Investigating the use of hub neuron identification for pruning sparse neural networks

Andrew Heath
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
a.j.heath@student.utwente.nl

ABSTRACT

Sparsity constraints decrease computation time and memory requirements for Artificial Neural Networks (ANN). Further research has shown that pruning during each epoch based on accuracy shows similar improvements. Research on pruning has brought ANNs closer in line with their biological counterparts. However, the formation and reinforcement of hub neurons, as seen in a brain, has not been explored entirely in ANNs. Reinforcement of said hub neurons could reduce the time and memory requirements of network training pruning algorithms. This research investigates the Laplacian centrality of neurons in ANNs and Sparse Neural Networks (trained using SET) during training, showing changes in the distribution of Laplacian centrality of the neurons. We propose an ANN training method, CenSET, that uses Laplacian centrality to instruct pruning of connections during the training of a sparse ANN. We show that this approach does not dramatically decrease the accuracy compared to training an ANN using a conventional MLP or SET approach.

Keywords

Artificial Neural Network Pruning, Brain Neuron Hubs, Laplacian Centrality, Network Science, Sparse Artificial Neural Network, Sparse Evolutionary Training

1. INTRODUCTION

Deep Learning (DL) has grown in its applications, in many fields, including medicine and social science. Despite growing applications, there are still limitations in the underlying methods, such as the memory and time requirements to run DL methods. Biological neural networks have served as inspiration for DL's Artificial Neural Networks (ANN). ANNs use fully connected neuron layers that do not mimic the sparsely connected neural networks of the brain or other networks seen in nature [35, 4]. These additional connections, present only in the artificial variant, increase its computational complexity without improving the ANNs accuracy [6, 19].

Initial attempts to improve the efficiency of ANNs focused on pruning [21, 29], the removal of unnecessary neurons from the dense neural network during training, and repeating the *prune train cycle* until an equilibrium is found between performance and ANN size [27]. Later work refined the pruning process further and showed that it was sufficient to repeat the *prune train cycle* twice if the network's weights are initialized randomly [8].

35th Twente Student Conference on IT Jul. 2nd, 2021, Enschede, The Netherlands.

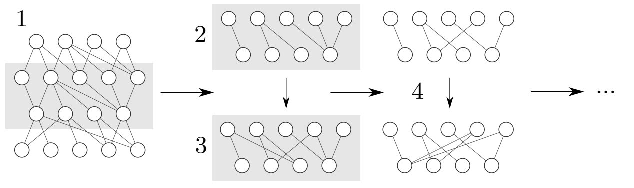


Figure 1: Visualization of SET. 1) For each sparse connected layer: 2) At the end of each training epoch, then absolute weights are removed. 3) For every weight removed a new weight is randomly added. 4) The procedure repeats as in normal ANN training

Pruning uses the approach of starting with a dense ANN and progresses towards a Sparse Neural Network (SNN) during training epochs. Recent work has focused on (static) sparse training, which in contrast to pruning, starts with an SNN and trains using gradient descent (or another method) afterwards [26, 17].

Sparse training reduces the memory requirements of the ANN and increases the training speed when compared with dense approaches to training [12, 26]. Although sparse training approaches are closer to their biological counterparts, they do not incorporate evolutionary sparsity, the replacement of neural synapses over time [15].

Sparse Evolutionary Training (SET) was proposed by Mocanu et al. as a method to introduce this evolutionary sparsity of synapses into SNN training [28]. With SET, a fraction of the smallest positive and largest negative weights are removed in each training epoch (visualised in Figure 1). The paper noted a decrease in training time with no significant decrease in accuracy using SET [28].

Further research by Lapshyna introduced the AccSET procedure, which further developed SET by cutting down on the number of connections during each training epoch while the accuracy is still increasing, bringing the SET procedure closer to synaptic pruning in the brain [20].

This research proposes using the identification of hub neurons [1] through measuring the Laplacian centrality of neurons within the network and pruning neurons that fall below a level of the centrality. Hub neurons are

[1]Highly connected and important neurons within a neural network.

typically identified as neurons with more than average connections, but this does not incorporate the weights of said connections. Laplacian centrality does account for connection weights and is therefore more suitable. The brain inspires the proposed approach, as the biological neural networks promote the growth of hub neurons over time [30].

The research began with an analysis of the Laplacian centrality of neural networks in a fully connected MLP as well as a sparse ANN trained using the SET method. This analysis aimed to answer the research question: *How does the Laplacian centrality of neurons within an ANN evolve during training?* We go on to propose a new training method, CenSET, that utilizes Laplacian centrality to prune non-central neurons. Answering the following research question: *How does pruning neurons based upon a Laplacian centrality threshold from a sparse ANN at each training epoch affect the ANN's accuracy?*

2. RELATED WORK

There are two groups of relevant literature to this research. Firstly, research into the brain's structure in relation to network science, and secondly research into the optimization of ANNs. The 2011 paper by Telesford et al.[35], shows that a graph can model the brain's internal networks, with vertex representing neurons and an edge representing a synapse. They show how network science [2] can be applied successfully as a model for the brain and that such an approach could improve understanding of the brain's functions. This idea is taken further in a 2019 paper by Oldham et al. [30] by modelling the development of the brains neural network using a graph model. The paper showed that the process of a (biological) neural network's development leads to the creation of hub neurons. An earlier conference paper by Bolanos et al. [3] in 2009 showed that network science centrality metrics could be used to find hub neurons within a brain's neural network. These three pieces of prior research make this research's premise, that centrality metrics can be used to improve the pruning of sparse neural networks, plausible.

We will now review the literature on the optimization of Artificial Neural Networks (ANN) with respect to the number of parameters in an ANN. In order to make ANNs more efficient, a parameter reduction is required [27]. Early work focused on transforming an initially dense network to a sparse one by removing neurons during training. This approach was termed *pruning* by LeCun et al [21], and Mozer et al [29].

The pruning approach involves removing connections with low weights during the training process and repeating this process each training epoch, for as many epochs as determined by the given applications time/performance tolerance. Later work by Han et al. [13] updated the pruning approach in the context of deep learning. Frankle et al. [8] showed that only 2 cycles of training and pruning are required if the ANN is initialised with random weights.

