Effect of Pruning on Mutual Information in Deep Neural Networks

Rudraharsh Tewary
University of Toronto

December 15, 2023

(**Note:-** As discussed with the professor this report will not cover the paper sent for the project and only the experiment performed)

1 Introduction

Our task in the proposed experiment is to investigate the impact that pruning has on the mutual information between the input and output layers of a neural network. The motivation for investigating this arises from the fact that in many cases, substantial pruning of neural networks leads to almost non-significant performance loss. However, there is little to no literature investigating how this impacts the mutual information between the input and output layers. So, we will attempt to showcase the effects of pruning on the information on a simple deep network.

2 Mutual Information

In statistics, Mutual information is the measure of 'dependence' between two random variables. That is, the information carried by both variables about each other. Now, it is calculated using the formula[2] :

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

In the case of neural networks[3], to estimate mutual information between layers we proceed to discretize them into bins. Therefore, we have the following mapping $\sigma : \mathcal{R}^d \to [-1,1]^d$ where d is the number of dimensions and σ is our non-linearity. Follwing that, we get the discretizations for our image and representation as:

$$\text{For image } X : [0,1]^d \to [m]^d$$
$$\text{For representation } H : [-1,1]^d \to [m]^d$$

here, m represents the number of bins we chose for our discretization. Following which, we get our expressions for mutual information for the data and label as :

$$I(X;H) = \sum_{x \in X} \sum_{h \in H} p(x,h) \log \left(\frac{p(x,h)}{p(x)p(h)} \right)$$
$$I(Y;H) = \sum_{y \in Y} \sum_{h \in H} p(y,h) \log \left(\frac{p(y,h)}{p(y)p(h)} \right)$$

3 Neural Network Pruning

Fully trained neural networks usually end up with a lot of unutilized/underutilized pathways. These excesses can be cut off or 'pruned' to make the model use less storage and also make it more computationally efficient. Now, there are multiple methodologies for pruning. However, for the purpose of the experiment we will be pruning at initialization. That is, we will be pruning the network first and then train it to completion. There are multiple ways we can prune a network:

- **Weight Based Pruning :** in weight based pruning, we trim the weight matrices of a given layer or the whole network depending on how its carried out
- **Neuron Based Pruning :** in neuron based pruning, we trim the individual neurons/channels in a layer or again, the whole network depending on how its carried out.

So, as discussed, we can trim the weights or neurons and, we can trim either layer-wise or prune the whole network globally. For the purpose of our experiment, we first look at the impacts of layer-wise weight and neuron pruning, and then the impact of global pruning.

4 Experiment Details

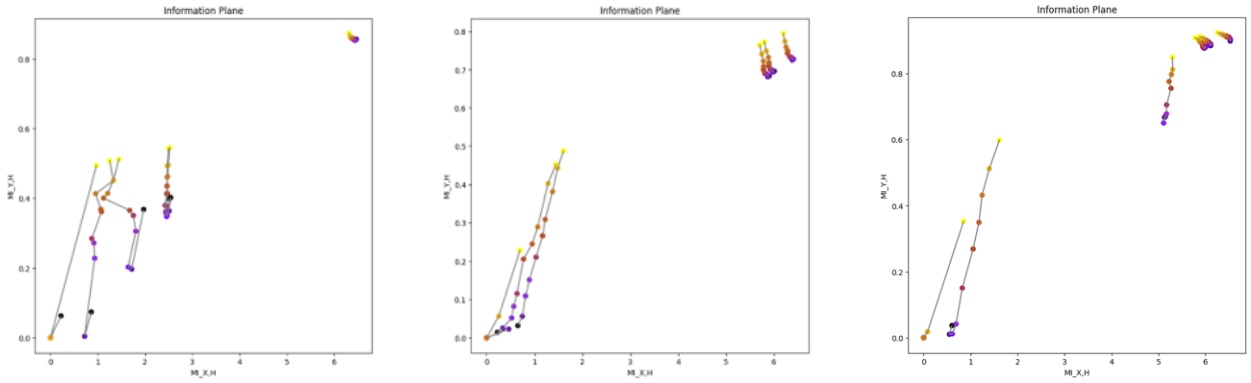
We use a 4-Hidden layer neural network with tanh activations. Our neural network was used for the task of binary classification on the MNIST dataset. It was built using *Pytorch* and pruning[1] was carried using the following Pytorch commands, *Prune.L1.Unstructured* which is used for weight pruning, here 'L1' stands for L1 norm and when a specified percentage is passed as a parameter, that percenatage of lowest performing weights are pruned. For example, if you prune at '50%', then the lowest '50%' weights will be eliminated. Similarly, *Prune.ln.Structured* is used for neuron pruning and cuts the percentage of weights specified by using L2 norm as the method of normalization.

