Evaluation of Network Bottlenecking and Centrality as Metrics for Effective Collaborative Networks

Naren Sivagnanadasan, Tanishq Dubey
University of Illinois at Urbana–Champaign

***Abstract*—Many of the interesting networks in the world are collaborative networks, where nodes work together to accomplish larger tasks than any one can complete by themselves. These networks describe how we build things, how bees survive the winter, how computers analyzes terabytes of data and how we and computers can perceive the world. However, designing these networks is difficult unless the task itself can be described concretely. For instance structuring a corporation to maximize profit may result in applying the current management trends then refining based on returns.**

In this work we want to explore the potential for topological metrics that network designers can apply in designing their systems. The metrics we chose to evaluate were betweenness centrality, PageRank centrality, in/out degree distribution and all pairs max flow. We use these metrics to evaluate networks of various effectiveness from the realms of artificial neural networks, naturally occurring networks and randomly generated ones.

We find that low variance in betweeness centrality and low mode and mean of PageRank are signs of effective networks and we see that there is a natural tendency for the flow of collaborative networks to narrow from source to sink instead of the flow spreading to use excess capacity later in the path. This leads us to support a common held intuition for feedforward neural network design, which is to determine the necessary capacity of the final hidden layer and make the preceding networks progressively wider. We believe that centrality is an important quality of effective networks and see this work as evidence that further work in this direction is a good idea.

I. Introduction

Collaborative networks appear in a huge variety of places. They show up in how we manufacture goods, how we get around, how we develop new knowledge and in naturally occurring networks ranging from swarm-like organizations. Much of what we depend on today is a result of collaboration between many individuals. Very often we can see that the efficiency or performance of collaborative systems is hindered by the structure of the network, anecdotally when one works in a team that has no clear leader or is too large and in the literature of management practices [1].

A. Collaborative Networks

We take the definition presented in Camarinha-Matos et al. as our definition of collaborative networks: "A collaborative network (CN) is constituted by a variety of entities (e.g.,organizations and people) that are largely autonomous, geographically distributed, and heterogeneous in terms of their: operating environment, culture, social capital, and goals. Nevertheless these entities collaborate to better achieve common or compatible goals, and whose interactions are supported by computer network." [2] and extend it to remove the limitation on purely networks backed by computer networks. Notable examples of collaborative networks include: Organizations, online collaboration networks such as open source projects, the ethereum network or the polymath project, natural symbiotic systems such as those describing the dependencies between plants and their pollinators and swarm systems. Artificially constructed examples of these networks include, distributed systems, and artificial neural nets.

1) Artificial Neural Networks (ANNs): In this work we evaluate ANNs as they are a collaborative network which is easy to design and evaluate. ANNs leverage very simple computations to develop incredibly sophisticated capabilities.

A common simple example of an ANN is the multilayer perceptron, which is an weighted directed acyclic computational graph. Each edge is weight corresponds to the importance of the activation of the node it originates from to the node it enters. Non-linear functions filter the activation to give the NN its function approximation capability. An example of a multilayer perceptron is as follows:

$$y' = w_5^T F(w_4, b_4, F(...)) + b_5$$

where $F(w, b, x) = relu(w^T x + b)$ and $relu(x) = max\{0, x\}$

The weights of the network are determined through gradient decent however as we show later the architecture of the network has an effect on where weights are high and where they are low. Designing effective neural network architectures is difficult and often relies on analogies to nature, framing of problems and intuition. There has been work on using neural networks to design other neural networks [3] but intentional design of Neural Nets is still a standing problem.

B. Design of Collaborative Networks

The design of many collaborative systems up has a lot of literature behind it, from observing successful organizations, the way natural systems interact and other existing systems, academia has come up with many theories on effective structures for collaboration. Often these have be summed up in common rules of thumb, like the importance of centralized decision making [4], the size of teams. [5] or adages like C-Level executives having an open door policy to improve productivity and moral [6]. Once a system has been created, it is also common for these networks to slowly refine themselves based on performance metrics (e.g. output, revenue, survival). For naturally occurring collaborative networks, these systems have been refined over generations through natural selection.

While this means that over time the efficiency of collaborative systems is improving, there may yet be techniques and structures unexplored due to the nature of iterative refinement.

C. Topological Metrics to Aid in the Design of Collaborative Networks

We in this work, examine the potential for creating topological metrics to aid in the design of collaborative networks so that instead of relying purely experience, intuition and refinement based on performance, designers of these networks have solid design parameters to tune for the results they are looking for. The features of networks are numerous and present many different tunable parameters.