Further research on pruning brought about more complex methods such as *Simultaneous training and pruning* proposed by Louizos et al. [25] and further developed by Liu et al. [24] which trains and prunes using L_0 regularization simultaneously starting from dense networks. Another pruning approach is called *one-shot pruning* [22, 38, 41]. These works showed that NNs could be pruned faster by training for a very short time while maintaining a similar performance.

[2]The field of study of complex networks.

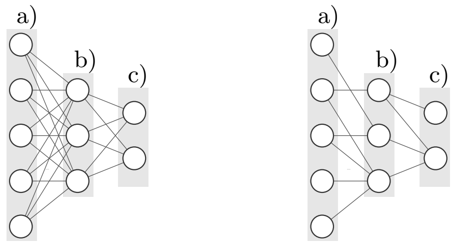


Figure 2: Example of a fully connected MLP (left) and a sparse ANN (right). a) Input layer b) Hidden layer(s) c) Output layers.

Recently the focus has shifted towards the possibility of starting with sparse neural networks and ending with one after pruning/training. This approach is termed *sparse training* [27]. Mocanu et al. proposed in 2016 a method for sparse training [26] starting from a sparse network initially and then training using gradient descent or another method. Further research [17] showed this to be a valid approach and it was later validated against other types of NN not explored in the initial study [36, 31, 2].

Although promising, this approach leads to worst performance compared to starting with a dense neural network. Later research by Mocanu et al. [28] proposed the Sparse Evolutionary Training (SET) in order to address these performance issues. With SET in addition to pruning connections, new connections are also added at each training cycle; the same number are added as were pruned in the same step. This approach proved to reduce the number of connections while obtaining a better accuracy than with a dense network.

SET is further developed in the 2020 paper by Lapshyna [20] where AccSET was proposed, building on SET with the addition of adding connections to the network during the training phase only if accuracy is decreasing.

Parallel work, such as a paper by Li et al. in 2020 [23] shows a pruning method for (dense) neural networks using Katz centrality as the pruning metric where neurons that do not meet the threshold centrality score are pruned. The paper shows this to be a valid metric to use for pruning, opening the possibility of combining a sparse training approach [28] with a network centrality metric as we will explore in our research.

3. BACKGROUND

It is assumed that the reader has basic knowledge of statistics, graph theory and linear algebra.

3.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are a type of computing system inspired by biological neural networks found in the brains of animals. Numerous kinds of ANNs exists; this research focuses on Multi-Layer Perceptrons (MLP) an example of which is seen in Figure 2. An MLP has three main structural parts: the input layer, the hidden layers and an output layer. Each layer has several neurons, and each layers neurons are connected to the neurons of the next layer by a connection with a weight. The input layer takes the initial input of information which could be data points or pixel information. The output layer classifies the input in one of the possible classes as defined by the model [33]. The hidden layers (of which there can be one or more) allow the network to classify the input correctly.

The network is trained through the supervised learning method of backpropagation. This entails adjusting the

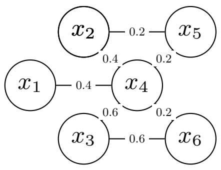


Figure 3: Example of a weighted graph.

weight of neurons depending on the error at the output of the network. This adjustment is made from the output layer to the input layer. It is important to note that MLPs are fully connected, meaning that each neuron in a layer has a weighted edge to every neuron in the next layer. This results in vast networks with many neurons and connections, making the backpropagation computationally very intensive.

3.2 Sparse ANN and SET

A sparse ANN is an ANN similar to an MLP, but it is not fully connected, Meaning each neuron in each layer is not connected to every neuron in the next layer. The difference between a fully connected MLP and a sparse ANN is visualized in Figure 2. The sparsity allows for decreased memory and computation requirements for the backpropagation of the ANN and is therefore desirable provided it does not reduce accuracy dramatically [27].

The Sparse Evolutionary Training (SET) procedure [28], is an MLP training procedure that generates an initial sparse topology for the MLP before beginning training, replacing the MLP's dense layers with sparse ones. During the training, the SET procedure removes a proportion of the connections with the smallest weights and then randomly regrows the number of removed connections. The SET procedure is visualized in Figure 1.

3.3 Graph theory & Network Metrics

A graph is made up of vertices connected by edges. An Artificial Neural Network can be modelled as a graph where the vertices are the neurons, and the edges are the connections, where edges have attributes corresponding to the connection's weight. Therefore we can denote the graph of an ANN as a weighted network, $G = (V, E, W)$, where V is the set of vertices $V(G) = v_1, v_2, ..., v_n$, E is the set of edges $E(G) = e_1, e_2, ..., e_n$ where each edge $e = (v_i, v_j)$ has an associated weight w_i, w_j.

Figure 3 shows an example weighted graph. The centrality of a vertex within a graph is a numerical score that indicates the relative importance within the network. The centrality score aims to reflect how important a given vertex is within the graph. The specific calculation for determining vertex centrality is varied as many different metrics can measure centrality. These measures differ in approach and what indicators they use to assess vertex centrality. A simple centrality metric is Betweenness centrality [9], which is calculated for a vertex as being the number of shortest paths between two other vertices that cross it [10]. The betweenness centrality (and Laplacian centrality discuss below) of the vertices of Figure 3 is given in Table 1. All vertices have betweenness 0 apart from vertex x_4 which has betweenness centrality of 0.8.

3.4 LaplacianCentrality

Laplacian centrality is a centrality metric that incorporates edge weight proposed by Xingqin et al. the explanation below will give a mathematical notation as seen in the relevant literature [32]. Given the graph G, where $G = (V, E, W)$, as defined in Subsection 3.3.

Vertex	BC	LC
x_1	0	12
x_2	0	20
x_3	0	48
x_4	8	20
x_5	0	20
x_6	0	20

Table 1: Laplacian (LC) and Betweenness centrality (BC) calculated for Figure 3 (both not normalized).

