REVISITING THE LOTTERY TICKET HYPOTHESIS: A RAMANUJAN GRAPH PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks often yield to weight pruning resulting in a sparse subnetwork that is adequate for a given task. Retraining these 'lottery ticket' subnetworks from their initialization minimizes the computational burden while preserving the test set accuracy of the original network. Based on our knowledge, the existing literature only confirms that pruning is needed and it can be achieved up to certain sparsity. We analyze the pruned network in the context of the properties of Ramanujan expander graphs. We consider the feed-forward network (both multi-layer perceptron and convolutional network) as a series of bipartite graphs which establish the connection from input to output. Now, as the fraction of remaining weights reduce with increasingly aggressive pruning two distinct regimes are observed: initially, no significant decrease in accuracy is demonstrated, and then the accuracy starts dropping rapidly. We empirically show that in the first regime the pruned lottery ticket sub-network remains a Ramanujan graph. Subsequently, with the loss of Ramanujan graph property, accuracy begins to reduce sharply. This characterizes an absence of resilient connectivity in the pruned sub-network. We also propose a new magnitude-based pruning algorithm to preserve the above property. We perform experiments on MNIST and CIFAR10 datasets using different established feed-forward architectures and show that the winning ticket obtained from the proposed algorithm is much more robust.

1 INTRODUCTION AND RELATED WORK

Neural Network (NN) and its recent advancements have made a significant contribution to solve various machine learning applications. The power of an over-parameterized NN lies on its capability to learn simple patterns and memorize the noise in the data (Neyshabur et al., 2018). However, the training of such networks requires enormous computational resources, and often the deployment onto low-resource environments such as mobile devices, or embedded systems becomes difficult. One obvious alternative to solve this is *Network Pruning* (LeCun et al., 1990; Blalock et al., 2020). Neural network pruning involves sparsification of the network. It identifies the weight parameters; removal of which incurs minimal effect on the generalization error. There exist different categories of pruning based on (i) how the pruning is performed like based on the weight magnitude (Han et al., 2015; Zhu & Gupta, 2017), gradient in the backpropagation, hessian of the weight (Hassibi et al., 1993; Dong et al., 2017), etc; (ii) whether the pruning is global or local; (iii) how often pruning should be applied like one-shot, iterative. One of the primary goals in the literature has been to reduce the computational footprint at the time of prediction, i.e., during post-training.

The current trend has shifted towards pre-training following the introduction of a remarkable contribution, named *Lottery Ticket Hypothesis (LTH)*, which hypothesize the existence of a highly sparse subnetwork and weight initialization to reduce the training resources as well (Frankle & Carbin, 2019). It provides a simple iterative, magnitude-based pruning algorithm, and empirically shows that even after removing approximately 90% of the weights, the subnetwork can preserve the original generalization error. In the subsequent studies, the focus goes on finding this lottery ticket for more competitive tasks by pruning with weight rewinding(Frankle et al., 2019a), fine tuning the learning rates (Renda et al., 2020), more efficient training (You et al., 2019; Brix et al., 2020; Girish et al., 2021), different natural language processing tasks (Chen et al., 2020a; Brix et al., 2020), finding winning ticket on graph neural network (Chen et al., 2021), etc. Evci et al. (2020b) explain empirically why the LTH works through the gradient flow at different stages of the training. Despite

previous attempts to explain why the Lottery Ticket Hypothesis works, the underlying phenomenon associated with the hypothesis still remains ill-understood.

All of these studies related to LTH identify that a sparse sub-network can be trained instead of a complete network and the network needs to be connected from input to output layers. However, none of them try to explain the LTH and the properties of the pruned network through the lens of spectral graph theory. The connectivity can be described from the *graph expansion* point of view, where any subset of vertices of size less than or equal to the half of the number of vertices in a graph, is adjacent to at least a fraction of the number of vertices in that set; for details, see (Lubotzky, 2010). In this paper, we initiate a study to observe the characteristics of a pruned sub-network from the eigen spectrum of its adjacency matrix, which, to the best of our knowledge, has not been reported previously. Note that, the *Ramanujan Graph* is a special graph in a bounded degree expander family, where the eigenbound is maximal (Nilli, 1991). This leads to a maximum possible sparsity of a network while preserving the connectivity. Hence, it is interesting to know if there is any connection between the eigenbound, and how long the pruning to be continued, preserving the test accuracy. We consider a feed-forward NN as a series of connected bipartite graphs and study the results of Hoory (2005) on the bound of spectral gap for the weight matrix of pruned network. Also other than test accuracy, it is necessary to observe the impact of noise into the data with its connection to Ramanujan graph characteristics.

Contributions The contributions of the paper are the followings:

- We propose a methodology for analyzing winning lottery ticket with respect to their Ramanujan graph properties.
- We empirically show that the LTH holds as long as the pruned sub-network follows a constraint on the second largest eigenvalue of the adjacency-matrices of each bi-partite layer. It provides the maximum possible sparseness maintaining the connectivity from input to output.
- We design an eigen-bound based network pruning algorithm, which attempts to preserve Ramanujan graph property and is robust to noise.

Related Work In recent LTH studies, the victim weights are determined by their value at the initialization, gradient of the error, and the network topology. Firstly, to understand weight initialization, Malach et al. (2020) show that pruning makes a stronger hypothesis with bounded weight distribution. The sparsity of the network is reduced from polynomial to logarithmic factor of the number of training variables (Orseau et al., 2020). Mocanu et al. (2018) suggest to consider the topology of the network from a network science point of view. The pruning algorithm starts from a random *Erdős Rényi* graph and returns a scale-free network of a high sparsity factor based on the number of neurons in each layer. The method is further evolved for convolution layers considering both the magnitude and gradient of the weights(Evci et al., 2020a).

Lottery ticket finding algorithm is further improved for specific tasks or datasets like continuous learning setting (Chen et al., 2020b), on linear models (Elesedy et al., 2020), graph neural networks (Chen et al., 2021), transformers (Brix et al., 2020), object recognition (Girish et al., 2021), etc. Lee et al. (2020) propose an adaptive layer-wise magnitude based pruning technique. Lee et al. (2019) reason the winning ticket from signal propagation.

2 Network Pruning and the Lottery Ticket Hypothesis

The lottery ticket hypothesis states that a randomly initialized, dense neural network contains a subnetwork which when trained independently using the same initialization achieves a test accuracy close to the original network after training for less or at most the same number of iterations Frankle & Carbin (2019). These subnetworks, denoted as 'winning tickets', can be uncovered by network pruning algorithms. Weight pruning is one such simple strategy. Let the original dense network be represented as the function $\mathcal{N}(x;\theta)$, where x is the input and θ are the weights. The weights have an initialization of θ_0. Weight pruning generates a mask $m \in \{0,1\}^{|\theta|}$ such that the pruned network can be represented by $\mathcal{N}(x; m \odot \theta)$ with initialization $m \odot \theta_0$.

Pruning can be either one-shot or iterative. In one-shot pruning the original network is trained to convergence, then $p\%$ of weights are pruned and the surviving weights are reinitialized to their values in θ_0 followed retraining/fine tuning the subnetwork. Here, the network training and pruning are not simultaneously performed and pruning occurs only after convergence is reached. Iterative pruning repeats one-shot pruning over several iteration. This often leads to higher pruning percentage while retaining test set accuracy. However, iterative pruning is more time consuming than one-shot pruning. After pruning the surviving weights may alternately be initialized to their values in θ_i, i denoting a small iteration number, rather than their initial values in θ_0. This process, as illustrated in Figure 1 (Frankle et al., 2019b), is denoted as rewinding and is effective for large networks. We adopt this in our study.

