Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting

Nima Dehmamy[†], Albert-László Barabási[‡], Rose Yu[†]

[†]Northeastern University, Boston, MA [‡]Harvard Medical School, Boston MA; Central European University, Budapest, Hungary

Problem

What types of functions are *Graph Convolutional Networks* (GCN) capcable of learning?
What design principles can make GCN-based models that are expressive, yet economical?

Our Results

Modular GCN design using multiple propagation rules and skip layers

- We find that GCN-based models are very restrictive, unable to learn non-smooth functions of the propagation rule (PR);
- GCN requires at least n layers to learn functions that have nth power of the PR;
- **Multiple PR** working in tandem can make GCN more expressive;
- **Skip layers** and fully connected layers mixing outputs of different PR between GCN layers also help;
- **Number of layers** is more influential than width of layers in the ability of our GCN-based model to discriminate graph topologies;
- We use **synthetic graphs** to precisely control the ground truth and level of difficulty of the task.

Graph Moments and Node Permutation Invariance

Define pth order "graph moment" M_p as the node average of an order p polynomial of the **adjacency matrix** A

$$M_p(A) = \prod_{q=1}^{p}(A \cdot W_q + B_q) \tag{1}$$

with W_q and B_q being $N \times N$ matrices. Under the constraint of **node permutation invariance** (NPI), W_q must be either proportional to the identity matrix, or a uniform aggregation matrix

$$f(A) = A \cdot W + B, \qquad \textbf{NPI} \Rightarrow \qquad W, B = cI, \quad \text{or} \qquad W, B = c\mathbf{1}\mathbf{1}^T \tag{2}$$

where $\mathbf{1}$ is a vector of ones. For instance, $M_p(A) = \sum_j (A^p)_{ij}$ aggregates graph powers, where A^p_{ij} is the number paths from node i to j of length p. Some NPI functions of A important in physical processes on graphs are

$$D_{ij} \equiv \delta_{ij} \sum_k A_{ik} \qquad \hat{A} \equiv D^{-1}A \qquad \hat{A}_s \equiv D^{-1/2}AD^{-1/2} \tag{3}$$

Problem Statement Graph moments encode much of the structural topology of the graph. We want to identify efficient architectures for learning graph moments. This will help us perform learning tasks such as classification of graph structure, as well as learning evolution of a graph.

Learning Graph Moments: Fully connected (FC) vs GCN

FC neural networks are universal approximators and can, in principle learn any function. Howerver, they generally require large amounts of data to train (Fig below).

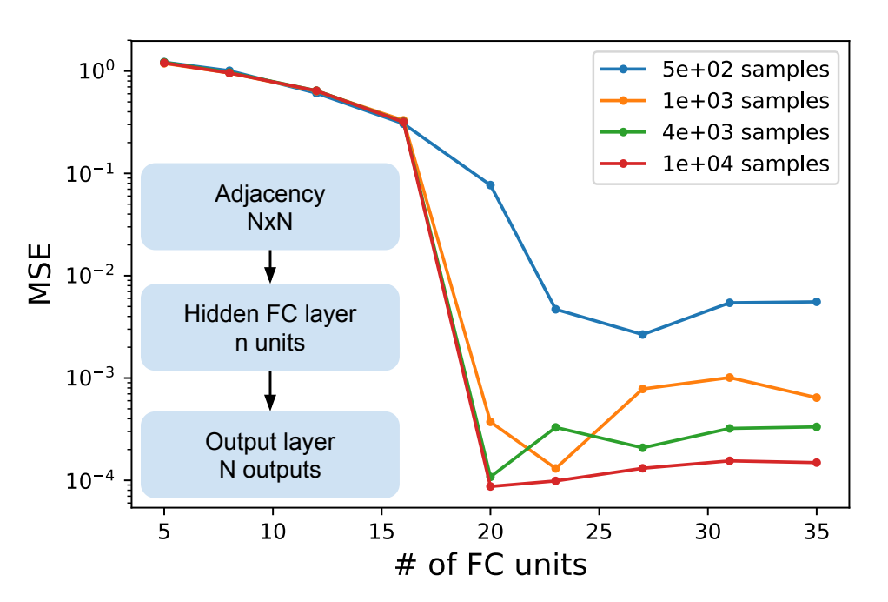


Figure: Learning graph moments (Erdős-Rényi graph) with single fully-connected layer. Best validation MSE w.r.t number of hidden units n and the number of samples in the training data (curves of different colors).

