GRAPH NEURAL NETWORK FOR TEXT CLASSIFICATION

FINAL EVALUATION

SWYAM PRAKASH SINGH

M.Tech, IISc

Problem Statement:

To use Graph Attention Network for text classification.

Task done in first evaluations:

- Using Text_GCN ,graph is constructed from documents provided.
- uses the same approach as in Text_GCN, weights are assigned to edges.

$$A_{ij} = \begin{cases} \text{PMI}(i,j) & i,j \text{ are words, PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

Source:Text_GCN by Yao et al 2019

Dataset used :

Dataset	R8	R52	ohsumed	MR
training	4937	5879	3022	6398
test	2189	2568	4043	3554
nodes	15362	17992	21557	29426
sentence length	65.72	69.8	135.82	20.3

Main Intuition

- Edge weights are considered by taking GCN as first layer.
- We have got first feature aggregation(weighted average) of nodes, which will be input to Attention layer.
- Most of weights in adjacency matrix are very small.
- So main aim is to refine this adjacency matrix by putting threshold, which can be learnt while training.

- Thinking behind this logic is, there are some edges which has very low weight and can act as noise.
- So by thresholding, it can be corrected.

GCN Propagation Rule:

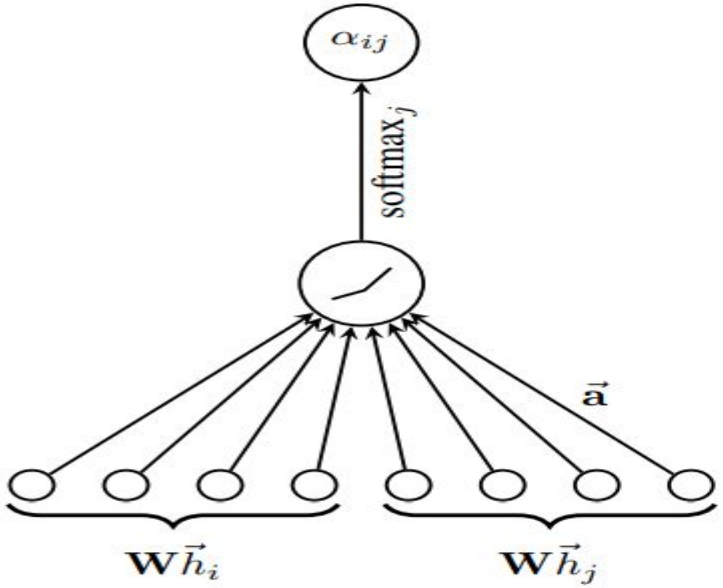
$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

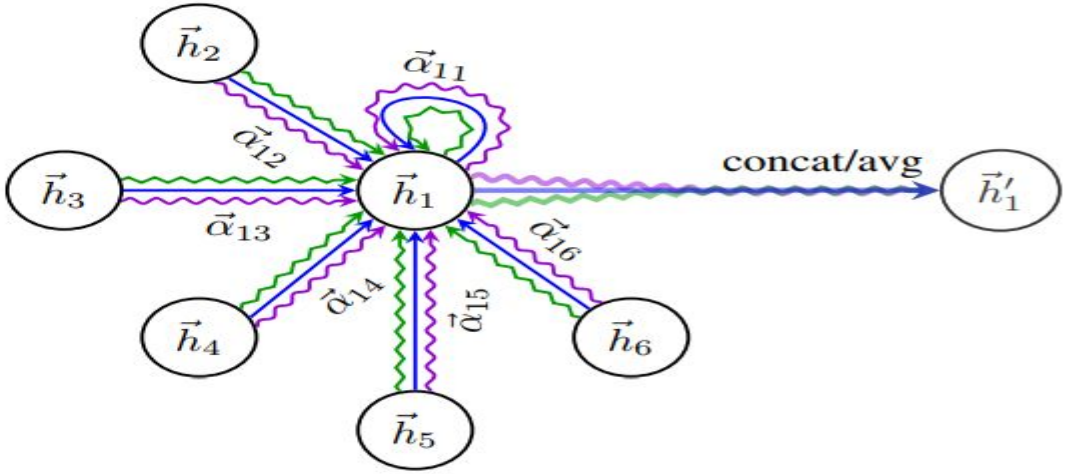
with $\hat{A} = A + I$, where A is Adjacency Matrix and $\hat{D}$ is degree of augmented Adjacency matrix.

Source:GCN,Kipf et al,2016

Graph Attention Network Propagation Rule:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$





Source:GAT,Velicovic et al 2018

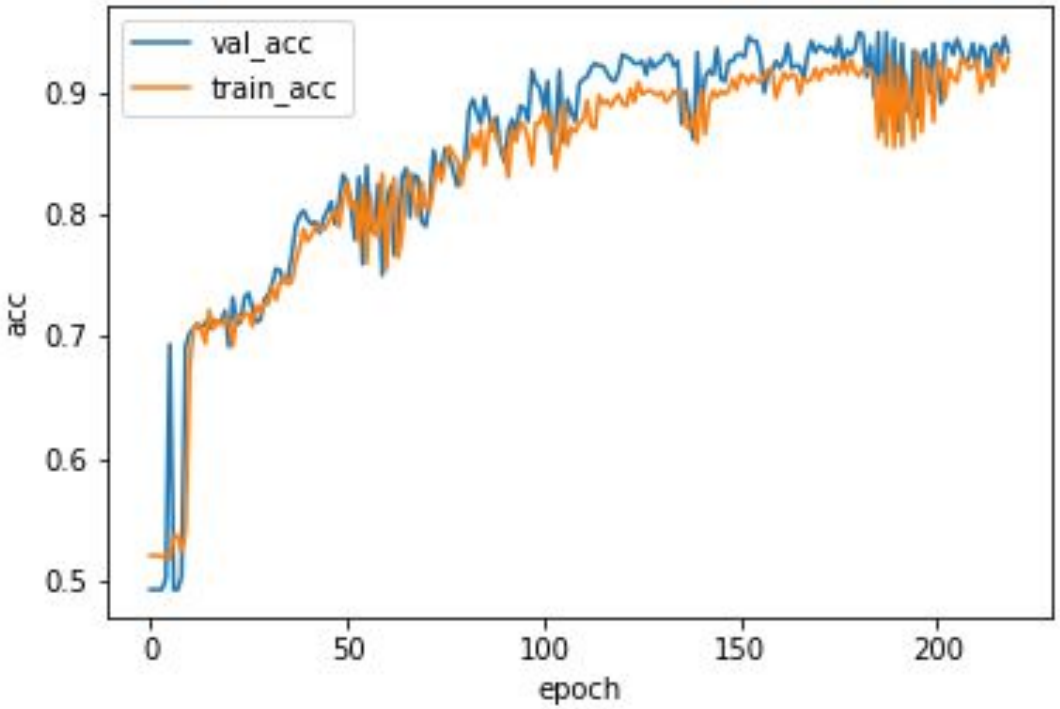
What I tried to include weights of edges:

