Pruning Neural Networks

Akranth Reddy
Mechanical Engineering Department
Indian Institute of Technology Madras
Chennai, India

Anshul Vaddiraju
Electrical Engineering Department
Indian Institute of Technology Madras
Chennai, India

Vignesh Datta
Electrical Engineering Department
Indian Institute of Technology Madras
Chennai, India

***Abstract*—This report investigates the presence of smaller sparse sub networks inside a neural network which when trained achieve comparable performance to the original network with far fewer parameters. We investigate the effectiveness of various pruning approaches in discovering these "winning tickets" across various network architectures and datasets.We investigate techniques like Iterarive magnitude pruning and random pruning Our findings provide light on the LTH and random pruning's potential for improving model efficiency and generalisation in deep learning applications.**

***Index Terms*—**

I. Introduction

Neural networks' proficiency in recognizing intricate patterns is often accompanied by a tendency toward overparameterization, sporting more weights than necessary. While their prowess spans diverse domains, criticisms arise due to their taxing computational needs and memory demands. These networks, excelling in tasks like image recognition and language processing, consume substantial computational resources during both training and deployment phases. As a response to these computational challenges, the practice of "pruning" has emerged as a pivotal technique aimed at mitigating the burdens imposed by these excessively large neural networks. When it comes to pruning neural networks, several elements can be targeted for reduction. These include neurons, attention heads, weights, or even entire blocks of the network. However, recent research has predominantly focused on pruning weights, recognizing them as a key area where significant reductions can be made without sacrificing functionality. The overarching goal of this report is to comprehensively examine the methodologies, challenges, and implications surrounding weight pruning in the context of large neural networks.

II. Lottery Ticket Hypothesis

The lottery ticket hypothesis suggests that a randomly-initialized, dense neural network contains a subnetwork that is initialized such that — when trained in isolation — it can match the test accuracy of the original network after training for at most the same number of iterations. These subnetworks, termed "**winning tickets**," achieve comparable performance compared to the original larger network.The winning tickets we find have won the initialization lottery: their connections have initial weights that make training particularly effective

A. Methodology

Identifying winning tickets. We identify a winning ticket by training a network and pruning its smallest-magnitude weights. The remaining, unpruned connections constitute the architecture of the winning ticket. Instead of randomly reinitializing the pruned network, we intialize them to the initial weights. Reinitializaton improves the accuracy as well the learning time of the pruned network. We consider the network as to have won the initiation lottery i.e if you take a same initial network and initialize them with different weights, we would observe that a different winning ticket would be generated. This forms our central experiment:

1) Randomly initialize a neural network $f(x;\theta_0)$.
2) Train the network for j iterations, arriving at parameters θ_j.
3) Prune $p\%$ of the parameters in θ_j , creating a mask m.
4) Reset the remaining parameters to their values in θ_0, creating the winning ticket $f(x; m \odot \theta_0)$.

As described, this pruning approach is one-shot: the network is trained once, $p\%$ of weights are pruned, and the surviving weights are reset. However, we focus on iterative pruning, which repeatedly trains, prunes, and resets the network over n rounds; each round prunes $p^{1/n}\%$ weights that survive the previous round. Our results show that iterative pruning finds winning tickets that match the accuracy of the original network at smaller sizes than does one-shot pruning.

Algorithm 1 Iterative Magnitude Pruning

Require: Initial network parameters $f(x;\theta_0)$, pruning percentages p, number of rounds n, training iterations j.
  Initialize $\theta \leftarrow \theta_0$
  for $k \leftarrow 1$ to n **do**
    Train the network for j iterations
    Prune $p^{1/n}\%$ of the parameters in θ, creating mask m_k
    .$m_{\text{combined}} \leftarrow m_1 \odot m_2 \odot m_3 \odot \odot m_k$
    Prune parameters in θ_0 using m_{combined} to create θ_{new}
    $\theta \leftarrow \theta_{\text{new}}$
  end for
  Return $f(x; m_{\text{combined}} \odot \theta_0)$

III. Random Pruning

Random pruning is a technique in machine learning used to create sparse models by selectively removing connections or parameters in a neural network. The idea behind it stems from "The Unreasonable Effectiveness of Randomness," suggesting that by randomly pruning connections without sophisticated criteria, surprisingly effective results can be achieved.