It should be noted that Laplacian centrality does not consider direction of edges so $w_{i,j} = w_{j,i}$. Loops are not considered. $A(G)$ is the adjacency matrix of graph G as is shown in Equation 1. $X(G)$ as shown in Equation 2 is the sum weight matrix of vertices of graph G where $x_i = \sum_{j=1}^{n} w_{i,j}$ (x_i is the sum weight of the vertex).

$$A(G) = \begin{bmatrix} 0 & w_{1,2} & ... & w_{1,n} \\ w_{2,1} & 0 & ... & w_{2,n} \\ . & . & . & . \\ w_{n,1} & w_{n,2} & ... & 0 \end{bmatrix} \tag{1}$$

$$X(G) = \begin{bmatrix} x_1 & 0 & ... & 0 \\ 0 & x_2 & ... & 0 \\ . & . & . & . \\ 0 & 0 & ... & x_n \end{bmatrix} \tag{2}$$

The Laplacian matrix of the graph G is defined as $L(G) = X(G) - A(G)$. Given that for graph G, $\lambda_1, \lambda_2, ..., \lambda_n$ are the eigenvalues of it's Laplacian matrix, $L(G)$. We denote the Laplacian energy of G as $E_L(G)$ where:

$$E_L(G) = \sum_{i=1}^{n} \lambda_i^2 \tag{3}$$

Given the definitions above and that G_i is G with v_i removed, we can define the Laplacian centrality for vertex v_i in G, $LC(v_i, G)$ as seen in Equation 4

$$LC(v_i, G) = (\Delta E)_i = E_L(G) - E_L(G_i) \tag{4}$$

This research uses the non-normalized version of Laplacian centrality, as we want to see the increase in Laplacian centrality between epochs.

4. CENSET

As proposed, the CenSET (Centrality SET) method is a variant of the SET method that uses Laplacian centrality (LC) to instruct pruning.

The CenSET is an ANN training algorithm. The method prunes connections in the ANN based on identifying neurons with low LC. This method has been benchmarked using an ANN initialized as sparse, but this is not a requirement for the method as the ANN could be initialized as fully connected although the process of re-adding connections would need to be adapted (we leave this exploration to further research).

CenSET initializes a sparse network (as in SET [28]) using a Erdős–Rényi model [7] and performs a standard training procedure. During each epoch, the weight layers are converted to a graph representation. LC of each neuron is then calculated, connections (weights) attached to a neuron with an LC that falls below the threshold $\mu - k\sigma$ (where k depends on the data-set being trained on) are removed. The same number of connections that are removed are then randomly added and the training continues.

This approach was motivated by the hypothesis that neurons with lower than average LC within the network are less important and therefore are good candidates to be pruned. The pseudo-code of CenSET can be found in Algorithm 1.

Algorithm 1: CenSET (based on SET [28] differences shown in red)

```
initialize ANN model;
notations are described in Section 3;
set $\epsilon$ and $\zeta$ ;
$k \leftarrow$ optimum threshold as seen in Section 6;
for each fully-connected (FC) layer of ANN do
    replace FC with a Sparse Connected (SC) layer;
end
Init training parameters;
for each training epoch $e$ do
    perform standard training procedure;
    perform weights update;
    G $\leftarrow$ graph representation of the ANN ;
    $R \leftarrow \{v | LC(v, G) < \mu - k\sigma\}$ ;
    for each bipartite SC layer of the ANN do
        remove connection if connected to neuron in $R$;
        if $e$ is not last training epoch then
            add randomly new weights (connections)
              the same amount as the ones removed
              previously;
        end
    end
end
```

4.1 Converting weight layers to a Graph

The weighted layer representation needs to be converted to an adjacency matrix to calculate Laplacian centrality on the neural network's graph. This allows it to be parsed into the NetworkX library [11], which can be further converted to the NetworKit library [34] to calculate the network's neurons Laplacian centralities. This conversion is done in the following way:

Given networks layers $L = l_1, l_2, ..., l_n$, Keras (see Subsection 5.1) stores the network weights in a matrix as in Equation 5, where $w_{1,1}$ is the weight of the connections between neuron 1 of layer l_k and neuron 1 of layer l_{k+1}.

$$K_{|l_k| \times |l_{k+1}|} = \begin{bmatrix} w_{1,1} & \dots & w_{1,k+1} \\ \vdots & \ddots & \vdots \\ w_{k,1} & \dots & w_{k,k+1} \end{bmatrix} \tag{5}$$

Given that the total number of neurons in the network is S where $S = \sum_{k=1}^{n} |l_k|$. NetworKit needs the ANN's graph to be in a adjacency matrix representation as in Equation 6. Where $w_{k,k}$ represents a connection between neruon k and neuron $k + 1$.

$$A_{S \times S} = \begin{bmatrix} w_{1,1} & \dots & w_{1,S} \\ \vdots & \ddots & \vdots \\ w_{S,1} & \dots & w_{S,S} \end{bmatrix} \tag{6}$$

In order to make the conversion to the representation in

Equation 5 to that in Equation 6 Algorithm 2 is used to perform the conversion.

5. METHODOLOGY

Algorithm 2: Converting from keras weights representation to an adjacency matrix

```
Result: $A_{S \times S}$ = adjacency matrix of ANN;
$A_{S \times S} = empty\ matrix$;
for $layer_i = 1$ to $n$ do
    calculate offsets for $layer_i$ to map layer to $A_{S \times S}$;
    Add to $A_{S \times S}$ offset weights of $layer_i$ ;
end
```

5.1 Instruments Used

For the analysis of Laplacian centrality and the creation and benchmarking of CenSET, the python machine learning libraries TensorFlow [1] and Keras [5] were used. Tensorflow is a machine learning library focusing on artificial neural networks, and Keras provides a more straightforward interface for using Tensorflow.

The GitHub repository [3] was used as a starting point. This provided an implementation of SET [28] which was used as a starting point for CenSET. The code base used to run the experiments of this research can be found here: `https://github.com/andrewjh9/CenBench`.

Other Python libraries used:

- Numpy [14] - Used to read results files, and for general matrix operations.
- scipy [37]- Used for sparse matrix operations.
- NetworKit [34] - Used for efficient implementation of Laplacian centrality.
- NetworkX [11] - Used to convert from adjacency matrix to network representation.
- matplotlib [16] - Used for creating of the plots.

All experiments (on MLP, SET, and CenSET) use the hyper-parameters [4] as given in Table 3.

5.2 Datasets

An overview of the datasets, FashionMNIST and CIFAR-10, is provided in Table 2. It should be noted that the structure of the ANN is dependent on the dataset. The sizes of each layer are listed under *Network architecture used* in Table 2. For example, 100-200-10 would refer to an input layer of 100 neurons, a single hidden layer of 200 neurons and an output layer of 10 neurons (each of which corresponds to a class).

