Power-law in Sparsified Deep Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

The power law has been observed in the degree distributions of many biological neural networks. Sparse deep neural networks resemble biological neural networks in many ways. In this paper, we study if these artificial networks also exhibit properties of the power law. Experimental results on two popular deep learning models, namely, multilayer perceptrons and convolutional neural networks, are affirmative. The power law is also naturally related to preferential attachment. To study the dynamical properties of deep networks in continual learning, we propose an internal preferential attachment model to explain how the network topology evolves. Experimental results in continual learning show that with the arrival of a new task, the new connections made follow this preferential attachment process. In biological neural networks with limited capacities, reuse of neural circuits is essential for learning multiple tasks, and scale-free networks have been found to learn faster than random and small-world networks. We showed empirically that making new connections in a deep neural network using preferential attachment leads to faster learning and smaller networks in the continual learning setting.

1 Introduction

In recent years, deep neural networks have achieved state-of-the-art performance in various tasks such as speech recognition, visual object recognition, and image classification [18]. By pruning unimportant connections and then retraining the remaining ones, they can often be significantly sparsified without performance degradation [11, 20]. This sparsification process is also analogous to how learning works in the mammalian brain [13]. In particular, pruning (resp. retraining) in deep neural networks resembles weakening (resp. strengthening) of functional connections in brains. Another similarity between deep and biological neural networks can be seen in the context of continual learning, in which learning proceeds progressively with the arrival of new tasks [8, 16, 25]. Continual learning by deep networks mimics the biological learning process in which new connections are made without loss of established functionalities in neural circuits [8]. In both biological and deep neural networks, sparsity allows a more economical representation of the learning experience to be obtained.

Biological neural networks are often scale-free, i.e., their degree distributions follow the power law. Specifically, the probability density function (PDF) of a power law distribution [3] is of the form:

$$f(x) \propto x^{-\alpha}, \tag{1}$$

where x is the measurement, and $\alpha > 1$ is the exponent. For example, Monteiro *et al.*[22] showed that the mean learning curves of scale-free networks resemble that of the biological neural network of the worm *Caenorhabditis Elegans*. They are also better than those generated from random and small-world networks. Eguiluz *et al.*[7] studied the functional networks connecting correlated human brain sites, and showed that the distribution of functional connections also follows the power law.

In this paper, we study if the sparsified deep neural networks also exhibit properties of the power law as observed in their biological counterparts (Section 3). It is well-known that the power law is related

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

to the dynamics of an evolving network via preferential attachment [3], in which new connections are preferentially made to the more highly connected nodes. We propose a preferential attachment model for deep neural networks, and verify that new connections added to the network in a continual learning setting follow this model (Section 4). Finally, we show how this can help efficient evolution of deep neural networks in continual learning (Section 5). Specifically, establishing new connections using preferential attachment can lead to smaller networks and faster learning.

2 Related Work: Power Law and Its Truncated Variant

While the power law distribution has been commonly used to describe the underlying mechanisms of a wide variety of physical, biological and man-made networks [3], few empirical phenomena obey it exactly for the whole range of observations [6]. Typically, the power law applies only for values greater than some minimum [6]. There is often also a maximum value above which the power law is no longer valid [5]. Such networks are sometimes called broad-scale or truncated scale-free networks [1], and have been observed for example in the human brain anatomical network [15]. Such an upper truncation may result from the finite size of data, temporal limitations of the collected data, or constraints imposed by the underlying physics [5]. For example, the size of forest fires is naturally limited by availability of fuel and climate [21].

To model this upper truncation effect, extensions of the power law have been proposed [5, 17]. In this paper, we will focus on the truncated power law (TPL) distribution [5, 17], which explicitly includes a lower threshold $x_{\min}$ and an upper threshold $x_{\max}$. Its PDF and complementary cumulative distribution function (CCDF)[1] are given by:

$$p(x) = \frac{1-\alpha}{x_{\max}^{1-\alpha} - x_{\min}^{1-\alpha}} x^{-\alpha}, \quad S(x) = \frac{x^{1-\alpha} - x_{\max}^{1-\alpha}}{x_{\min}^{1-\alpha} - x_{\max}^{1-\alpha}}. \tag{2}$$