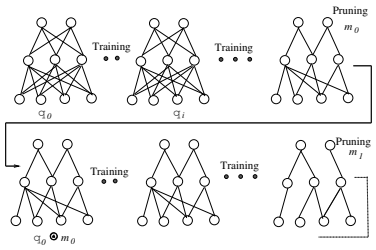


Figure 1: Iterative Network Pruning with Rewinding Frankle et al. (2019b)

Various scoring functions are used to prioritize the weights for pruning. They may be based on weight magnitudes, gradient, or information flow (Blalock et al., 2020; Hoefler et al., 2021). Magnitude pruning provides a simple method for obtaining the pruning mask by retaining the top $p\%$ weights $w_i \in \theta$ that have the highest values of $|w_i|$. For large networks magnitude pruning has been empirically found to perform as well or better than more complex pruning algorithms (Gale et al., 2019). Hence, we use magnitude pruning in this paper.

Magnitude pruning algorithms usually consider globally top weights. The role of the weights in local computation in the network layers is not considered. A higher pruning efficiency may be achieved by algorithms that account for connectivity structures in the individual layers. In the next section we describe graph parameters of the network that determine such connectivity.

3 Expanders and Ramanujan Graphs

Expanders are highly connected, and yet sparse graphs which have proved useful in a number of applications. In this work, we shall be considering finite, connected, undirected, but not necessarily regular graphs. Recall that the degree of a vertex v in a graph is the number of half edges emanating from v.

Definition 1 ((n, d, ε)-expander) *Let $\varepsilon > 0$. An (n, d, ε)-expander is a graph $\mathcal{G} = (V, E)$ on $|V| = n$ vertices, having maximal degree d, such that for every set $\emptyset \neq U \subseteq V$ satisfying $|U| \leq \frac{n}{2}$, $|\delta(U)| \geq \varepsilon|U|$ holds.*

Here, $\delta(U)$ denotes the vertex boundary of U, i.e., the set of vertices which are adjacent to U but not in U. The quantity $\frac{|\delta(U)|}{|U|}$ measures the rate of expansion and the infimum $\frac{|\delta(U)|}{|U|}$ as U varies among the non-empty subsets of V with $|U| \leq \frac{|V|}{2}$ is called the (vertex) Cheeger constant $h(\mathcal{G})$ of the graph $\mathcal{G}$. The higher the value of $h(\mathcal{G})$, the more expansion property it exhibits and vice versa. Expansion and the Cheeger constant quantifies the connectivity properties of a graph as a high value of $h(\mathcal{G})$ signifies that the graph is strongly connected.

Definition 2 (Expander family) *A sequence of finite, connected graphs $\{\mathcal{G}_i = (V_i, E_i)\}_{i=1,2,\cdots}$ on V_i vertices and E_i edges is called an expander family if there exists an uniform $\epsilon > 0$ such that each graph in the sequence is an $(|V_i|, d_i, \epsilon)$ expander for some d_i's.*

We are mainly interested in bounded degree expanders i.e., there exist $D > 0$ such that $d_i < D \ \forall i$.

The study of expansion properties of graphs is closely related to the study of the spectrum (distribution of eigenvalues) of the adjacency operator defined on them. Given a finite r-regular graph of size $|V| = n$, the eigenvalues t_i of the adjacency matrix are all real and they satisfy, $-r \leq t_n \leq \ldots \leq t_2 \leq t_1 = r$. The graph is connected iff $t_2 < t_1$ and is bipartite iff t_i's are symmetric about 0 (in particular $t_n = -r$). The quantity $t_1 - t_2$ is known as the *spectral gap* and it is related to the Cheeger constant via the discrete Cheeger-Buser inequality, discovered independently by Dodziuk (1984) and by Alon & Milman (1985). In our context, we need to consider a stronger notion of the spectral gap. Let $t := \max\{|t_i| : 1 < i \leq n, |t_i| < t_1\}$. For us, the quantity $t_1 - t$ will denote the spectral gap.

The more connected a graph is, the larger is the spectral gap and ideally, a graph with strong expansion properties has a very large spectral gap. However, for a bounded degree expander family, this spectral gap cannot be arbitrarily large. This brings us to the notion of Ramanujan graphs.

Definition 3 (Ramanujan graph) *Let $\mathcal{G}$ be a r-regular graph on n vertices, with adjacency eigenvalues $\{t_i\}_{i=1,2,\ldots n}$, satisfying $-r \leq t_n \leq \ldots \leq t_2 \leq t_1 = r$. Let $t(\mathcal{G}) := \max\{|t_i| : 1 < i \leq n, |t_i| < t_1\}$. Then $\mathcal{G}$ is a Ramanujan graph if $t(\mathcal{G}) \leq 2\sqrt{r-1} = 2\sqrt{t_1 - 1}$.*

The fact that in a bounded degree expander family, the eigenvalue bound in Ramanujan graphs is maximal can be deduced from the following result due to Alon, see Nilli (1991), $t(\mathcal{G}) \geq 2\sqrt{r-1} - \frac{2\sqrt{r-1}-1}{\lfloor m/2 \rfloor}$ where m denotes the diameter of the graph. As $m \to +\infty$ in a bounded degree expander family, we obtain the result (this also follows from the Alon-Bopanna theorem).

For our applications, we shall encounter not necessarily regular graphs, thus we need a notion of irregular version of Ramanujan graphs. The two ways we shall exploit Definition 3 for irregular graphs will be:

1. Replace the regularity r by t_1, the largest eigenvalue of the adjacency matrix.
2. Substitute the average degree d_{avg} in place of the regularity.

Note that a motivation for considering the above bounds comes from the following generalisation of the definition of Ramanujan graphs to irregular graphs. For a finite, connected graph $\mathcal{G}$ (not necessarily regular) consider its universal cover $\tilde{\mathcal{G}}$. A Ramanujan graph is a graph satisfying $t(\mathcal{G}) \leq \rho(\tilde{\mathcal{G}})$ where $\rho(\tilde{\mathcal{G}})$ denotes the spectral radius of $\tilde{\mathcal{G}}$. For a short discussion see Marcus–Spielman–Srivastava (Marcus et al., 2015, sec 2.3). A result of Hoory, see Hoory (2005) implies that $2\sqrt{d_{avg} - 1} \leq \rho(\tilde{\mathcal{G}})$. Thus, it makes sense to consider $t(\mathcal{G}) \leq 2\sqrt{d_{avg} - 1} \leq \rho(\tilde{\mathcal{G}})$. The above consideration also result in extremal families, Hoory (2005). Further, for irregular bipartite graphs, with minimal degree at least two and an average degree d_{avgL} on the left and d_{avgR} on the right, we can further consider the sharper estimate $t(\mathcal{G}) \leq \sqrt{d_{avgL} - 1} + \sqrt{d_{avgR} - 1} \leq \rho(\tilde{\mathcal{G}})$ see Hoory (2005).

Upto now, we had only spoken about unweighted graphs. A weighted graph is a graph with a weight function $w : E \to \mathbf{R}_{\geq 0}$ attached on the edges. Here $\mathbf{R}_{\geq 0}$ denotes the set of non-negative real numbers. A weighted graph can be looked upon as a generalisation of unweighted graphs in the sense that in the unweighted case, the function w takes values in the set $\{0, 1\}$. In this article, in the case of weighted graphs, we shall interpret the previous definitions naively, i.e., if there is an edge of weight $w(u, v)$ between u and v in the graph, then we interpret it as $w(u, v)$ edges going from u to v. This also means that in the case of weighted graphs, we need to modify the definition of the edge set of the graph to incorporate multiple (as well as fractional) edges. An unweighted graph will sometimes be called a simple graph.

