Chapter 6

RESIDUAL GRAPH CONVOLUTIONAL NETWORK

6.1 Introduction

In Chapter 5, two deep auto-encoders frameworks were described - Graph Auto-Encoder with Fully Convolutional layers (G.A.E.-F.C.) and Variational Graph Auto-Encoder with Fully Convolutional layers (V.A.E.-F.C.). Experiments performed on network data-sets demonstrated that for network data-sets with features < 50, standard G.C.N. based auto-encoders such as Graph Auto-Encoder with Fully Convolutional layers (G.A.E.-F.C.) and Variational Graph Auto-Encoder with Fully Convolutional layers (V.A.E.-F.C.) achieved similar test accuracy as Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.). However, as the scale and dimensionality of the data increased the performance of Graph Auto-Encoder (G.A.E.-F.C.) and Variational Graph Auto-Encoder (V.A.E.-F.C.) was better.

The deep graph neural network architectures in the literature, as well as proposed in Chapter 5 are two layers deep. Therefore, these G.C.N. based architectures have a small depth compared to deep learning networks in image and natural language processing [35, 52]. However, to improve training accuracy and reduce the loss on data, a much deeper architecture would be required. Increasing the depth of these architectures by increasing the hidden layers causes a dramatic drop in model performance. Hence,

to build deeper models with many layers of neighbourhood aggregation without hampering training time and accuracy, Chapter 6 investigates the effects of addition of a residual connection to the Graph Convolutional Network architecture.

6.1.1 Summary of contribution

- The main contribution in this chapter is to propose a residual connection to the Graph Convolutional Network architecture.
- Extensive experiments were performed on network data-sets to validate the proposed architecture viz. Residual Graph Convolutional Network architecture for latent space representation of networks.
- The advantages of the proposed approach were demonstrated over the standard Graph Convolutional Network by performing experiments on network data-sets.

The rest of the paper is organized as follows: In Section 6.2 describes the issues in deep learning architectures for L.S.R., Section 6.3 provides a background of Residual Neural Architectures, Section 6.4 describes the performance of the proposed method and discusses the results.

6.2 Issues in Deep Learning Architectures

Theoretically, deep neural networks are accurate, and this accuracy increases with additional hidden layers [35]. However, practically in deep neural networks, the problem of vanishing or exploding gradients is observed. This hampers convergence [37]. As the depth of the networks increases, the number of parameters also increases ($\sim 10^6 - 10^7$), and the accuracy during optimization gets saturated. Adding additional layers in such a scenario affects the training accuracy while increasing the training time. The convolutional implementations in Chapter 5 address the issue related to excess parameters. However, to improve the graph convolutional network's receptive field, there is a need to increase its depth.

6.2.1 Drawbacks of the existing architectures:

- Existing G.C.N. based architectures in literature are two to three layers deep.
- Receptive field of standard G.C.N. based architectures is low.
- G.C.N. based architectures, when stacked together, were observed to have limited effect in model performance.

6.2.2 Motivation for the proposed approach

In G.C.N. based networks, a graph convolution operation aggregates node attribute information. In the first convolution layer, only information of directly connected nodes is aggregated. Hence, the receptive field of a standard G.C.N. is less. To increase the receptive field, it is necessary to transfer the activations (learned representations) of the initial convolutions to deeper layers. To achieve this, multiple graph convolution layers could be stacked in a single architecture. However, as the graph convolution is a particular form of Laplacian smoothing, stacked graph convolutions create concerns of over smoothing [33, 36, 125]. To overcome the G.C.N. model's limits with shallow architectures, the current investigation proposes a Residual Graph Convolutional Network architecture.

6.3 Residual Graph Convolutional Network

6.3.1 Background of residual blocks

For machine learning tasks, it was observed that as the neural networks' depth was increased, accuracy saturated, and with further training degraded rapidly. This problem was called vanishing or exploding gradients. It was also aggravated with increasing network depth [126]. Residual networks could avoid these problems by utilizing "shortcuts" or skip-connections in the network, as shown in Figure 6.1.

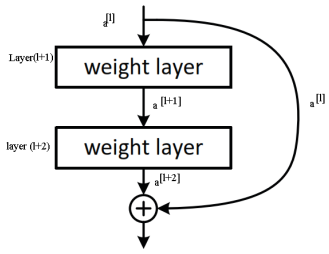

Figure 6.1: A Residual neural network architecture. The skip-connection is used to transfer the activation's to deeper parts of the network.