It should also be noted that both datasets had 10 classes; further research should check to see if the findings of this paper hold for a different amount of classes.

FashionMNIST [40] is a dataset made up of 28×28 images of Zalando's products. The dataset is made up of 60,000 training examples and 10,000 testing examples. CIFAR-10 [19] is a dataset made up of 32×32 images. The dataset comprises of 50,000 training examples and 10,000 testing examples.

5.3 Analyzing Centrality

For analyzing Laplacian centrality, two ANNs were trained, one using an MLP approach and another the SET approach. Each approach was run on each dataset. During each epoch, the conversion as described in Subsection 4.1 is performed, then the $A_{S \times S}$ matrix

[3]`https://github.com/dcmocanu/sparse-evolutionary-artificial-neural-networks`
[4]A hyper-parameter is a parameter that controls the learning process in a machine learning model

Dataset	Type	# Classes	Training Set Size	Testing Set Size	Network architecture used
FashionMNIST [39]	Image	10	60000	10000	784-256-128-100-10
CIFAR-10[18]	Image	10	50000	10000	3072-4000-1000-4000-10

Table 2: Information on the datasets used in analysis of centrality and benchmarking CenSET.

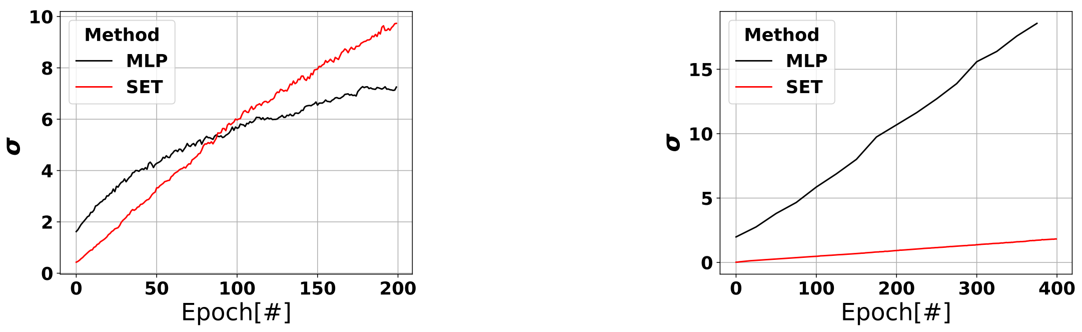


Figure 4: Standard deviation (σ) of the Laplacian centrality of the neurons for MLP and SET. (left) FashionMNIST over 200 epochs, (right) CIFAR-10 over 400 epochs.

Hyper-parameter	Value	Hyper-parameter	Value
activation function	SRelu	momentum	0.9
batch size	100	optimiser	SGD
dropout rate	0.3	ζ	0.05
learning rate	0.01	ϵ	20
loss function	Categorical cross-entropy		

Table 3: Hyperparameters used in all experiments

is read by the NetworKit library [34]. The NetworKit's Laplacian centrality function is then called on the network, and this returns each neuron's Laplacian centrality. These Laplacian values were then saved to CSV files for later analysis. During the analysis of the Laplacian centrality, the conversion as described in Subsection 4.1 and calling of the Laplacian centrality function was done each epoch.

5.4 Finding Optimum Pruning Threshold in CenSET

To find an optimum pruning level for each dataset in CenSET, trials were conducted on the two datasets. Each dataset started with a wide search pruning between $\mu - 0\sigma$ and $\mu - 4\sigma$ at intervals of 0.5σ. The number of epochs varied per dataset, with FashionMNIST's wide search running for 50 epochs and CIFAR-10's for 100 epochs. This difference in the number of epochs reflects the different dataset sizes. The accuracy for each trial was calculated as the mean of the last 10 epochs.

Further trials for FashionMNIST consisted of a narrow search between $\mu - 2.1\sigma$ and $\mu - 3\sigma$ at intervals of 0.1σ. This range was chosen as the wide search showed this to be the range in which the highest accuracy lies.

CIFAR-10's further trials consisted of two parts; a *semi-narrow* search and a *narrow search* (this was due to the long time required to conduct searches on CIFAR-10 at a small interval). The semi-narrow search consisted of trials between $\mu - 2\sigma$ and $\mu - 4\sigma$ at intervals of 0.4σ. Finally the narrow search consisted of trials between $\mu - 2.8\sigma$ and $\mu - 4\sigma$ at intervals of 0.1σ.

6. RESULTS

6.1 Results of Analyzing Centrality

As seen in Figure 8, results of the analysis of Laplacian centrality clearly shows an observable pattern. The figure shows the Laplacian centrality distribution across the neurons at 25 epoch intervals for Multi-Layered-Perceptron (MLP). It can be seen that initially, the distribution has a low standard deviation (σ), and as the number of epochs increases, σ also increases. This is true for both CIFAR-10 and FashionMNIST as seen in Figure 4. This visual intuition can be confirmed by Table 9 which shows $\sigma = 1.162$ in epoch 0 and $\sigma = 7.074$ by epoch 175 an average increase of 26.9% per epoch for FashionMNIST. Also in Table 10 we see a similar pattern for CIFAR-10 with an initial $\sigma = 1.984$ and by epoch 375, $\sigma = 18.563$ an average increase of 16.5% per epoch for CIFAR-10.

For SET [28] a similar picture emerges, but the centrality is affected by the changes in training that SET imposes. The sparse topology that it imposes (see Subsection 3.2 for details) decreases the number of neurons in the network, and therefore the initial σ for FashionMNIST and CIFAR-10 is 0.429 and 0.02, respectively. For FashionMNIST, σ shows a 72% increase from epoch 0 to epoch 175 and CIFAR-10 shows a 66% increase between epoch 0 and epoch 375.

6.1.1 *Centrality and Accuracy*

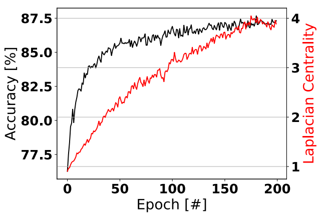
It can be seen in Figure 5 that a similar centrality curve is observed in both SET and MLP across both datasets. Laplacian centrality displays linear growth over epochs. Although accuracy and centrality are both increasing, it cannot be said from this investigation that they are correlated although the data does imply this.