- Different combination are:
  - single layer of GCN with single layer of attention.
  - double layer of GCN with single layer of attention.
  - single layer of GCN with double layer of attentions

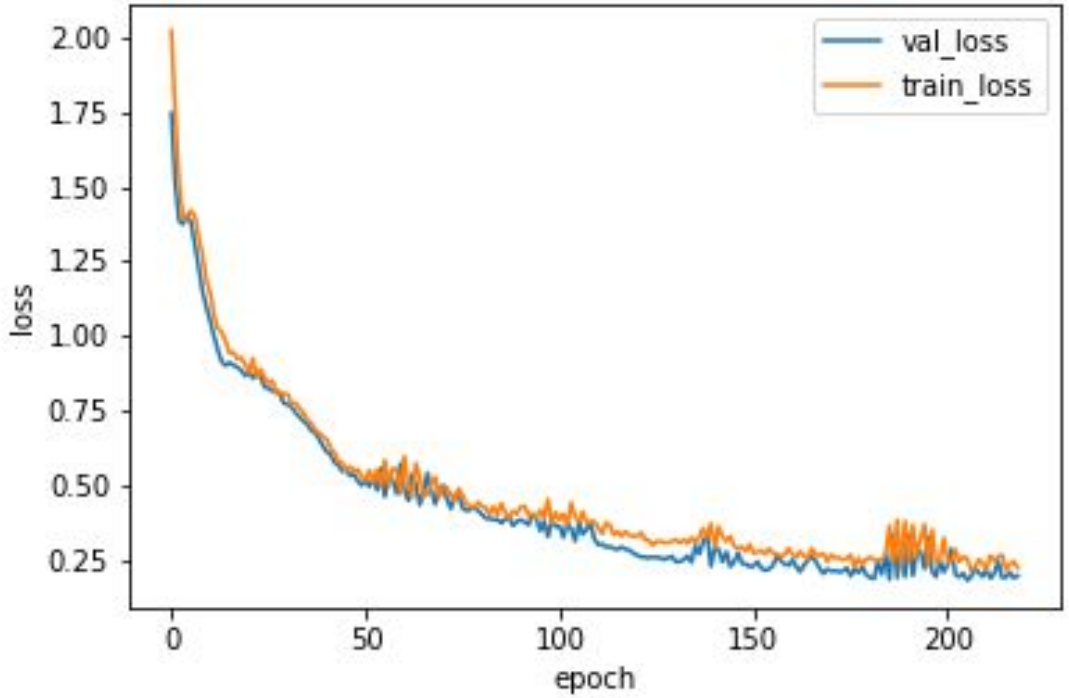
Results and Analysis:

- In first evaluation i tried with single and double attention layer.
- After analysis, i found that results are bad due to not considering weight of edges in smoothing.
- Most of results are on R8 dataset.