6.3.2 Architecture

Residual Graph Convolutional Networks architecture is shown in Figure 6.2. Input of layer $l+1$ is given by $a^{[l]}$, then the two-step activation function computation at layer $l+1$ is given by $z^{[l+1]} = W^{[l]}a^{[l]} + b^{[l+1]}$ followed by $a^{[l+1]} = g(z^{[l+1]})$. $g(.)$ is a non-linear activation function. Similarly, in layer $l+2$ the two-step activation function computation is given by $z^{[l+2]} = W^{[l+1]}a^{[l+1]} + b^{[l+2]}$ followed by $a^{[l+2]} = g(z^{[l+2]})$. The change in a residual network is the activation of previous layers $a^{[l]}$ is added to deeper layers, changing $a^{[l+2]}$ to $a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$. To avoid dimension mismatch, the activation of the previous layers $a^{[l]}$ is passed through a dense layer to resize $a^{[l]}$ for addition to $(z^{[l+2]}$. This dense layer does not apply a nonlinear activation function as its role is only to resize the input to match the dimensions of the destination. Using such residual blocks it is possible to build deep networks that avoid problems of over-fitting or vanishing/exploding gradients. During experimentation, batch normalization layer and ReLU activation were found to give optimal results.

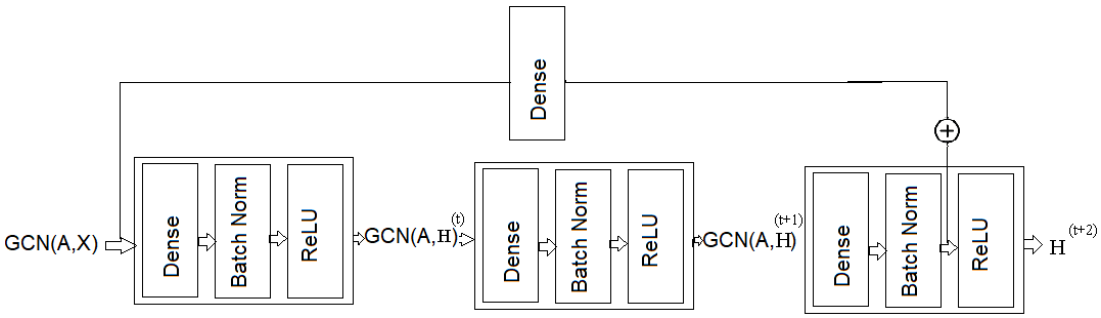

Figure 6.2: Residual graph convolutional network block

6.3.3 Understanding neighbourhood-aggregation encoder algorithm of G.C.N. and Residual G.C.N.

Algorithm 7 gives the neighbourhood aggregation strategy of a standard G.C.N.. There are two operations: AGGREGATE and COMBINE. The AGGREGATE operation adds the layer k node embedding of all nodes in the neighbourhood of node u to u. COMBINE operations applies a nonlinearity to the aggregated node embedding of node u.

Algorithm 7: neighbourhood aggregation algorithm of G.C.N. Adapted from [40]

Inputs: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\{\mathbf{W}^k, \forall k \in [1, K]\}$; non-linearity σ; differentiable aggregator functions $\{\text{AGGREGATE}_k, \forall k \in [1, K]\}$; neighbourhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Outpt : Vector embedding $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \text{NORMALIZE}(\mathbf{h}_v^k), \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

The neighbourhood aggregation algorithm of a residual G.C.N. is given in Algorithm 8. For every third layer, the AGGREGATE operation adds the layer k and layer $k-2$ node embedding of all nodes in the neighbourhood of node u to u. The complexity analysis, for an input graph having N nodes and feature vectors of length C, the adjacency matrix A will be in R^{N*N} and input feature matrix V^a will be of size R^{N*C}. For a given graph convolution layer, to obtain an output $V^{out} = R^{N*F}$, the time complexity is $O(N^2CF)$. For the residual G.C.N., the time complexity is $O(N^2CF + N^2C'F')$ where component $N'^2C'F'$ is due to the aggregation operation requiring additional computations of the

residual block.

Algorithm 8: neighbourhood aggregation algorithm of Residual G.C.N. Adapted from [40]

Inputs: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\{\mathbf{W}^k, \forall k \in [1, K]\}$; non-linearity σ; differentiable aggregator functions $\{\text{AGGREGATE}_k, \forall k \in [1, K]\}$; neighbourhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Outpt : Vector embedding $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    if $k\%3 == 0$ **then**
      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}, \mathbf{h}_{\mathcal{N}(v)}^{k-2})$;
      $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
    else
      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
      $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
    end
  end
  $\mathbf{h}_v^k \leftarrow \text{NORMALIZE}(\mathbf{h}_v^k), \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

For the residual block, it is assumed the input graph has N nodes and feature vectors of length C', the adjacency matrix A will be in R^{N*N} and input feature matrix V^a will be of size $R^{N*C'}$.