6.2 Finding optimum pruning level for CenSET

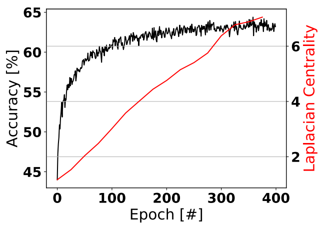
CenSET, requires a pruning level to be found per dataset. Multiple trials where conducted as described in Subsection 5.4. The results of these trials will be discussed per dataset. The graph G will be considered as the graph representation of the an ANN, the notation will be as given in Section 3.

Model	**CenSET**				**SET**			**MLP**	
Dataset	Acc %	Conn #	LC	Prune Threshold	Acc %	Conn #	LC	Acc %	LC
FashionMNIST	84.87	33337	2.595 ± 5.599	$< \mu - 2.6\sigma$	86.00	33230	5.090 ± 9.732	87.3	3.886 ± 7.252
CIFAR-10	64.34	341124	0.452 ± 2.573	$< \mu - 3.2\sigma$	67.08	341664	0.676 ± 1.807	63.10	7.050 ± 18.563

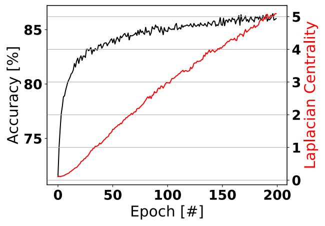
Table 4: Comparison between CenSET (at optimum pruning threshold for the dataset), SET and MLP. Values given are the mean of the final 10 epochs. FashionMNIST was run for 200 epochs and CIFAR-10 was run for 400 epochs.



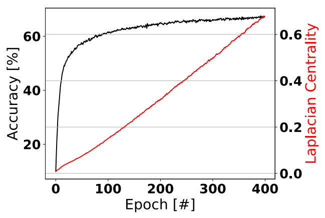
(a) FashionMNIST (MLP)



(b) CIFAR-10 (MLP) *



(c) FashionMNIST (SET)



(d) CIFAR-10 (SET)

Figure 5: Accuracy vs Laplacian Centrality for MLP and SET. Showing FashionMNIST over 200 epochs and CIFAR-10 over 400 epochs. *(For CIFAR-10 MLP, LC was sampled at 25 epoch intervals and a line was fitted)

	FashionMNIST	**CIFAR-10**
Threshold	**Accuracy (%)**	**Accuracy (%)**
$\mu - 0\sigma$	61.63	10.00
$\mu - 0.5\sigma$	61.63	47.54
$\mu - \sigma$	80.11	49.01
$\mu - 1.5\sigma$	82.11	50.09
$\mu - 2\sigma$	82.59	51.40
$\mu - 2.5\sigma$	83.45	58.75
$\mu - 3\sigma$	83.24	59.17
$\mu - 3.5\sigma$	83.09	59.17
$\mu - 4\sigma$	83.06	59.31

Table 5: Wide search for optimum value pruning for CenSET. Mean accuracy of last 10 epochs is given. FashionMNIST was run for 50 epochs and CIFAR-10 was run for 100 epochs.

6.2.1 *FashionMNIST*

FashionMNIST's wide search can be seen in Figure 6 left, this results in a optimum range being identified between 2.1σ and 3σ. A further narrow search between 2.1σ and 3σ is seen in Figure 6 right. From this search, we found the optimum pruning threshold to be 2.6σ (It should be noted that the difference in accuracy between the thresholds in the narrow search are not significant and may represent noise between the different trials, but the optimum does sit within the narrow search range). Therefore the pruning

Threshold	**Accuracy (%)**	**Threshold**	**Accuracy (%)**
$\mu - 2\sigma$	84.43	$\mu - 2.6\sigma$	**84.87**
$\mu - 2.1\sigma$	84.21	$\mu - 2.7\sigma$	84.76
$\mu - 2.2\sigma$	84.50	$\mu - 2.8\sigma$	84.45
$\mu - 2.3\sigma$	84.29	$\mu - 2.9\sigma$	84.71
$\mu - 2.4\sigma$	84.46	$\mu - 3\sigma$	84.83
$\mu - 2.5\sigma$	84.62		

Table 6: Narrow search for optimum value pruning for CenSET on FashionMNIST, over 200 Epochs. Mean accuracy across epochs last 10 epochs on is given.

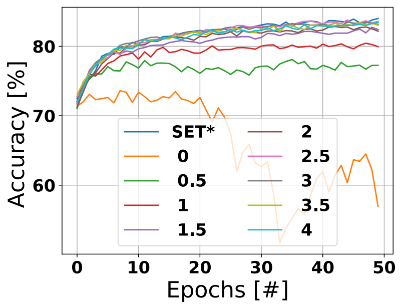
function p of neuron v where $v \in$ V is:

$$p(v) = \begin{cases} \text{prune } v, & \text{if } LC(v,G) < \mu - 2.6\sigma \\ \text{don't prune } v, & \text{otherwise} \end{cases}$$

The full results of the narrow and wide search can be seen in Table 6 and Table 5 respectively.

6.2.2 *CIFAR10*

CIFAR-10's wide search can be seen in Figure 7 (left), this results in a optimum range being identified between 2σ and 4σ. A secondary semi-narrow search as seen in Figure 7 (middle) between the previously stated interval resulting in a another range being found between 2.8σ and 4σ. A final search as seen in Figure 7 (right) resulted



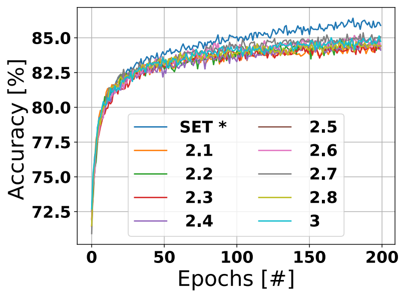


Figure 6: Finding optimum pruning level for FashionMNIST. Pruning neuron v *if* $LC(v,G) < \mu - x\sigma$ (where x is given in the legend). (left) Wide search over 50 epochs, at intervals of 0.5σ between 0 and 4σ. (right) Narrows search over 200 epochs, at intervals of 0.1σ between 2.1σ and 3σ. * SET is a baseline.