We have chosen to examine the role of bottlenecking and betweenness centrality, to which we compare another centrality metric - PageRank. In addition we also examine in/out degree distribution and all pairs max flow on a variety of networks each exhibiting a different level of effectiveness at collaborating. Using these metrics we hope to see qualities that emerge differentiating effective collaborative networks from the rest. The ability to show that these metrics signify effective collaboration could lead to a new set of tools to help designers in the creation of new networks.

II. Related Work

The metrics we have chosen to examine stem from evidence shown in the literature, notable cases include the evaluation of protein networks and of various artificial neural nets lead us to believe that bottlenecking signifies effective collaboration

A. The Role of Bottlenecks in Protein Regulation Networks

Yu et al. [7] show through an analysis of protein regulation networks in yeast that bottlenecks in the network are correlated with essential proteins. This was known about protein networks before Yu et al.'s work however due to analysis approaches it was not clear if the correlation was due the degree of nodes in the protein network or the betweenness of nodes. Yu et. al shows that it is indeed a factor correlated with betweenness in particular.

B. Development of Specialized Nodes and Sub-Graphs and Their Importance in Artificial Neural Nets

Radford et al. [8], Zhou et al. [9] and Guimar et al. [10] show in their respective work that as a result of training, nodes of artificial neural networks begin to segment the high level task into smaller sub tasks, resulting in specific nodes and sub-graphs tuned to recognize specific stimuli Guimar et al. shows that in a network designed to predict chemical properties of organic compounds specific nodes will "[be] assigned a specific task, either as a main predictor of the output or as a correction factor". Zhou et al. shows that in the training of deep convolution neural nets, that they will "automatically discover meaningful object detectors, representative of learned scene categories". And Radford et al. 2017 demonstrates this again with the analysis of recurrent neural nets used for text generation, that through the training process a specific node gets assigned the task of sentiment analysis. Furthermore, Radford et al. shows that simply by fixing the output value of the network, a positive or negative sentence can be generated out of the entire network. This implies that specific nodes are extremely important in the result of a neural networks evaluation and gives us more confidence in the importance of bottlenecks in networks

III. Methods

To evaluate this hypothesis, we measured a set of metrics on three classes of networks, artificial networks in the form of ANNs, natural occurring networks and randomly generated networks.

A. Measured Metrics

1) Betweenness Centrality: Betweenness centrality measures the number of geodesic paths that flow through a particular node. This measure is analogous to the the amount of bottlenecking is caused by the node. This metric is defined by the following equation:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Where σ_{st} is the total number of geodesic paths from s to t and $\sigma_{st}(v)$ is the number of geodesic paths from s to t which pass through v [11] [12]

2) PageRank Centrality: PageRank centrality measures the probability of arriving at specific node by traversing edges. Thus the higher the value of a node, the more "important" it is. In particular the PageRank centrality of a node is derived from the PageRank centrality of its neighbors, proportional to their out-degree. The metric is defined by the following equation:

$$C_p(v) = \alpha \sum_{u \in B_v} \frac{C_p(u)}{L_u} + \frac{1-\alpha}{N}$$

Where B_v is the set of nodes with edges pointing to v, L_u is the number of outgoing edges from u and N is the total number of nodes. α is a tuning parameter which controls the residual probability that a node is randomly connected to other nodes, to account for nodes which are just sinks (i.e. no outbound edges). [13]

3) In/Out Degree Distribution: In degree, the measure of how many directed edges are coming into a node, and out degree, the measure of how many directed edges are starting from a node, are used as comparative metrics to quantify the differences between network structure.

4) All Pairs Max Flow: All pairs max flow was computed by finding all sinks and sources within a network, and then computing the flow between the pair of source and sink. To measure all pairs max flow, we employ two algorithms, the Edmonds-Karp Algorithm [14] and the Push-Relabel Algorithm [15]. The Edmonds-Karp Algorithm is defined in X-A

and runs in $O\left(VE^2\right)$. The Push-Relabel Algorithm is defined in X-B and runs in $O\left(V^3\right)$.