6.4 Performance of Residual Graph Convolutional Network for Network Representation Learning

The standard Graph Convolutional Network and Residual Graph Convolutional Network architectures were compared in the experimental study. These architectures were

used to convert network data of Blog-Net, Cora-Net, Cite-Net, Flickr-Net, Protein-Net, and Wiki-Net from their original dimensions into vector space of dimension = 32. The quality of the vertex embeddings is measured using distance-based statistics obtained from clustering the representations. The parameters are separation index, widest within-cluster gap, average silhouette width, the average distance between clusters, and the Dunn index. Twenty iterations are performed on vertex embeddings for calculating each performance metric. The values of mean along with limits of two standard deviations $2SD$ of the mean (upper and lower bounds of the 95% confidence interval) are given for quantitative comparison. Additionally, the architectures are compared based on their parameters and execution time on the data-sets. T Kipf *et al.* [36] implementation of G.C.N. is used. Residual G.C.N. is built using Keras package in Python [124].

6.4.1 Data-sets

Network data-sets with $|V| > 10^3$ were chosen for the evaluation of Residual G.C.N..

Table 6.1: Description of Network Data-sets with binary attributes

Description	**Blog-Net**	**Flickr-Net**	**Protein-Net**	**Wiki-net**	**Cora-net**	**Cite-net**
$\|V\|$	5196	7575	3890	4777	2708	3312
V^a	8189	12047	50	40	1434	3704
$\|E\|$	171743	239738	37845	54810	5429	4732
c	0.08	0.1	0.09	0.43	0.14	0.13

6.4.2 Architectures

The number of skip-connections used in the Residual Graph Convolutional Network architecture for BlogCatalog, CiteSeer, Cora, Flickr, Protein, and Wiki data-sets is given in Table 6.2. Dimensions of the fully connected layers used in the G.C.N. and Residual G.C.N. architectures is summarized in Table 6.3 and Table 6.4 respectively.

Table 6.2: Number of skip-connections used residual graph convolutional network architectures

Sr No	Data-set	Dense layers	Skip connections
1	Blog-Net	6	2
2	Cora-Net	6	2
3	Cite-Net	5	1
4	Flickr-Net	6	2
5	Protein-Net	3	1
6	Wiki-Net	3	1

Table 6.3: Dimensions of the fully connected layers in the G.C.N. architecture

layer name	Blog-Net	Cora-Net	Cite-Net	Flickr-Net	Protein-Net	Wiki-Net
X	5196*8189	2708*1434	3312*3704	7575*12047	3890*50	4777*40
A	5196*5196	2708*2708	3312*3312	7575*7575	3890*3890	4777*4777
dense_encode	8189*4096	1434*1024	3703*1024	12047*5196	50*128	40*128
dense_encode	4096*1024	1024*512	1024*256	5196 * 2048	128*64	128*64
dense_encode	1024*512	512*256	256*128	2048 * 1024	64*32	64*32
dense_encode	512*128	256*128	128*64	1024 * 512	-	-
dense_encode	128*64	128*64	64*32	512 * 128	-	-
dense_encode	64*32	64*32	-	128 * 32	-	-

Table 6.4: Dimensions of the fully connected layers in the Residual G.C.N. architecture

layer name	Blog-Net	Cora-Net	Cite-Net	Flickr-Net	Protein-Net	Wiki-Net
X	5196*8189	2708*1434	3312*3704	7575*12047	3890*50	4777*40
A	5196*5196	2708*2708	3312*3312	7575*7575	3890*3890	4777*4777
dense_encode	8189*4096	1434*1024	3703*1024	12047*5196	50*128	40*128
dense_encode	4096*1024	1024*512	1024*256	5196 * 2048	128*64	128*64
dense_encode	1024*512	512*256	256*128	2048 * 1024	64*32	64*32
dense_residual	4096*512	1024*256	1024*128	5196*1024	128*32	128*32
dense_encode	512*128	256*128	128*64	1024*512	-	-
dense_encode	128*64	128*64	64*32	512*128	-	-
dense_encode	64*32	64*32	-	128*32	-	-
dense_residual	128*32	128*32	-	512*32	-	-

Table 6.5 gives the number of parameters in the G.C.N. and Residual G.C.N. architectures. A higher number of parameters leads to more computations and increases the

time required to obtain the L.S.R.