Results for first combinations:1GCN+1GAT



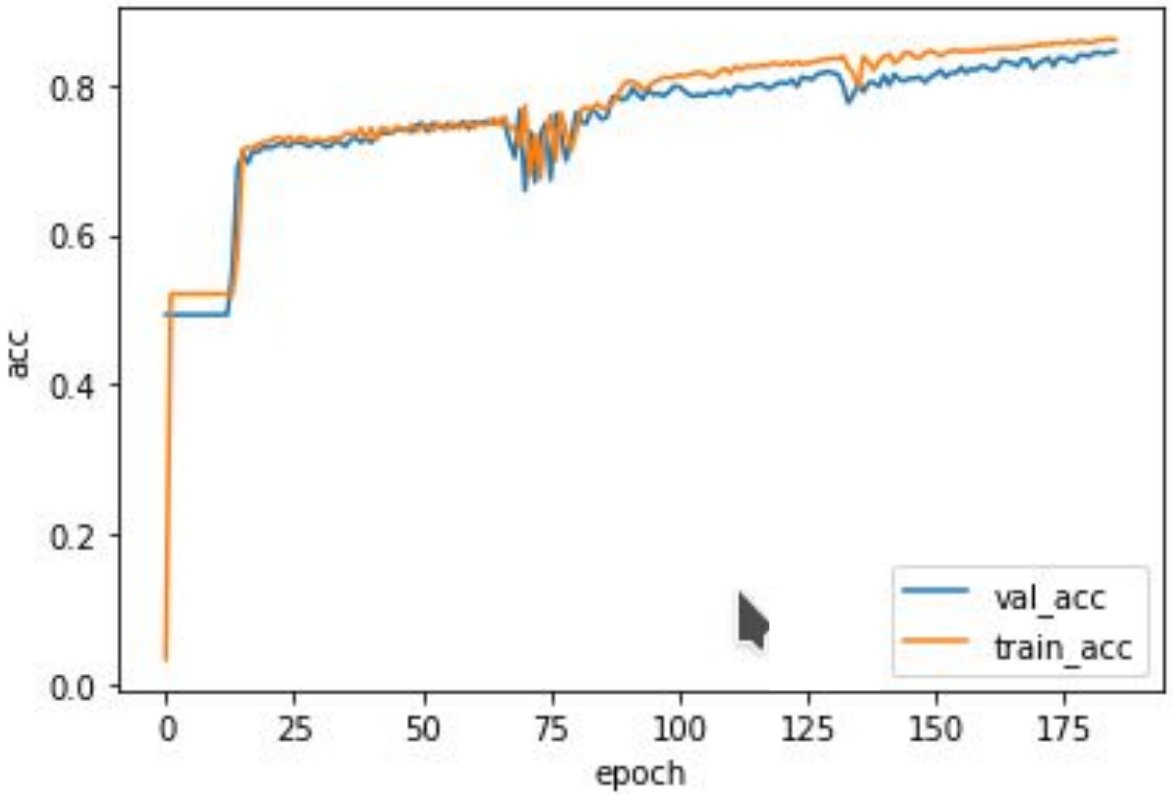
Accuracy plots



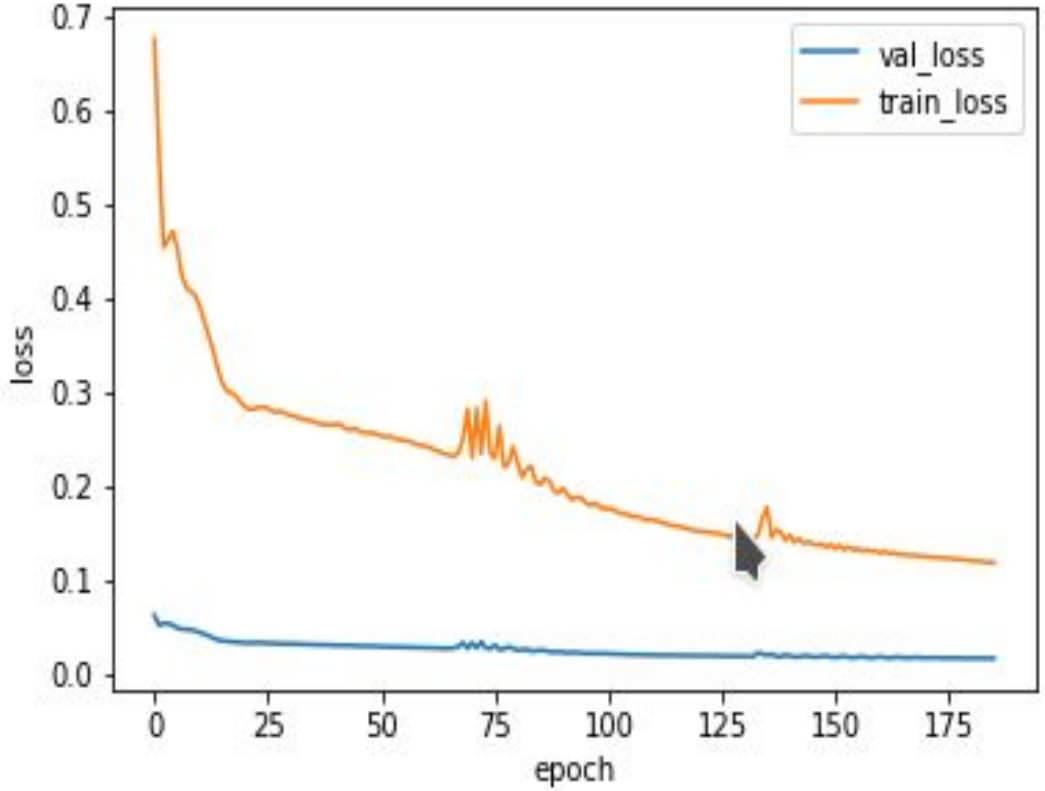
Loss Plots

```
Loading 217th epoch
Test set results: loss= 0.1780 accuracy= 0.9429
```

Results for first combinations:2GCN+1GAT



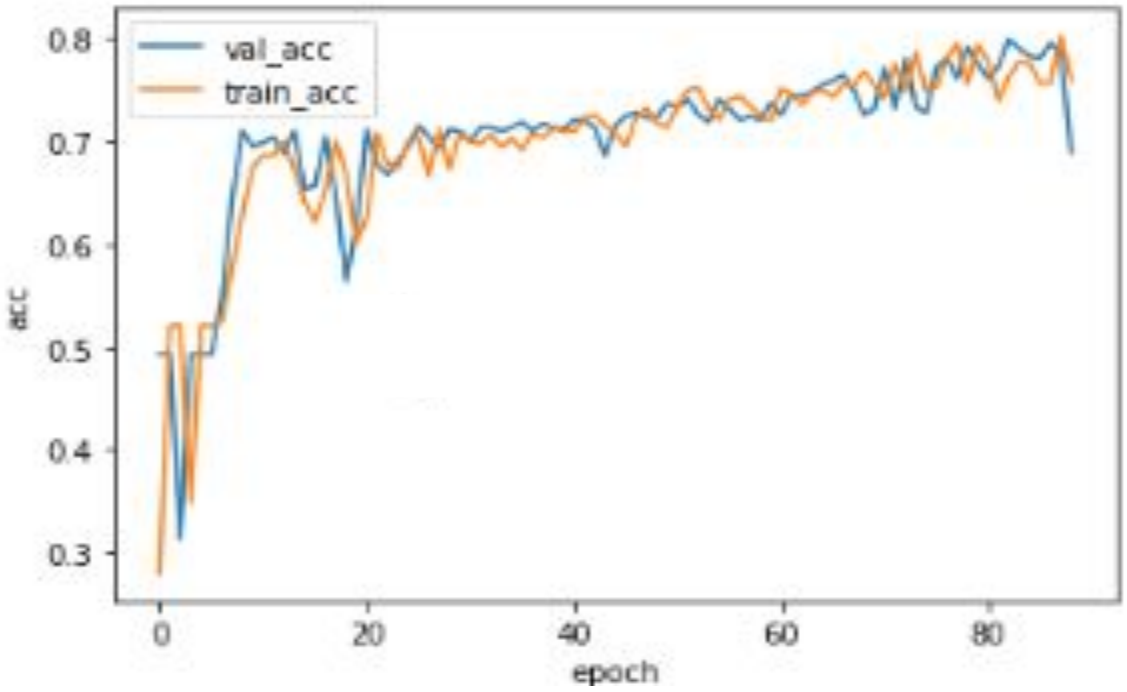
Accuracy plots



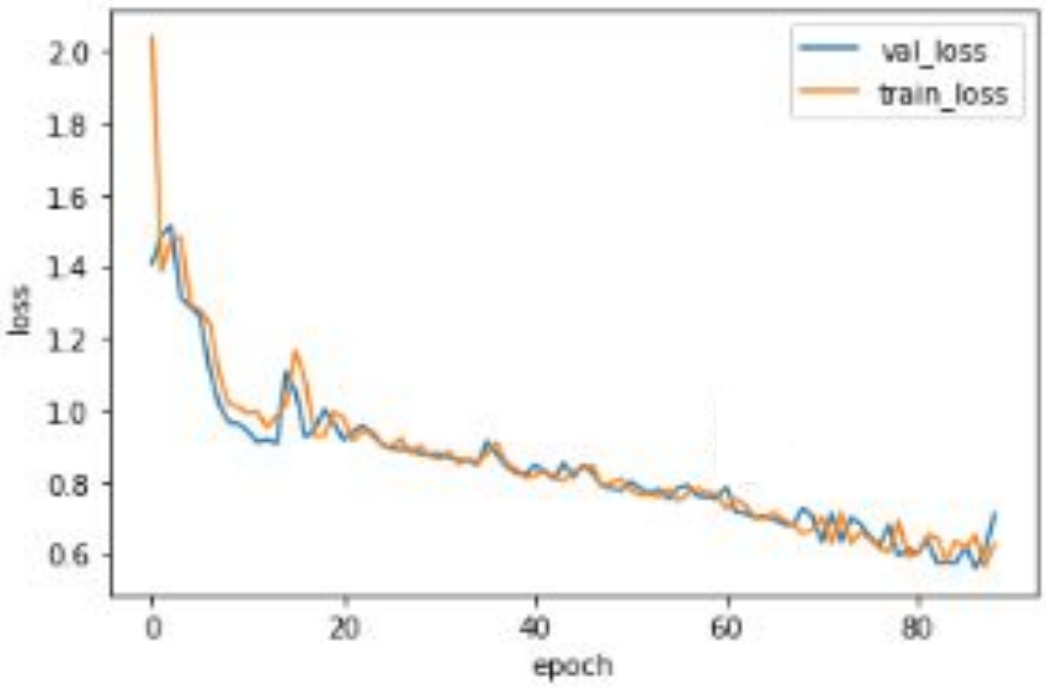
Loss Plots

```
Loading 183th epoch
Test set results: loss= 0.0532 accuracy= 0.8598
```