The Edmonds-Karp algorithm is an implementation of the Ford-Fulkerson method which takes a graph $G(V, E)$ and for an edge from vertex u to vertex v, there is a defined capacity for the edge $c(u, v)$ and $f(u, v)$. To find the maximum flow from a source s to a sink t, the Ford-Fulkerson algorithm maintains flow conservation, and the value of the flow, defined as $\sum_{(s,u)\in E} f(s, u) = \sum_{(v,t)\in E} f(v, t)$, or the flow leaving the source, must arrive at the sink. By applying this idea across all paths from source to sink, we can find the path of maximum flow. [16]

Push-Relabel max flow has two main operations, a push operation during which a flow is pushed to a local neighborhood of edges, and a relabel operation, during which edges are actually modified to maintain maximum flow paths. The push operation takes an admissible edge (u, v) and moves $\texttt{min}(x_f(u), c_f(u, v)$ flow from u to v. The relabel operation analyzes all pushes from the previous operation and applies the pushes such that sink conservation is as low as it can be. This always increases the values of the edges, and finally allows for a max flow path to be built into the network.

All of our network analysis was conducted with a software package called Graph Tool [17]

B. Evaluated Networks

1) Artifical Networks: The artificial networks we evaluated were 5 layer fully connected ANNs. The network was trained on the task of converting a binary value of length 10 to a decimal value [18]. For example, 1010101001 to 681. We chose ANNs as our example of an artifical network over something like an organizational hierarchy because ANNs can be designed very precisely, have very clear performance metrics and are quick to develop.

Three networks of this class were created with a constant set of nodes: Narrow, Constant, and Wide. Each network has the same amount of input (10) and output nodes (1). The differences in the network were the number of nodes in each hidden layer and as a result the number of edges in each graph. The **Narrow Net** *(as shown in Figure 1)* starts wide in the first hidden layer with 25 nodes and progressively gets narrower by five nodes per layer for four layers ($10 \rightarrow 25 \rightarrow 20 \rightarrow 15 \rightarrow 10 \rightarrow 1$).

The **Constant Width Net** (as shown in Figure 2) starts wide in the first hidden layer with 18 nodes and stays that width for four layers ($0 \rightarrow 18 \rightarrow 18 \rightarrow 18 \rightarrow 18 \rightarrow 1$).

Finally, the **Widening Width Net** *(as shown in Figure* 3) starts wide in the first hidden layer with 10 nodes and gets wider by five nodes for four layers ($10 \rightarrow 10 \rightarrow 15 \rightarrow 20 \rightarrow 25 \rightarrow 1$).

All of our networks were implemented in and trained with PyTorch [19] with 150 iterations of a dataset of binary and decimal digits from 1 to 1024.

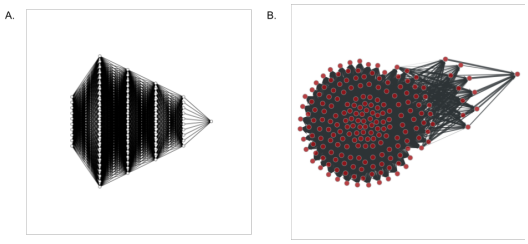

Figure 1. A. Narrowing neural network architecture B. Graph visualization of the narrowning neural network

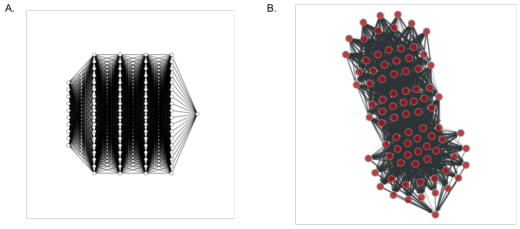

Figure 2. A. Constant width neural network architecture B. Graph visualization of the constant width neural network

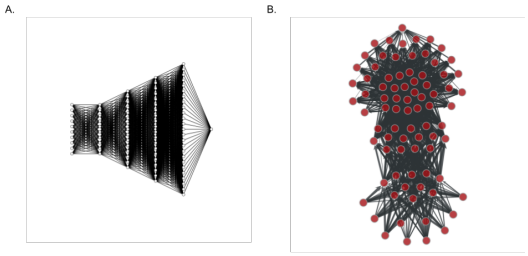

Figure 3. A. Widening neural network architecture B. Graph visualization of the widening neural network

As a cursory performance evaluation to inform the discussion below we also implemented MLPs with the same type of layer scaling for the MNIST task [20]. The architectures we used were:

- **Narrowing Network:** ($784 \rightarrow 800 \rightarrow 600 \rightarrow 400 \rightarrow 200 \rightarrow 10$) Accuracy: 85% on MNIST
- **Constant Width Network:** ($784 \rightarrow 500 \rightarrow 500 \rightarrow$

$500 \rightarrow 500 \rightarrow 10$) Accuracy: 58% on MNIST

- **Widening Network:** ($784 \rightarrow 200 \rightarrow 400 \rightarrow 600 \rightarrow 800 \rightarrow 10$) Accuracy: 61% on MNIST

This leads us to consider the narrowing network as the better collaborating network of the three.

