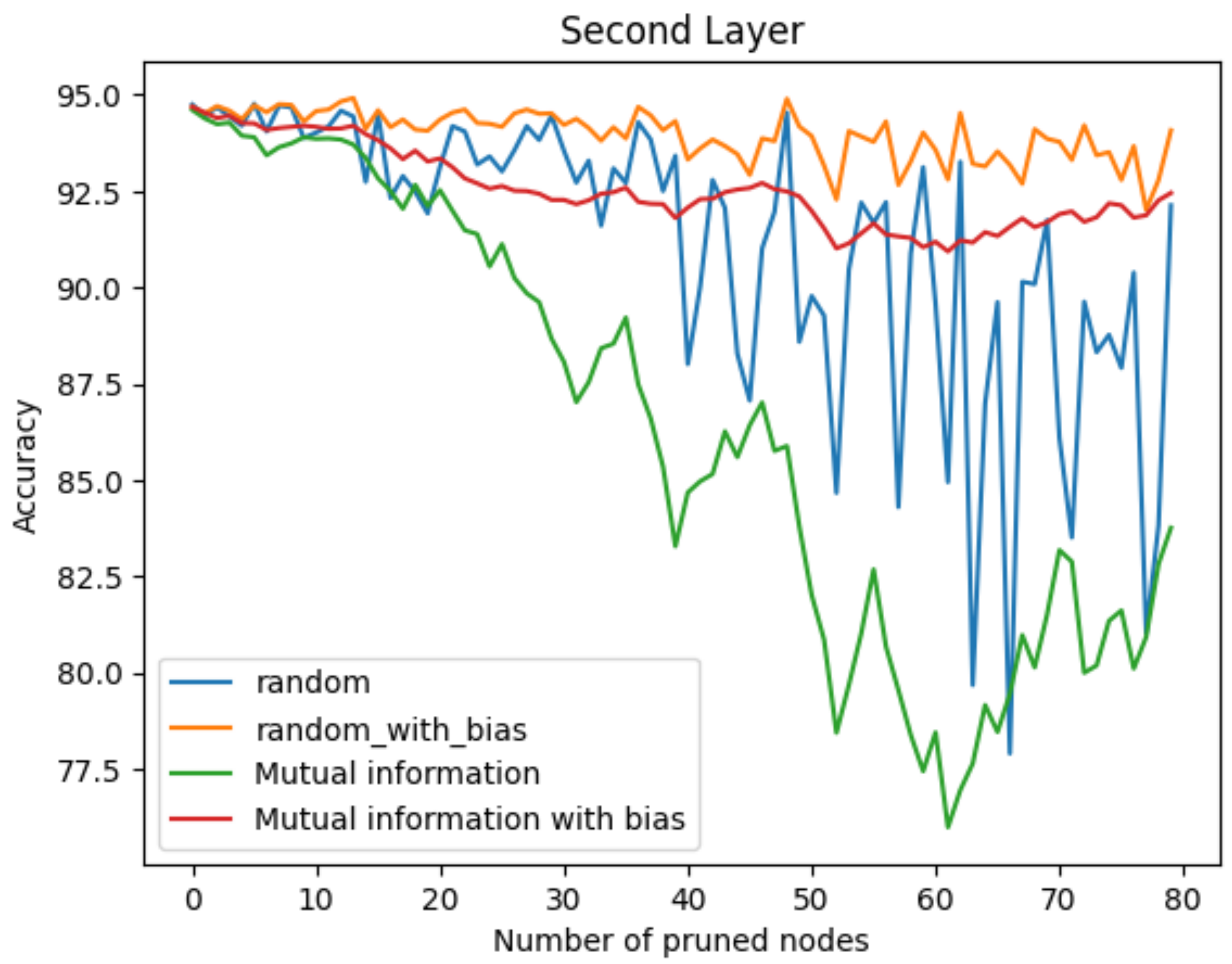


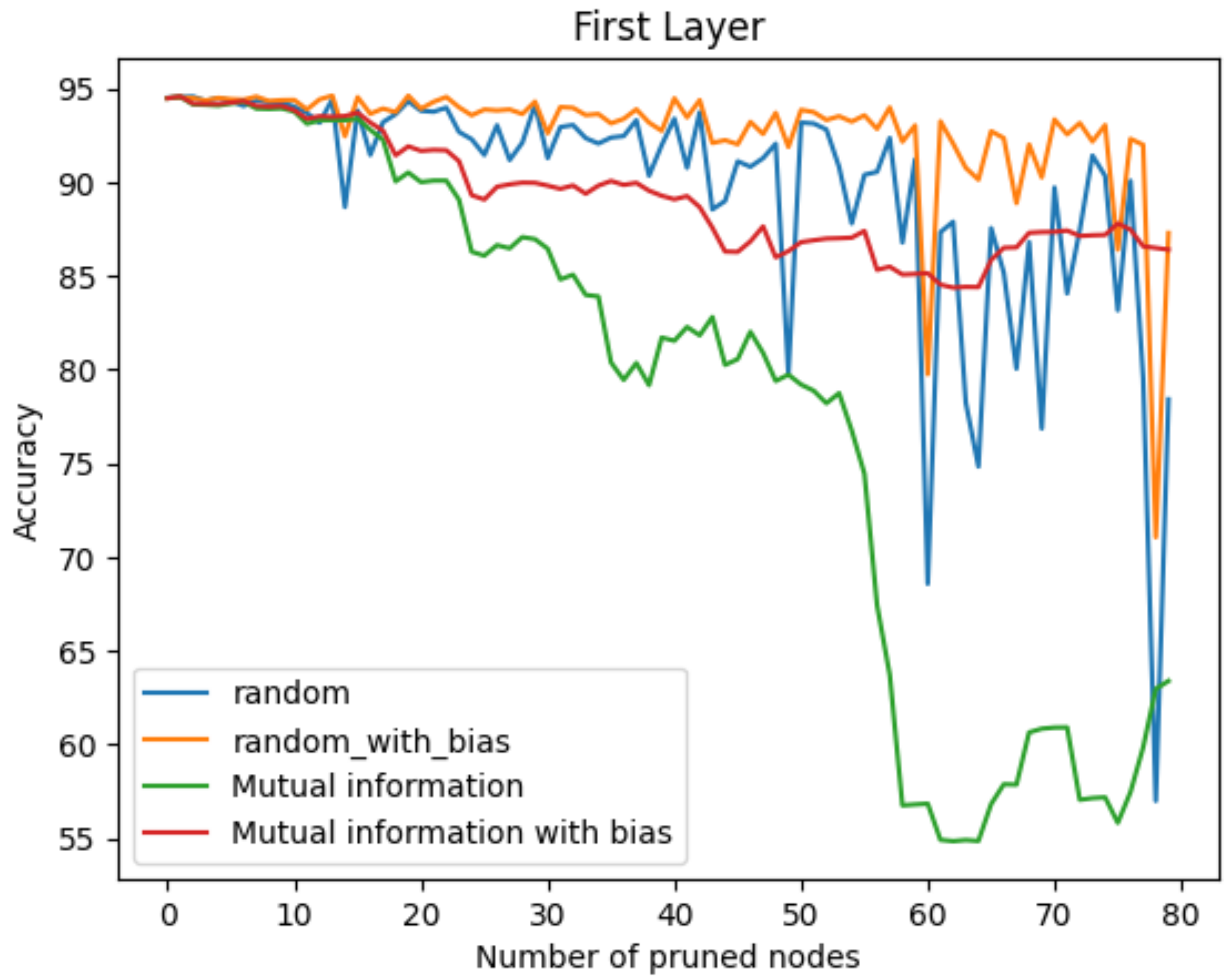
Experiment : Result

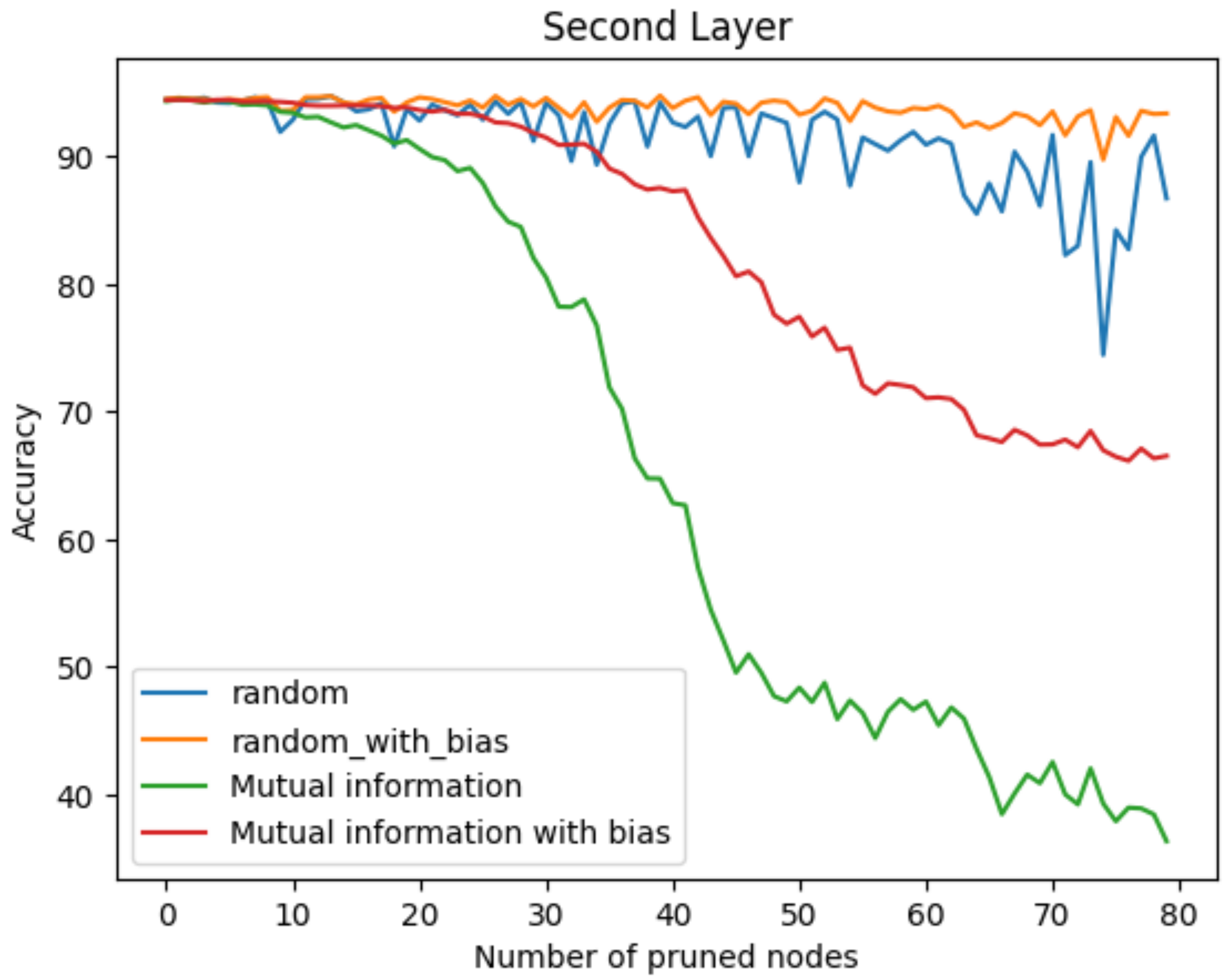