4 Ramanujan Graph Characterization of Pruned Neural Networks

We can represent any network using graphs (possibly weighted, depending on the context) and in this article, we will be dealing with a sequence of finite, bipartite graphs. This is because if $\mathbf{N}$ denotes a neural network having l layers $\mathbf{N}_1, \mathbf{N}_1, \ldots, \mathbf{N}_l$ respectively, then each $\mathbf{N}_i$ $(i = 1, 2, \ldots, l)$ is a complete bipartite graph to start with. A pruned subnetwork results in edge sparsification of the underlying graphs. For that purpose, we need to approximate complete, bipartite graphs by sparse,

proper subgraphs. The motivation to study pruning based on expander characteristics stems from the fact that complete graphs can be approximated using expanders, see Spielman (2018). The notion of Ramanujan graphs allow us to quantify the pruning limit, and we empirically justify our technique in the following.

4.1 PRUNING ALGORITHM

In this section, we describe the proposed pruning algorithms which preserve the Ramanujan graph property of the neural network. The pruning approach considered here is iterative. The pruning algorithm is employed once the training is complete (i.e. after it converged to an optimum or it reached certain epochs). The pruned network is trained again with its initial weight values to perform the next pruning iteration. Also, we consider a simple magnitude based pruning technique, where the weights are made to zero if their absolute values are less than a certain percentile p. Now, based on the application of percentile, there are two common techniques namely, *layer-wise*, and *global*. Start with the neural network $\mathbf{N}$ having l layers $\mathbf{N}_1, \mathbf{N}_1, \ldots, \mathbf{N}_l$ respectively, with each $\mathbf{N}_i$ $(i = 1, 2, \ldots, l)$ representing a complete bipartite graph (ignoring weights). As the name suggests, in layer-wise pruning the percentile is applied to each layer $\mathbf{N}_i$ of the neural network, whereas, it is computed from the entire network $\mathbf{N}$, in case of global pruning. In both the cases, our proposed pruning method maintains the Ramanujan graph property by ensuring the eigen-bound for each layer of the neural network.

Algorithm 1: LAYER-WISE EIGEN-BOUND BASED PRUNING

Input: Trained Network $\mathbf{N}$, Pruning percentile p, Pruning bound type bt, pruning depth dep
Output: Pruned Network $\mathbf{N}^\star$

1 $\mathbf{N}^\star \leftarrow \mathbf{N}$
2 **for** $i \leftarrow 1$ ***to*** l **do**
3 $\quad k \leftarrow 1$
4 $\quad$ **while** $k \leq dep$ **do**
5 $\quad\quad \mathbf{N}_i^{temp} \leftarrow$ PRUNEBYPERCENTILE$(p/2^k, \mathbf{N}_i)$
6 $\quad\quad$ CALCULATE THE EIGENVALUES t_1 AND t_2 FROM $\mathbf{N}_i^{temp}$
7 $\quad\quad$ **if** $bt == eb$ **then**
8 $\quad\quad\quad bound \leftarrow 2 * \sqrt{t_1 - 1}$
9 $\quad\quad$ **else**
10 $\quad\quad\quad$ CALCULATE d_{avgL} AND d_{avgR}
11 $\quad\quad\quad bound \leftarrow \sqrt{d_{avgL} - 1} + \sqrt{d_{avgR} - 1}$
12 $\quad\quad$ **end**
13 $\quad\quad$ **if** $t_2 \leq bound$ **then**
14 $\quad\quad\quad \mathbf{N}_i^\star \leftarrow \mathbf{N}_i^{temp}$
15 $\quad\quad\quad$ **BREAK**
16 $\quad\quad$ **end**
17 $\quad\quad k \leftarrow k + 1$
18 $\quad$ **end**
19 **end**
20 **return** $\mathbf{N}^\star$

Eigen-Bound Based Pruning Algorithm 1 describes the *layer-wise eigen-bound pruning*, which takes the trained network $\mathbf{N}$, pruning percentile p, bound type bt, pruning depth dep as inputs and returns the pruned network $\mathbf{N}^\star$ as output. As discussed in Section 3, we mainly study two types of bounds on the second largest eigenvalue of the bipartite graph $t_2 = t(\mathcal{G})$, i.e., (i) **eb**- based on the largest eigenvalue(t_1) and (ii) **db**- based on the average degree (d_{avgL} and d_{avgR}). Here, $\mathcal{G}$ will be replaced by the structure or weights of a neural network layer $\mathbf{N}_i$. Now, after performing the pruning by percentile at each layer, the bound is computed from Line 7 to 12 (in algorithm 1) based on the bound type bt. If the pruned network $\mathbf{N}_i^{temp}$ follows the Ramanujan graph property given by Line 13 (in algorithm 1), then the algorithm simply proceeds to the next layer of pruning operation. If the the bound is not met by t_2, the pruning percentile is halved and search for a better approximation of $\mathbf{N}_i$ is resumed. The pruning depth $k \in [dep]$ acts as a controlling parameter to limit the search iteration.

Table 1: Different bound criteria on the second largest eigenvalue t_2

BI-ADJACENCY	EIGENVALUE (eb)	AVERAGE DEGREE (db)
Unweighted ($\mathbf{M}_i$)	UE: $t_2(\mathbf{M}_i) \leq 2\sqrt{t_1(\mathbf{M}_i) - 1}$	UD: $t_2(\mathbf{M}_i) \leq \sqrt{d_{avgL}(\mathbf{M}_i) - 1} + \sqrt{d_{avgR}(\mathbf{M}_i) - 1}$
Weighted ($\mathbf{W}_i$)	WE: $t_2(\mathbf{W}_i) \leq 2\sqrt{t_1(\mathbf{W}_i) - 1}$	WD: $t_2(\mathbf{W}_i) \leq \sqrt{d_{avgL}(\mathbf{W}_i) - 1} + \sqrt{d_{avgR}(\mathbf{W}_i) - 1}$

This algorithm is called at each pruning iteration, and continued till the stopping criterion is met, i.e., $\mathbf{N}_i^\star = \mathbf{N}_i, \forall i$. This process ensures the connectivity property as established by Ramanujan graph characteristics. Though it may increase the density of the network, it gives a robust lottery ticket at the end. We will show this in the experimentation. The top-2 largest magnitude eigenvalues at layer i are computed from the symmetric matrix $\mathbf{N}_i^T\mathbf{N}_i$ with implicitly restarted Arnoldi methods (Lehoucq et al., 1998) and, the average degree is computed from the connected component of the bipartite-graph. Hence, the computational complexity is mainly driven by the pruning depth dep, the number of input and output neurons at each layer, and the number of layers. The additional space will be required to store the bipartite graph of a layer.

Compared to layer-wise one, in *global eigen-bound based pruning*, the depth of pruning, dep, is not considered and the whole network is pruned using a global pruning percentile p. Based on the bound type bt, the eigen-bound is computed at each layer. If the bound is not met by the second largest eigenvalue, t_2, of a particular layer i, the weights are simply reset to the values before pruning.

4.2 Bipartite Graph Structure

In this work, we focus on the feed-forward neural network only. There exist several configurations such as, fully-connected layers, convolution layer, dropout layer, pooling layer etc, which create variety of feed-forward networks. In each case, it is interesting to know what is the structure of its corresponding bipartite graph. Since, pruning is a part of network compression, we only consider the trainable layers here.