2) Naturally Occurring Networks: The two naturally occurring networks were used were an airport connection network, where directed edges show connections between airports [21], and the weights of the edges represent the amount of traffic between the nodes. The second network is an ecology network [21], where nodes are species in an environment, edges represent dependence of one species on another, and the weight is the amount of dependence of one species on another. We see the airport network as a more collaborative network as flights must share common resources (gates, runways, airspace) and work together to get people where they are going, as opposed to the ecology network which describes the interactions between animals (i.e. which animals each which others)

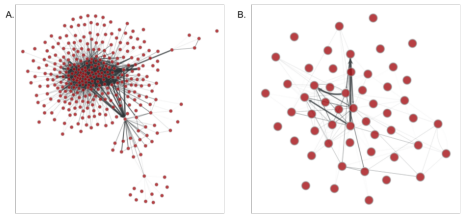

Figure 4. Visualization of the two natural networks evaluated in this study. A. is the Airport connection network. B. is the ecological network.

3) Random Networks : Random networks were built using a Poisson random distribution that determined both the degree of each node, and the weight of each is defined as the inverse of the Euclidean distance between two nodes in a Delaunay triangulation layout. We use this network to represent an optimized collaborative network.

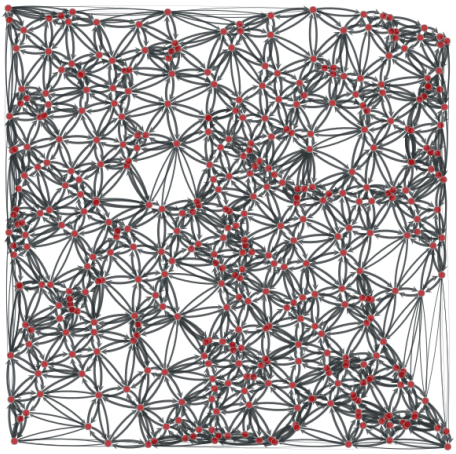

Figure 5. An example of a random network analyzed in the study

IV. Results

A. Betweenness Centrality and Bottlenecking

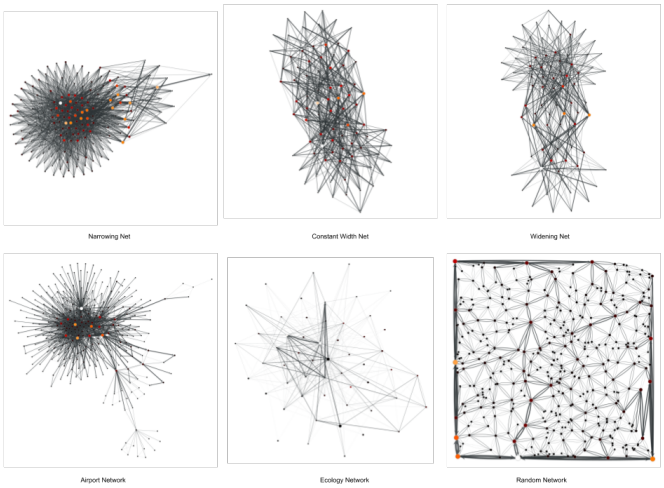

Figure 6. Vertex and Edge Betweenness for the networks analyzed in this work. From the top left: Narrowing NN, Constant Width NN, Widening NN, Airport Network, Ecology Network and Random Network

As can be seen from Figure 7, most nodes in the tested networks have a very low betweenness score, indicating that there are only a few nodes through which many paths pass. This means that there is a definite bottlenecking phenomenon that is occurring in the tested networks, indicating a few nodes control most of the flow through the network, i.e. that that a few nodes can control the flow of information.

We also see that the betweeness edge distribution for the two networks we see as the best collaboration networks have smaller variance, signifying that most flow is unconstained but there are a few nodes that see slightly more flow.