The upper is the 2-layer model.
The bottom is the 3-layer model

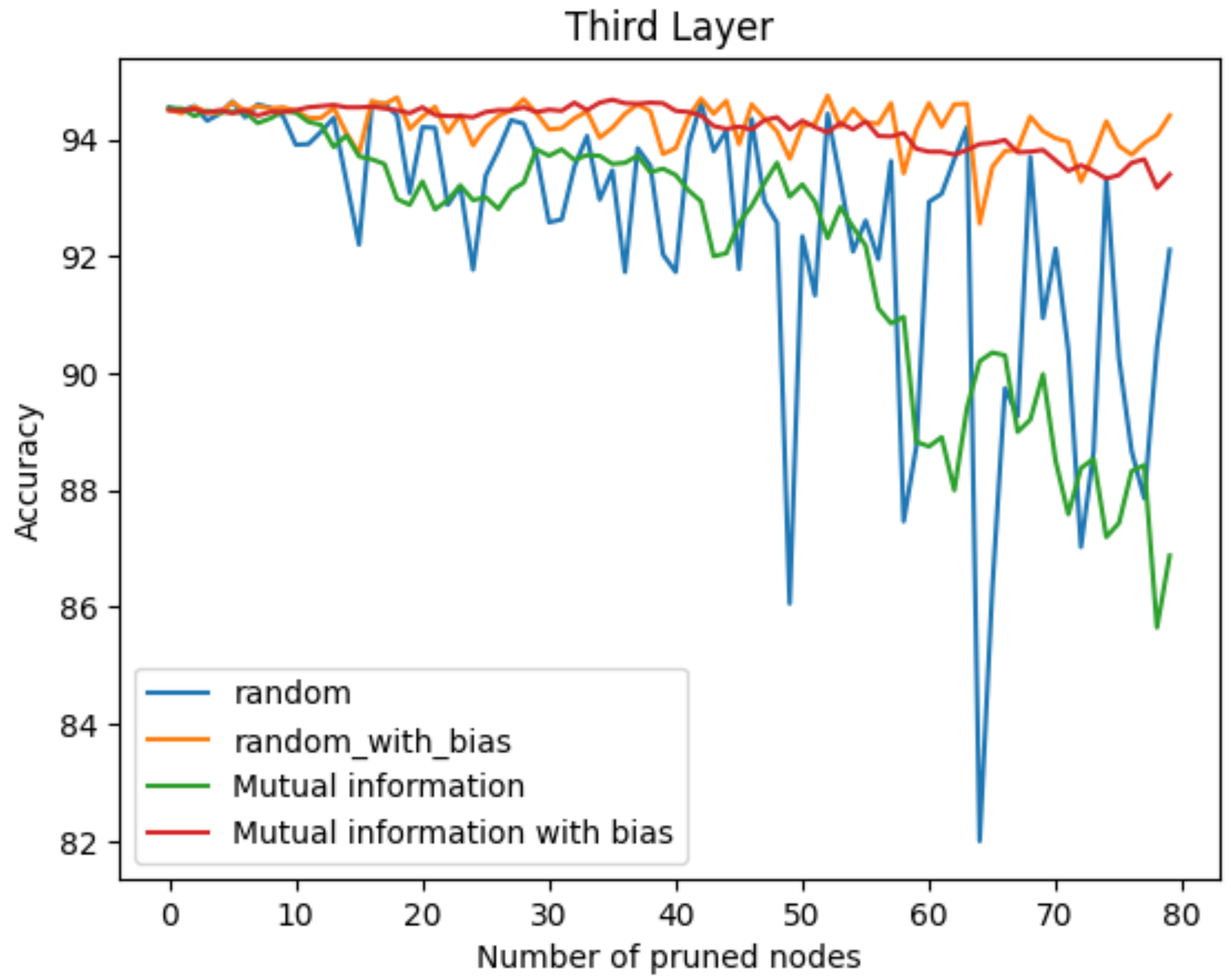
Decide whether to apply bias balancing or not