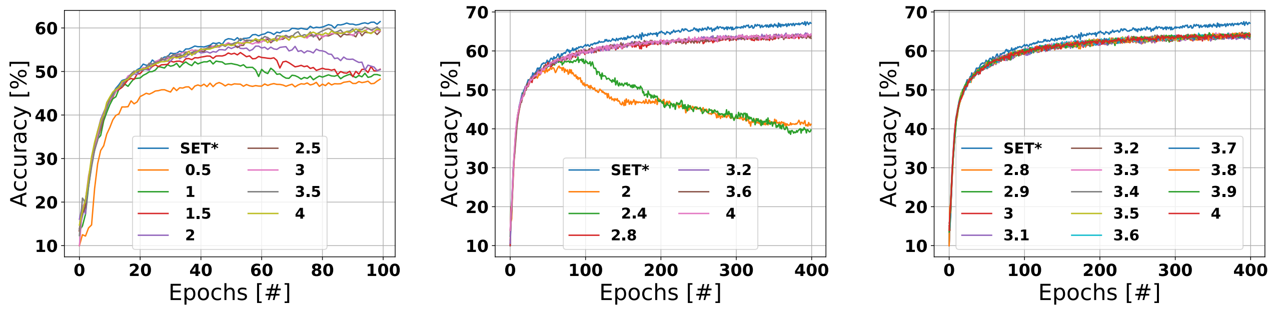
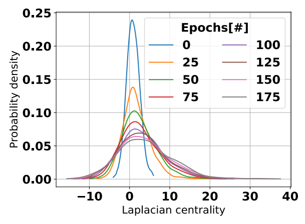
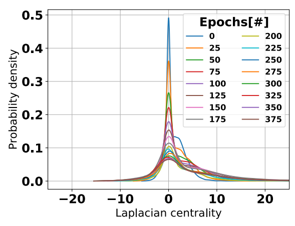


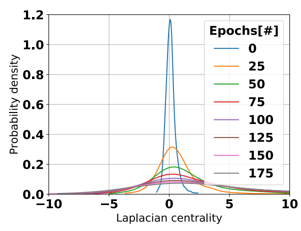
Figure 7: Finding optimum pruning level for CIFAR-10. Pruning neuron v *if* $LC(v,G) < \mu - x\sigma$ (where x is given in the legend). (left) Wide search over 100 epochs at intervals of 0.5σ between 0.5σ and 4σ. (middle) Semi-narrow search over 400 epochs between 2σ and 4σ at intervals of 0.4σ. (right) Narrow search over 400 epochs between 2.8σ and 4σ at intervals of 0.1σ. * SET is a baseline.



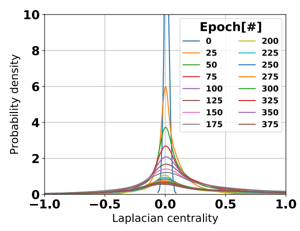
(a) FashionMNIST (MLP)



(b) CIFAR-10 (MLP)



(c) FashionMNIST (SET)



(d) CIFAR-10 (SET)

Figure 8: Probability density of MLP and SET distributions of Laplacian centrality (LC) of neurons shown at every 25 epochs. 175 epochs trials for FashionMNIST and 375 epochs for CIFAR-10. LC range is limited for readability.

Threshold	Accuracy (%)
$\mu - 2\sigma$	41.19
$\mu - 2.4\sigma$	39.59
$\mu - 2.8\sigma$	63.75
$\mu - 3.2\sigma$	64.14
$\mu - 3.6\sigma$	63.59
$\mu - 4\sigma$	64.02

Table 7: Semi-narrow search trials for optimum pruning threshold for CenSET over 400 epochs for CIFAR-10. Mean accuracy last 10 epochs given.

Threshold	Accuracy (%)	Threshold	Accuracy (%)
$\mu - 2.8\sigma$	63.75	$\mu - 3.5\sigma$	63.88
$\mu - 2.9\sigma$	63.60	$\mu - 3.6\sigma$	64.06
$\mu - 3\sigma$	63.89	$\mu - 3.7\sigma$	63.90
$\mu - 3.1\sigma$	63.30	$\mu - 3.8\sigma$	64.14
$\mu - 3.2\sigma$	**64.34**	$\mu - 3.9\sigma$	64.16
$\mu - 3.3\sigma$	64.18	$\mu - 4\sigma$	64.02
$\mu - 3.4\sigma$	64.02		

Table 8: Narrow search trials for optimum CenSET pruning threshold for CIFAR-10. Mean accuracy over last 10 epochs given.

in the optimum pruning threshold to be 3.2σ (It should be noted as with FashionMNIST that the difference in accuracy between the thresholds in the narrow search are not significant and may represent noise between the different trials, but the optimum does sit within the narrow search range). Therefore the pruning function p of neuron v where $v \in$ V is:

$$p(v) = \begin{cases} \text{prune } v, & \text{if } LC(v,G) < \mu - 3.2\sigma \\ \text{don't prune } v, & \text{otherwise} \end{cases}$$

The full results of the narrow, semi-narrow and wide search can be seen in Table 8, Table 7 and Table 5 respectively.

6.3 Benchmarking CenSET

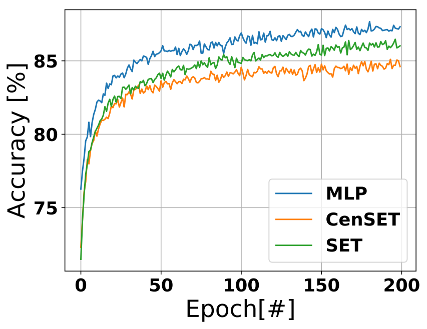
With the optimum prune thresholds of 2.6σ for FashionMNIST and 3.7σ for CIFAR-10. CenSET was benchmarked against SET and an MLP. The results of this benchmarking is summarized in Table 4 and plotted in Figure 9. The benchmarking shows that CenSET underperforms both MLP and SET training approaches; for FashionMNIST, CenSET under-performs SET by 2.3% and MLP by 3.8% on MLP. As for CIFAR-10, a different picture emerges, CenSET underperforms SET by 3.8% but performs better than MLP by 1.25%. Table 4 shows the full comparison between the methods.

7. DISCUSSION

7.1 Discussion of Analyzing Centrality

The findings of an increase in the spread of the distribution of Laplacian centrality as the ANN trains is exciting. This result implies the existence of hub neurons within the ANN. Furthermore, the standard deviation increases over the training epochs, this implies a small set of neurons are becoming more important than others.