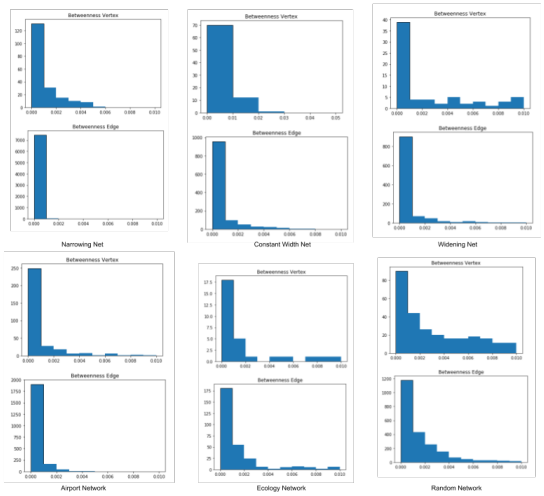

Figure 7. Edge and Betweenness distributions for the six evaluated networks. From the top left: Narrow NN, Constant Width NN, Widening NN, Airport network, Ecology network and random network

B. PageRank

We see when evaluating the PageRank centrality distribution (Figure 9) that our two good collaborative networks have a mean and mode much lower than the other networks. This further reinforces the idea put forward by the betweenness centrality that there are a few important nodes but these nodes do not dominate all flow in the network, but may nudge the result significantly. In addition, visual inspection (Figure 8) showed that similar bottlenecking structures existed within all the analyzed networks, showing that flow through the networks had similar dynamics.

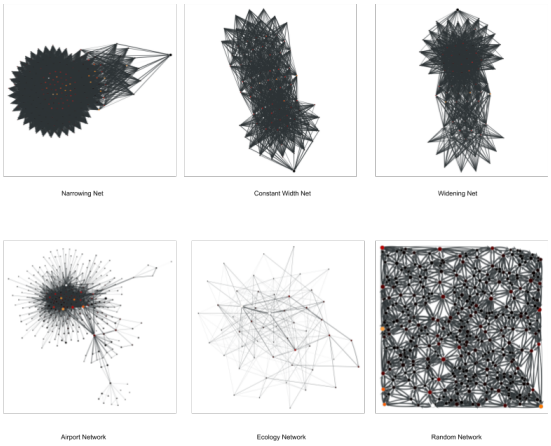

Figure 8. Visualization of the PageRank of nodes in the graphs analyzed. From the top left: Narrowing NN, Constant Width NN, Widening NN, Airport Network, Ecology Network and Random Network

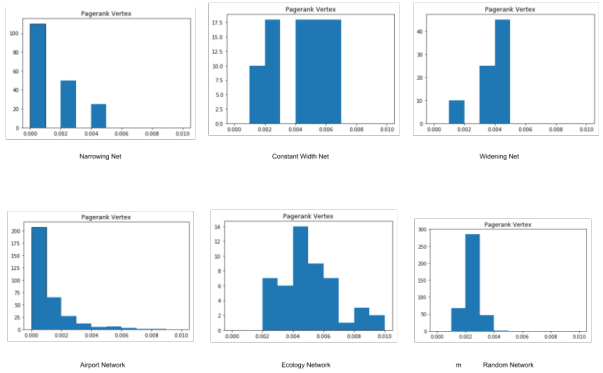

Figure 9. PageRank distribution for the nodes in the networks we analyzed. From the top left: Narrowing NN, Constant Width NN, Widening NN, Airport Network, Ecology Network and Random Network

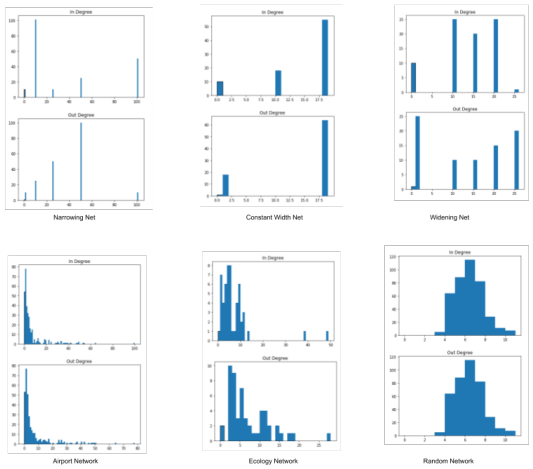

Figure 10. In and out degree distributions for the networks analyzed in this study. From the top left: Narrowing NN, Constant Width NN, Widening NN, Airport Network, Ecology Network and Random Network

C. In/Out Degree Distributions

The In/Out Degree distributions for the analyzed networks varied quite a bit, but this was beneficial as it showed that even though the structure of the networks differed, the other metrics remained constant between the networks. This shows that our flow analysis, betweenness analysis, and PageRank analysis can be verified between networks of various structures as well.