5 Results

Upon pruning, it was noticed that most of the time, unless the parameters were not extreme the network would usually train to the expected accuracy of 98%, interesting events only happened when the pruning was done at extreme parameters to bring layers to extreme sparsity. More information about accuracy at these levels and ideal scenarios are available in the appendix

5.1 Layer Pruning

We get the following information planes when we prune the weights of a given layers to extreme sparsity:



(a) Pruning at Layer 2 (b) Pruning at Layer 4 (c) Pruning at Layer 5

Figure 1: Information planes at 95% weight pruning for different layers

We get to see some things that are very surprising, we see that the model conserves a good portion of accuracy $\sim 90\%$ while the mutual information is significantly lower than what we expected. However, when we do neuron pruning, we see that network decay is much faster, therefore, we only get a interpretable result when we prune neurons of layer 2:

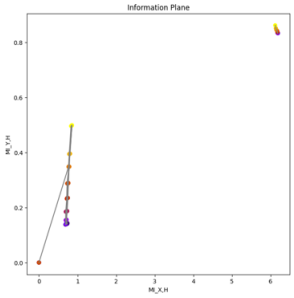
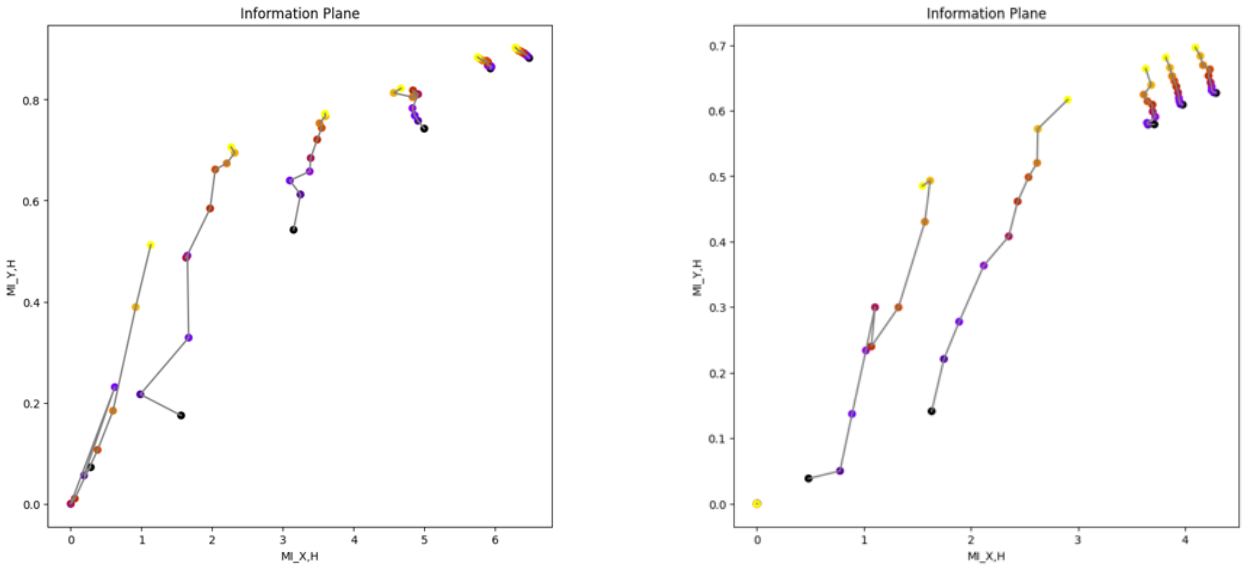


Figure 2: Information plane after doing neuron pruning on layer 2 to 90% sparsity

We see that the information plot is even more bizarre than the previous examples. One possible explanation for the low mutual information but still stable accuracy could be that partial, but usable information is still passing through the leftover channels, which gives the model just enough of an advantage to perform classification, but restricts generalization.