The formation of the higher distribution spread seems to be important to the learning of the ANN as the accuracy is increasing the most during the epochs that see the biggest changes in distribution. The first four epoch intervals for FashionMNIST show the most distribution change as seen in Figure 8 (this holds for both SET and MLP), this distribution change over the first 100 epochs happens in approximately the same interval as the largest accuracy increase as seen in Figure 5.



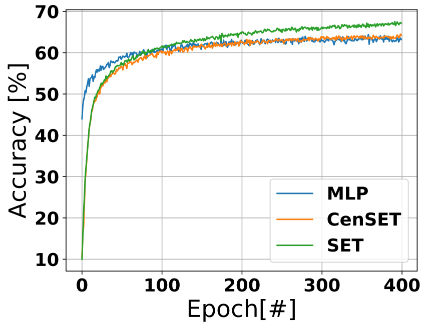


Figure 9: MLP, SET and CenSET at optimum pruning accuracy compared. (left) FashionMNIST over 200 epochs. (right) CIFAR-10 over 400 epochs.

Although we do not formally demonstrate a link between the increase in standard deviation and accuracy in this research, our results empirically show it. Based on this observation, further lines of research are mentioned in Section 9.

7.2 Discussion of CenSET

CenSET currently underperforms SET on both datasets and performs worst than MLP on FashionMNIST and approximately matches MLP's accuracy on CIFAR-10. We believe this performance to be due to two reasons: the decreasing removal rates as CenSET trains longer and possible knock-on effects of removing certain non-central neurons. The underperformance against SET is less than a 5% decrease in accuracy across both datasets tested. We would think that with a refined version of CenSET, it may be possible to match or surpass the other methods on accuracy.

As CenSET trains with a fixed pruning threshold, the standard deviation becomes stable quicker than in SET and MLP. This is because lower outliers are being pruned. Once the standard deviation stabilizes, very few (if any) neurons will be removed in a given epoch. This lack of removal prevents the network from learning. SET, unlike CenSET, will always remove a fixed amount of connections in a given epoch, ensuring that the network has enough new connections to continue increasing accuracy.

Another possible explanation for CenSET's performance is the knock-on effects of the removal of neurons of low centrality. The removal of neurons with low centrality could lead to an overall drop of centrality in the whole network in a given epoch. As the neurons being removed could be increasing the centrality of neurons they are connected to. A different possible approach would be to only remove low centrality neurons while the centrality of the whole network is not increasing. This approach is similar to the AccSET method [20], which uses inter epoch accuracy change as a metric for deciding when to re-add connections after pruning or not. Deciding not to re-add could lead to a reduction in the a number of connections in the network.

8. CONCLUSION

Within this research, a first attempt to sparsify a neural network using inspiration from network science and graph theory is made. Prior works [28, 13] were focusing on pruning criteria like magnitude pruning and gradient-based pruning. In this work, we introduce the use of centrality metrics as a new pruning criterion in a dynamic sparse training algorithm. The proposed method is named CenSET. CenSET is extending the sparse evolutionary training algorithm [28] and uses Laplacian centrality (LC) as a pruning criterion.

This research shows that the LC of an MLP changes as it trains and that this also holds for networks trained using SET [28](an MLP based approach that uses sparse topologies as well as a prune re-grow cycle during training). Most notably, the standard deviation of the network increases linearly. We also propose a training approach, CenSET, similar to SET, but it uses LC as a metric for deciding which neurons to prune. To this end, LC was proven to provide very interesting insights into the existence of hub neurons in dense and sparse neural networks. The results show that the proposed method (CenSET) has the ability to account for hub neurons at a similar or slightly lower accuracy, with a good level of stability.

9. FURTHER RESEARCH

This paper brings to light several new directions for research into the structure and centrality of neurons in an ANN as it is being trained. Firstly, more specific analysis should be done to identify if there are hub neurons being formed in the network during training. This research implied this but did not prove it directly. Further development of CenSET could also be possible, in which a non-random method of re-adding connections could be explored. This re-adding method could use information regarding what new connections would increase the network's overall centrality. As discussed in the conclusion, centrality could also be used to tell whether the networks are learning along with the increase of accuracy and could be used to instruct a pruning approach that only re-adds connections if centrality is decreasing between epochs.

Finally, research could be undertaken to create a more appropriate centrality metric for Artificial Neural Networks. Our research used Laplacian Centrality, which does not take into account the direction of edges (connections). It is also a metric created for graphs generally and doesn't consider topological considerations of ANNs, such as the distinction between input, output, and hidden layers. The different types of layers could well be used to make distinctions when calculating neuron centrality. The development of a more appropriate centrality metric for neurons in an ANN could lead to better identification of hub neurons, which could be used to prune or sparsify the ANN in novel ways.

ACKNOWLEDGMENTS

I would like to give special thanks to my supervisor Elena Mocanu for her all-around support during this research project. I would also like to thank Freek Nijweide for his helping in discussing theoretical matters and finally, to my small army of proofreaders.

10. REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang. TensorFlow: A system for large-scale machine learning. *CoRR*, abs/1605.08695, 2016. _eprint: 1605.08695.

[2] N. Ailon, O. Leibovich, and V. Nair. Sparse Linear Networks with a Fixed Butterfly Structure: Theory and Practice. *arXiv:2007.08864 [cs, stat]*, July 2020. arXiv: 2007.08864.

[3] M. E. Bolanos and S. Aviyente. Identifying centralized hubs within neural functional connections. In *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, pages 25–28, 2009.

[4] D. M. Busiello, S. Suweis, J. Hidalgo, and A. Maritan. Explorability and the origin of network sparsity in living systems. *Scientific Reports*, 7(1):12323, Sept. 2017.

[5] F. Chollet and others. *Keras*. 2015.

[6] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas. Predicting parameters in deep learning. *CoRR*, abs/1306.0543, 2013.

[7] P. Erd. os and A. R enyi, On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[8] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[9] L. C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, 1977. Publisher: [American Sociological Association, Sage Publications, Inc.].

[10] R. P. Grimaldi. *Discrete and Combinatorial Mathematics: an Applied Introduction*. Pearson Addison Wesley, Boston, 2004.