D. All Pairs Max Flows

After all flows were computed, the maximum of these max flows was graphed for analysis. A curious result was gained from this analysis, as we found that networks naturally tended to narrow as they went from source to sink instead of

spreading to utilize the extra capacity given to the network. This means that information flow is condensed from a wide array of nodes, into just a few nodes, usually ones that can be identified through PageRank and Betweenness.

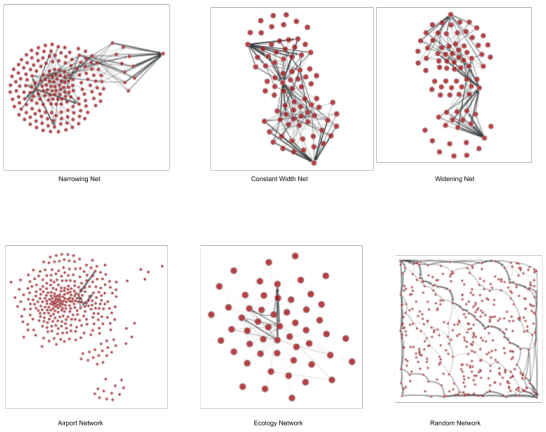

Figure 11. All pairs max flow for the networks analyzed in this study. From the top left: Narrowing NN, Constant Width NN, Widening NN, Airport Network, Ecology Network and Random Network

V. Discussion

From these results, we can infer a few ideas. Centrality is a important sign that separates more effective networks from less effective ones. We see in both Betweenness and PageRank a tendency for most nodes to have a low centrality, and a few nodes to have a slightly higher centrality as opposed to the other networks where the mean PageRank centrality and the variance in the betweenness centrality is larger. The exact nature of centrality and its role in effective networks is an area for future study and it would be good to see if there is a way to separate the correlation between PageRank and betweenness centrality before we can purpose a particular course of action for network designers. This leads us back to evidence found in the literature about the role of bottlenecks. We saw that effective networks do not bottleneck a large amount flow through a few nodes, however they do select a few nodes to be more "important" than the rest. While this is not what we were expecting, it still is consistent with what is shown in Radford et al. [8] and Guimar et al. [10] where there is assignment "main predictors and correction factors."

In addition to this, we saw through the maximum flow path of the networks, that a network narrows, as it goes from source to sink naturally. This was a surprising but welcomed result as it serves to confirm some of the intuition applied to neural network design. We see that instead of leveraging the capacity provided later in the networks, the networks continue to condense. This gives designer a metric to work with, which is to make the hidden layer before the output layer has enough capacity to encode the representation of the input data and make the earlier hidden layers wider so that the network may condense to a larger and more precise encoding. The narrowing of the network also explains the larger varience in betweeness centrality and the larger mean and mode of PageRank.

Combined, these results show that perhaps there are qualities about the topology of effective networks that distinguish them from less effective counter parts and a promising platform for future work.

VI. Limitations of This Work

There are clear limitations of this work mostly caused by the lack of computational power available. We were not able to test a network representing the null hypothesis (a random DAG [22]) for this experiment (our runtime kept crashing when trying to run analysis on it). We would also like to have run the full suite of metrics on a full sized neural network (e.g. the MNIST network we describe in Section III-B1) instead of a toy one and on large scale natural analogs like the mouse visual cortex connectome. In order to do so we would most likely need to implement a more powerful analysis suite using something like a compute cluster or GPU computing. We also only examined 4 classes of networks, to make a larger claim about the nature of collaborative networks we would need to analyze many more.

VII. Conclusion

In this work we set out to find a set of topological metrics that could help inform network designers on ways to construct their collaborative networks. We identified betweenness centrality, and PageRank centrality as two measures worth looking at. In addition we also mapped the distribution of the in and out degrees of the networks we analyzed and we also determined the all pairs max flow of the networks.