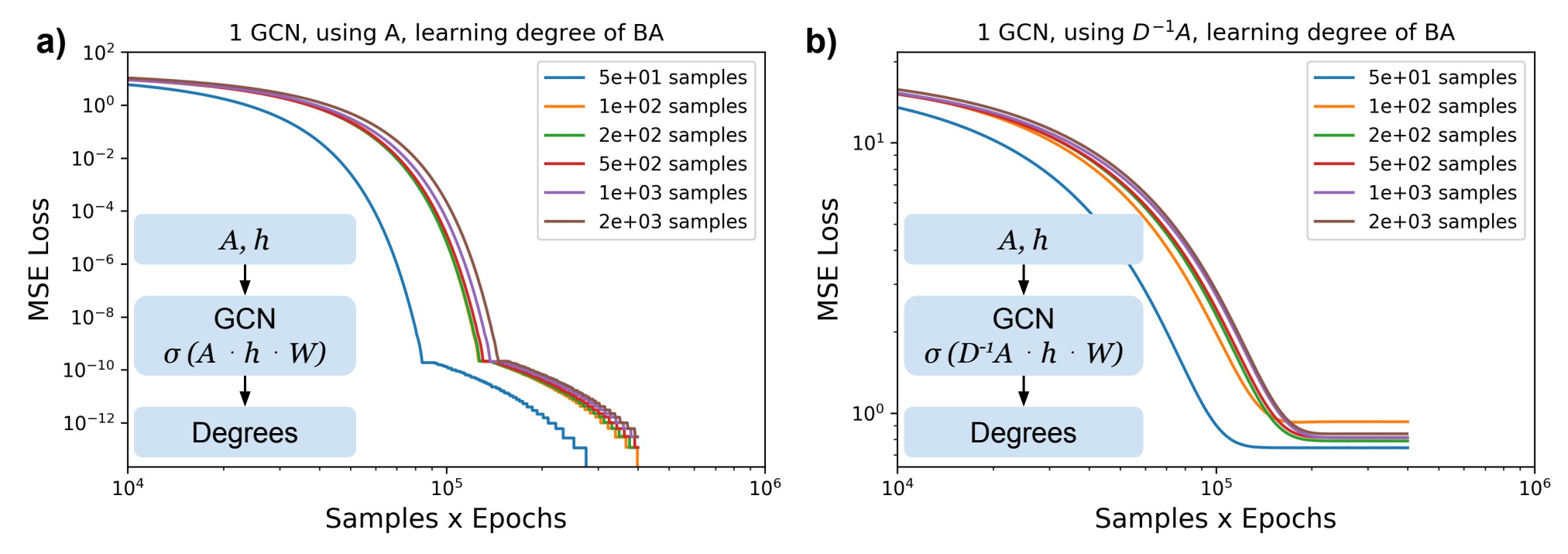


Figure: Single layer of GCN with a single weight learning the degree of nodes in a graph. When the GCN layer is designed as $\sigma(A \cdot h \cdot W)$ with linear activation function $\sigma(x) = x$, the network easily learns the degree (a). However, if the network is designed as $\sigma(D^{-1}A \cdot h \cdot W)$, it fails to learn degree, with very high MSE loss (b). The training data were instances of Barabasi-Albert graphs (preferential attachment) with $N = 20$ nodes and $m = 2$ initial edges.

Theorem 1: A fully connected neural network with one hidden layer requires $n > O(C_f^2) \sim O(p^2N^{2q})$ number of neurons in the best case with $1 \leq q \leq 2$ to learn a graph moment of order p for graphs with N nodes. Additionally, it also needs $S > O(nd) \sim O\left(p^2N^{2q+2}\right)$ number of samples to make the learning tractable. (Follows Directly from [2])

GCN is Explicitly Node Permutation Invariant: A single layer GCN propagates node attributes h using a function $f(A)$ (the propagation PR) and has output

$$F(A,h) = \sigma\left(f(A) \cdot h \cdot W + b\right) \tag{4}$$

W is the weight matrix and b is the bias. As we only consider graph topology and ignore the node attributes, we set $h_i = 1$.

Lemma 1: A graph convolutional network with $n < p$ layers cannot, in general, learn a graph moment of order p for a set of random graphs. (We prove this by showing a counterexample.)

Proposition 1: A graph convolutional network with n layers, and no bias terms, in general, can learn $f(A)_i = \sum_j (A^n)_{ij}$ only if $n = p$ or $n > p$ if the bias is allowed.

