Graphlet Count Estimation via Convolutional Neural Networks

Xutong Liu, Yu-Zhen Janice Chen, John Lui, Konstantin Avrachenkov

Graphlet Count Estimation via Convolutional Neural Networks

Xutong Liu[1]*, Yu-Zhen Janice Chen[1]*, John C.S. Lui[1], and Konstantin Avrachenkov[2]

[1] The Chinese University of Hong Kong [2] Inria Sophia Antipolis, France

1 Introduction. Graphlets are defined as k-node connected induced subgraph patterns. For an undirected graph, 3-node graphlets include close triangle (△) and open triangle (⟑). When $k = 4$, there are six different types of graphlets, e.g., tailed-triangle (⟒) and clique (⧆) are two possible 4-node graphlets. The number of each graphlet, called graphlet count, is a signature which characterizes the *local network structure* of a given graph. Graphlet count plays a prominent role in network analysis of many fields, most notably bioinformatics [4] and social science [3].

However, enumerating exact graphlet count is inherently difficult and computational expensive because the number of graphlets grows exponentially large as the graph size and/or graphlet size k grow [3]. To deal with this difficulty, many sampling methods were proposed for estimating graphlet count with bounded error [2, 3, 5]. Nevertheless, these methods require large number of samples to be statistically reliable, which is still computationally demanding. Moreover, they have to repeat laborious counting procedure even if a new graph is similar or exactly the same as previous studied graphs.

Intuitively, learning from historic graphs can make estimation more accurate and avoid many repetitive counting to reduce computational cost. Based on this idea, we propose a *convolutional neural network* (CNN) framework and two *preprocessing techniques* to estimate graphlet count.[1] Extensive experiments on two types of random graphs and real world biochemistry graphs show that our framework can offer substantial speedup on estimating graphlet count of new graphs with high accuracy.

2 Method. Given a set of undirected graphs and a particular type of k-node graphlet, our objective is to develop a CNN which will be trained using part of dataset with known graphlet counts. After training, the CNN can quickly and accurately predict graphlet counts of other unseen graph samples in the set. Our framework takes the graph adjacency matrix as input and outputs the graphlet count of the input graph. Let us define some notations for our CNN. Let $\mathbf{O}^{(l)} \in \mathbb{R}^{N^{(l)} \times N^{(l)} \times C^{(l)}}$ be the output tensor at layer l, where $l = 0, 1, 2, 3$, $N^{(l)}$ denotes the width (and height) along each channel and $C^{(l)}$ denotes the channel size. Let $\mathbf{O}^{(l)}_{i,j,t}$ be the $(i, j)^{\text{th}}$ element along the t^{th} channel. We assign $\mathbf{O}^{(0)}$ as the graph adjacency matrix. Mathematically, our CNN structure can be described as follows:

$$\mathbf{O}^{(l)}_{i,j,t} = \text{ReLU}(\mathbf{W}^{(l)}_t \cdot \mathbf{O}^{(l-1)}[i : i + H^{(l)} - 1, j : j + H^{(l)} - 1, :] + b^{(l)}_t), \quad l = 1, 2, \quad (1)$$

$$\mathbf{O}^{(3)} = \text{ReLU}(\text{Flatten}(\mathbf{O}^{(2)})^T \mathbf{W}^{(3)} + b^{(3)}). \quad (2)$$

Equation (1) corresponds to two convolution layers. Each layer applies $C^{(l+1)}$ filters over the input feature map $\mathbf{O}^{(l-1)}$, and the t^{th} filter is parameterized by a trainable 3D

*Both authors contributed equally to this work

[1] Our code is accessible at https://github.com/jjanicechen/GraphletCountEstimationCNN.git

weight tensor $\mathbf{W}_t^{(l)} \in \mathbb{R}^{H^{(l)} \times H^{(l)} \times C^{(l)}}$, where $H^{(l)}$ denotes the width (and height) of the filter. $[a:b, c:d, :]$ is a slicing function which extracts subset of elements indexing from a to b in width, c to d in height and all in channel to form a new tensor. $\cdot$ is the sum of element wise product of two tensors. After adding bias term $b_t^{(l)}$, we apply ReLU ($\max(0,x)$) as the activation function to obtain the output feature map $\mathbf{O}^{(l)}$. Equation (2) is associated with the fully connected layer. It flattens the output $\mathbf{O}^{(2)}$ into a column vector, applies $\mathbf{W}^{(3)}$, $b^{(3)}$ and ReLU to obtain the estimated graphlet count. Finally, our CNN is trained with back propagation and mean squared error as the loss function.

The above CNN structure inherits the learning power for local structural information of graphs. However, we still need to address the following challenges: (1) The input adjacency matrix is not consistent because graphs in the training set may have different sizes. (2) In practice, real world network dataset may not contain sufficient amount of graph samples for training, which will cause overfitting problem. To address these challenges, we introduce two preprocessing techniques:

Adjacency Matrix Zero Padding. To preserve edge connectivity information of all training graphs, we consider the largest graph in the training set, and use its dimension (say N) as the dimension of the input adjacency matrix ($N \times N$). For other graphs in the training set, we take each adjacency matrix and pad it with zero till we have an input matrix of dimension $N \times N$. This solves the varying input size problem.