In doing so we saw that centrality is a metric which differentiates effective networks from less effective ones, where networks with low variance for betweenness centrality, and low mean and mode for PageRank do better. This leads us to believe that centrality is important in the effectiveness of these networks and that highly bottlenecked networks are not as effective as networks which are slightly less bottlenecked. We saw evidence to support the findings of Radford et al. and Guimar et al. that there are nodes that do much of the work and others that serve to nudge the result slightly. We also saw that these networks have a tendency to narrow flow as they move further from the source, explaining why the varience and mode and mean for betweennes centrality and PageRank centrality respectively are higher in less performant networks. Because of this flow result some of the intuition commonly applied to network design, that being of start wide and narrow/condense decision making is supported. Therefore we recommend with designing at least feed-forward networks to start by determining the necessary capacity for the output layer then making each preceding layer progressively wider. We think that this work may signal that there is potential to add formalism to collaborative network design.

VIII. Future Work

We think there is a lot of potential for future work in this direction. First of all evaluating more networks specifically ANNs where every network is ensured that it has at least the

capacity of a performant network at each hidden layer, but in general more collaborative networks is important to confirm the result. It would be nice to see if there are cases where the betweeness and PageRank centrality metrics disagree to determine what actually matters in the network, the flow or the number and "quality" of connections. We would also like to evaluate more complicated and more varied networks like the nature of skip connections in layered systems or investigating ideas like churn in collaborative networks (i.e. dropout, job change etc.). There are organizational hierarchy datasets out there like those assembled by the US Government, however a suitable performance metric should be determined for these networks before analyzing them. The main variable we altered in our artificial networks, the width of hidden layers had a large impact on the number of edges in the network, so analyzing constant edge networks would be interesting. There are other metrics to measure like cohesion indices [23] and the fractal scaling found in self organizing networks [24] and it would be interesting to compare the qualitative findings of this study with the measure of algebraic topological capacity shown in Guss et al. [25] We would also be interested in looking at developing or applying a notion of role centrality to analyzing these networks. There also interesting applications of this work, including identifying nodes susceptible to adversarial attacks and designing mitigation factors, intelligent dropout for the training of neural networks in addition to applying the lessons learned to collaborative network design.

IX. Acknowledgements

We would like to acknowledge Professor Lav Varshney for teaching the course on Network Science and for the advice given in conducting this work. We would also like to acknowledge the other students of ECE498LV whose helpful suggestions helped us refine this work's direction.

X. Appendix A

A. Edmonds-Karp Algorithm for Max Flow

@default

```
algorithm EdmondsKarp(G, s, t) -> (F)
    // input:
    //    G   (G[v] should be the list of edges coming out of vertex v.
    //             Each edge should have a capacity, flow, source and sink as parameters,
    //             as well as a pointer to the reverse edge.)
    //    s       (Source vertex)
    //    t       (Sink vertex)
    // output:
    //    F    (Value of maximum flow)

    F := 0   (Initialize flow to zero)
    do
        // Run a bfs to find the shortest s-t path.
        // We use 'pred' to store the edge taken to get to each vertex,
        // so we can recover the path afterwards)
        q := queue()
        q.push(s)
        pred := array(graph.length)
        while not empty(q)
            cur := q.poll()
            for Edge e in graph[cur]
                if pred[e.t] = null and e.t [?] s and e.cap > e.flow
                    pred[e.t] := e
                    q.push(e.t)

        if not (pred[t] = null)
            // We found an augmenting path.
            // See how much flow we can send
            df := [?]
            for (e := pred[t]; e [?] null; e := pred[e.s])
                df := min(df, e.cap - e.flow)
            (And update edges by that amount)
            for (e := pred[t]; e [?] null; e := pred[e.s])
                e.flow  := e.flow + df
                e.rev.flow := e.rev.flow - df
            flow := flow + df

    while pred[t] is not null  //(i.e., until no augmenting path was found)
    return flow
```