Theorem 2: With the number of layers n greater or equal to the order p of a graph moment $M_p(A)$, where A is the adjacency matrix, graph convolutional networks with residual connections can learn a graph moment M_p with $O(p)$ number of neurons, independent of the size of the graph.

Theorem 2 highlights the importance of residual connections. By introducing residual connections into multiple GCN layers, we can learn any polynomial function of graph moments. Interestingly, Graph Isomophism Network (GIN) proposed in [4] uses the following propagation rule:

$$F(A,h) = \sigma\left([(1+\epsilon)I + A] \cdot h \cdot W\right) \tag{5}$$

which is a special case of our GCN design with one residual connection between two modules.

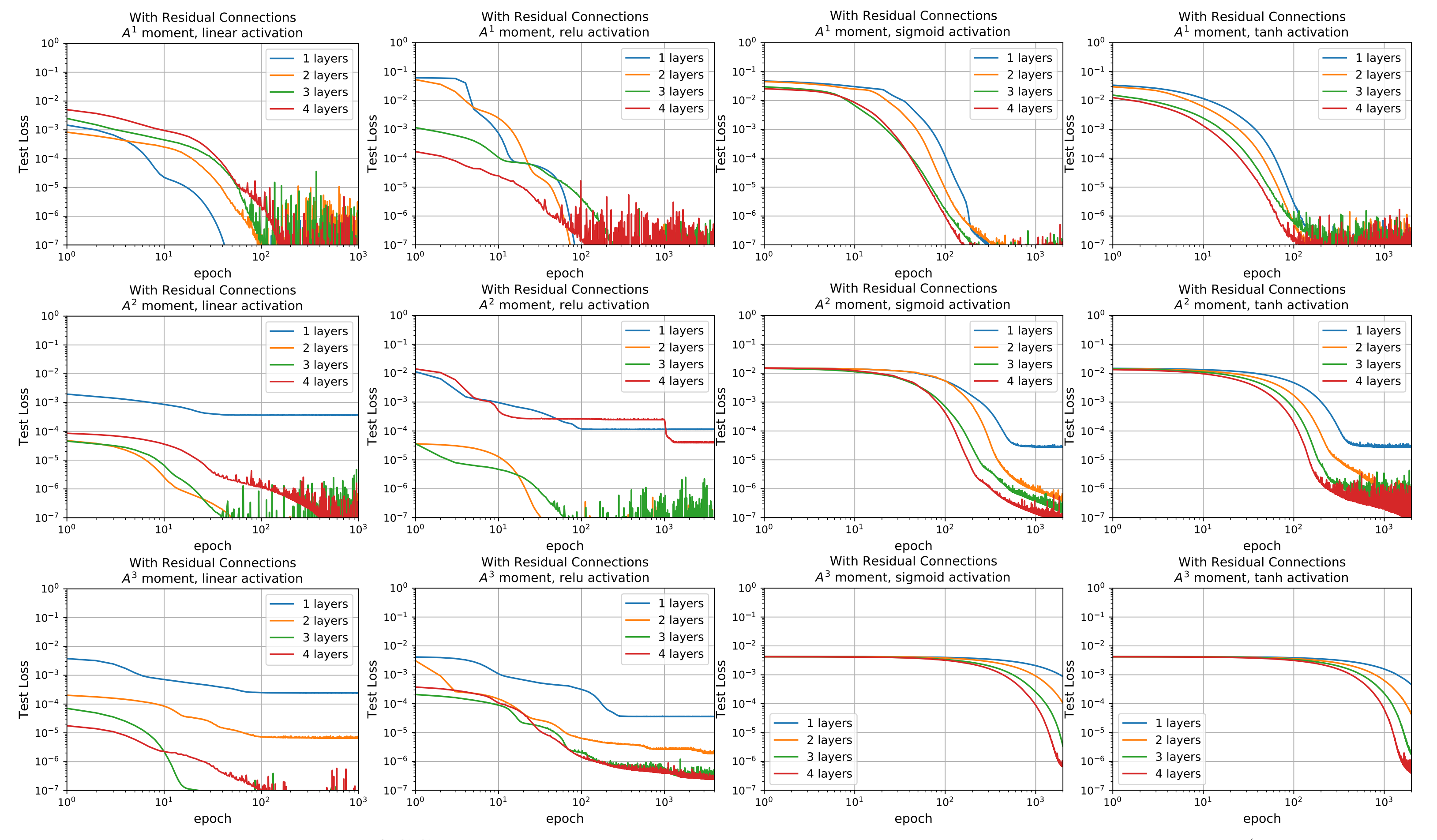