Experiment : Result

Decide whether to apply bias balancing or not (cont'd)

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.

These figures are for 2-layer model

Experiment : Result

Layer-wise Ablation

- Denote HVF as “high value first”
- Denote 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.

Hint: The sequence of pruning depends totally on value of metric

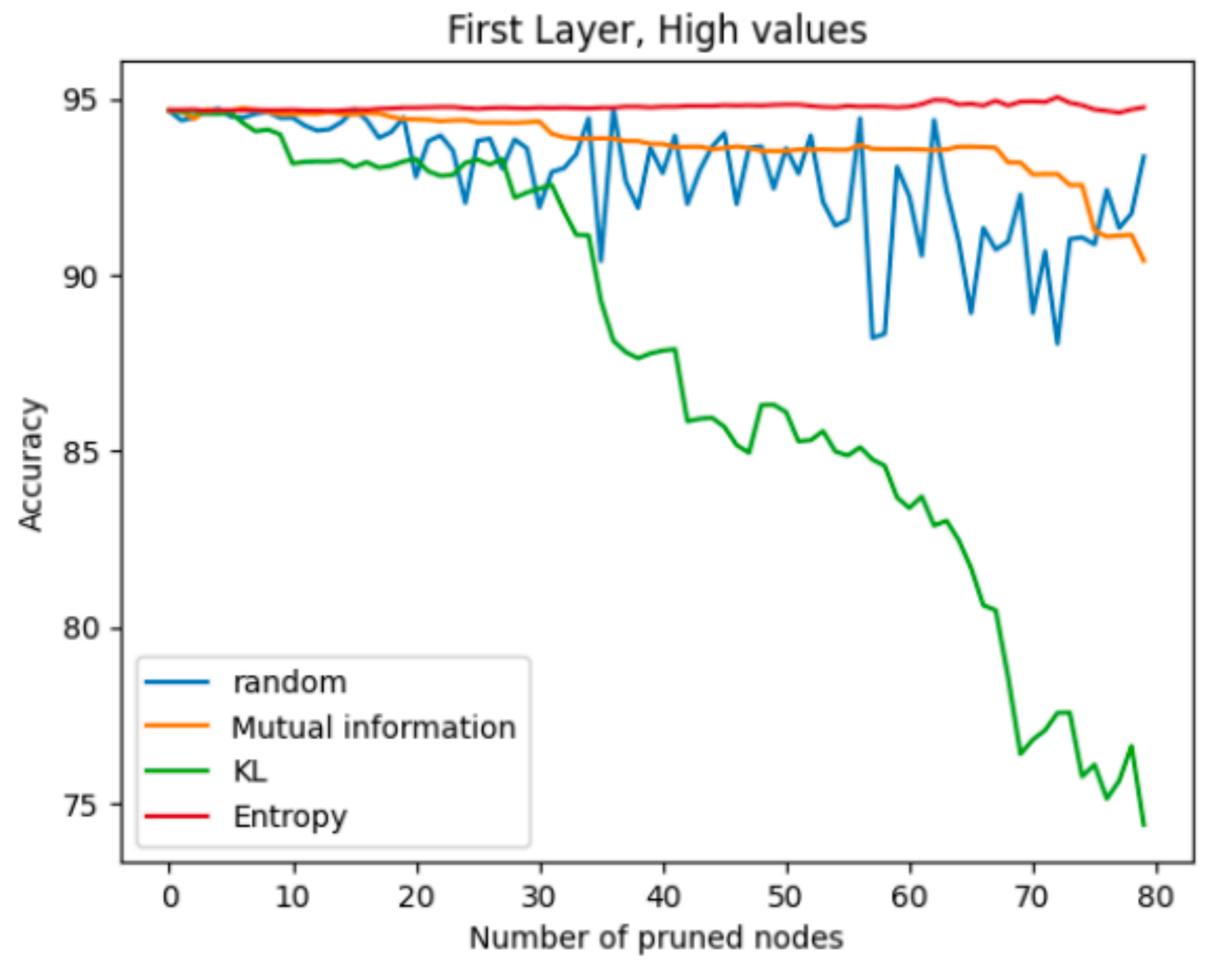


Fig 11. 1-th HVF

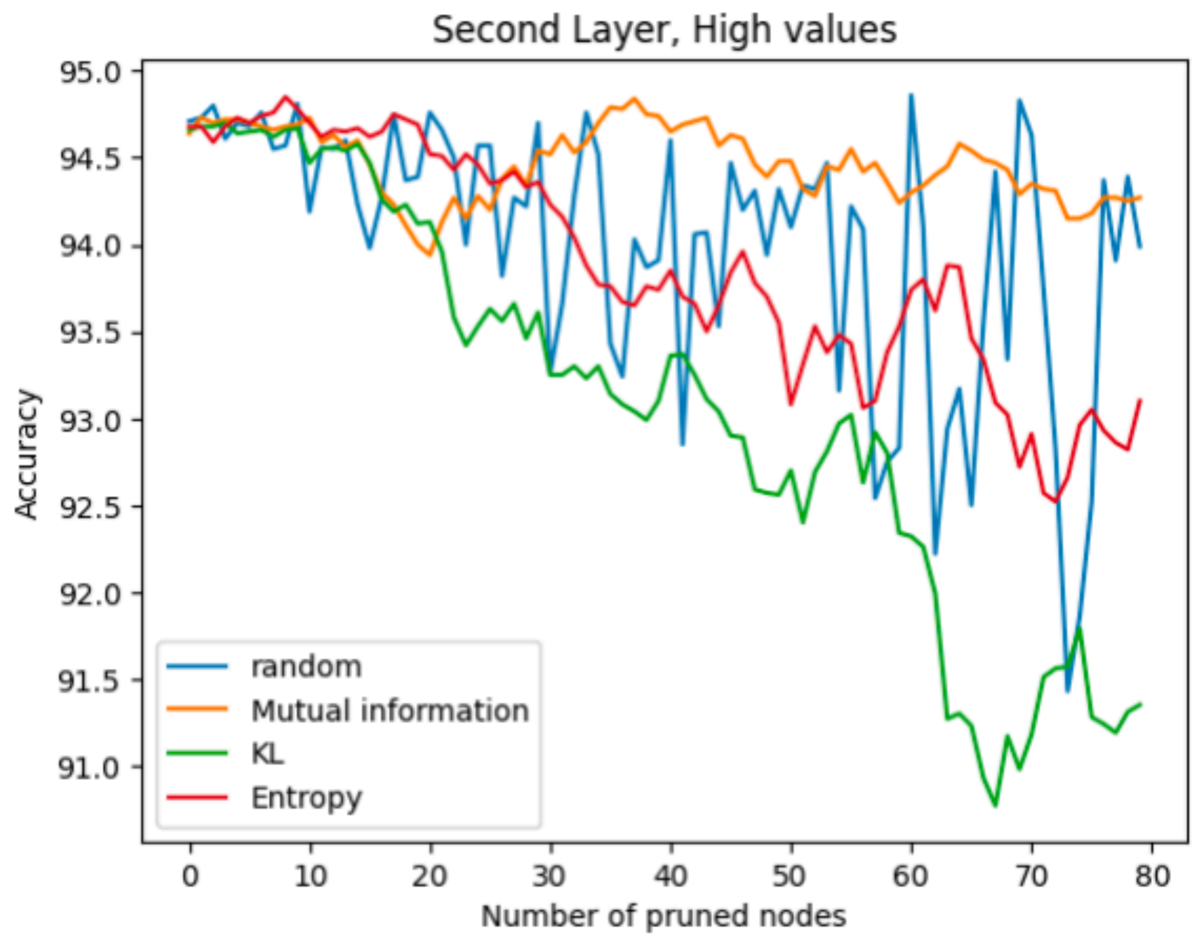


Fig 12. 2-th HVF

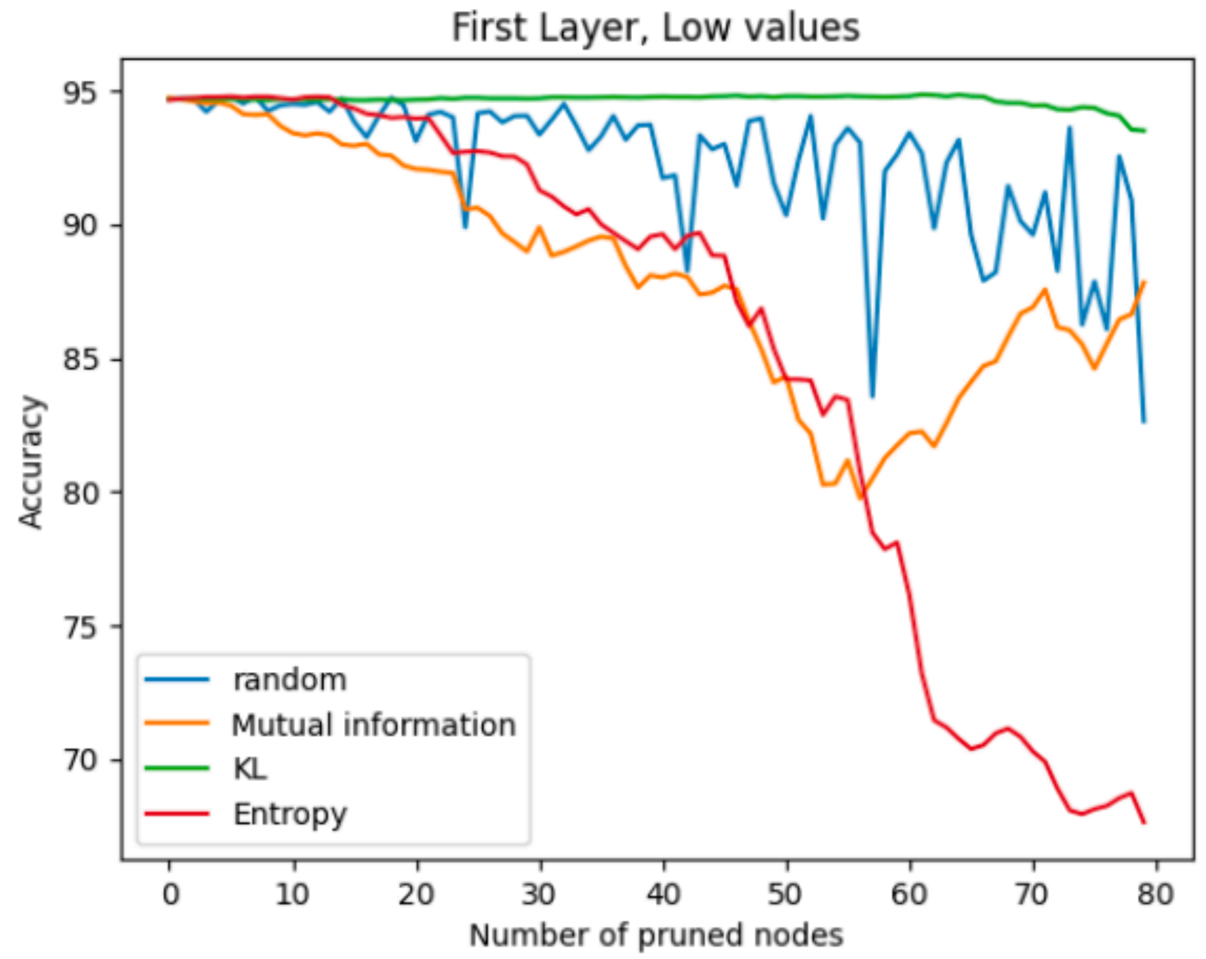


Fig 13. 1-th LVF

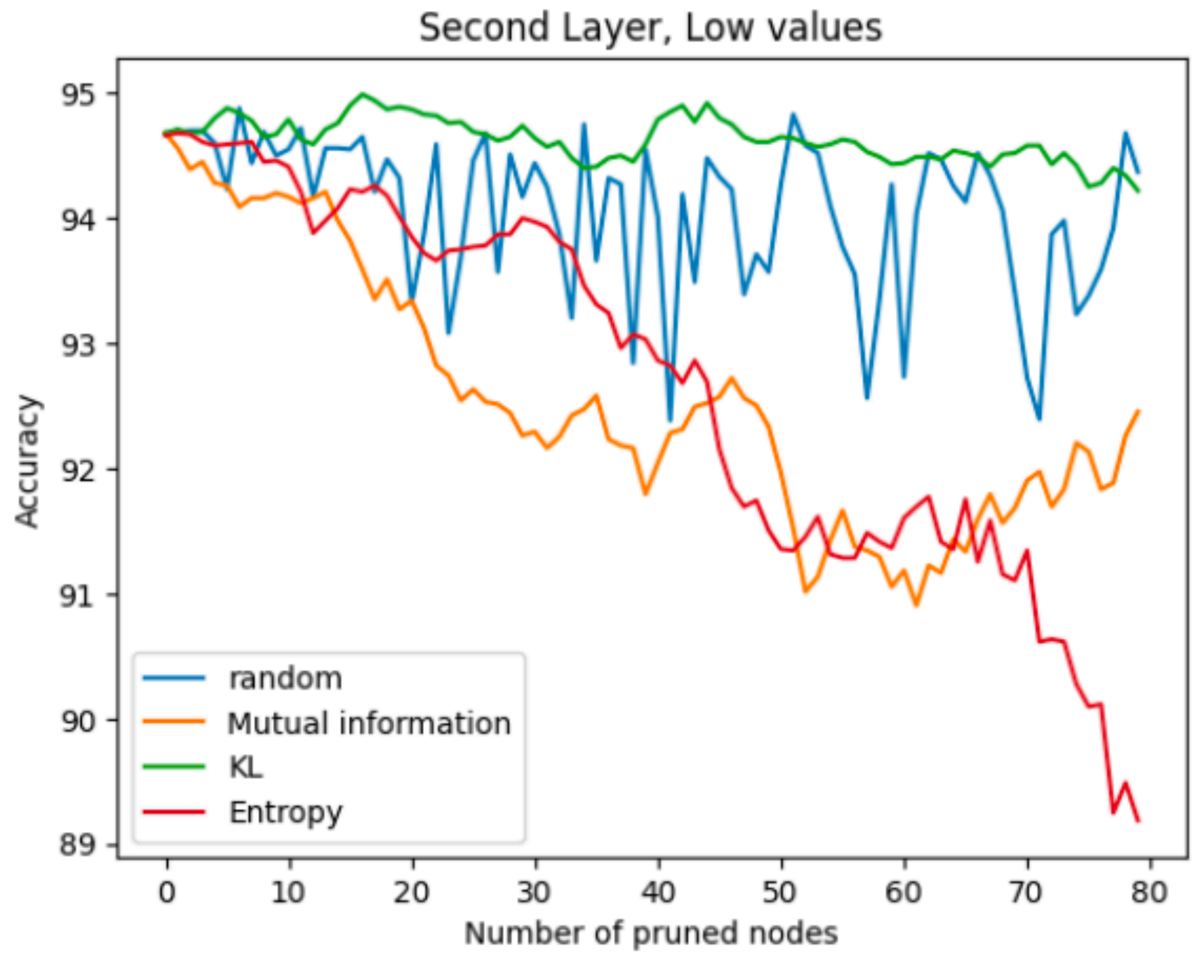


Fig 14. 2-th LVF

Experiment : Result

These figures are for 3-layer model

Layer-wise Ablation (cont'd)