Results for first combinations:1GCN+2GAT

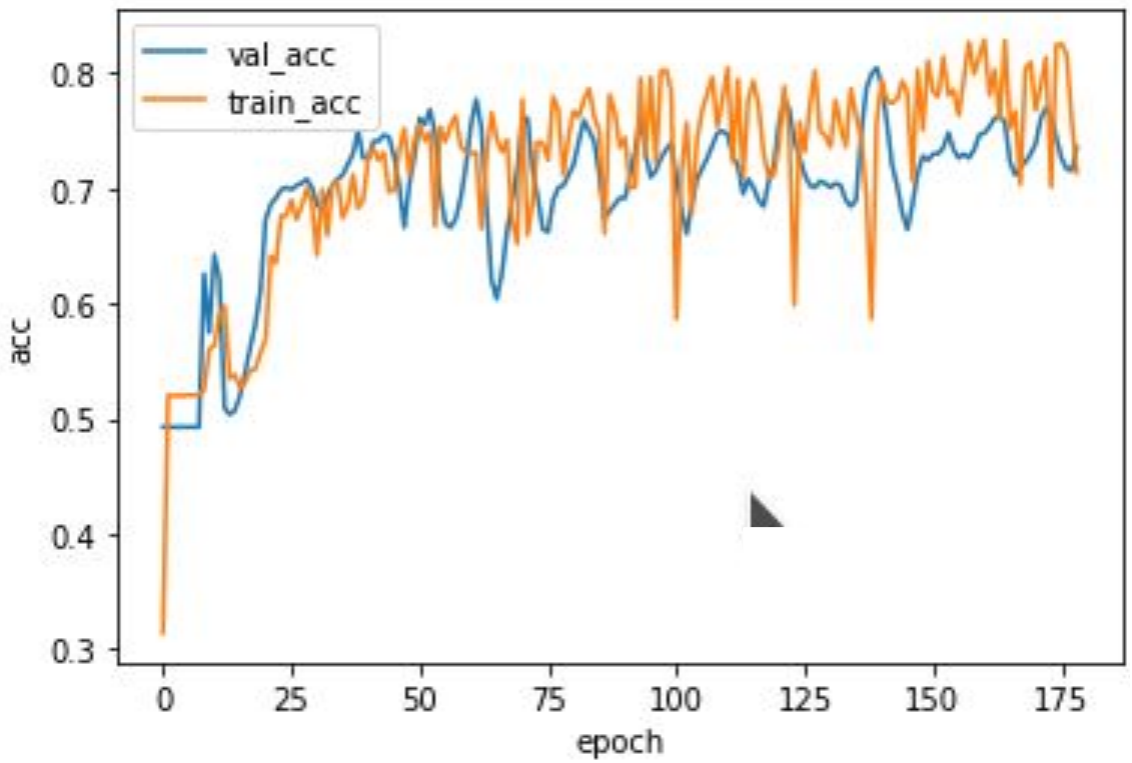


Accuracy plots

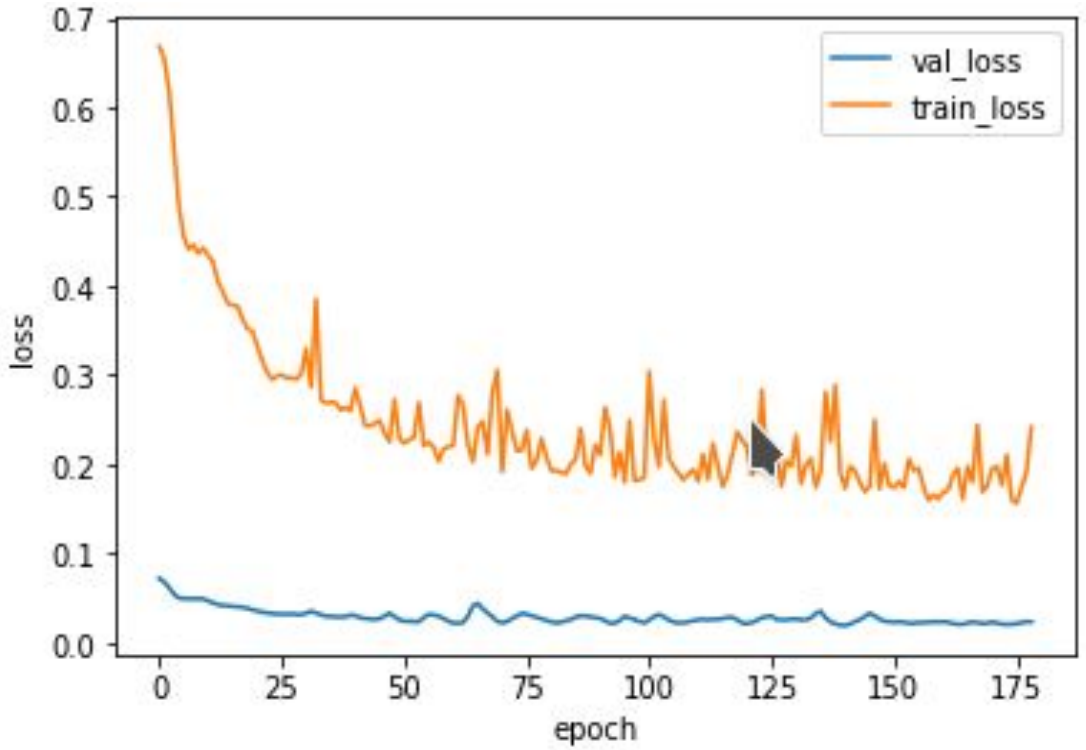


Loss Plots

Results for first combinations:2GAT



Accuracy plots

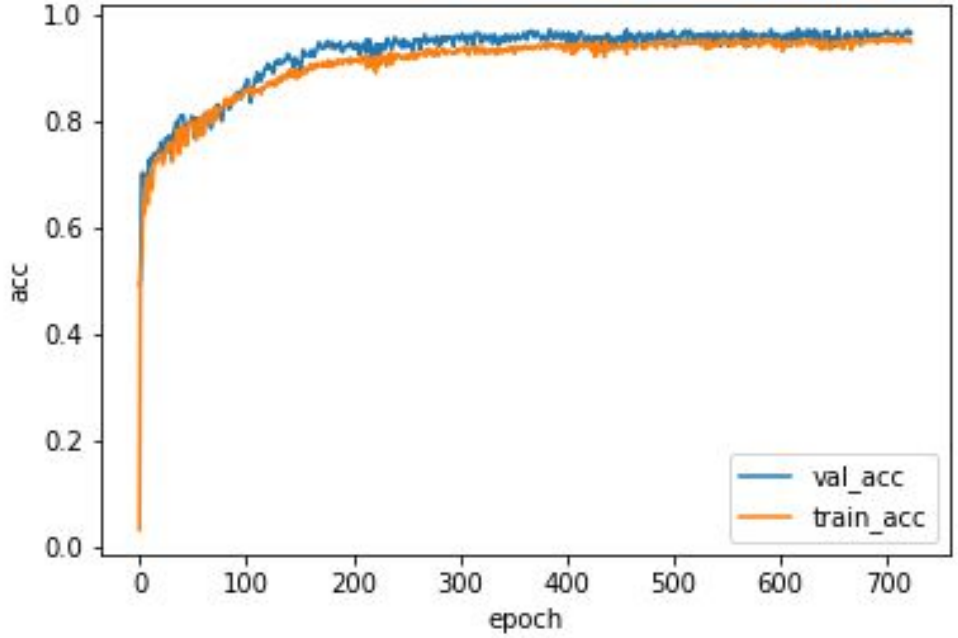


Loss plots

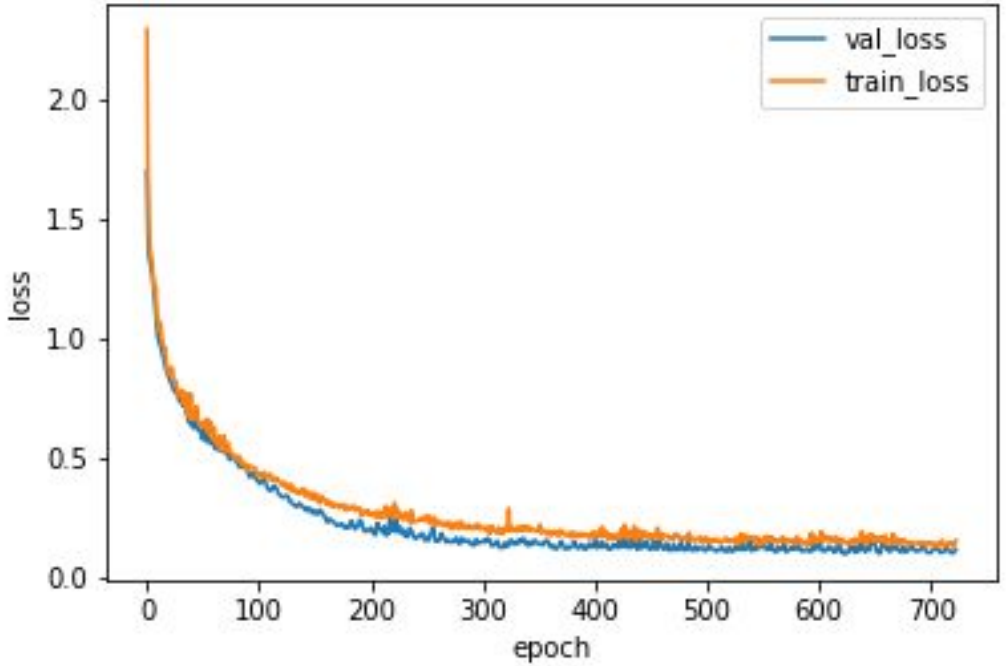
Test set results: loss= 0.0724 accuracy= 0.7889

Problem is weight is not considered so its results is poor . Although its improved by stacking 2 layers compare to single layer

Same combination 1GCN+1GAT:



Accuracy plot

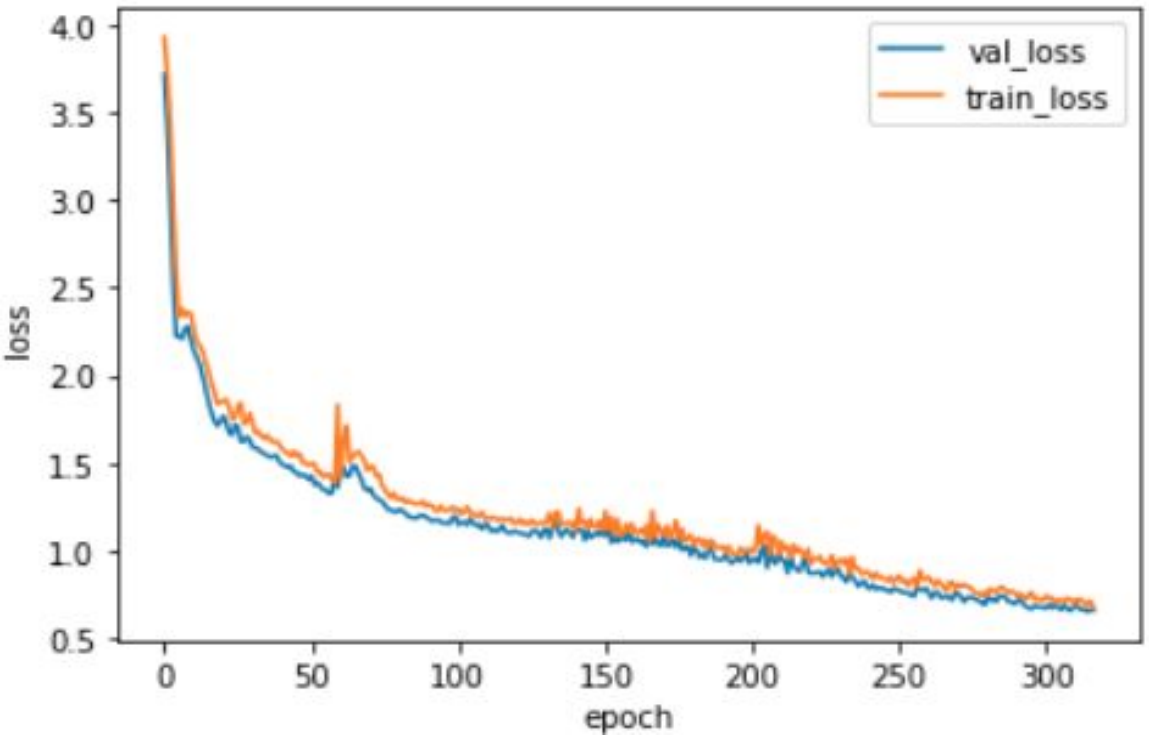


Loss Plot

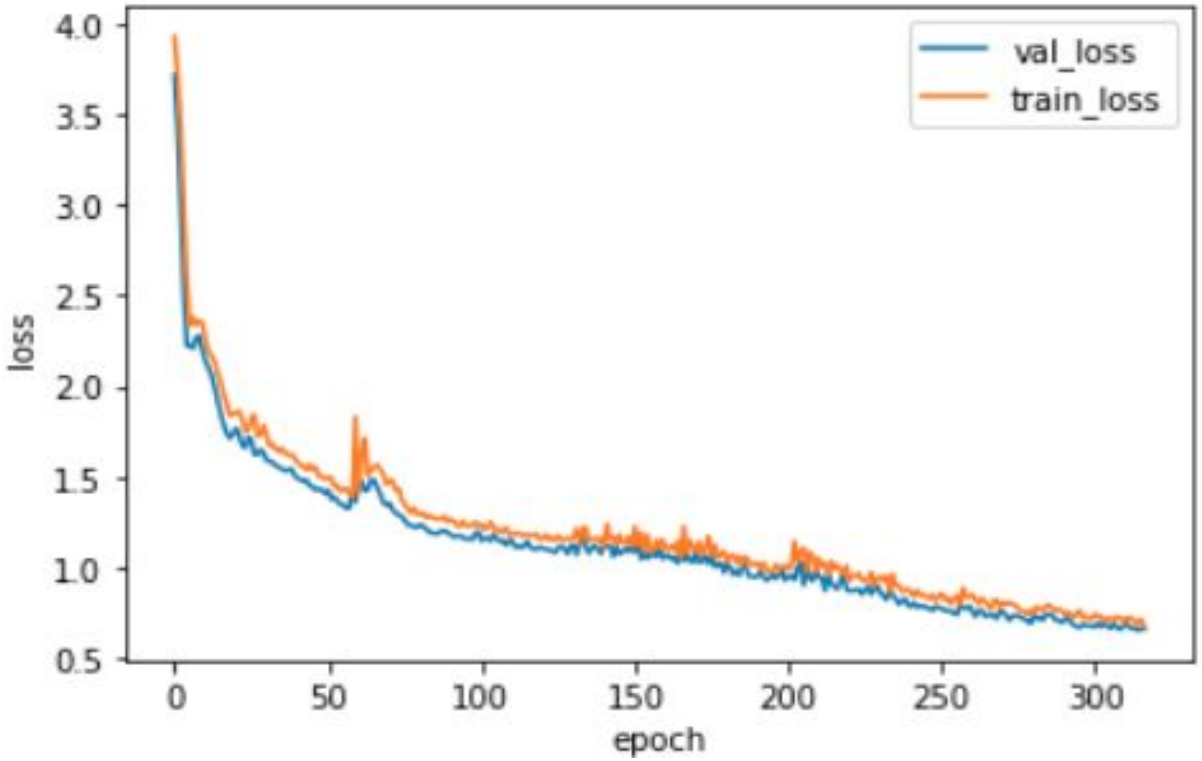
```
Test set results: loss= 0.0999 accuracy= 0.9708
```

- This combination give good result
- but input(features and adjacency) are not normalised(self loop is also not used).
- Train smoothly on R8 , create nan in case of R52