It is well-known that the log-log CCDF plot for the standard power law distribution is a line. However, from (2), the log-log CCDF for a TPL is

$$\log(S(x)) = \log(x^{1-\alpha} - x_{\max}^{1-\alpha}) - \log(x_{\min}^{1-\alpha} - x_{\max}^{1-\alpha}). \tag{3}$$

As $\lim_{x \to x_{\max}} \log(S(x)) = -\infty$, the log-log CCDF plot for TPL has a fall-off near $x_{\max}$. When $x_{\max}$ is large, $\log(S(x)) \simeq (1-\alpha)\log(x) - (1-\alpha)\log(x_{\min})$, and the log-log CCDF plot reduces to a line. When $x_{\max}$ gets smaller, the linear region shrinks and the fall-off starts earlier.

When x takes only integer (instead of continuous) values, the probability function of the TPL distribution becomes [14]

$$f(x) = \frac{x^{-\alpha}}{\zeta(\alpha, x_{\min}, x_{\max})}, \tag{4}$$

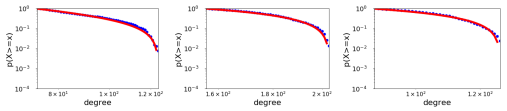
where $\zeta(\alpha, x_{\min}, x_{\max}) = \sum_{k=x_{\min}}^{x_{\max}} k^{-\alpha}$. The corresponding CCDF is: $S(x) = \frac{\zeta(\alpha, x, x_{\max})}{\zeta(\alpha, x_{\min}, x_{\max})}$.

Power law distributions can originate from the process of preferential attachment [3], which can be either external or internal [4]. External preferential attachment refers to that a new node is more likely to connect to an existing node with high degree; while internal preferential attachment means that existing nodes with high degrees are more likely to connect to each other. Let N be the number of nodes in the network, and a be the number of new internal connections created in unit time per existing node. The total number of internal connections all nodes created per unit time is $2Na$. For two nodes with degrees d_1 and d_2, the expected number of new connections created between them per unit time via internal preferential attachment is [4]:

$$\Delta(d_1, d_2) = 2Na \frac{d_1 d_2}{\sum_{s,m \neq s} d_s d_m}, \tag{5}$$

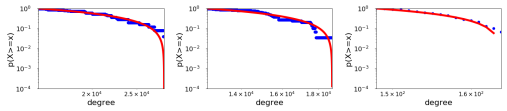
where s and m are indices to all nodes in the network.

[1] The CCDF is defined as one minus the cumulative distribution function, i.e., $1 - \int_{x_{\min}}^{x} f(x)\, dx$.



(a) input. (b) fc1. (c) fc2.

Figure 1: Log-log CCDF and TPL fit (red) for MLP on MNIST. “fc1" and “fc2" denote the first and second FC layer, respectively.



(a) conv1. (b) conv2. (c) fc3.

Figure 2: Log-log CCDF and TPL fit (red) for CNN on MNIST. Here, “conv1" and “conv2" denote the first and second convolutional layers.

3 Power Law in Deep Neural Networks

In this section, we study whether the degree distributions in artificial neural networks follow the power law. Network connectivities are often fixed by design, redundant, and not learned from data. Hence, we will study networks that have been sparsified, in which only important connections are kept. Specifically, we use sparse networks produced by the state-of-the-art three-step network pruning method in [13], and also pre-trained sparse convolutional neural networks. The three-step network pruning method involves: (i) train a dense network; (ii) for each layer, a fraction of $s \in (0, 1)$ connections with the smallest magnitudes are pruned; (iii) the unpruned connections are then retrained. To avoid potential performance degradation, connections to the output layer are always left unpruned, as is common in network pruning [20] and continual learning [8].

Obviously, neural networks are of finite size and the connections each node can make is limited. This suggests upper truncation and the TPL is more appropriate for modeling connectivity. In particular, we will use the discrete TPL in (4) as the degree is integer-valued. Moreover, as different layers can differ greatly in the number of nodes, the maximum number of connections each node can make, and the features extracted, it is thus more appropriate to study connectivity in a layer-wise manner. Given the degree distribution of the deep network nodes, the TPL parameters are fitted with maximum likelihood and the Kolmogorov-Smirnov statistic as in [6]. Details can be found in Appendix A.