- **Fully Connected Layers (FC)**: For a fully connected layer $\mathbf{N}_i$, with n^{i-1} number of inputs and n^i number of outputs the weighted bi-adjacency matrix of the corresponding graph is $\mathbf{W}_i \in \mathbb{R}^{[n^{i-1} \times n^i]}$, and the corresponding pruning mask is $\mathbf{M}_i \in \{0,1\}^{[n^{i-1} \times n^i]}$.
- **Convolution Layers (Conv)**: Here, we consider the kernel size, the number of input and output channels to unfold the layer into a complete bipartite graph. For a convolution layer $\mathbf{N}_i$ with the kernel of size k, n^{i-1} input, and n^i output channels, the weighted bi-adjacency matrix of the corresponding graph will be $\mathbf{W}_i \in \mathbb{R}^{[(n^{i-1}.k.k) \times n^i]}$, and the corresponding pruning mask is $\mathbf{M}_i \in \{0,1\}^{[(n^{i-1}.k.k) \times n^i]}$. An example is shown in Figure 2.

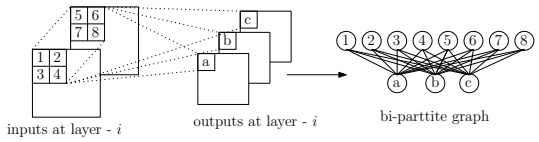


Figure 2: An example of bipartite graph computation for a particular convolution layer with kernel size 2×2, 2 input channels, and 3 output channels

Now, the bounds are analyzed for both unweighted ($\mathbf{M_i}$) and weighted ($\mathbf{W}_i$) bipartite graphs at each layer $\mathbf{N}_i$. Table 1 describes different bounding constraint depending on the type of the considered graph and bound type.

5 Experimental Results

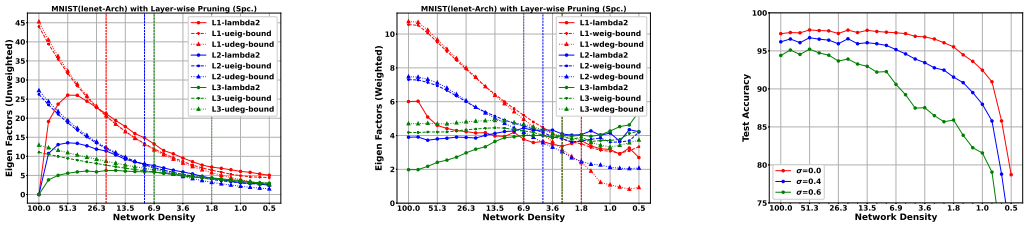
In this section, we discuss the experimental results. We have used the datasets, MNIST and CIFAR10. As an evaluation measure we use the classification accuracy for both clean and noisy test

sets. We report results for the Lenet, Conv2, Conv4 network architectures following the methodology adopted in Frankle & Carbin (2019). We also perform the studies on other networks which are presented in Appendix.

5.1 THE EXPERIMENTAL SETUP

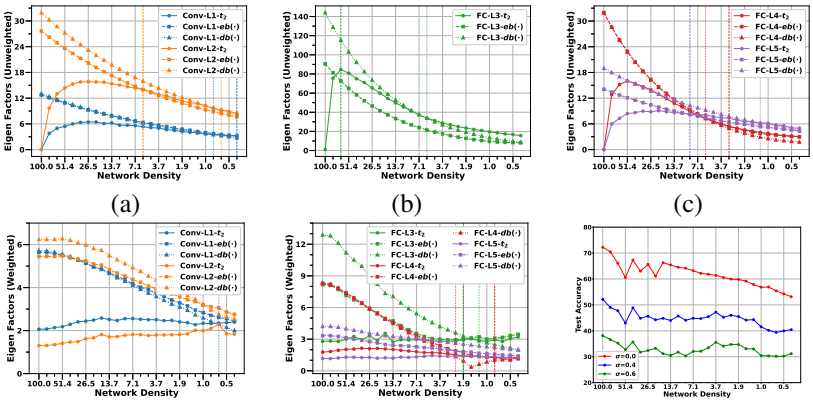
We study the variation of classification accuracy with the network density as measured by the pruning percentile p. For different layers the percentile can be different, as denoted by p_{fc} for FC layers, p_{conv} for convolution layers, and p_{out} for the output layers. For each of the studies, we plot four parameters; (i) eigen bound (eb), (ii) degree bound (db), (iii) network density for each layer, and (iv) test accuracy on both clean and noisy data. We consider Gaussian noise in the training images. We study the two cases for the bound specific to unweighted ($\mathbf{M}_i$) and weighted graph ($\mathbf{W}_i$).

Relationship between Ramanujan graph properties and LTH The primary goal of our study to identify the LTH regimes, determined by the spectral properties of the bipartite-network. This motivates the proposed *eigen-bound based pruning* algorithm. The existence of LTH regime is illustrated for three representative established networks. The results for Lenet on MNIST dataset is shown in Figure 3; for Conv2 on CIFAR10 dataset is shown 4; and for Conv4 is shown in 5. In each of the plots, we show three curves representing the second largest eigenvalue t_2, eigen-bound and degree-bound. We also highlight the boundaries after which the layer does not maintain Ramanujan graph property. Similar results for other networks and different pruning settings are discussed in Appendix.



(a) (b) (c)

Figure 3: The results for Lenet using MNIST dataset with pruning percentile $p_{fc} = 0.2$ and $p_{out} = 0.1$ (layer-wise-pruning); (a) eigen factors for the unweighted graph, (b) eigen factors for the weighted graph, (c) test accuracy



(a) (b) (c)

(d) (e) (f)

Figure 4: The results for Conv2 architecture using CIFAR10 dataset with pruning percentile $p_{conv} = 0.1$, $p_{fc} = 0.2$, and $p_{out} = 0.1$ (layer-wise-pruning); (a-c) eigen factors for the unweighted graph, (d-e) eigen factors for the weighted graph, (f) test accuracy

In all the three figures, we observe that the accuracy on clean data falls sharply when the layers do not satisfy the bounds on t_2 for the weighted graphs. The bounds for the unweighted graph allow

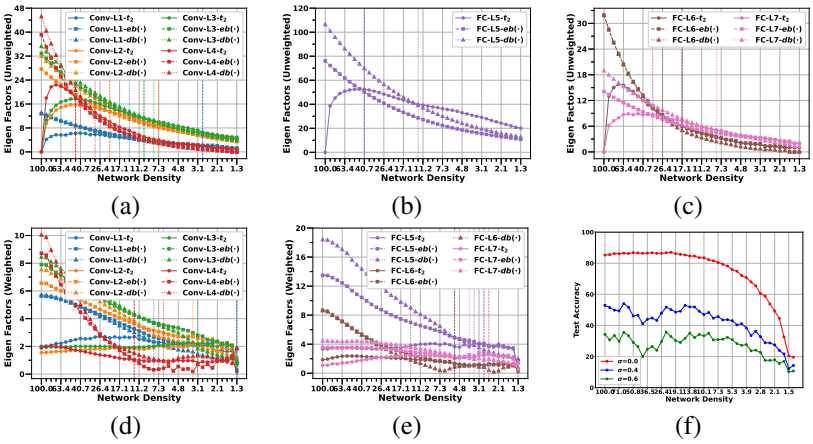


Figure 5: The results for Conv4 architecture using CIFAR10 dataset with pruning percentile $p_{conv} = 0.1$, $p_{fc} = 0.2$, and $p_{out} = 0.1$ (layer-wise-pruning); (a-c) eigen factors for the unweighted graph, (d-e) eigen factors for the weighted graph, (f) test accuracy

only the robust winning tickets, having high accuracy even on noisy data. For Lenet, we see that accuracy drops for the network density varying from $6 \sim 2\%$, which supports the boundaries on the curve for weighted graph (refer Figure 3b). However, the boundaries for the unweighted graphs are demarcated for more dense networks (around 26%), where the accuracy falls sharply while adding noise (refer Figure 3a, c). The same boundaries are observed for Conv2 at around $7 \sim 13\%$ for the unweighted graph in Figure 4, and for conv4 around 40% in Figure 5.