This approach involves randomly eliminating a certain percentage of weights or connections in a neural network, effectively creating a sparse structure. Surprisingly, despite its simplicity, random pruning has demonstrated its effectiveness as a baseline technique for sparse training. This reduction in size often leads to faster inference times, reduced memory requirements, and potentially lower computational costs, making it an appealing technique in the pursuit of efficient and effective machine learning models.

A. Methodology

Denote s^l as the sparsity of layer l. Random pruning, namely, removes weights or filters in each layer randomly to the target sparsity s^l. Unlike iterative magnitude pruning, the layer-wise sparsities of random pruning is pre-defined before training. Many pre-defined layer-wise sparsity ratios in the literature are suited for random pruning, while they may not initially be designed for random pruning. We study the three below.

ERK. Erdos-Renyi Kernel (ERK) sparsifies the neural network in which larger layers are allocated with higher sparsity then smallerlayer.The sparsity of the convolutional layer is scaled proportional to $1 - \frac{n_{l-1}+n_l+w_l+h_l}{n_{l-1}\times n_l \times w_l \times h_l}$ where n_l refers to the number of neurons/channels in layer l, where w_l and h_l represent the corresponding width and height of the convolutional kernel if present, respectively.

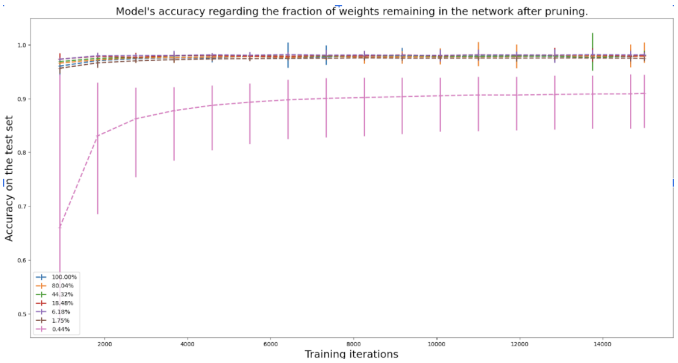
Uniform. Each layer is pruned with the same pruning rate so that the pruned network ends up with a uniform sparsity distribution.

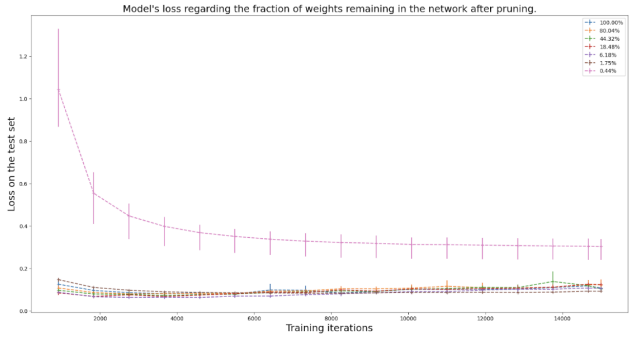
SNIP ratio. This is PaI (prune at initialisation) method where select weights based on connection sensitivity score $|w \odot g|$, where w and g is the network weight and gradient, respectively. The weights with the lowest scores in one iteration are pruned before training. While not initially designed for random pruning, we adjust SNIP for random pruning by only keeping its layer-wise sparsity ratios, while discarding its mask positions.

IV. Results and Observations

A. Lottery Ticket Hypothesis

We observe that a network pruned to about 1.7% of the weights in network on MNIST gives about a similar result as compared to the dense network. This shows the effectiveness of the winning tickets for the given task.