In the following, we examine the degree distributions of two popular deep learning models: multilayer perceptrons (Section 3.1) and convolutional neural networks (Section 3.2).

3.1 Multilayer Perceptron (MLP)

We first perform experiments on the MNIST data set, which contains 28×28 grayscale images from 10 digit classes. $50,000$ images are used for training, another $10,000$ for validation, and the remaining $10,000$ for testing. Following [13], we first train a dense MLP (with two hidden layers)

$$input - 1024FC - 1024FC - 10softmax$$

using the cross-entropy loss in the Lasagne package[2]. Here, $1024FC$ denotes a fully-connected layer with 1024 units and $10softmax$ is a softmax layer for the 10 classes. The optimizer is stochastic gradient descent (SGD) with momentum. All hyperparameters are the same as in the Lasagne package. The maximum number of epochs is 200. After training, a fraction of $s = 0.9$ connections are pruned in each layer. The testing accuracies of the dense and (retrained) pruned networks are comparable (98.09% and 98.21%).

For each node, we obtain its degree by counting the total number of connections (after pruning) to nodes in its upper and lower layers.[3] Figure 1 shows the CCDF plot and TPL fit for each layer. We do not show the softmax layer, which is not pruned. As can be seen, the TPL fits the degree distributions well.

3.2 Convolutional Neural Network (CNN)

Next, we experiment with the CNN. We consider each feature map in the convolutional layer as a node. As in Section 3.1, the node degree is obtained by counting the total number of connections (after pruning) to its upper and lower layers. A detailed example is shown in Appendix B. As will be seen from Figures 2-5, the TPL fits the degree distributions well.

[2] `https://github.com/Lasagne/Lasagne/blob/master/examples/mnist.py`

[3] For a node in the input (resp. last pruned) layer, we only count its connections to nodes in the upper (resp. lower) layer.

CNN on MNIST First, we experiment with the CNN in the Lasagne package on MNIST. It is similar to LeNet-5 [19], and has two convolutional layers followed by two fully-connected layers:

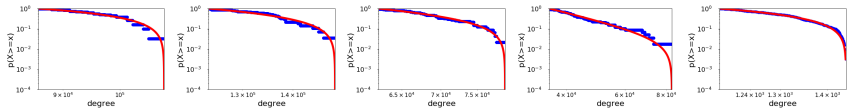
$$\textit{input} - 32C5 - \textit{MP}2 - 32C5 - \textit{MP}2 - 256\textit{FC} - 10\textit{softmax}.$$

Here, $32C5$ denotes a ReLU convolution layer with 32 5×5 filters, and $\textit{MP}2$ is a 2×2 max-pooling layer. We use SGD with momentum as the optimizer. The other hyperparameters are the same as in the Lasagne package. The maximum number of epochs is 200. The (dense) CNN has a testing accuracy of 98.85%. This is then pruned using the method in [13], with $s = 0.7$. After retraining, the sparse CNN has a testing accuracy of 98.78%, and is comparable with the dense network. Figure 2 shows the CCDF plot and TPL fit for each CNN layer.

CNN on CIFAR-10 Next, we perform experiments on the CIFAR-10 data set, which contains 32×32 color images from ten object classes. We use $45,000$ images for training, another $5,000$ for validation, and the remaining $10,000$ for testing. The following CNN from [28] is used[4]:

$$\textit{input} - 32C3 - 32C3 - \textit{MP}2 - 64C3 - 64C3 - \textit{MP}2 - 512\textit{FC} - 10\textit{softmax}.$$

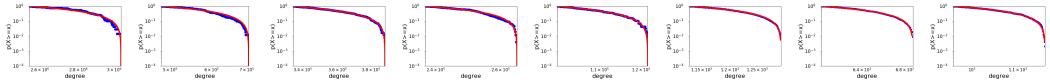
We use RMSprop as the optimizer, and the maximum number of epochs is 100. The (dense) CNN has a testing accuracy of 81.90%. This is then pruned using the method in [13], with $s = 0.7$. After retraining, the testing accuracy of the sparse CNN is 80.46%, and is comparable with the original network. Figure 3 shows the CCDF plot and TPL fit for each layer.