Efficacy of the proposed pruning algorithm The performance of the pruning algorithm is discussed considering the bound types mentioned in Table 1. Here, we consider the same pruning percentile for all the layers. We present the results for Lenet (Figure 6 and 7), Conv2 (Figure 8).

We observe that for the same density of the pruned network, the proposed pruning algorithm provides a higher accuracy than simple magnitude based pruning, for all the datasets. Among the eigen factors as bound used by the propesd pruning algorithm, eb on unweighted graph return the maximum density in the wining ticket and minimum dispersion in the accuracy on the noisy dataset (refer Figure 7a). Compared to the bound on the unweighted graph, the weighted graph versions can lead to more sparse network. We also show the results for **UD** setting for Conv2 architecture in Figure 8.

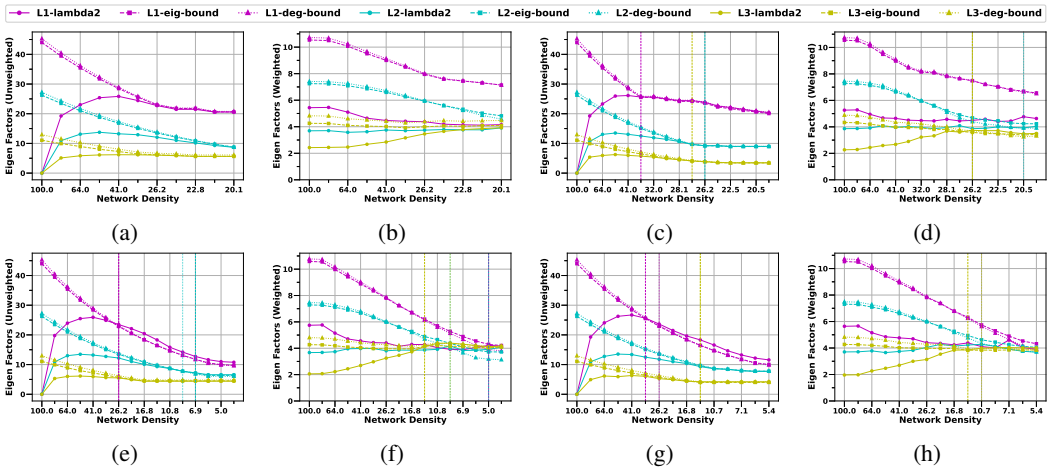
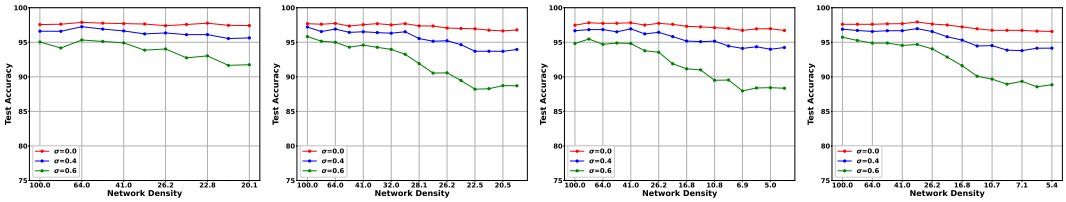
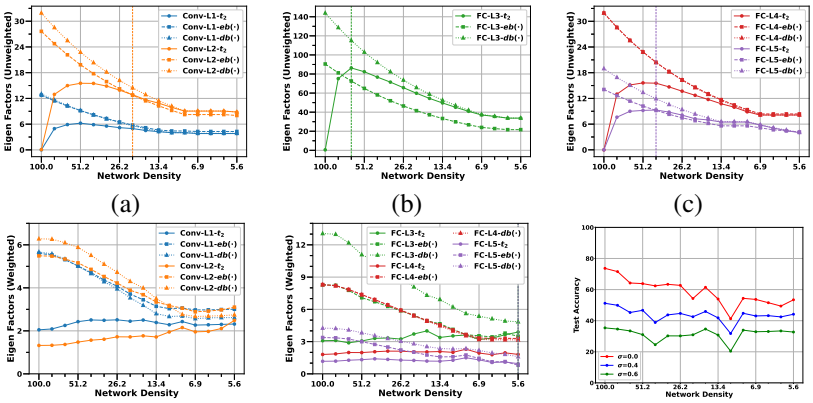


Figure 6: The eigen-factor results for the proposed pruning algorithm on MNIST dataset using Lenet architecture and different types of bound criteria, mentioned in Table 1; (a-b) for **UE**, (c-d) for **UD**, (e-f) for **WE**, (g-h) for **WD**



(a) (b) (c) (d)

Figure 7: The test accuracy results of the proposed pruning algorithm on MNIST dataset using Lenet architecture with different noise levels in the input image using and bound criteria, mentioned in Table 1; (a) for **UE**, (b) for **UD**, (c) for **WE**, (d) for **WD**



(a) (b) (c)

(d) (e)

Figure 8: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv2 architecture using different types of bound criteria **UD** on the second largest eigenvalue, t_2

6 Conclusion

In this work, we attempt to justify the validity of the LTH based on spectral characteristics of the network under consideration. The lottery ticket hypothesis ensures that the pruned sub-networks can be easily refined to achieve a high accuracy and we empirically demonstrate this using MNIST and CIFAR10 datasets on several feed-forward architectures. The pruned network has the property of being simultaneously sparse and yet highly connected, which is a characterization of expander graphs in general. As a consequence, we considered robust expanders, i.e., Ramanujan graphs and motivated to us propose a new pruning technique following the spectral bounds of the individual layers of the network. Further refinement of this approach of pruning can result in more efficient winning ticket search, which will be the basis of future research.

References

N. Alon and V. D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *J. Combin. Theory Ser. B*, 38(1):73–88, 1985. ISSN 0095-8956. doi: 10.1016/0095-8956(85)90092-9. URL https://doi.org/10.1016/0095-8956(85)90092-9.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

Christopher Brix, Parnia Bahar, and Hermann Ney. Successfully applying the stabilized lottery ticket hypothesis to the transformer architecture. *arXiv preprint arXiv:2005.03454*, 2020.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*, 2020a.

Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations*, 2020b.

Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning*, pp. 1695–1706. PMLR, 2021.

Jozef Dodziuk. Difference equations, isoperimetric inequality and transience of certain random walks. *Trans. Amer. Math. Soc.*, 284(2):787–794, 1984. ISSN 0002-9947. doi: 10.2307/1999107. URL `https://doi.org/10.2307/1999107`.

Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *arXiv preprint arXiv:1705.07565*, 2017.

Bryn Elesedy, Varun Kanade, and Yee Whye Teh. Lottery tickets in linear models: An analysis of iterative magnitude pruning. *arXiv preprint arXiv:2007.08243*, 2020.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020a.

Utku Evci, Yani A Ioannou, Cem Keskin, and Yann Dauphin. Gradient flow in sparse neural networks and how lottery tickets win. *arXiv preprint arXiv:2010.03533*, 2020b.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019a.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. The lottery ticket hypothesis at scale. *CoRR*, abs/1903.01611, 2019b.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019.

Sharath Girish, Shishira R Maiya, Kamal Gupta, Hao Chen, Larry S Davis, and Abhinav Shrivastava. The lottery ticket hypothesis for object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 762–771, 2021.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28:1135–1143, 2015.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, 2021.