Result for 1GCN+1GAT: R52 dataset



Accuracy plot

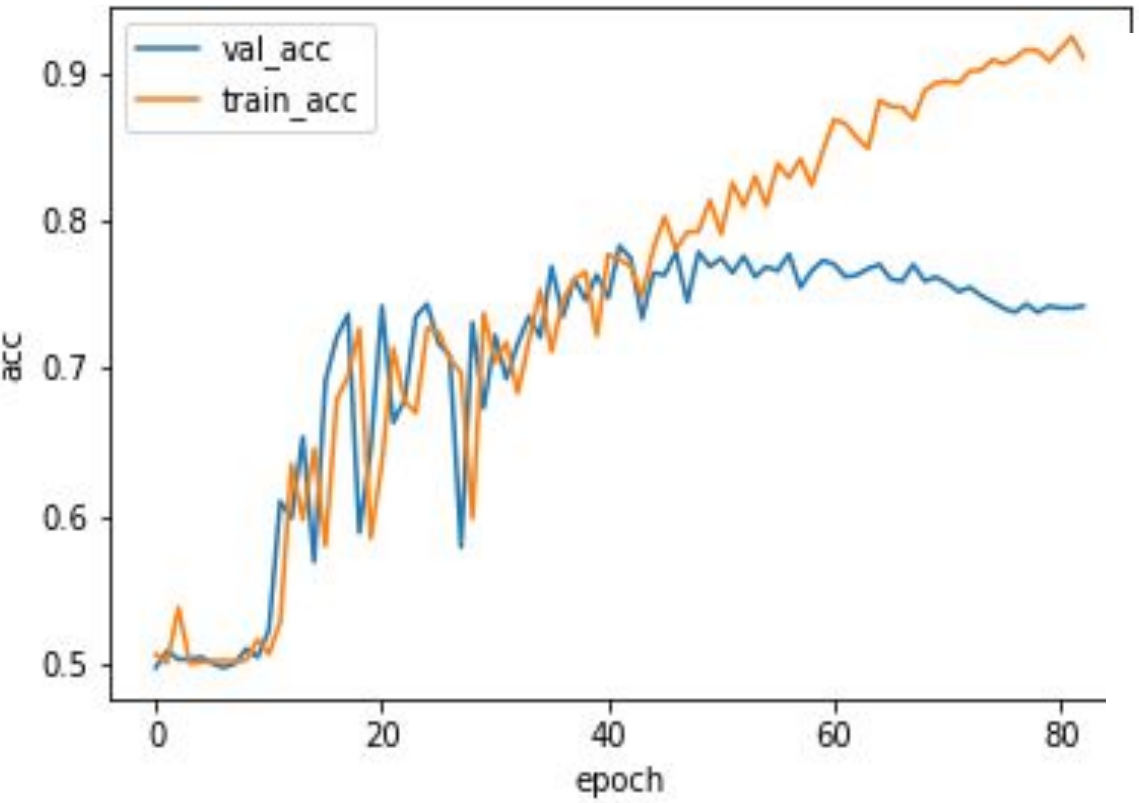


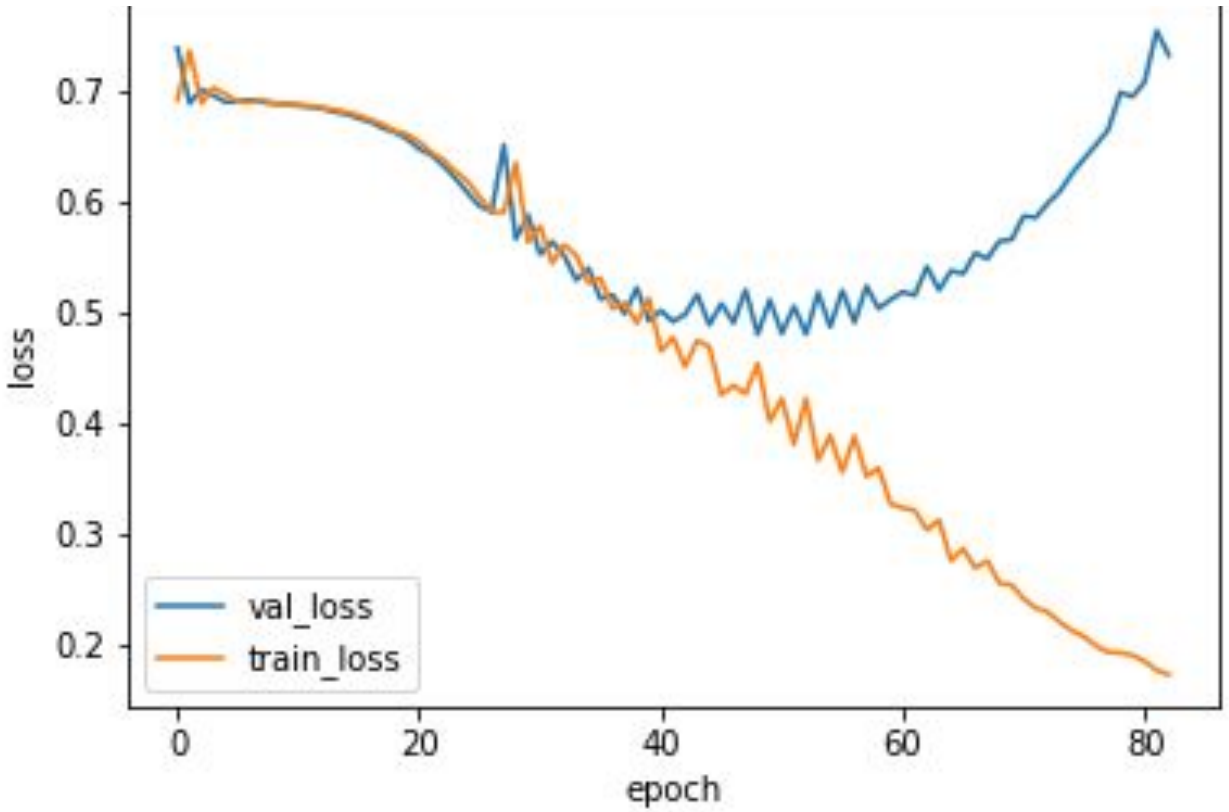
Loss plot

- Nan was introduced at 317th epoch in R52 dataset

```
Loading 314th epoch