(a) conv1. (b) conv2. (c) conv3. (d) conv4. (e) fc5.

Figure 3: Log-log CCDF and TPL fit (red) for CNN on CIFAR-10.

AlexNet on ImageNet We perform experiments on the ImageNet data set [24], which has 1,000 categories, over 1.2 million training images, 50,000 validation images, and 100,000 test images. We use the sparsified AlexNet[5] (with 5 convolutional layers and 3 fully connected layers) from [11]. Figure 4 shows the CCDF plot and TPL fit for the various AlexNet layers.



(a) conv1. (b) conv2. (c) conv3. (d) conv4. (e) conv5. (f) fc6. (g) fc7. (h) fc8.

Figure 4: Log-log CCDF and TPL fit (red) for sparse AlexNet (naming of layers follows Caffe).

VGG-16 on ImageNet We use the (dense) VGG-16 model[6] (with 13 convolutional layers and 3 fully connected layers) obtained by the dense-sparse-dense (DSD) procedure of [12]. We then prune a fraction of $s = 0.3$ connections with the smallest magnitudes as suggested in [12]. The top-1 and top-5 validation accuracies for the original dense CNN are 68.50% and 88.68%, respectively, while those of the sparse CNN are 71.81% and 90.77%. Figure 5 shows the CCDF plot and TPL fit for various layers.

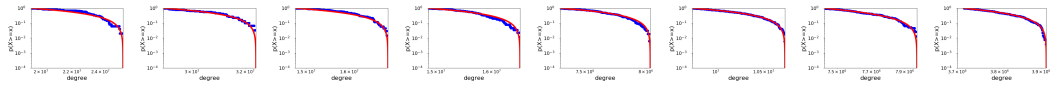
4 Preferential Attachment in Deep Neural Networks

As discussed in Section 1, the power law is related to the dynamics of an evolving network via preferential attachment [3]. In this section, we study this by considering the continual learning setting in which consecutive tasks are learned [16]. Specifically, at time $t = 0$, an initial sparse network is trained to learn the first task. At each following timestep $t = 1, 2, \ldots$, a new task is encountered, and the network is re-trained on the new task. We assume that only new connections, but not new nodes, can be added. Hence, preferential attachment, if exists, can only be internal but not external.

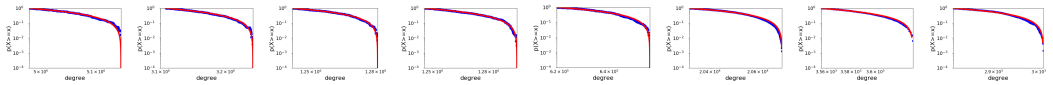
[4] https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py

[5] https://github.com/songhan/Deep-Compression-AlexNet

[6] https://github.com/songhan/DSD/tree/master/VGG16



(a) conv1_1. (b) conv1_2. (c) conv2_1. (d) conv2_2. (e) conv3_1. (f) conv3_2. (g) conv3_3. (h) conv4_1.



(i) conv4_2. (j) conv4_3. (k) conv5_1. (l) conv5_2. (m) conv5_3. (n) fc6. (o) fc7. (p) fc8.

Figure 5: Log-log CCDF and TPL fit (red) for VGG-16 (naming of layers follows Caffe).

4.1 Modeling the Evolution of Degree Distribution

In this section, we assume that only nodes in adjacent layers can be connected, as is common in feedforward neural networks. Consider a pair of nodes at time t, one from layer l with degree $d_1(t)$ and the other from layer $(l+1)$ with degree $d_2(t)$. If internal preferential attachment exists, the increase in the number of connections between this node pair is proportional to the product $d_1(t)d_2(t)$. Analogous to (5), the number of connections created per unit time between this node pair at time t is:

$$\Delta_t^l(d_1(t), d_2(t)) = N^l a^l \frac{d_1(t)d_2(t)}{\sum_{s,m} d_s(t)d_m(t)}, \tag{6}$$

where s and m are indices to all the nodes in the two layers involved, N^l is the number of nodes in layer l, and a^l is the number of new internal connections created in unit time from a node in layer l to layer $(l+1)$. Note that unlike (5), there is no factor of 2 here because the total number of connections created between nodes in layer l and $l+1$ is $N^l a^l$ at time t.