Shlomo Hoory. A lower bound on the spectral radius of the universal cover of a graph. *J. Combin. Theory Ser. B*, 93(1):33–43, 2005. ISSN 0095-8956. doi: 10.1016/j.jctb.2004.06.001. URL `https://doi.org/10.1016/j.jctb.2004.06.001`.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. Layer-adaptive sparsity for the magnitude-based pruning. In *International Conference on Learning Representations*, 2020.

Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip HS Torr. A signal propagation perspective for pruning neural networks at initialization. *arXiv preprint arXiv:1906.06307*, 2019.

Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

Alex Lubotzky. *Discrete groups, expanding graphs and invariant measures*. Springer Science & Business Media, 2010.

Eran Malach, Gilad Yehudai, Shai Shalev-Shwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. *arXiv preprint arXiv:2002.00585*, 2020.

Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: Bipartite Ramanujan graphs of all degrees. *Ann. of Math. (2)*, 182(1):307–325, 2015. ISSN 0003-486X. doi: 10.4007/annals.2015.182.1.7. URL https://doi.org/10.4007/annals.2015.182.1.7.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

A. Nilli. On the second eigenvalue of a graph. *Discrete Math.*, 91(2):207–210, 1991. ISSN 0012-365X. doi: 10.1016/0012-365X(91)90112-F. URL https://doi.org/10.1016/0012-365X(91)90112-F.

Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33, 2020.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

Daniel A. Spielman. Properties of expander graphs. http://www.cs.yale.edu/homes/spielman/561/lect17-18.pdf, 2018.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A Appendix

A.1 Global Eigen-Bound Based Pruning

Algorithm 2 describes the *global eigen-bound based pruning*, which takes the trained network $\mathbf{N}$, pruning percentile p, bound type bt as inputs and return the pruned network $\mathbf{N}^\star$ as output. Compared to layer-wise one, here pruning depth is not considered and the whole network is pruned prior to eigen-bound establishments by each layer. Based on the bound type bt, if the bound is not met by t_2 of a layer $\mathbf{N}_i$, it is simply made to the weights before pruning.

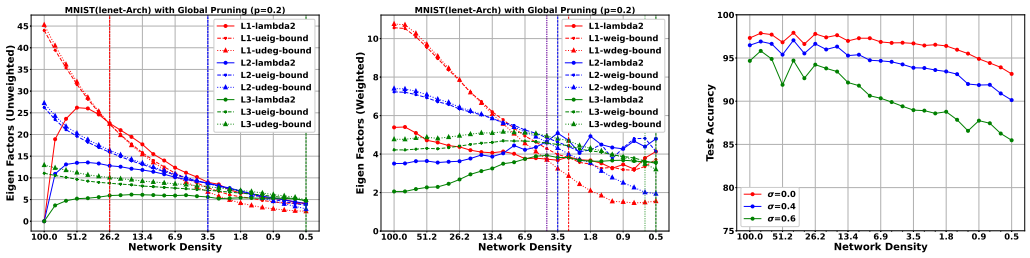
Algorithm 2: GLOBAL EIGEN-BOUND PRUNING

Input: Trained Network $\mathbf{N}$, Pruning percentile p, Pruning bound type bt
Output: Pruned Network $\mathbf{N}^\star$

$\mathbf{N}^\star \leftarrow \mathbf{N}$
$\mathbf{N}^{temp} \leftarrow \text{PRUNEBYPERCENTILE}(p, \mathbf{N})$
for $i \leftarrow 1$ ***to*** l **do**
  CALCULATE THE EIGENVALUES t_1 AND t_2 FROM $\mathbf{N}_i^{temp}$
  if $bt == eb$ **then**
    $bound \leftarrow 2 * \sqrt{t_1 - 1}$
  else
    CALCULATE d_{avgL} AND d_{avgR}
    $bound \leftarrow \sqrt{d_{avgL} - 1} + \sqrt{d_{avgR} - 1}$
  end
  if $t_2 \leq bound$ **then**
    $\mathbf{N}_i^\star \leftarrow \mathbf{N}_i^{temp}$
  end
end
return $\mathbf{N}^\star$

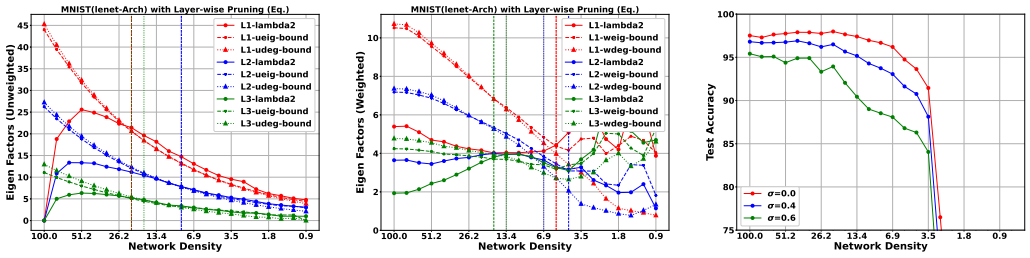
We use the code from `https://github.com/facebookresearch/open_lth` for our experiments. Also, all the hyper parameter settings to train the networks are followed from Frankle & Carbin (2019).

A.2 MORE RESULTS FOR LENET ARCHITECTURE ON MNIST DATASET



(a) (b) (c)

Figure 9: The eigen factors for global pruning with pruning percentile $p = 0.2$ on MNIST dataset using Lenet architecture



(a) (b) (c)

Figure 10: The eigen factors for layer-wise pruning with pruning percentile $p = 0.2$ on MNIST dataset using Lenet architecture

A.3 MORE RESULTS FOR CONV2 ARCHITECTURE ON CIFAR10 DATASET

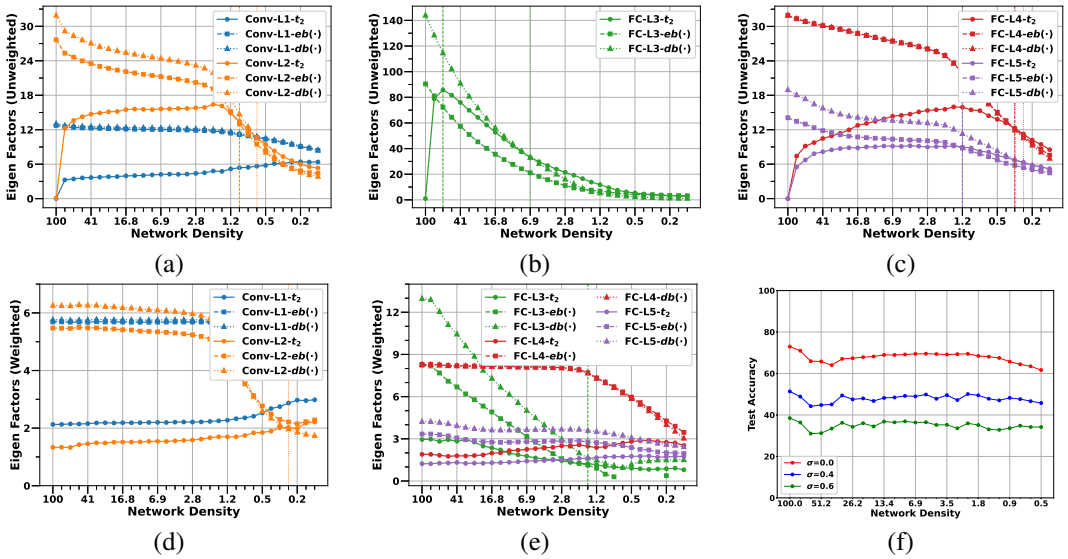


Figure 11: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv2 architecture; (a-e) Global Pruning with $p = 0.2$