Test set results: loss= 0.6622 accuracy= 0.8248
```

Result for 1GCN+1GAT: MR dataset

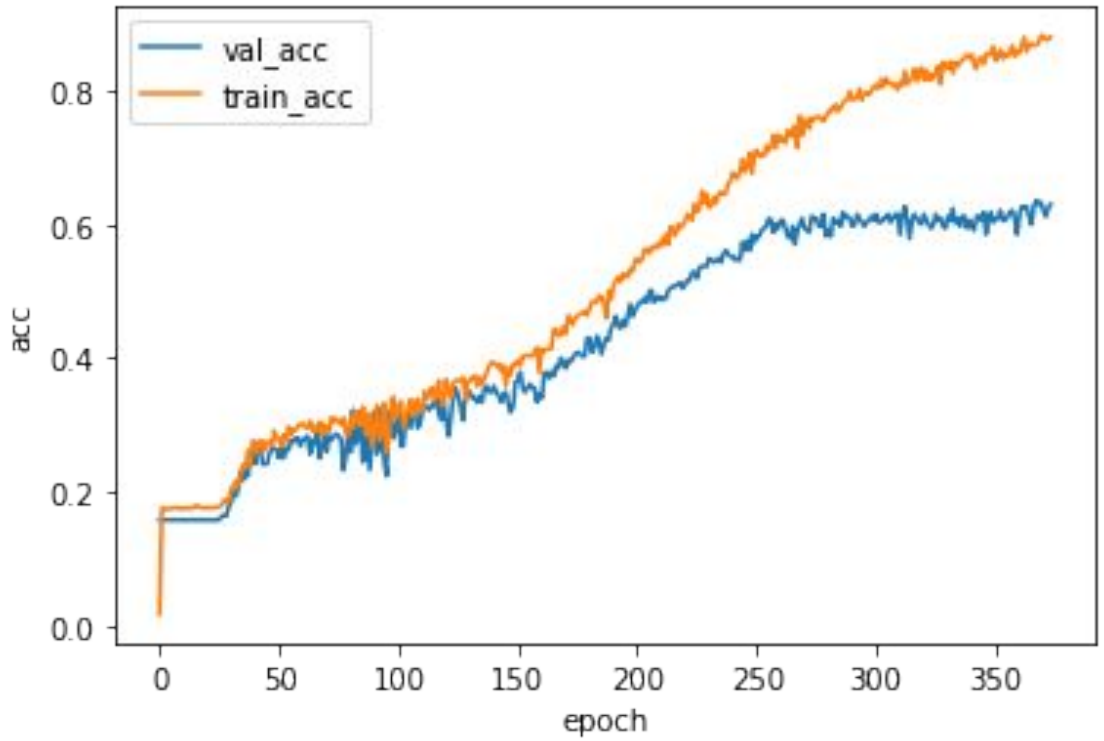




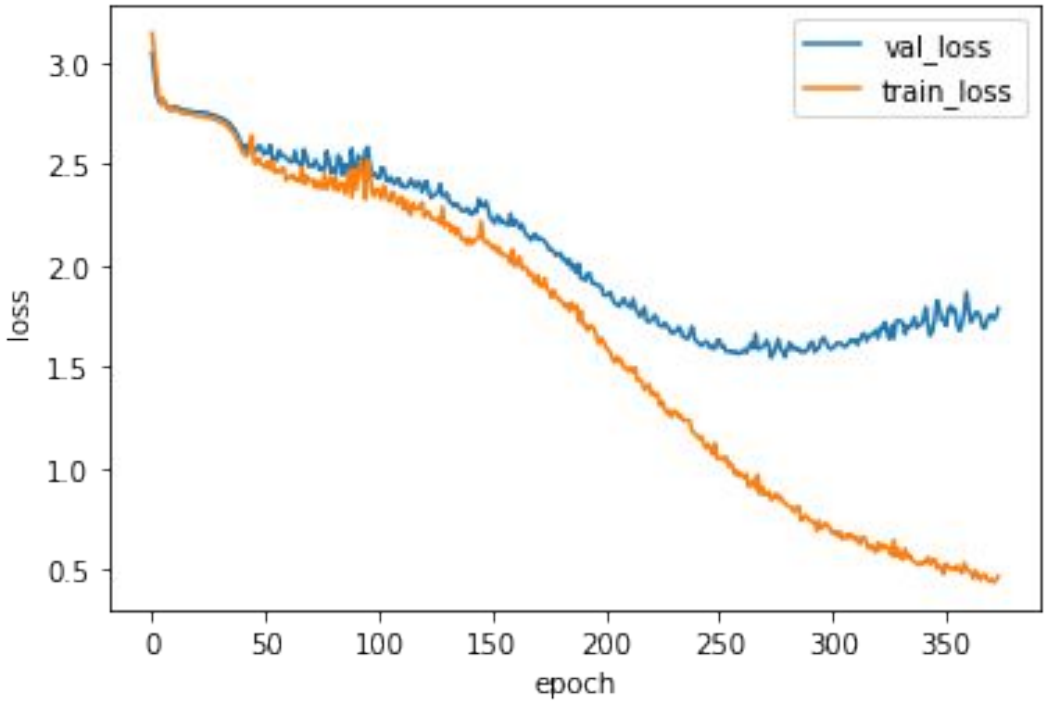
Accuracy plot

```
Loading 48th epoch
Test set results: loss= 0.4864 accuracy= 0.7645
```