Remark 4.1 *Note that (6) assumes the same number of connections being created per unit time at all time t. However, as we assume that new nodes cannot be added to the network, the number of connections cannot grow infinitely. Hence, (6) will not hold for large t.*

For node i at layer l, let its degree at time t be $d_i(t)$. The following Proposition shows how the degree evolves. In particular, $d_i(t)$ is linear w.r.t. the node's initial degree $d_i(0)$. Proof can be found in Appendix C.

Proposition 4.1 $\frac{\mathrm{d}d_i(t)}{\mathrm{d}t} = \frac{(N^l a^l + N^{l-1}a^{l-1})d_i(t)}{\sum_{\text{node } s \text{ in layer } l} d_s(0) + (N^l a^l + N^{l-1}a^{l-1})t}$, *and* $d_i(t) = d_i(0)c^l(t)$, *where* $c^l(t) = 1 + \frac{(N^l a^l + N^{l-1}a^{l-1})t}{\sum_s d_s^l(0)}$.

Corollary 1 *For any layer, if its initial degree distribution follows the (standard or truncated) power law, its degree distribution at time t also follows the same power law.*

4.2 Experimental Evidence for Preferential Attachment

Experiments are performed on the MNIST data set. For illustration, we consider continual learning with only two tasks. Task A uses the original MNIST images, while task B uses images in which pixels inside a central 8×8 square are permuted. This permuted MNIST task has been commonly used in the continual learning literature [10, 16, 27, 28].

We use a MLP with 1024 nodes in each of the two hidden layers as in Section 3.1. Similar to PathNet[7] [8], the number of nodes is fixed, and new connections are created when a new task arrives. The training procedure is shown in Figure 6. We first train a dense network on task A. 90% of the connections are pruned from each layer (except for the last softmax layer) as in Section 3.1. The remaining 10% connections are fine-tuned on task A, and then frozen (as in PathNet). Next, we reinitialize the pruned connections and train a dense network on task B. Connections in each layer

[7] In PathNet, each module is often a convolutional or linear network. Here, each module is simply a node.

are again pruned so that only 20% remain. Recall that connections learned on task A are frozen, and so they are not pruned or retrained, while the newly added 10% connections are fine-tuned on task B. The other parts of the setup are the same as in Section 3.1.

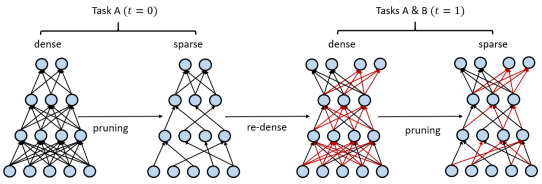
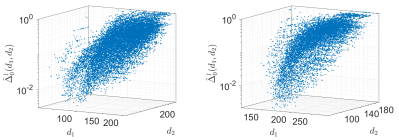


Figure 6: The continual learning setup.



(a) input-to-fc1. (b) fc1-to-fc2.

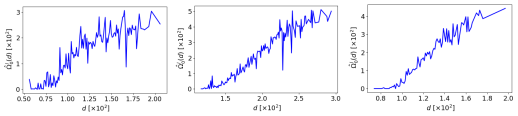
Figure 7: $\Delta_0^l(d_1, d_2)$ vs d_1 and d_2.

Figure 7 shows $\Delta_0^l(d_1, d_2)$ in (6) at time $t = 0$ versus the degrees d_1 and d_2. As can be seen, $\Delta_0^l(d_1 d_2)$ increases with d_1 and d_2, indicating the presence of internal preferential attachment.

From Proposition 4.1, for a node with degree $d_i(0)$ at $t = 0$, the number of connections added to it at $t = 1$ is

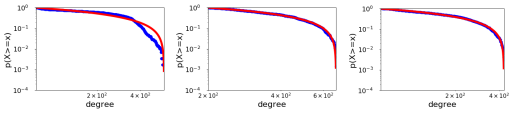
$$\left.\frac{\mathrm{d}d_i(t)}{\mathrm{d}t}\right|_{t=0} = d_i(0)\frac{N^l a^l + N^{l-1}a^{l-1}}{\sum_s d_s(0)}. \tag{7}$$