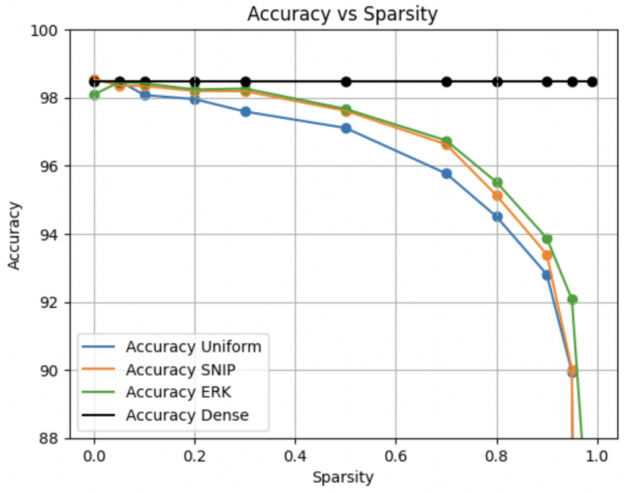




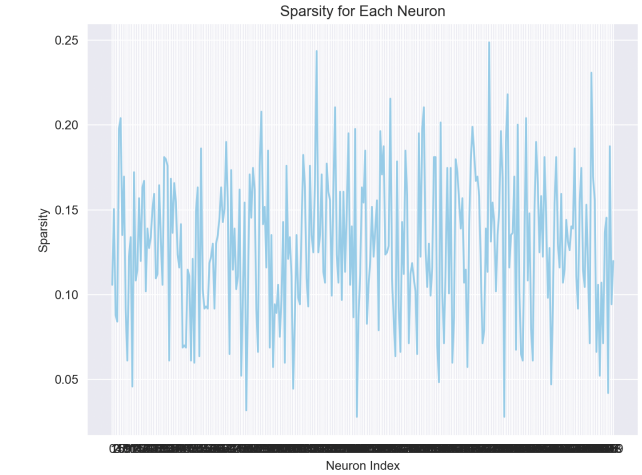
B. Random Pruning

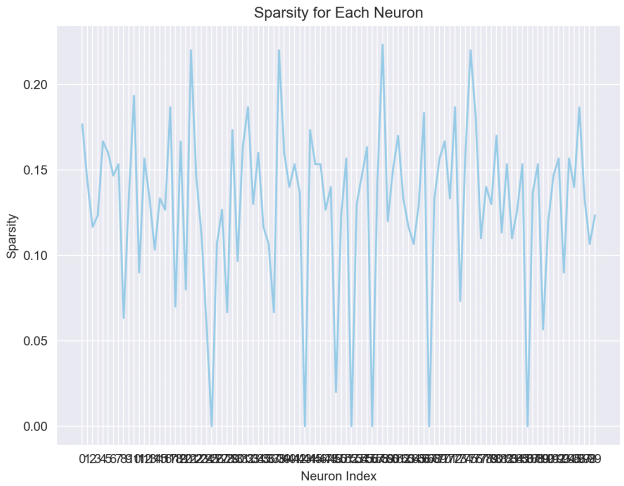
When all layers are pruned at the same rate, these smaller layers become bottlenecks, preventing us from identifying the smallest possible winning tickets. Global pruning makes it possible to avoid this pitfall. Hence we observe that uniform pruning fares worse as compared to the other pruning techniques. Another interesting observation is that ERK performs better than complex method like SNIP.

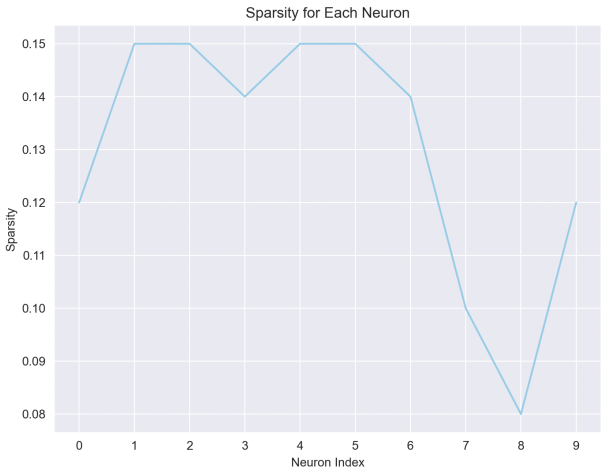
We don't observe the astonishingly high accuracy which was a trademark of IMP at >40% sparsities with random pruning. This goes along the expected lines as we prune the weights randomly as compared to IMP which uses a magnitide based pruning approach but is computationally more expensive as compared to random prunning. But the findings are positive and warrant further potential for exploration and development in the area. The larger the network the better the result of random pruning on the network.



C. Neuron Pruning







At an overall sparsity level of 13% during IMP, we've observed neuron-wise sparsity ratios. There seems to be minimal pruning of neurons in the network. It is not optimal to prune neurons as it would lead to the loss of information encoded in those neurons which when done at higher sparsity would lead to a sharp decease in the accuracy of the pruned network.