Table 6.5: Number of parameters in the G.C.N. and Residual G.C.N. architectures

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	38336512	40437760
Cora-Net	2166784	2433024
Cite-Net	4096934	4228008
Flickr-Net	75895924	81233012
Protein-Net	16640	20736
Wiki-Net	15360	19456

Configuration of the system on which an experimental study was conducted is Intel(R) Core(T.M.) i5-6402P CPU@2.8GHz with four cores and 8GB DDR3 RAM. Table 6.6 gives the time for execution (in secs) of the G.C.N. and Residual G.C.N. architectures.

Table 6.6: Time for execution (in secs) of the G.C.N. and Residual G.C.N. architectures

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	5.4 (±0.36)	7.6 (±0.91)
Cora-Net	7.1 (±0.52)	9.3 (±0.8)
Cite-Net	5 (±0.67)	6.4 (±1.09)
Flickr-Net	11.2 (±0.28)	16.3 (±0.98)
Protein-Net	4.9 (±0.76)	6.2 (±1.03)
Wiki-Net	4.3 (±0.3)	6.7 (±0.28)

6.4.3 Results

The quality of the node embedding generated by the standard Graph Convolutional Network and the Residual Graph Convolutional Network architecture was evaluated using node clustering. The aim was to obtain clusters with high compactness and high separation. Table 6.7 gives the results of parameter separation index for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on separation index compared to the standard G.C.N. as mentioned: Wiki-Net (8%), Cora-Net (-1%), Cite-Net (-1%), Blog-Net (-11%), Flickr-Net (7%) and Protein-Net (-6.2%).

Table 6.7: Result of separation index of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	**1.43** (±0.061)	1.56 (±0.048)
Cora-Net	**3.15** (±0.083)	3.17 (±0.37)
Cite-Net	**4.29** (±0.49)	4.34 (±0.78)
Flickr-Net	2.84 (±0.028)	**2.68** (±0.41)
Protein-Net	**1.59** (±0.089)	1.67 (±0.078)
Wiki-Net	2.76 (±0.34)	**2.67** (±0.19)

Table 6.8 gives the results of parameter widest within-cluster gap for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on widest within-cluster gap compared to the standard G.C.N. as mentioned: Wiki-Net (6%), Cora-Net (5%), Cite-Net (-30%), Blog-Net (6%), Flickr-Net (5%) and Protein-Net (-8%).

Table 6.8: Result of widest within-cluster gap of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	2.37 (±0.059)	**2.49** (±0.084)
Cora-Net	4.61 (±0.053)	**4.82** (±0.048)
Cite-Net	**1.64** (±0.06)	1.28 (±0.16)
Flickr-Net	5.83 (±0.57)	**6.08** (±0.6)
Protein-Net	**4.29** (±0.3)	3.97 (±0.46)
Wiki-Net	6.18 (±0.51)	**6.43** (±1.02)

Table 6.9 gives the results of the parameter average silhouette width for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on average silhouette width compared to the standard G.C.N. as mentioned: Wiki-Net (-4%), Cora-Net (1%), Cite-Net (6%), Blog-Net (7%), Flickr-Net (-4%) and Protein-Net (2%).

Table 6.9: Result of average silhouette width of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	0.21 (±0.0064)	**0.34** (±0.0097)
Cora-Net	0.37 (±0.0048)	**0.39** (±0.0094)
Cite-Net	0.16 (±0.002)	**0.22** (±0.0088)
Flickr-Net	**0.43** (±0.0109)	0.38 (±0.0029)
Protein-Net	0.35 (±0.0043)	**0.38** (±0.0097)
Wiki-Net	**0.57** (±0.0046)	0.53 (±0.0046)

Table 6.10 gives the results of the parameter average distance between clusters for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on the average distance between clusters compared to the standard G.C.N. as mentioned: Wiki-Net (5%), Cora-Net (5%), Cite-Net (-68%), Blog-Net (3%), Flickr-Net (-11%) and Protein-Net (2%).

Table 6.10: Result of average distance between clusters of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	0.57 (±0.0028)	**0.62** (±0.0064)
Cora-Net	1.58 (±0.003)	**1.65** (±0.0074)
Cite-Net	**0.38** (±0.0091)	0.24 (±0.0027)
Flickr-Net	**3.16** (±0.0206)	2.94 (±0.071)
Protein-Net	3.46 (±0.26)	**3.51** (±0.4)
Wiki-Net	1.67 (±0.156)	**1.93** (±0.074)

Table 6.11 gives the results of parameter Dunn index for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on Dunn index compared to the standard G.C.N. as mentioned: Wiki-Net (-5%), Cora-Net (-5%), Cite-Net (-12%), Blog-Net (11%), Flickr-Net (4%) and Protein-Net (0.12%).