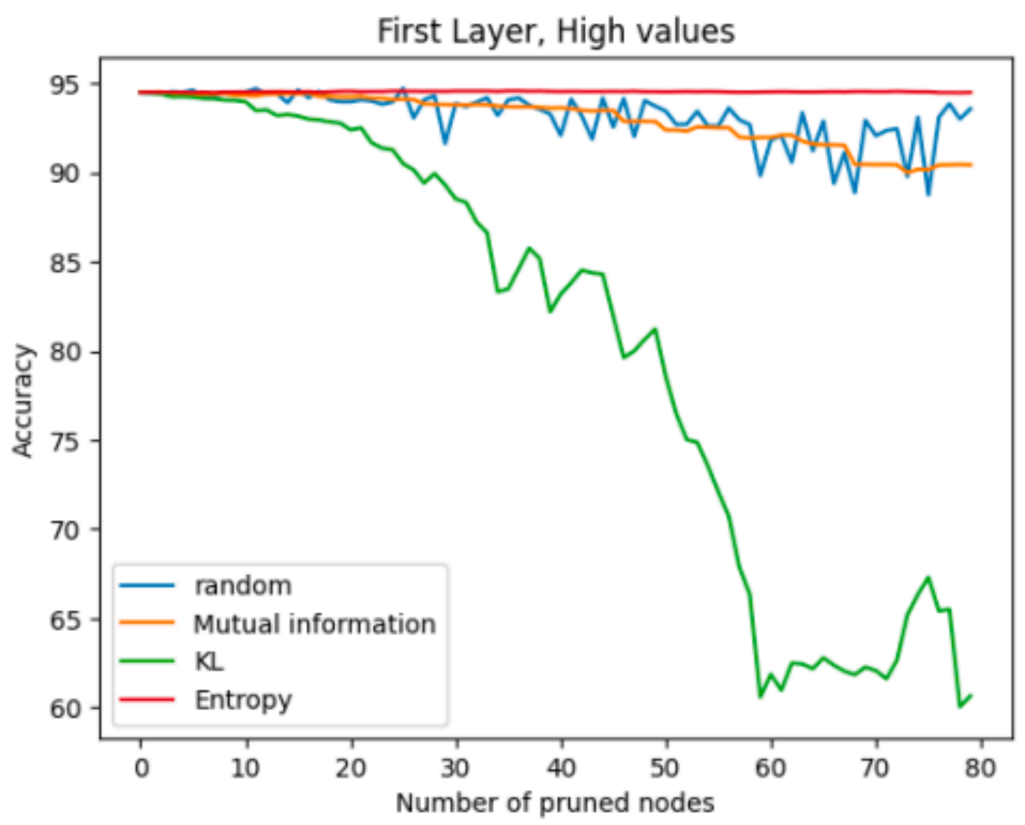


Fig 15. 1-th HVF

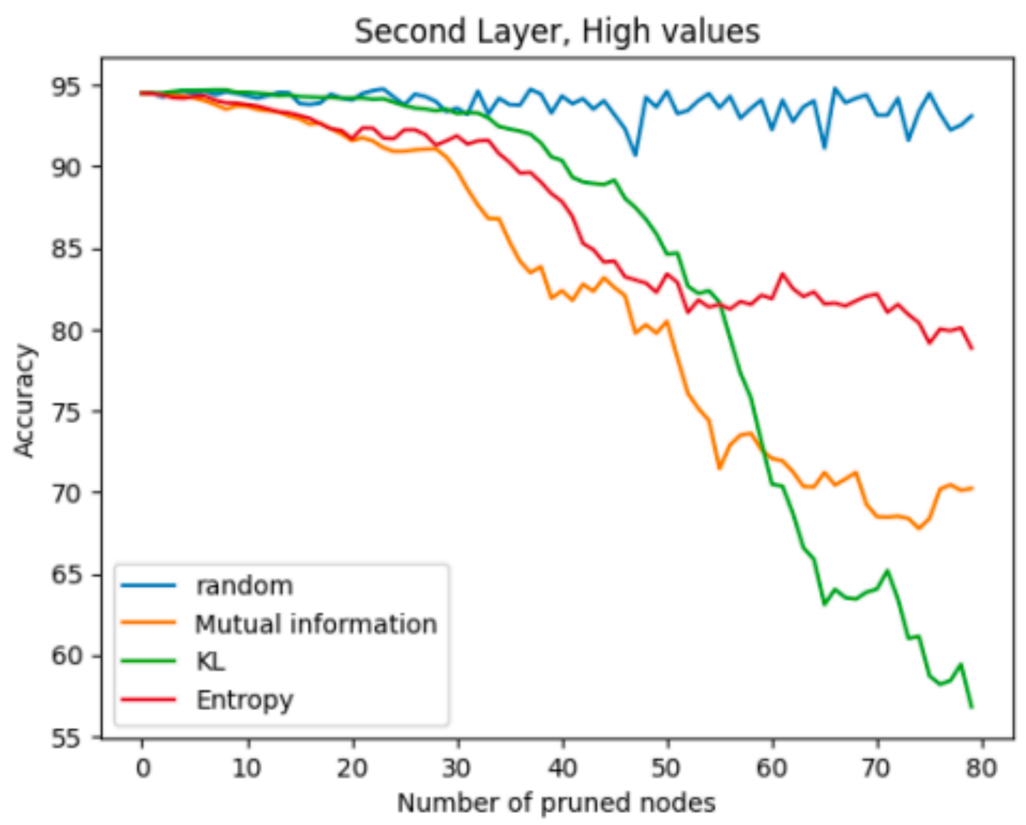


Fig 16. 2-th HVF

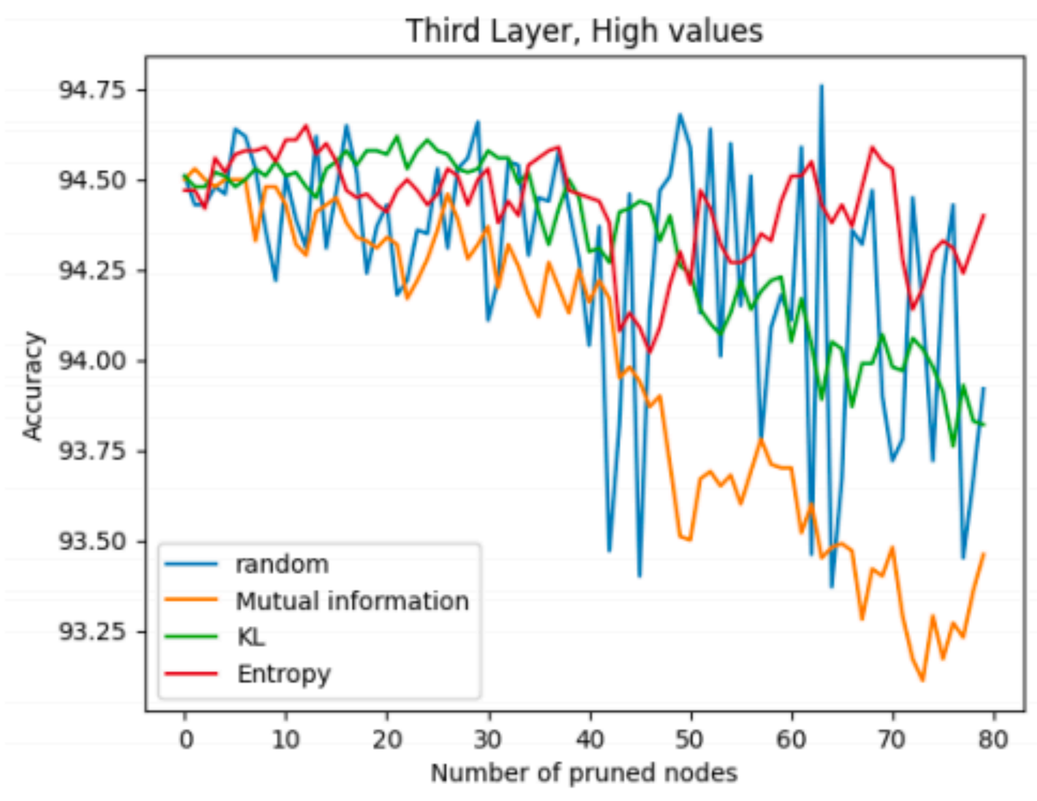


Fig 17. 3-th HVF

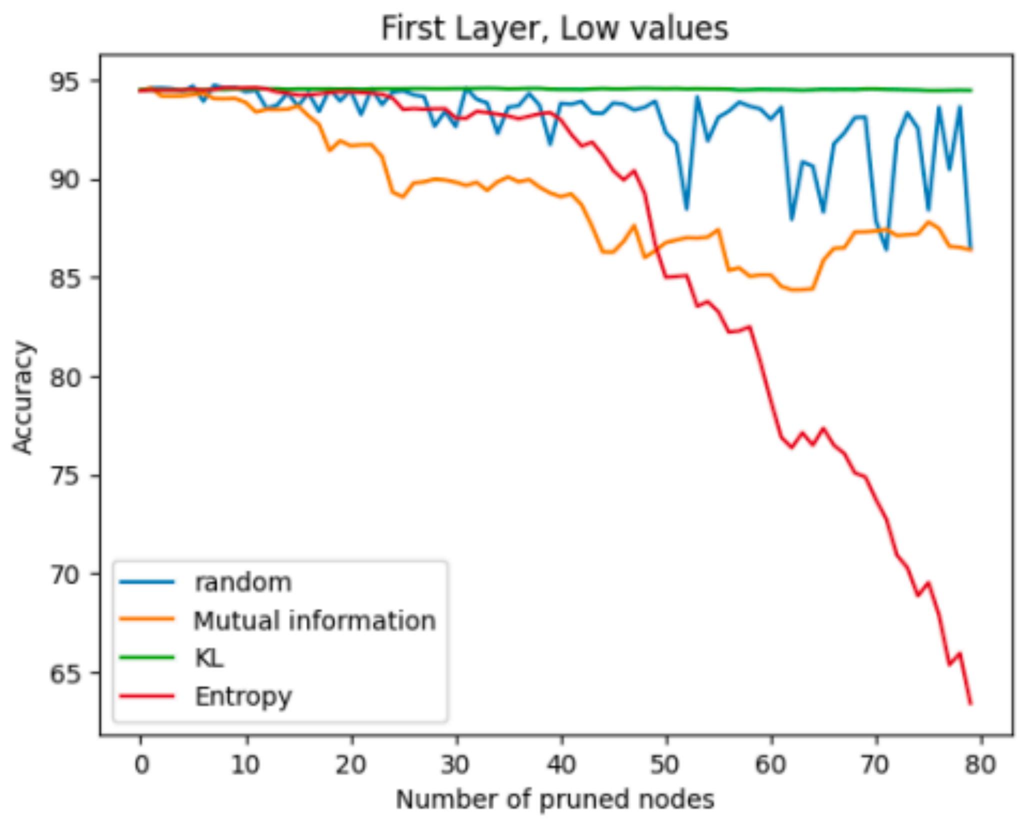


Fig 18. 1-th LVF

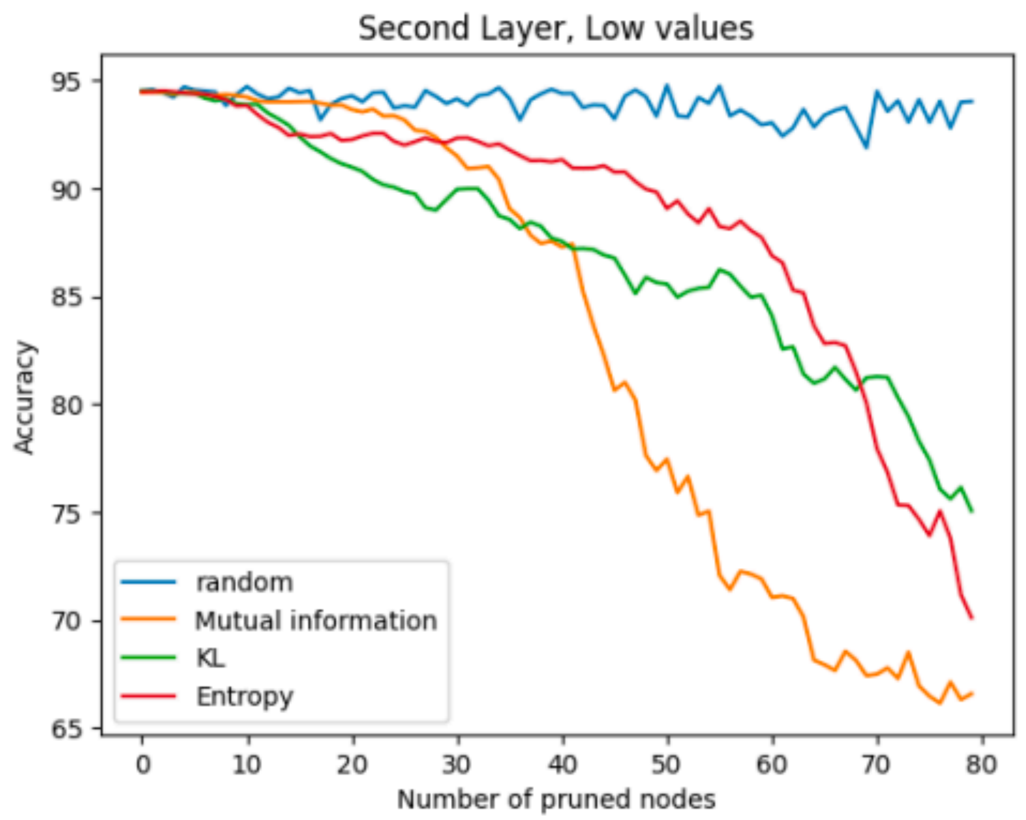


Fig 19. 2-th LVF

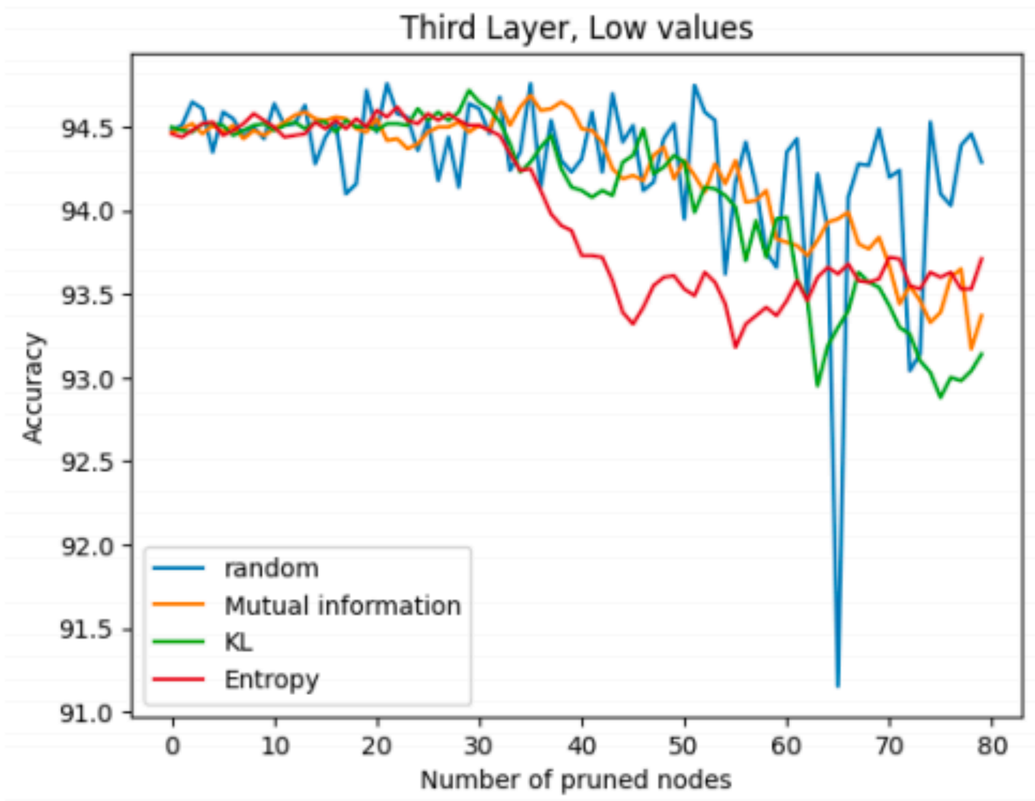


Fig 20. 3-th LVF

Experiment : Result

Layer-wise Ablation (cont'd)

I have given a concrete analysis of the result in the paper. The counterintuitive showed in "2" may due to overfitting or else

- Analysis (personal insights):

1. Obviously, random pruning is a moderate choice regardless of the level of pruning.
2. Consider MI and KL-Selectivity. For shallow layers, LVF is a better choice than HVF, which intuitively makes sense because high KL-Selecitivity and high MI 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.
3. 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.

Experiment : Result

Whole-Network Ablation

I perform whole network ablation in order to get more insights about the DNN globally.