[11] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx.

[12] S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv:1510.00149 [cs]*, Feb. 2016. arXiv: 1510.00149.

[13] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both Weights and Connections for Efficient Neural Networks. *arXiv:1506.02626 [cs]*, Oct. 2015. arXiv: 1506.02626.

[14] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[15] J. Hawkins. Special report : Can we copy the brain? - What intelligent machines need to learn from the Neocortex. *IEEE Spectrum*, 54(6):34–71, 2017.

[16] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. Publisher: IEEE COMPUTER SOC.

[17] J. Kepner and R. A. Robinett. Radix-net: Structured sparse matrices for deep neural networks. pages 268–274, 2019.

[18] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

[19] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*, 2012.

[20] V. Lapshyna. Sparse Artificial Neural Networks: Adaptive Performance-based Connectivity inspired by Human-Brain processes, 2020. Bachelor Tshesis.

[21] Y. LeCun, J. Denker, and S. Solla. Optimal Brain Damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1990.

[22] N. Lee, T. Ajanthan, and P. H. S. Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity. *arXiv:1810.02340 [cs]*, Feb. 2019. arXiv: 1810.02340.

[23] W. Li, M. Chu, and J. Qiao. A pruning feedforward small-world neural network based on Katz centrality for nonlinear system modeling. *Neural Networks*, 130:269–285, 2020.

[24] J. Liu, Z. Xu, R. Shi, R. C. C. Cheung, and H. K. H. So. Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers. Sept. 2019.

[25] C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

[26] D. C. Mocanu, E. Mocanu, P. H. Nguyen, M. Gibescu, and A. Liotta. A topological insight into restricted Boltzmann machines. *Machine Learning*, 104(2):243–270, Sept. 2016.

[27] D. C. Mocanu, E. Mocanu, T. Pinto, S. Curci, P. H. Nguyen, M. Gibescu, D. Ernst, and Z. A. Vale. Sparse Training Theory for Scalable and Efficient Agents. *arXiv:2103.01636 [cs]*, Mar. 2021. arXiv: 2103.01636.

[28] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):2383, June 2018.

[29] M. C. Mozer and P. Smolensky. Using Relevance to Reduce Network Size Automatically. *Connection Science*, 1(1):3–16, Jan. 1989. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/09540098908915626.

[30] S. Oldham and A. Fornito. The development of brain network hubs. *Developmental Cognitive Neuroscience*, 36:100607, 2019.

[31] A. Prabhu, G. Varma, and A. M. Namboodiri. Deep expander networks: Efficient deep networks from graph theory. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, volume 11217 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2018.

[32] X. Qi, E. Fuller, Q. Wu, Y. Wu, and C.-Q. Zhang. Laplacian centrality: A new centrality measure for weighted networks. *Information Sciences*, 194:240–253, July 2012.

[33] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[34] C. L. Staudt, A. Sazonovs, and H. Meyerhenke. Networkit: A tool suite for large-scale complex network analysis, 2015.

[35] Q. K. Telesford, S. L. Simpson, J. H. Burdette, S. Hayasaka, and P. J. Laurienti. The Brain as a Complex System: Using Network Science as a Tool for Understanding the Brain. *Brain Connectivity*, 1(4):295–308, Oct. 2011. Publisher: Mary Ann Liebert, Inc., publishers.

[36] K.-a. Tessera, S. Hooker, and B. Rosman. Keep the Gradients Flowing: Using Gradient Flow to Study Sparse Network Optimization. *arXiv:2102.01670 [cs]*, Feb. 2021. arXiv: 2102.01670.

[37] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17.

[38] C. Wang, G. Zhang, and R. Grosse. Picking Winning Tickets Before Training by Preserving Gradient Flow. *arXiv:2002.07376 [cs, stat]*, Aug. 2020. arXiv: 2002.07376.

[39] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747, 2017. _eprint: 1708.07747.

[40] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[41] S. Zhang and B. C. Stadie. One-shot pruning of recurrent neural networks by jacobian spectrum evaluation. In *International Conference on Learning Representations*, 2020.

APPENDIX

A. LAPLACIAN CENTRALITY

Epoch	**MLP**	**SET**	**CenSET**
0	0.90 ± 1.62	0.12 ± 0.42	0.15 ± 0.50
25	1.29 ± 3.38	0.27 ± 1.74	0.34 ± 1.17
50	1.72 ± 4.28	0.66 ± 3.32	0.74 ± 2.05
75	2.06 ± 5.09	1.07 ± 4.55	1.12 ± 2.75
100	2.41 ± 5.64	1.55 ± 5.97	1.49 ± 3.47
125	2.63 ± 6.00	1.90 ± 7.05	1.76 ± 4.10
150	2.72 ± 6.54	2.23 ± 7.95	2.04 ± 4.66
175	2.95 ± 7.07	2.60 ± 8.84	2.35 ± 5.16

Table 9: Mean Laplacian centrality across different methods on FashionMNIST

Epoch	**MLP**	**SET**	CenSET
0	1.16 ± 1.98	0.01 ± 0.02	0.01±0.02
25	1.16 ± 2.76	0.02 ± 0.16	0.05±0.39
50	1.53 ± 3.80	0.04 ± 0.26	0.03±0.48
75	1.53 ± 4.65	0.06 ± 0.37	0.03±0.61
100	2.03 ± 5.85	0.07 ± 0.47	0.04±0.74
125	2.03 ± 6.88	0.09 ± 0.59	0.06±0.89
150	2.48 ± 8.00	0.11 ± 0.59	0.08±1.03
175	2.48 ± 9.74	0.13 ± 0.81	0.11±1.17
200	3.02 ± 10.68	0.15 ± 0.92	0.15±1.36
225	3.02 ± 11.62	0.16 ± 1.04	0.18±1.51
250	3.582± 12.70	0.19 ± 1.15	0.22±1.65
275	3.58 ± 13.88	0.21 ± 1.26	0.26±1.81
300	4.01 ± 15.58	0.23 ± 1.38	0.30±1.97
325	4.01 ± 16.38	0.25 ± 1.48	0.34±2.16
350	4.44 ± 17.58	0.27 ± 1.60	0.38±2.32
375	4.44 ± 18.56	0.29 ± 1.72	0.42±2.51

Table 10: Mean Centrality across different methods on Cifar10