Let $d_i(0) = d$, an empirical estimate of $\left.\frac{\mathrm{d}d_i(t)}{\mathrm{d}t}\right|_{t=0}$ (denoted $\hat{\Omega}_0^l(d)$) can be obtained by counting the number of new connections that all nodes (from layer l) with degree d made at time $(t + 1)$, and then divide it by the number of degree-d nodes (in layer l) at time t. Figure 8 shows this empirical estimate (denoted $\hat{\Omega}_0^l(d)$) versus the node degree d at time $t = 0$. As can be seen, the input layer shows a weaker linear relationship compared to the other two layers. This is because with permutation, the network must learn to associate new collections from pixels to penstrokes, and connections from the input layer to the first hidden layer have to be modified. On the other hand, as discussed in [10], once the input layer has established new associations from pixels to penstrokes, the higher-level feature extractors (i.e., fc2) are only dependent on these penstroke features (extracted at fc1) and thus less affected by the input permutation.



(a) input. (b) fc1. (c) fc2.

Figure 8: Number of new connections $\hat{\Omega}_0^l(d)$ vs degree d.



(a) input. (b) fc1. (c) fc2.

Figure 9: Log-log CCDF and TPL fit (red) for MLP at $t = 1$.

Recall from Section 3.1 that the degree distribution of the sparse MLP trained on task A ($t = 0$) follows the TPL. From Corollary 1, we expect the degree distribution after training on task B ($t = 1$) also follows the TPL. Figure 9 shows the CCDF plot and TPL fit for each layer of the sparse network at $t = 1$. As can be seen, all the layers fit well except for the input layer. As in the discussions on Figure 8, this is again because the input layer has to establish new associations to map from the permuted pixels to penstrokes.

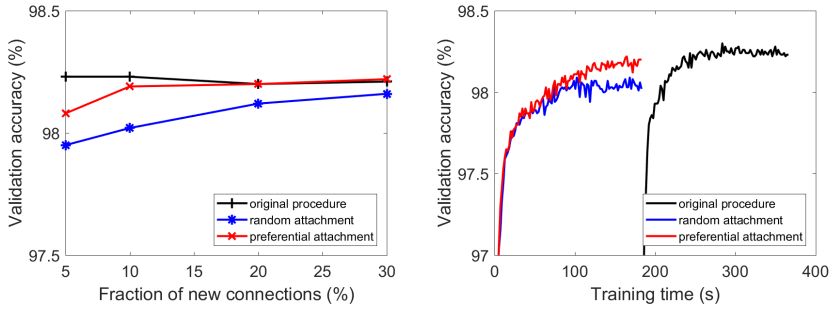
5 Applications to Continual Learning

In biological neural networks, reuse of neural circuits is essential for learning new tasks with limited capacity. Scale-free networks have been shown to resemble biological neural networks and achieve fast learning [22]. As another important aspect in learning a diverse range of tasks in real-life, humans can transfer what they learned to new tasks through exaptation of existing neural circuits [2]. In this section, we demonstrate how preferential attachment can help speed up training (Section 5.1), and lead to more economical representations (Section 5.2) in continual learning.

5.1 Preferential Attachment with PathNet-like Network

Recall that in Section 4.2, in order to obtain a sparse network for task B, a dense network (by reinitializing all the pruned connections) is first trained, pruned and then fine-tuned. This can be computationally expensive, especially when the dense network is large. In this experiment, we consider constructing a sparse network directly by preferential attachment. Two approaches are used to select the connections to be added: (i) (internal) preferential attachment, in which the probability that a connection is added between two nodes is proportional to the product of their degrees; and (ii) random attachment, which selects the new connections randomly.

Figure 10(a) shows the validation accuracies when different fractions of new connections (out of the total number of connections in the dense network) are added. As can be seen, preferential attachment always outperforms random attachment. Moreover, adding 10% new connections via preferential attachment achieves similar accuracy as the original procedure. When the network is very sparse, the original procedure can have higher accuracy. However, it is much more expensive[8] (Figure 10(b)).



(a) Different fractions of new connections. (b) Training time.

Figure 10: Comparison of different approaches to obtain a sparse network in continual learning. In Figure 10(b), the time measured is for preferential/random attachment with 10% new connections, and the original procedure with 5% new connections.