Swapping Augmentation. To acquire sufficient data for training, we take advantage of the graph isomorphism property, where a graph can be expressed by different input adjacency matrices having the same underlying network structure. Our approach is to randomly pick indices i and j, then swap the i^{th} row with j^{th} row and i^{th} column with j^{th} column of the adjacency matrix. We can repeat the swapping operation for each graph m times to create m more training data. Analogous to flipping or rotation of images, we improve CNN's generalization ability and thus improve the accuracy of our model.

3 Data and Metric. Here, we introduce our testing datasets, benchmarking works, and evaluation metrics.

Random Graph. We synthesize datasets with two random graph models: random geometric graph (RGG) and Erdos-Renyi (ER) graph. A RGG is constructed by placing nodes uniformly at random in a unit cube and connecting two nodes by an edge if and only if their distance is within a given radius r. In a ER graph, the edge between every two nodes exists with probability p. In each synthetic dataset, we have 3000 training graphs, 300 validation graphs, and 300 testing graphs.

Empirical Network. We test on three real world biochemistry datasets: MUTAG [6], NCI1 and NCI109 [7]. MUTAG dataset contains 188 mutagenic compound graphs. NCI1 and NCI109 each has 4110 and 4127 chemical compound graphs tested on lung and ovarian cancer cells respectively. For MUTAG, we use *swapping augmentation* to increase the number of training samples. We also apply *adjacency matrix zero padding* to make all graphs in each dataset have the same size.

Benchmark. We compare CNN with three existing frameworks: GRAFT [5], CC2 [2], GUISE [1], which are based on edge sampling, color coding, and Markov Chain Monte Carlo method respectively.

Relative Error. Let c_i be the ground truth graphlet count of sample graph i, c_i' be its estimated count, and there are S samples in the dataset. We compute the mean absolute

error of the estimations, mae $= \Sigma_{i=1}^{S} |c_i' - c_i|/S$, and mean of ground truth counts, $\mu = \Sigma_{i=1}^{S} c_i/S$. We take relative error as $e = \text{mae}/\mu$.

Speed. To ensure a fair comparison, we do not choose running time as the performance metric since it highly depends on hardware and implementation (e.g. running on GPU/CPU). Instead, we measure the number of arithmetic operations they use. For CNN model, we compute the number of floating-point operations (FLOPs). For benchmarking works, we calculate the number of comparison operations in the algorithms.

4 Result. We test our framework on random graph datasets. For approximating 4-clique counts, our CNN model achieves less than 8% relative error on 50-node RGGs with radius 0.45 and less than 5% relative error on 50-node ER graphs with edge existing probability 0.5. We also train our CNN models for estimating 4-path (), 3-star (), 5-path () on the empirical biochemistry datasets. The relative errors on all three datasets are less than 20% of the ground truth counts. For estimating 4-path on MUTAG dataset, our model performs especially well making only 6% relative error.

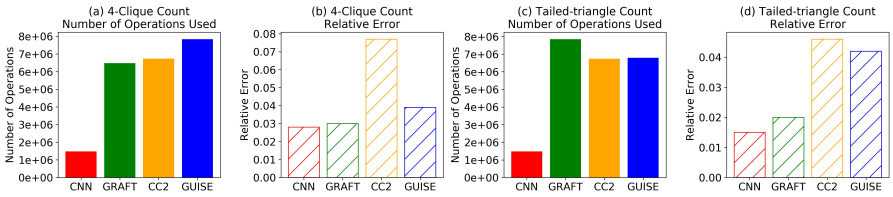

Fig. 1. Comparison of the number of arithmetic operations used for estimating 4-clique counts, tailed-triangle counts on 50-node ER graphs with edge existing probability 0.5. (a, c) The number of operations used by each framework. (b, d) The relative error each framework makes.

To compare the speed of our CNN with existing methods, the number of arithmetic operations used are calculated. For a fair comparison, we tune the number of iterations for all benchmarking sampling methods, so that they obtain as close relative errors to that of CNN as possible. Figure 1 (a, c) shows that the numbers of arithmetic operations used by GRAFT, CC2, or GUISE are significantly more than that used by CNN. This result demonstrates that our CNN based graphlet count estimation approach offers remarkable speedup on predicting graphlet counts while still maintaining high accuracy.

References

1. Bhuiyan, M. A., Rahman, M., Rahman, M., Al Hasan, M.: Guise: Uniform sampling of graphlets for large graph analysis. In: 2012 IEEE 12th ICDM. (pp. 91-100). IEEE. (2012)
2. Bressan, M., Chierichetti, F., Kumar, R., Leucci, S., Panconesi, A.: Counting graphlets: Space vs time. In: Proc. Int. Conf. Web. Search. Data. Min. (pp. 557-566). ACM. (2017)
3. Chen, X., Li, Y., Wang, P., Lui, J.: A general framework for estimating graphlet statistics via random walk. Proceedings VLDB Endowment 10(3), 253-264 (2016)
4. Prulj, N.: Biological network comparison using graphlet degree distribution. Bioinformatics, 23(2), e177-e183 (2007)
5. Rahman, M., Bhuiyan, M. A., Al Hasan, M.: Graft: An efficient graphlet counting method for large graph analysis. IEEE Trans. Knowl. Data. Eng., 26(10), 2466-2478 (2014)
6. Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., Borgwardt, K. M.: Graph kernels. J. Mach. Learn. Res, 11(Apr), 1201-1242 (2010)
7. Wale, N., Watson, I. A., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. KAIS, 14(3), 347-375 (2008)