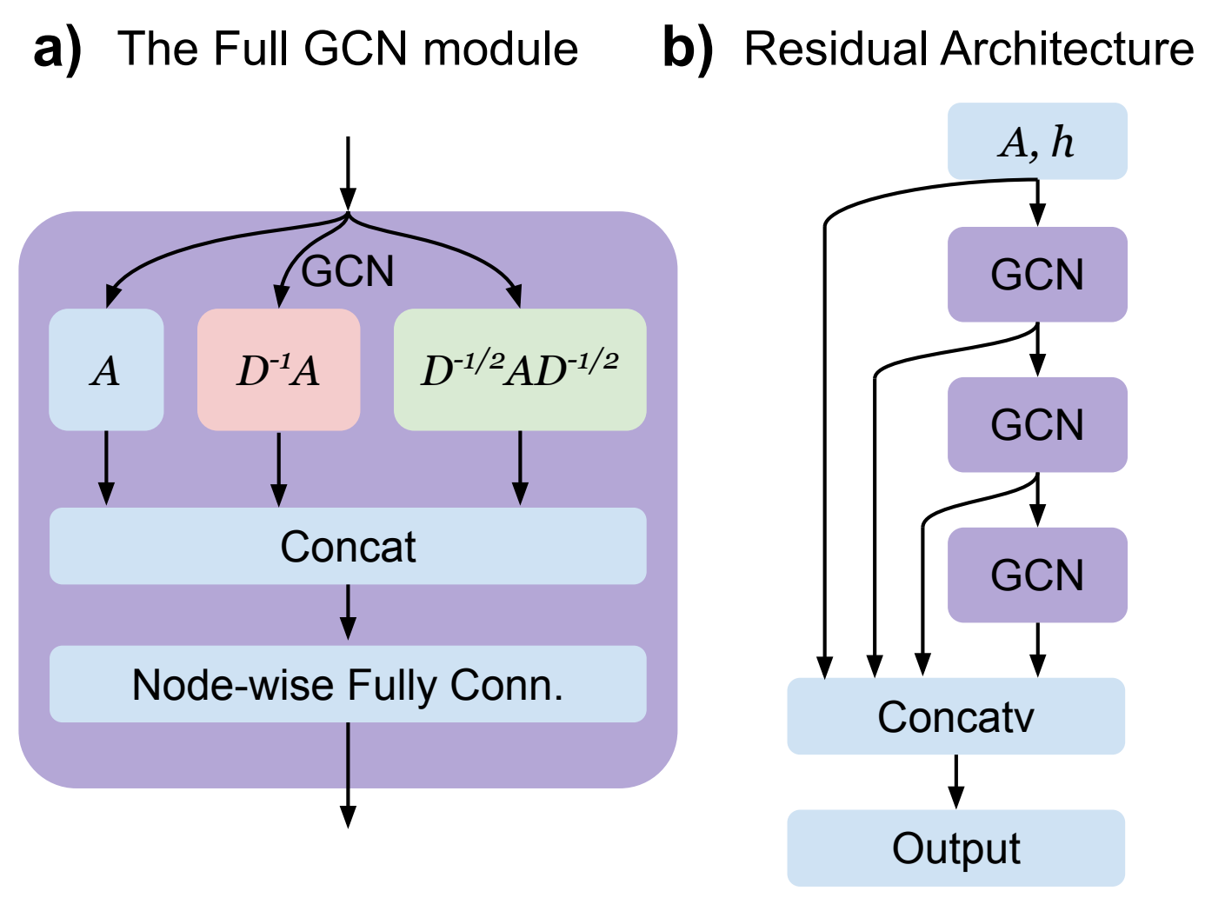


Figure: Representation power of GCN *with* residual connections: Using residual outputs through skip layers (passing the output of lower layers directly to the last layer), we can ensure that a system of GCN layers interlaced with FC layers is capable of learning a large class of polynomial functions $f(A) = \sum_{i,m} a_m \left(A^m\right)_{ij}$ of the graph adjacency matrix A.

Modular Design Increases Expressive Power

We suggest using a GCN layer with three different propagation rules and a node-wise FC layer to improve expressivity (Fig. a). Using residual connections (Fig. b) allows a n-layer modular GCN to learn any polynomial function of order n of its constituent operators.