5.2 Preferential Attachment with Progressive Neural Network

The progressive neural network [25] is a popular deep learning model for continual learning [9, 26] (a brief introduction is in Appendix D). It overcomes catastrophic forgetting and leverages prior knowledge via lateral connections to previously learned features. For illustration, we again consider the simple setting with two tasks, A and B. A sparse network is first trained on task A (here, using the method in [13]), and then fixed. A second narrow network, with the same number of layers, is constructed for task B. Each new hidden layer receives input from the previous layers in both the old and new networks. In the standard progressive neural network, lateral connections are dense. As each new task needs to establish lateral connections to all previously learned networks, the number of weights thus grows quadratically with the number of tasks.

In this experiment, we use sparse lateral connections selected by (i) (external) preferential attachment [3]; and (ii) random attachment. For layer l of the task A network, let N_l be its number of nodes, and $d_l^{(1)}(j)$ be the degree of one of its nodes j. For (external) preferential attachment, a node at layer l of the task B network is connected to a node j at layer $(l-1)$ of the task A network with probability $d_{l-1}^{(1)}(j) / \sum_{k=1}^{N_{l-1}} d_{l-1}^{(1)}(k)$. For random attachment, a node at layer l of the task B network is connected to a node at layer $(l-1)$ of the task A network with probability $1/N_{l-1}$. As a further baseline, we also compare with the case where networks for tasks A and B are not connected, such that B does not utilize knowledge learned from A.

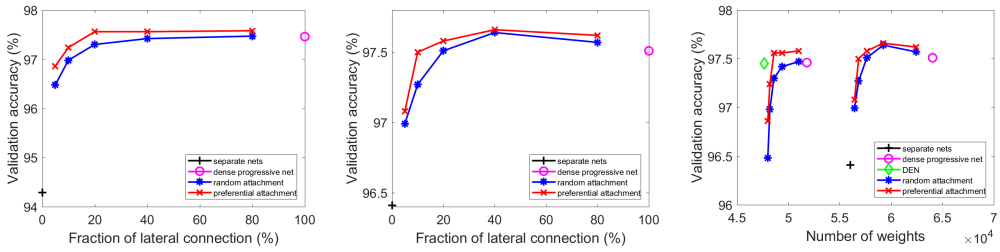
MLP on MNIST The first network is a sparse MLP trained on the original MNIST data set (task A) using the method in [13] as in Section 3.1. Each of the two hidden layers has 200 nodes, and the

[8] In the implementation, we use masking to handle sparse weights. A higher speedup is expected when targeted operations for sparse weights are used [23].

sparsity for each layer is 0.8. A second network, with K nodes in each hidden layer, is then learned on the permuted MNIST data set (task B).

Figures 11(a) and 11(b) show the validation accuracies for $K = 10$ and 20, respectively. As can be seen, preferential attachment consistently outperforms random attachment. Accuracy drops when a large fraction of lateral connections are used (e.g., 80%), indicating redundancy in connectivity. With only 5% lateral connections to task A, preferential attachment has significantly better accuracy than using separate networks for the two tasks. With 20% lateral connections, preferential attachment has similar or even better accuracies than the dense progressive neural network.

Figure 11(c) shows the total number of weights after task B is learned. We also compare with the very recent dynamically expandable network (DEN) [27]. As can be seen, though DEN has a slightly smaller number of weights, its accuracy is worse than preferential attachment. Moreover, DEN relies on expensive optimization to select connections and expand networks, while the preferential attachment mechanism is computationally very simple.



(a) K=10. (b) K=20. (c) Number of weights.

Figure 11: Performance of the progressive neural network variants on MNIST.

CNN on CIFAR In this experiment, we use the two-task continual learning setting in [28]. The first network is a sparse CNN (each convolutional layer has a sparsity of 0.7) trained on the CIFAR-10 data set (task A) as in Section 3.2. A second network is then learned using the first 10 classes of the CIFAR-100 data (task B). The new network has a structure analogous to that of the task A network. Each of its first two convolutional layers has K feature maps, and each of the next two convolutional layers has 2K features maps. The number of nodes in the FC layer is kept the same as 512.