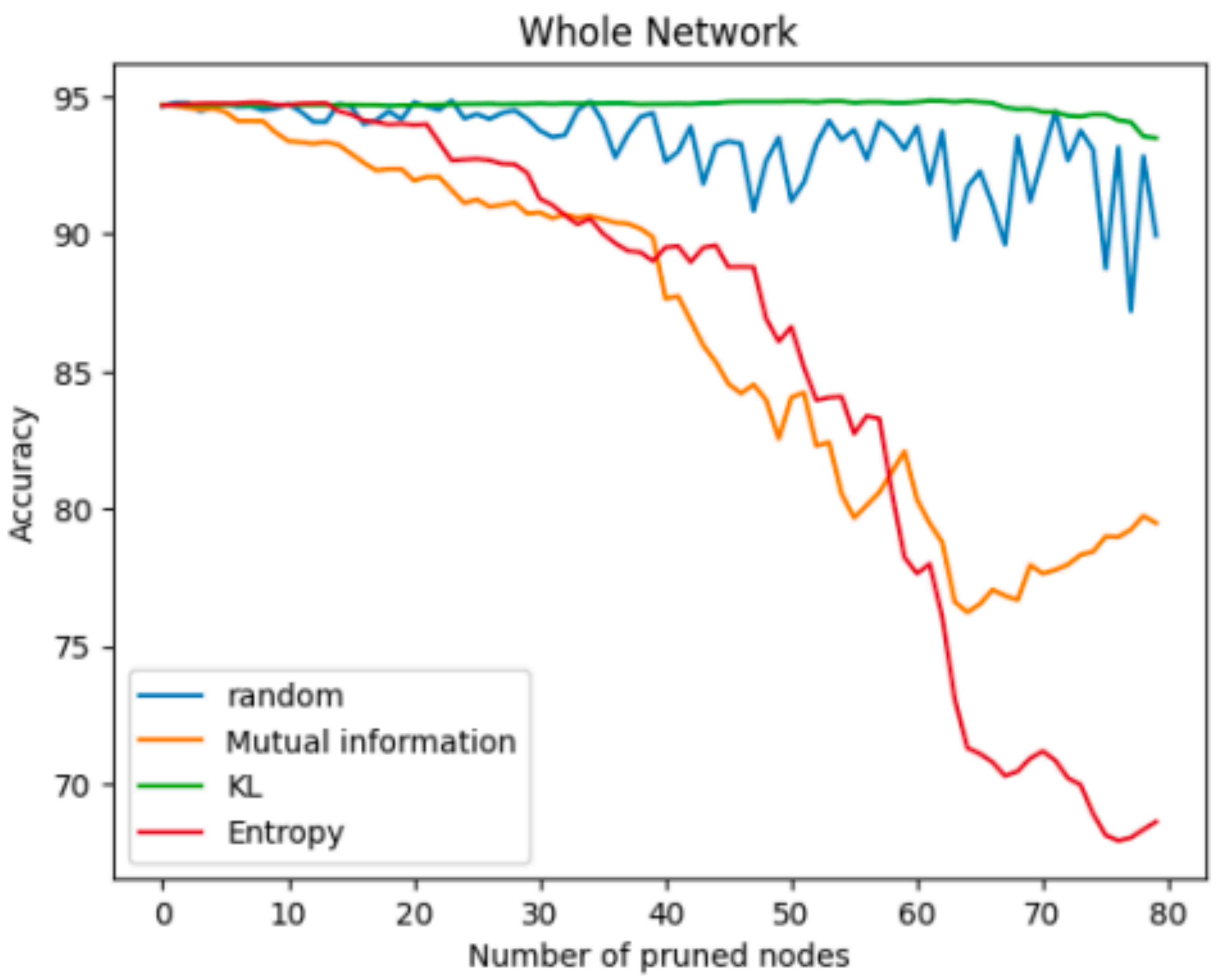


Fig 21. 2-layer: Ablation on whole NN

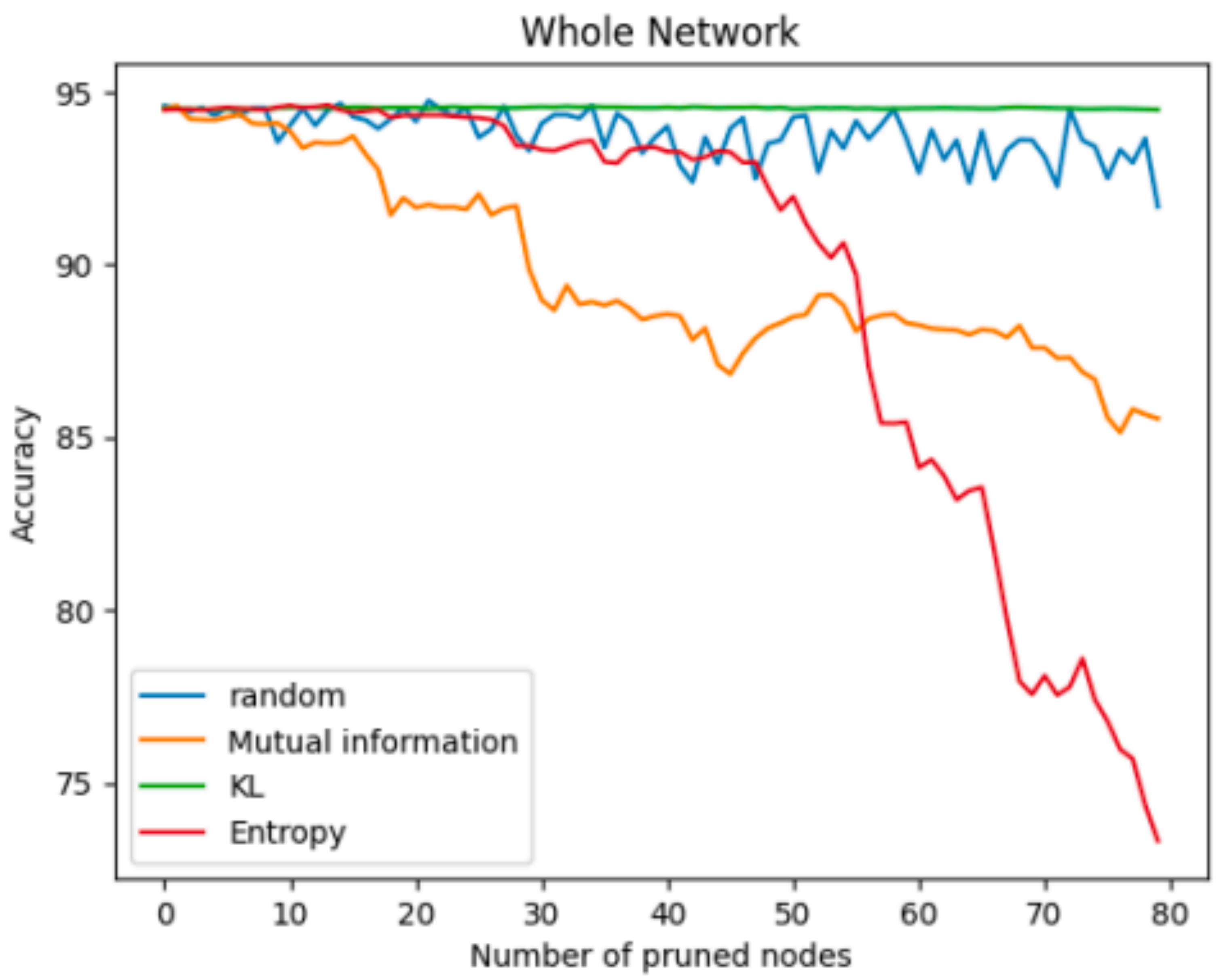


Fig 22. 3-layer: Ablation on whole NN

Experiment : Result

Whole-Network Ablation (cont'd)

- Analysis (personal insights):

- When node pruning is performed on the whole NN, the curves of Entropy LVF and MI LVF decrease greatly.

- 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.

- (Possible)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.

Conclusion

1. The distribution of the proposed metrics changes from layer to layer.
2. We can thus formulate hypotheses about the interactions of neurons.
3. Deeper layers may have larger redundancy.
4. The correlation between metrics considered and neurons has dependency.
5. Counterintuitive behavior of the MI and KL-Selectivity in layer-wise ablation is possibly due to overfitting problem and important features shared between different neurons.

Thanks for your attention

Kr.Cen SJTU
Email Addr: kr2256671169@sjtu.edu.cn

You can reach my codes directly at my open repository
https://github.com/Kr-Panghu/UNN-on-MNIST