Ablation Study: The importance of multiple PR can be seen in classification task of disciminating a Erdős-Rényi (ER) random graph from Barabasi-Albert (BA) [1] preferential attachment model, with similar nodes and edges. The table below shows the change of test accuracy when we use different combinations of modules. A single PR is not enough. Having all three PR leads to almost perfect discrimination between BA and ER graphs.



Modules	Accuracy
f_1	53.5 %
f_3	76.9 %
f_1, f_3	89.4 %
f_1, f_2, f_3	98.8 %

Table: Test accuracy with different modules combinations for BA-ER. $f_1 = A$, $f_2 = D^{-1}A$, and $f_3 = D^{-1/2}AD^{-1/2}$.

Experiments

1) BA vs. ER. Classification
2) BA vs. Configuration Model Distinguishing BA from ER for small graphs may still be easy.
Harder task: Rewire the edges of BA graphs using a *configuration model* [3] (Config) to generate "fake" BA graphs. The resulting graphs share exactly the same degree distribution, and even mirror the real BA in higher graph moments.

Distinguishing BA and Config graphs is very difficult using standard methods such as a Kolmogorov-Smirnov (KS) test, measuring the distributional differences of a statistical measure between two graphs and uses hypothesis testing. Figure below shows the KS test values for pairs of real-real BA (blue) and pairs of real-fake BA (orange) w.r.t different graph moments (dashed black lines are mean of the KS test values for real-real pairs), showing it is very difficult to distinguish between real and Config BA.

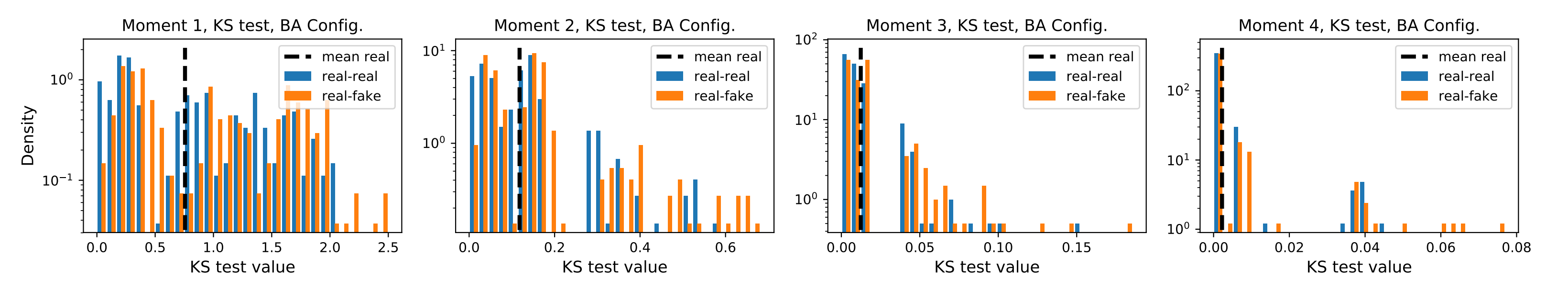


Figure: Distribution of Kolmogorov-Smirnov (KS) test values for differences between graph the first four graph moments $\sum_i (A^p)_{ij}$ in the dataset. "real-real" shows the distribution of KS test when comparing the graph moments of two real instances of the BA. All graphs have $N = 30$ nodes, but varying number of links. The "real-fake" case does the KS test for one real BA against one fake BA created using the configuration model.

Classification Using our GCN Module Our architecture consists of layers of our GCN module, linear activation. The output is passed to a fully connected layer with softmax activation, yielding and $N \times c$ matrix (N nodes in graph, c label classes). The final classification is found by mean-pooling over the N outputs. We find that **depth is more influential than width:** increasing one layer can improve the test accuracy by at least 5%, whereas increasing the width has very little effect. Smaller graphs are harder to learn, while for $N \geq 50$ nodes is enough for 100% accuracy in BA-ER case. BA-Config is a much harder task, with the highest accuracy of 90%.