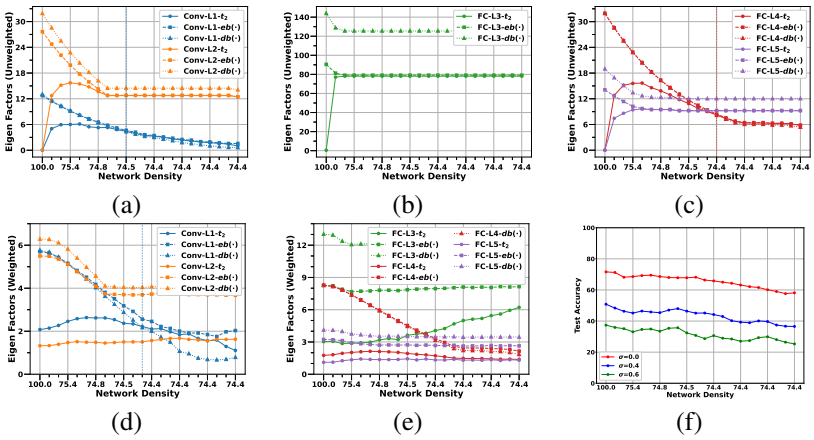


Figure 12: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv2 architecture using different types of bound criteria **UE** on the second largest eigenvalue, t_2

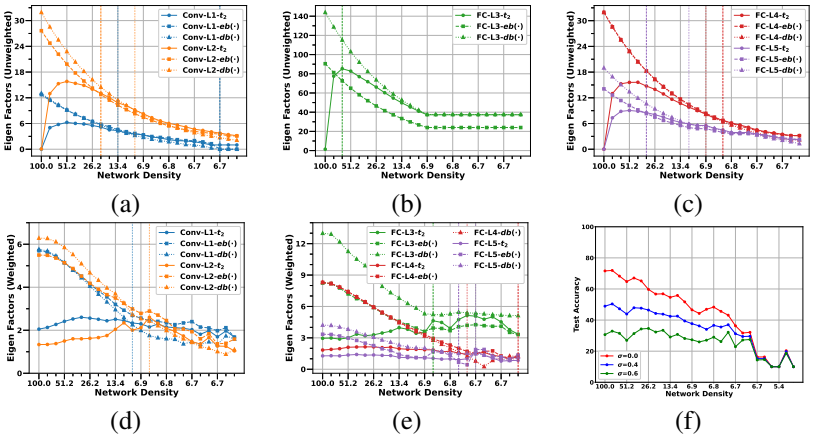


Figure 13: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv2 architecture using different types of bound criteria **WE** on the second largest eigenvalue, t_2

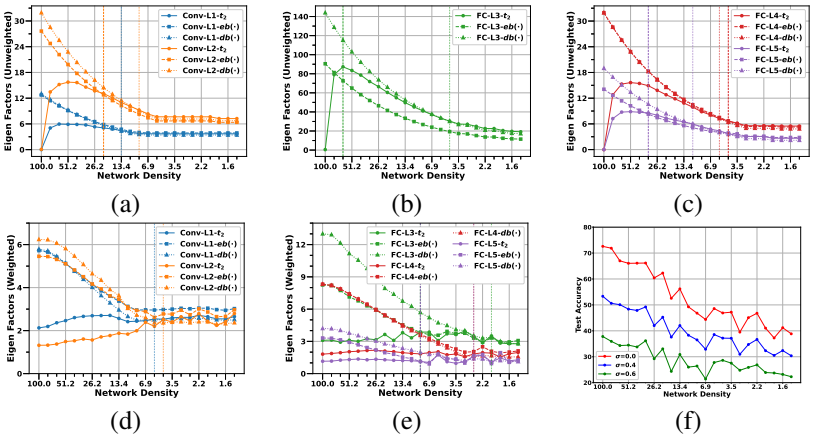


Figure 14: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv2 architecture using different types of bound criteria **WD** on the second largest eigenvalue, t_2

A.4 MORE RESULTS FOR CONV4 ARCHITECTURE ON CIFAR10 DATASET

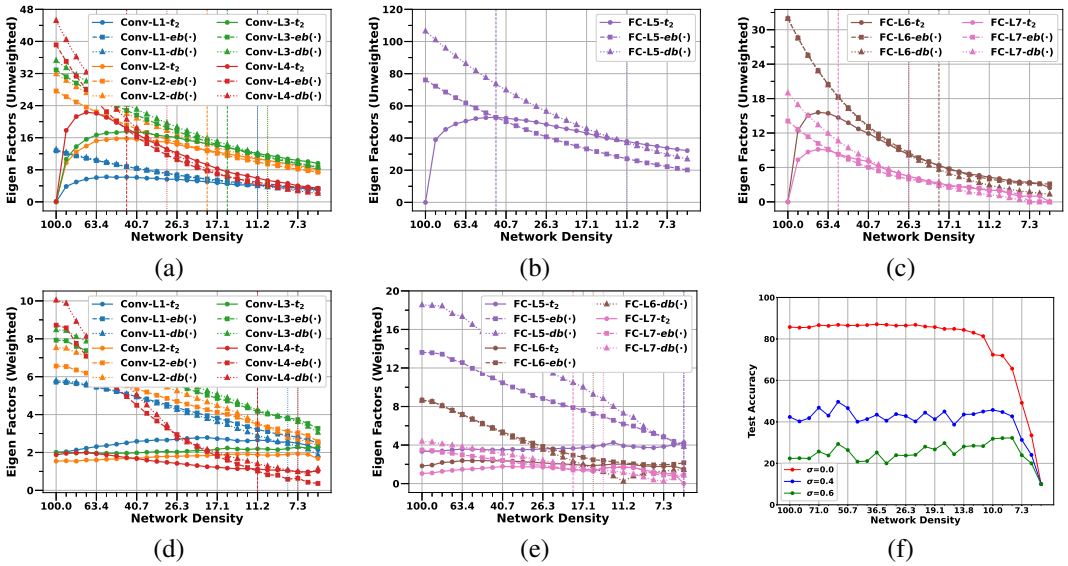


Figure 15: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv4 architecture; (a-e) Layer-wise Pruning with $p_{conv} = 0.1$, $p_{fc} = 0.2$, and $p_{out} = 0.2$

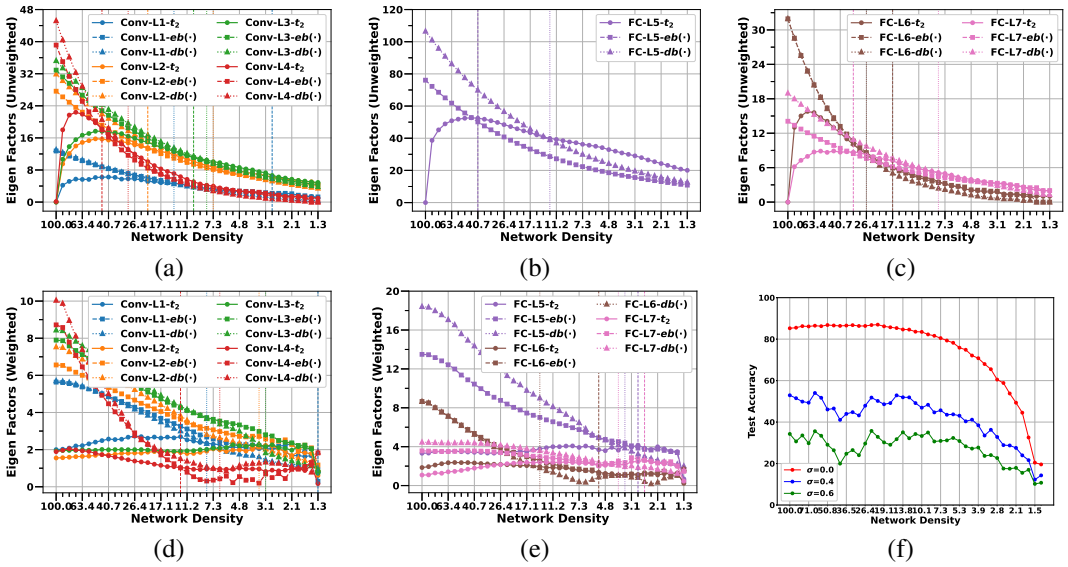


Figure 16: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv4 architecture; (a-e) Layer-wise Pruning with $p_{conv} = 0.1$, $p_{fc} = 0.2$, and $p_{out} = 0.1$

