BRGAN: Generating Graphs of Bounded Rank

William Shiao  
University of California Riverside  
wshia002@ucr.edu

Evangelos E. Papalexakis  
University of California Riverside  
epapalex@cs.ucr.edu

1 Extended Abstract

Introduction: Graph generation is a common task, and many models exist for this task. Traditional statistical methods (such as the Barabási–Albert model [1]) usually attempt to model specific attributes of the graph. While this can lead to generated graphs being similar to the target graph with respect to one or more attributes, it requires the user to accurately identify characteristics of the graph they wish to mimic. This issue has been addressed with the introduction of neural network models (like GraphRNN [9]), which learn directly from a set of graphs without requiring the user to explicitly declare any graph attributes.

Recent advances in the field include the use of Generative Adversarial Networks (GANs) for graph generation. A current state-of-the-art model LGGAN [3] learns to directly generate the adjacency matrices of graphs and its corresponding labels given a set of graphs as input. Alternatively, GraphRNN [9] models the graph as a sequence and uses a RNN to generate realistic graphs.

However, most existing graph generation techniques tend to produce a graph of full or high rank. While this may be sufficient for most use cases, generating a graph of a known rank can also be useful. The rank of a community graph roughly corresponds to the number of communities within the graph. Because of this, most real-world graphs are not of full rank. Bounding the rank of the generated graph therefore allows a user to generate graphs with bounded number of communities, which is difficult or impossible with existing graph generation models and can result in more realistic graphs.

Determining the rank of a matrix is computationally tractable, and can be found by computing the Singular Value Decomposition (SVD). However, the problem becomes difficult when we move to higher-order structures like time evolving graphs, since finding the rank of three-dimensional tensor is NP-hard [5]. Generating realistic data where the rank of the data is known can further promote research in developing and testing methods for approximating the rank of higher order tensors like AutoTen [6].

We propose a new model, BRGAN, that first generates factor matrices of rank at most $\rho$ and uses these factor matrices to construct the adjacency matrix. This allows us to bound the rank of the generated graphs. This work focuses on generating graphs (adjacency matrices) rather than higher-dimensional structures because graphs are easier to evaluate and the effectiveness of our bound can be evaluating using SVD.

Proposed Method & Architecture: Let the two factor matrices be denoted $A$ and $B$, where $A \in \mathbb{R}^{n \times \rho}$ and $B \in \mathbb{R}^{\rho \times n}$. Then, their product $C = AB \in \mathbb{R}^{n \times n}$ and $\text{rank}(C) \leq \min(\text{rank}(A), \text{rank}(B))$. By definition, $\text{rank}(A) \leq \min(n, \rho)$ and $\text{rank}(B) \leq \min(n, \rho)$. If we define $\rho$ such that $\rho \leq n$, we know that $\text{rank}(A), \text{rank}(B) \leq \rho$. Then, $\text{rank}(C) \leq \rho$. We exploit this property to generate a graph of bounded rank. Note that this approach does not guarantee that $C$ is of exactly rank $\rho$ because the vectors in $A$ and $B$ are not guaranteed to be linearly independent.

A GAN consists of two main models: the generator $G$ and the discriminator $D$. The generator maps a sample from a latent space ($z$) to an adjacency matrix $M$. The discriminator takes an adjacency

Preprint. Under review.
matrix $\mathbf{M}$ and outputs the probability $p$ of a sample being fake. These two models are then trained in unison in the hope that the generator will improve at generating realistic graphs and the discriminator will improve in identifying fake graphs. We use the CT-GAN [8] framework, which improves upon WGAN-GP [4], which is in turn an improvement on WGAN [2].

$\mathcal{G}$ consists of a multi-layer perceptron (MLP) network followed by two MLP networks. The two MLP networks take the output of the first MLP network and use it to generate the factor matrices. This reduces the number of parameters and results in faster convergence and better performance when compared to a single MLP network.

$\mathcal{D}$ accepts an $n \times n$ adjacency matrix and consists of several Graph Convolutional Network (GCN) layers followed by a single fully-connected layer. We take advantage of the residual GCN connections by performing a max pool across all of the GCN layer outputs. As Fan and Huang [3] found, the network performs better with these residual connections.

**Experimental Evaluation:** For this work, we performed a cursory evaluation of our model using the Citeseer citation graph dataset [7]. We split the graph into a small dataset, which consists of 2-egonets and 3-egonets with 30-50 nodes. While BRGAN’s discriminator uses a GCN and is node permutation invariant, BRGAN’s generator is affected by node ordering because it uses a MLP. As such, we use the approach used by You et al. [9] and Fan et al. [3] and generate all possible BFS orderings of the graph. This allows us to only have $n^2$ permutations per graph rather than the full $n!$ possible permutations. We then train on this augmented dataset.

We evaluated the effectiveness of our bound on the rank of the generated graphs. As theoretically expected, we found that our bound was effective and the rank of generated matrices were equal to or less than $\rho$. We also found that as $\rho$ increases, the median rank of the generated graphs generally also increases. However, the median rank of the generated matrices is closer to $\rho$ when $\rho$ is small. This means that $\rho$ becomes a tighter bound on the true rank of the generated graphs as $\rho \rightarrow 1$.

**Table 1:** Evaluation results for the small Citeseer dataset. Note that the numbers for other models are the ones reported by Fan and Huang [3].

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Deg.</th>
<th>Clust.</th>
<th>Orbit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erdos-Renyi</td>
<td>0.63</td>
<td>0.86</td>
<td>0.12</td>
</tr>
<tr>
<td>Barabási–Albert</td>
<td>0.37</td>
<td>0.18</td>
<td>0.11</td>
</tr>
<tr>
<td>MMSB</td>
<td>0.17</td>
<td>0.5</td>
<td>0.11</td>
</tr>
<tr>
<td>DeepGMG</td>
<td>0.27</td>
<td>0.36</td>
<td>0.2</td>
</tr>
<tr>
<td>GraphRNN</td>
<td>0.19</td>
<td>0.2</td>
<td>0.39</td>
</tr>
<tr>
<td>LGGAN</td>
<td>0.17</td>
<td>0.13</td>
<td>0.04</td>
</tr>
<tr>
<td>BRGAN ($\rho = 10$)</td>
<td>0.08</td>
<td>0.18</td>
<td>0.12</td>
</tr>
<tr>
<td>BRGAN ($\rho = 15$)</td>
<td>0.11</td>
<td>0.22</td>
<td>0.08</td>
</tr>
<tr>
<td>BRGAN ($\rho = 20$)</td>
<td>0.07</td>
<td>0.23</td>
<td>0.08</td>
</tr>
<tr>
<td>BRGAN ($\rho = 30$)</td>
<td>0.09</td>
<td>0.28</td>
<td>0.07</td>
</tr>
<tr>
<td>BRGAN ($\rho = 50$)</td>
<td>0.03</td>
<td>0.27</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Figure 1: BRGAN architecture. $\mathcal{G}$ is on the left and $\mathcal{D}$ is on the right.
References