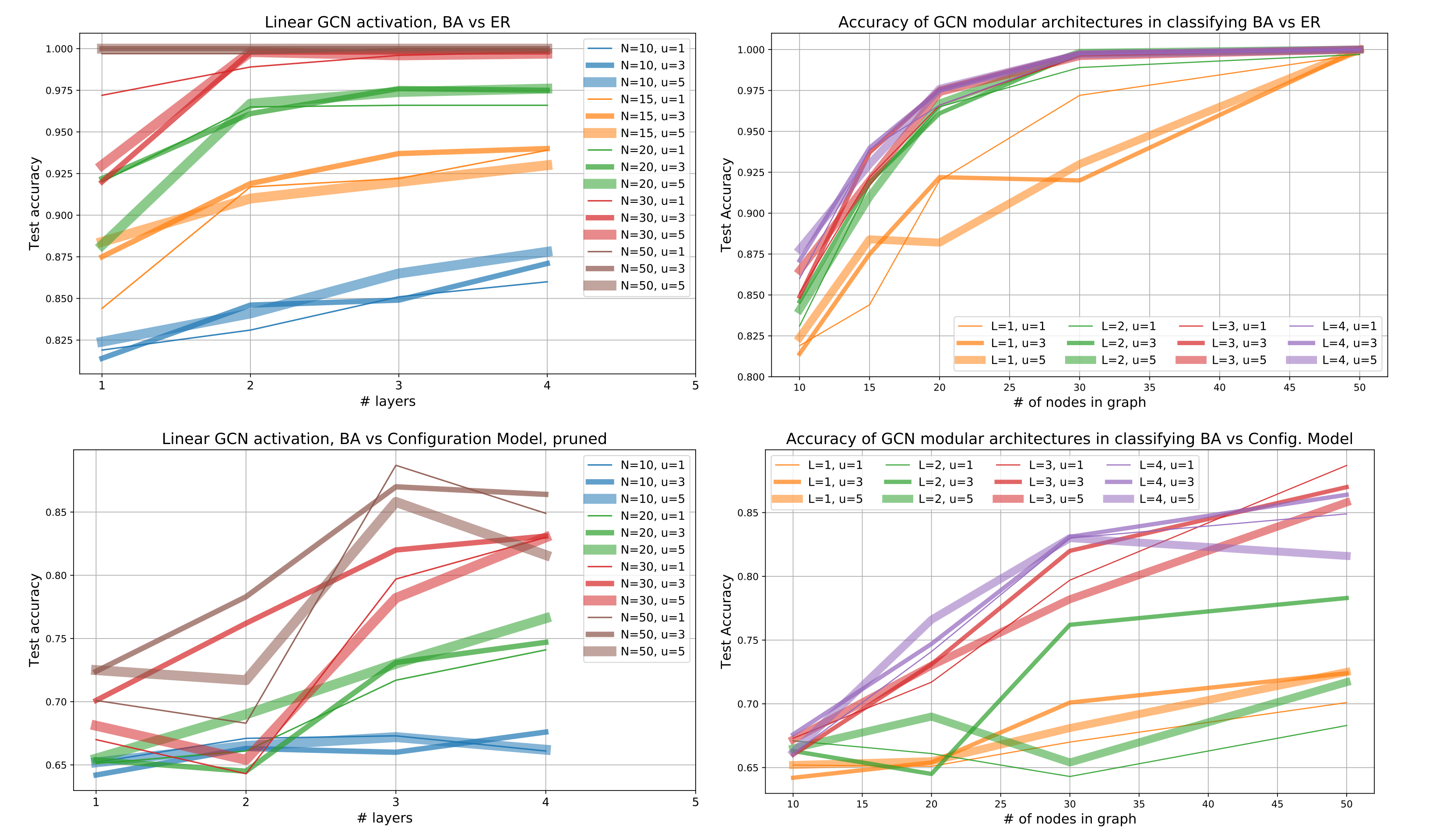


Figure: Classify graphs of **Barabasi-Albert** model vs. **Erdos-Renyi** model (top) and **Barabasi-Albert** model vs. **configuration** model (bottom). Left: test accuracy with respect to network depth for different number of nodes (N) and number of units (U). Right: test accuracy with respect to graph size for different number of layers (L) and number of units (U).

Data statistics

Different experiments with datasets of $N = 10$ to $N = 60$ nodes.

$2,500$ Barabasi-Albert (BA, rich gets richer) graphs are evenly split with $m = 1, N/8, N/4, 3N/8, N/2$.
$2,500$ configuration model BA matching the number of edges in the BA dataset. To avoid bias from the order of nodes we shuffle the node labels.

$2,500$ Erdos-Renyi (ER) random graphs with edge probability p uniformly between $1/N$ and $N/2$, matching density of our BA graphs.

References

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] A. R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

[3] M. Newman. *Networks: an introduction*. Oxford university press, 2010.

[4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.