Table 1 shows the validation accuracies for $K = 4$ and 12. We do not compare with DEN as only its MLP implementation is publicly available. Again, selecting connections via preferential attachment consistently outperforms random attachment and the use of separate networks. By using only 10% of the lateral connections, it achieves better accuracy than the dense progressive neural network.

Table 1: Performance of the progressive neural network variants on CIFAR. Here, “fraction" denotes the fraction of lateral connections used.

	K=4		K=12	
	accuracy (%)	fraction	accuracy (%)	fraction
separate networks	42.44	0%	55.56	0%
dense progressive net	76.22	100%	76.67	100%
random attachment	74.89	10%	73.79	10%
preferential attachment	**76.89**	10%	**77.44**	10%

6 Conclusion

In this paper, we showed that a number of sparse deep learning models exhibit the (truncated) power law behavior. We also proposed a preferential attachment model to explain how the network topology evolves, and verified in a continual learning setting that new connections added to the network indeed have this preferential bias. Scale-free networks have been shown to resemble biological neural networks and achieve faster learning. We similarly showed that in a deep neural network, making new connections using preferential attachment also leads to faster learning and smaller networks.

References

[1] L. A. N. Amaral, A. Scala, M. Barthelemy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152, 2000.

[2] M. L. Anderson. Neural reuse: A fundamental organizational principle of the brain. *Behavioral and Brain Sciences*, 33(4):245–266, 2010.

[3] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[4] A. L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3):590–614, 2002.

[5] S. M. Burroughs and S. F. Tebbens. Upper-truncated power laws in natural systems. *Pure and Applied Geophysics*, 158(4):741–757, 2001.

[6] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[7] V. M. Eguiluz, D. R. Chialvo, G. A. Cecchi, M. Baliki, and A. V. Apkarian. Scale-free brain functional networks. *Physical Review Letters*, 94(1):018102, 2005.

[8] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. Preprint arXiv:1701.08734, 2017.

[9] J. Gideon, S. Khorram, Z. Aldeneh, D. Dimitriadis, and E. M. Provost. Progressive neural networks for transfer learning in emotion recognition. Preprint arXiv:1706.03256, 2017.

[10] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. Preprint arXiv:1312.6211, 2013.

[11] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference of Learning Representations*, 2016.

[12] S. Han, J. Pool, S. Narang, H. Mao, S. Tang, E. Elsen, B. Catanzaro, J. Tran, and W. J. Dally. DSD: Regularizing deep neural networks with dense-sparse-dense training flow. In *International Conference of Learning Representations*, 2017.

[13] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[14] R. Hanel, B. Corominas-Murtra, B. Liu, and S. Thurner. Fitting power-laws in empirical data with estimators that work for all exponents. *PloS One*, 12(2):e0170920, 2017.

[15] Y. Iturria-Medina, R. C. Sotero, Erick J. Canales-Rodríguez, Y. Alemán-Gómez, and L. Melie-García. Studying the human brain anatomical network via diffusion-weighted mri and graph theory. *Neuroimage*, 40(3):1064–1076, 2008.

[16] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[17] D. Kolyukhin and A. Torabi. Power-law testing for fault attributes distributions. *Pure and Applied Geophysics*, 170(12):2173–2183, 2013.

[18] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *International Conference of Learning Representations*, 2017.

[21] B. D. Malamud, G. Morein, and D. L. Turcotte. Forest fires: An example of self-organized critical behavior. *Science*, 281(5384):1840–1842, 1998.

[22] R. L. S. Monteiro, T. K. G. Carneiro, J. R. A. Fontoura, V. L. da Silva, M. A. Moret, and H. B. de Barros Pereira. A model for improving the learning curves of artificial neural networks. *PloS One*, 11(2):e0149874, 2016.

[23] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally. SCNN: An accelerator for compressed-sparse convolutional neural networks. In *Annual International Symposium on Computer Architecture*, pages 27–40, 2017.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, A. Khosla, M. Bernstein, A. C. Berg, and F. Li. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[25] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. Preprint arXiv:1606.04671, 2016.

[26] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. Preprint arXiv:1610.04286, 2016.

[27] J. Yoon, E. Yang, J. Lee, and S. Hwang. Lifelong learning with dynamically expandable networks. In *International Conference of Learning Representations*, 2018.

[28] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.