5.2 Network Pruning

Now, we observe similar, but at the same time slightly different results when we prune the network globally:



(a) information plane after globally pruning 65% of weights

(b) Information plane after globally pruning 40% of neurons

Figure 3: Results of Network Pruning

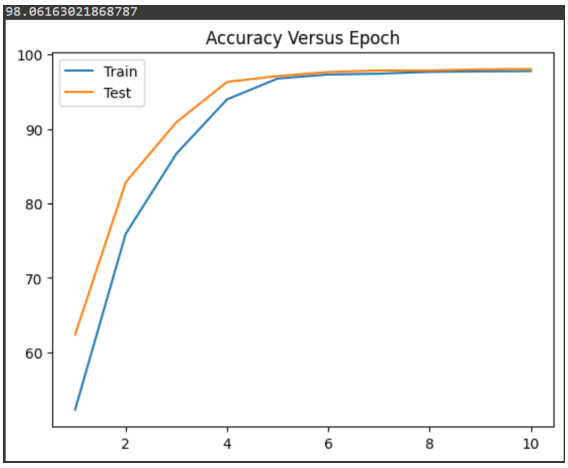
The reason for us choosing these specific limits of pruning are the fact that below these percentages, the model trains to completion, while above these values, the model fails to train. Now, analyzing the results we see that for the case of weight pruning in the network, we get similar results to the layer-wise experiment, the information plane looks similar while at the same time, we also get decent accuracy. However, for neuron pruning, we get a very bizarre result, wherein the information plane shows no mutual information between the final layer and the input layer. However, we also get a bit lesser accuracy than the other results where we did have *some* mutual information. A possible explanation for this scenario would be the number of bins was not ideal for MI calculation. However, during my experiments, in most cases the number of bins did not impact the information plane, especially in the ideal no-pruning case.

So, looking over the results as a whole, we see that our information plane is odd, and that compression is absent even at decent levels of accuracy. This could mean that either compression as a phenomenon only shows up when the network is incredibly accurate and generalizes well over training data, or, pruning itself is causing compression in the network. However, in my experiments, whenever the mutual information was high and the plot showed compression, the network showed extremely high accuracy, even with pruned networks, so there is something else causing this specific phenomenon to occur. Also, it must be noted that these information planes can be reproduced, however they are infrequent and take multiple attempts to occur. A possible future direction for this work would be to perform the same experiment on bigger and more complex networks.

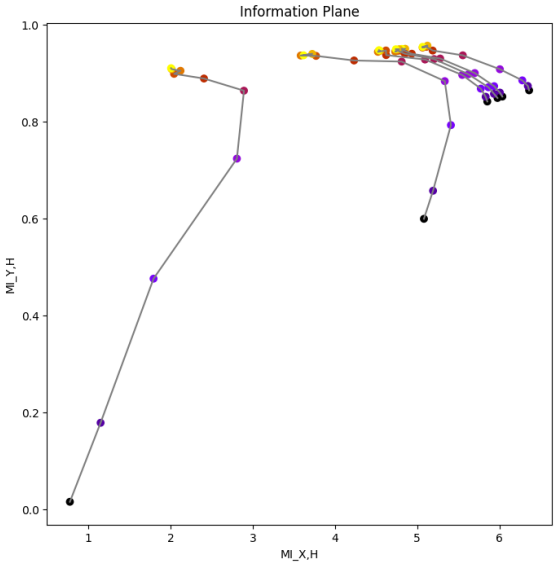
Appendix

A Ideal Information Plane and Accuracy

The ideal information plane and accuracy level for a fully trained network are as follows :



(a) Ideal Accuracy of our trained model

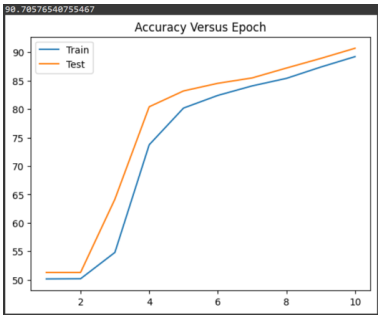


(b) Ideal Information Plane of fully trained model

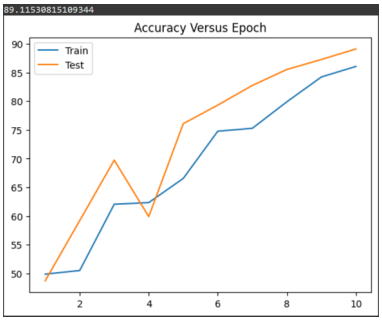
Figure 4: Accuracy plots for global network pruning

B Layer Wise Accuracy

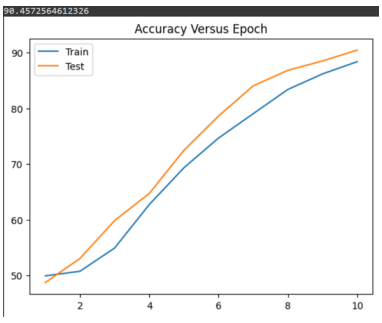
Here are the accompanying accuracy plots for each mutual information plot shown in the layer-wise experiment:



(a) Accuracy for Layer 2



(b) Accuracy for Layer 4



(c) Accuracy for Layer 5

Figure 5: Accuracy plots for shown information planes

Similarly, the accuracy plot for neuron pruning of layer 2 is :

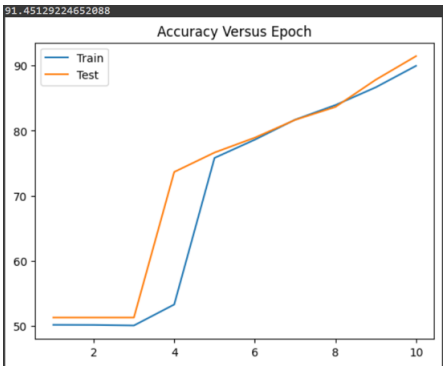
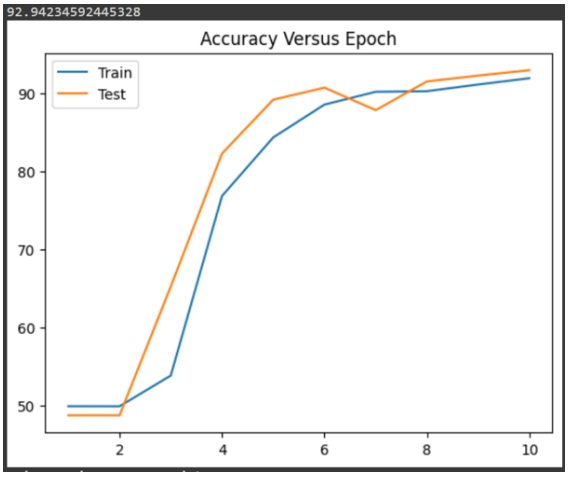


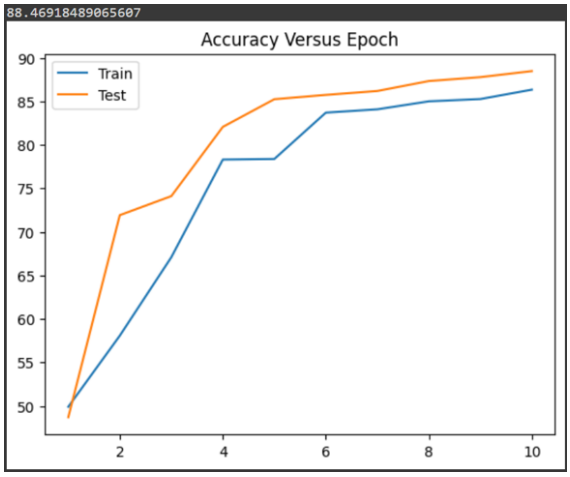
Figure 6: Accuracy of layer 2 neuron pruning

C Network Accuracy

The accuracy plots of the network at the shown levels of pruning are :



(a) Accuracy at 65% Global weight pruning



(b) Accuracy at 40% Global Neuron Pruning

Figure 7: Accuracy plots for global network pruning

D Code and additional information

As discussed earlier, these results, although reproducible are infrequent and only gotten through multiple attempts of training,also it was tested and seen that changing the number of bins did not have an impact on the information plane, other scenarios observed but not discussed in this report are :-

- The network training to completion and showcasing an ideal accuracy and information plot as displayed in section A
- The network fails to train and gets stuck in a loop, causing the model to end up with an accuracy of 51% at the end of training, equivalent to random classification.

The code used for this project is nearly identical to the code provided for HW3 in MAT1510, with only modifications performed being the addition of pruning, the code is available at this link :
`https://colab.research.google.com/drive/1M6z84ScJyNagPxzI12QOV3wA-8GfqFpa?usp=sharing`

References

[1] *Pruning Tutorial &x2014; PyTorch Tutorials 2.2.0+cu121 documentation — pytorch.org*. https://pytorch.org/tutorials/intermediate/pruning_tutorial.html. [Accessed 20-12-2023].

[2] Claude E. Shannon. "A mathematical theory of communication". In: *Bell Syst. Tech. J.* 27 (1948), pp. 623–656. URL: https://api.semanticscholar.org/CorpusID:55379485.

[3] Ravid Shwartz-Ziv and Naftali Tishby. *Opening the Black Box of Deep Neural Networks via Information*. 2017. arXiv: 1703.00810 [cs.LG].