Table 6.11: Result of Dunn index of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	0.87 (±0.0059)	**0.73** (±0.0027)
Cora-Net	**0.63** (±0.0049)	0.68 (±0.0067)
Cite-Net	**0.81** (±0.0042)	0.94 (±0.004)
Flickr-Net	0.34 (±0.0087)	**0.26** (±0.0091)
Protein-Net	0.29 (±0.0082)	0.29 (±0.0014)
Wiki-Net	**0.44** (±0.0037)	0.46 (±0.0084)

6.4.4 Discussion of results and summary

Graph Convolutional Networks (G.C.N.) are capable of encoding both graph structure and node features in the vector representations. The standard G.C.N. and its variants are two layers deep. Hence, to overcome the limitations of shallow architectures, the current investigation proposes a Residual Graph Convolutional Network architecture. Adding a residual block to plain neural networks had proven useful in increasing their depth without affecting the performance. However, adding a residual block to a G.C.N. to improve its depth was not observed in the literature. In the current experimental setting, it was noted that Residual G.C.N. had comparable results with the standard G.C.N.

- For the Blog-Net data-set, the performance of Residual G.C.N. was lower by 11% on the separation index, higher by 6% on the widest within-cluster gap, higher by 7% on average silhouette width, higher by 3% on the average distance between clusters and higher by 11% on Dunn index.
- For the Cora-Net data-set, the performance of Residual G.C.N. was lower by 1% on the separation index, higher by 5% on the widest within-cluster gap, higher by 1% on average silhouette width, higher by 5% on the average distance between clusters and lower by 5% on Dunn index.
- For the Cite-Net data-set, the performance of Residual G.C.N. was lower by 1% on separation index, lower by 30% on the widest within-cluster gap, higher by

6% on average silhouette width, lower by 68% on the average distance between clusters and lower by 12% on Dunn index.

- For the Flickr-Net data-set, the performance of Residual G.C.N. was higher by 7% on the separation index, higher by 5% on the widest within-cluster gap, lower by 4% on average silhouette width, lower by 11% on the average distance between clusters and higher by 4% on Dunn index.
- For the Protein-Net data-set, the performance of Residual G.C.N. was lower by 6.2% on separation index, lower by 8% on the widest within-cluster gap, higher by 2% on average silhouette width, higher by 2% on the average distance between clusters and higher by 0.12% on Dunn index.
- For the Wiki-Net data-set, the performance of Residual G.C.N. was higher by 8% on separation index, higher by 6% on the widest within-cluster gap, lower by 2% on average silhouette width, higher by 5% on the average distance between clusters and lower by 5% on Dunn index.

Table 6.12 gives the result of the transitivity of G.C.N. and Residual G.C.N. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on transitivity compared to the G.C.N. as mentioned: Wiki-Net (12%), Cora-Net (35%), Cite-Net (39%), Blog-Net (38%), Flickr-Net (42%) and Protein-Net (27%).

Table 6.12: Result of transitivity of G.C.N. and Residual G.C.N.

Data-set	**Actual** c	**G.C.N.**	**Residual G.C.N.**
Blog-Net	0.08	0.26 (±0.064)	0.17 (±0.019)
Cora-Net	0.09	0.19 (±0.027)	0.11 (±0.02)
Cite-Net	0.13	0.24 (±0.032)	0.16 (±0.087)
Flickr-Net	0.1	0.22 (±0.096)	0.13 (±0.076)
Protein-Net	0.09	0.19 (±0.103)	0.14 (±0.084)
Wiki-Net	0.43	0.64 (±0.036)	0.57 (±0.094)

Graph convolution is a particular form of Laplacian smoothing. Multiple stacked graph convolutions in the standard G.C.N. lead to over smoothing of the vector representation.

This causes the vector representations of the original graph nodes to be mapped to close to each other in the embedded space. Hence, the reconstructed graph from a standard G.C.N. has higher transitivity than seen in the original graph. However, with the residual blocks, the initial layers' representations are added to the deeper layers. This addition overcomes, to a certain extent, the smoothing effect of the graph convolution. Therefore, the reconstructed graph's transitivity is lower in the Residual G.C.N compared to the standard G.C.N.

6.5 Publications

Discussions and results in this chapter are communicated in:

6.5.1 Journals

1. Nerurkar, P., Chandane, M. and Bhirud, S., (2019). Investigations of Residual Graph Convolutional Network for Representation Learning. Neural Computing and Applications. (Springer)