Listing 1. Edmonds-Karp Algorithm

B. *Push-Relabel Algorithm for Max Flow Implementation*

@default

```
def relabel_to_front(C, source, sink):
    n = len(C) # C is the capacity matrix
    F = [[0] * n for _ in xrange(n)]
    # residual capacity from u to v is C[u][v] - F[u][v]

    height = [0] * n # height of node
    excess = [0] * n # flow into node minus flow from node
    seen   = [0] * n # neighbours seen since last relabel
    # node "queue"
    nodelist = [i for i in xrange(n) if i != source and i != sink]

    def push(u, v):
        send = min(excess[u], C[u][v] - F[u][v])
        F[u][v] += send
        F[v][u] -= send
        excess[u] -= send
        excess[v] += send

    def relabel(u):
        # find smallest new height making a push possible,
        # if such a push is possible at all
        min_height = [?]
        for v in xrange(n):
            if C[u][v] - F[u][v] > 0:
                min_height = min(min_height, height[v])
                height[u] = min_height + 1

    def discharge(u):
        while excess[u] > 0:
            if seen[u] < n: # check next neighbour
                v = seen[u]
                if C[u][v] - F[u][v] > 0 and height[u] > height[v]:
                    push(u, v)
                else:
                    seen[u] += 1
            else: # we have checked all neighbours. must relabel
                relabel(u)
                seen[u] = 0

    height[source] = n # longest path from source to sink is less than n long
    excess[source] = [?] # send as much flow as possible to neighbours of source
    for v in xrange(n):
        push(source, v)

    p = 0
    while p < len(nodelist):
        u = nodelist[p]
        old_height = height[u]
        discharge(u)
        if height[u] > old_height:
            nodelist.insert(0, nodelist.pop(p)) # move to front of list
            p = 0 # start from front of list
        else:
            p += 1

    return sum(F[source])
```

Listing 2. Python Push-Relabel Implementation

REFERENCES

[1] Q. Hao, H. Kasper, and J. Muehlbacher, "How does organizational structure influence performance through learning and innovation in Austria and China," *Chinese Management Studies*, vol. 6, no. 1, pp. 36–52, 2012.

[2] L. M. Camarinha-Matos, H. Afsarmanesh, N. Galeano, and A. Molina, "Collaborative networked organizations–Concepts and practice in manufacturing enterprises," *Computers & Industrial Engineering*, vol. 57, no. 1, pp. 46–60, 2009.

[3] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[4] X. Zhang, V. Venkatesh, and S. A. Brown, "Designing collaborative systems to enhance team performance," *Journal of the Association for Information Systems*, vol. 12, no. 8, p. 556, 2011.

[5] R. E. Smith, E. Carraher, and P. DeLisle, "Maintaining Collaborative Teams," *Leading Collaborative Architectural Practice*, pp. 77–86, 2017.

[6] J. R. Detert and E. R. Burris, "Leadership behavior and employee voice: Is the door really open?" *Academy of management journal*, vol. 50, no. 4, pp. 869–884, 2007.

[7] H. Yu, P. M. Kim, E. Sprecher, V. Trifonov, and M. Gerstein, "The importance of bottlenecks in protein networks: correlation with gene essentiality and expression dynamics," *PLoS computational biology*, vol. 3, no. 4, p. e59, 2007.

[8] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *arXiv preprint arXiv:1704.01444*, 2017.

[9] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," *arXiv preprint arXiv:1412.6856*, 2014.

[10] P. Guimar, C. McGreavy *et al.*, "Flow of information through an artificial neural network," *Computers & chemical engineering*, vol. 19, pp. 741–746, 1995.

[11] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[12] J. M. Anthonisse, "The rush in a directed graph," *Stichting Mathematisch Centrum. Mathematische Besliskunde*, no. BN 9/71, 1971.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[14] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM (JACM)*, vol. 19, no. 2, pp. 248–264, 1972.

[15] A. V. Goldberg, *A new max-flow algorithm*. Laboratory for Computer Science, Massachusetts Institute of Technology, 1985.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms second edition," 2001.

[17] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014. [Online]. Available: http://figshare.com/articles/graph_tool/1164194

[18] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" *arXiv preprint arXiv:1610.04161*, 2016.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] R. A. Rossi and N. K. Ahmed, "The Network Data Repository with Interactive Graph Analytics and Visualization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. [Online]. Available: http://networkrepository.com

[22] B. Karrer and M. E. Newman, "Random graph models for directed acyclic networks," *Physical Review E*, vol. 80, no. 4, p. 046110, 2009.

[23] J. Belau, "Consequences of connection failure-centrality and the importance for cohesion," 2014.

[24] P. J. Laurienti, K. E. Joyce, Q. K. Telesford, J. H. Burdette, and S. Hayasaka, "Universal fractal scaling of self-organized networks," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 20, pp. 3608–3613, 2011.

[25] W. H. Guss and R. Salakhutdinov, "On Characterizing the Capacity of Neural Networks using Algebraic Topology," *arXiv preprint arXiv:1802.04443*, 2018.