A.5 MORE RESULTS FOR CONV6 ARCHITECTURE ON CIFAR10 DATASET

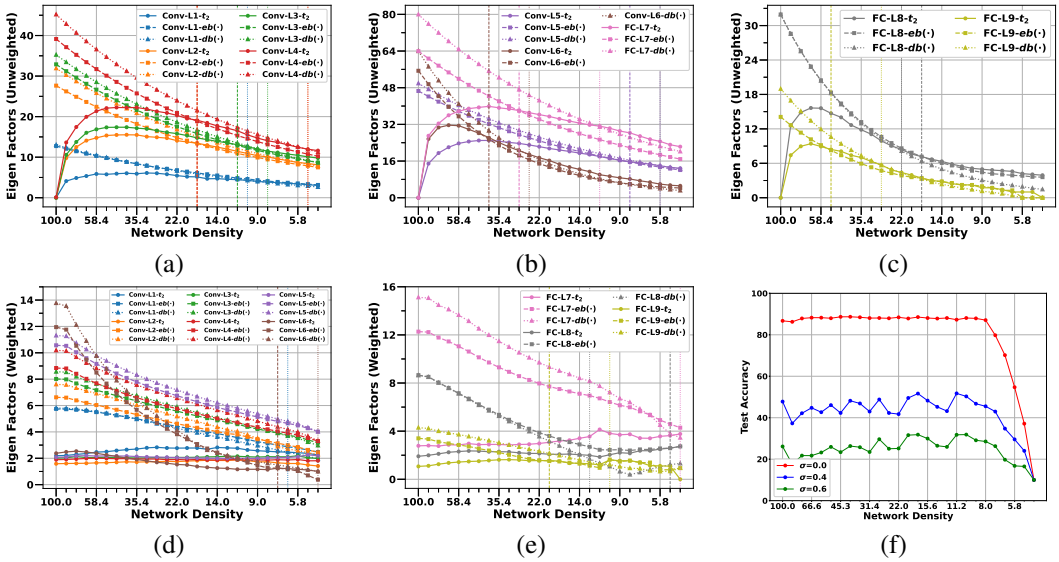


Figure 17: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv6 architecture; (a-e) Layer-wise Pruning with $p_{conv} = 0.1$, $p_{fc} = 0.2$, and $p_{out} = 0.2$

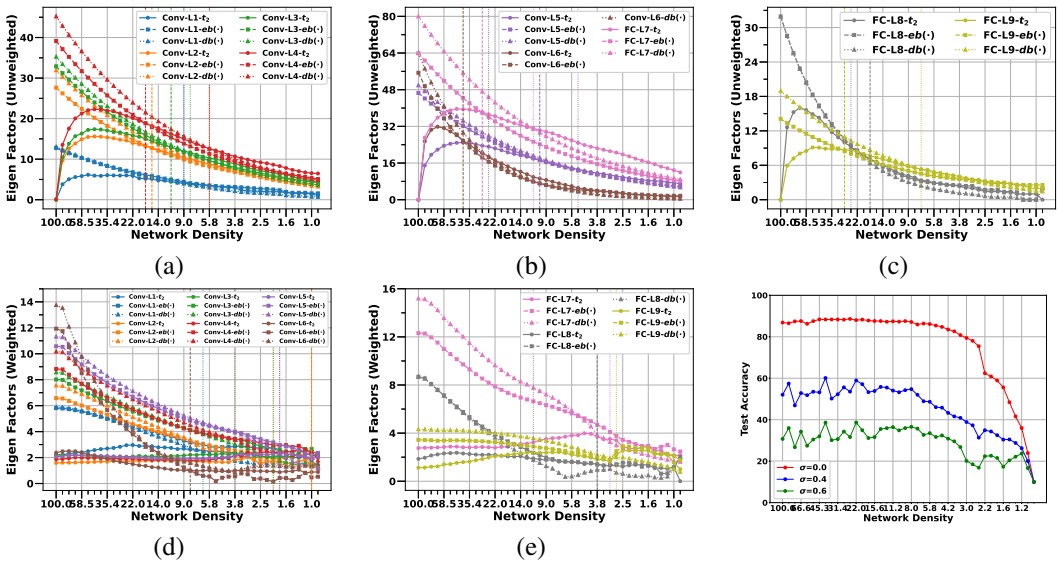


Figure 18: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using Conv6 architecture; (a-e) Layer-wise Pruning with $p_{conv} = 0.1$, $p_{fc} = 0.2$, and $p_{out} = 0.1$

A.6 RESULTS FOR VGG19

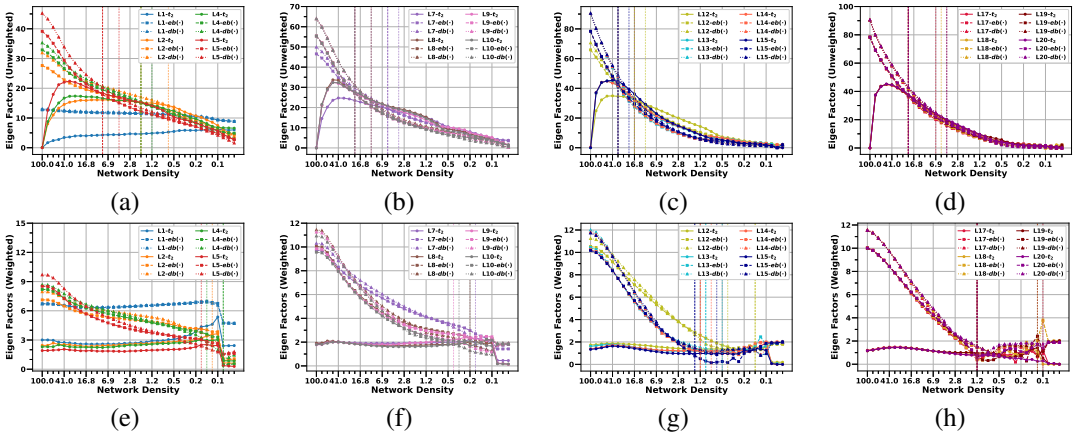


Figure 19: The eigen factors for the proposed pruning algorithm on CIFAR10 dataset using VGG19 architecture and Global Pruning with $p = 0.2$ and $lr = 0.01$

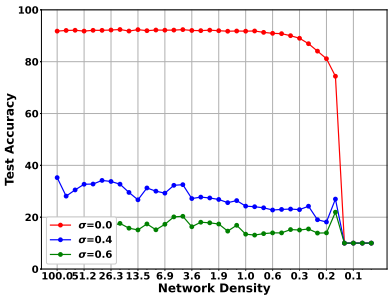


Figure 20: Results for the test accuracy of VGG19 on different noise level