Result for 1GCN+1GAT: ohsumed dataset



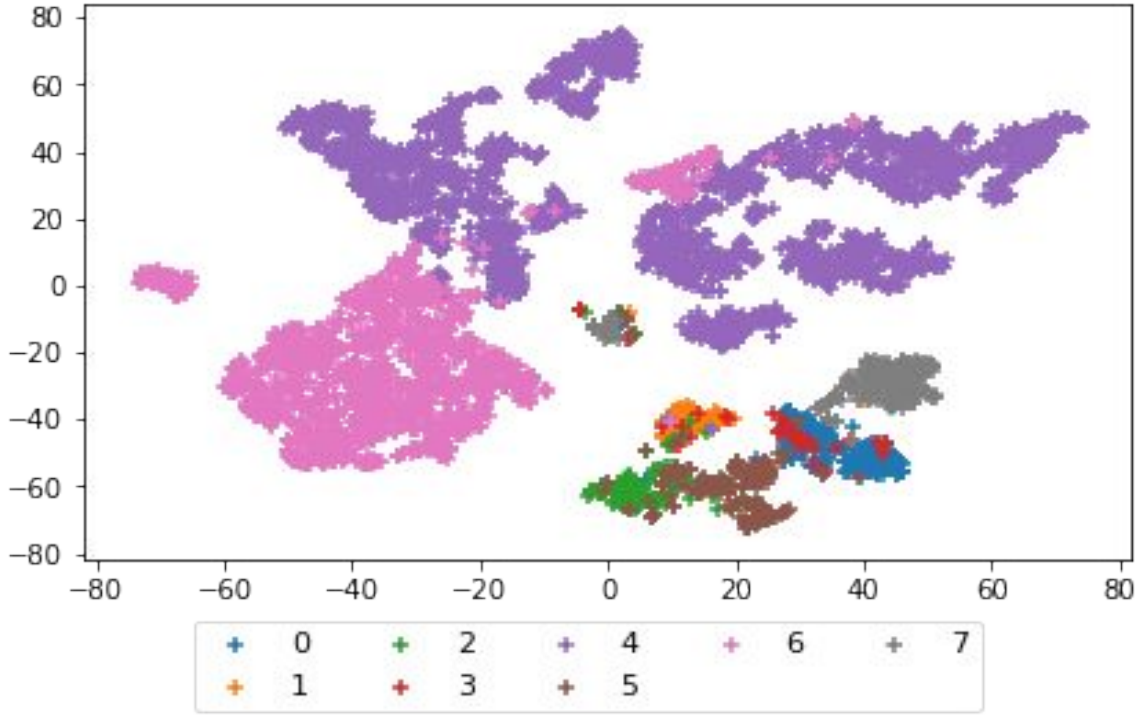
Accuracy plot



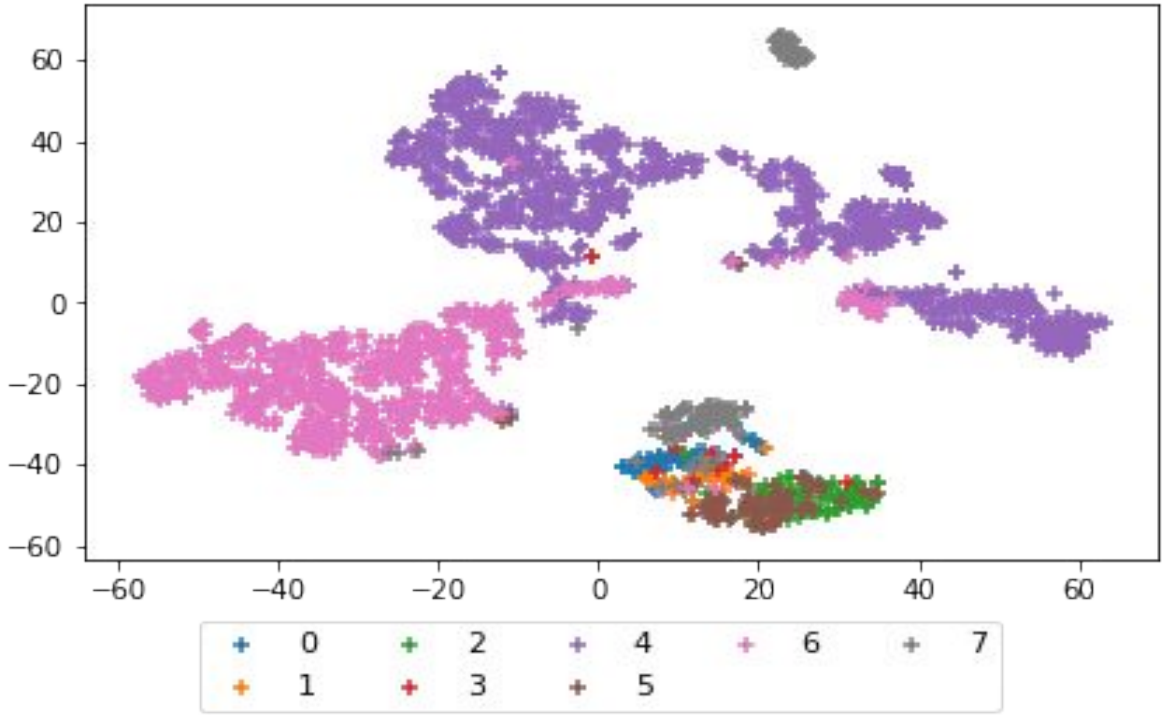
Loss plot

```
Loading 273th epoch
Test set results: loss= 1.6259 accuracy= 0.5842
```

TSNE for R8

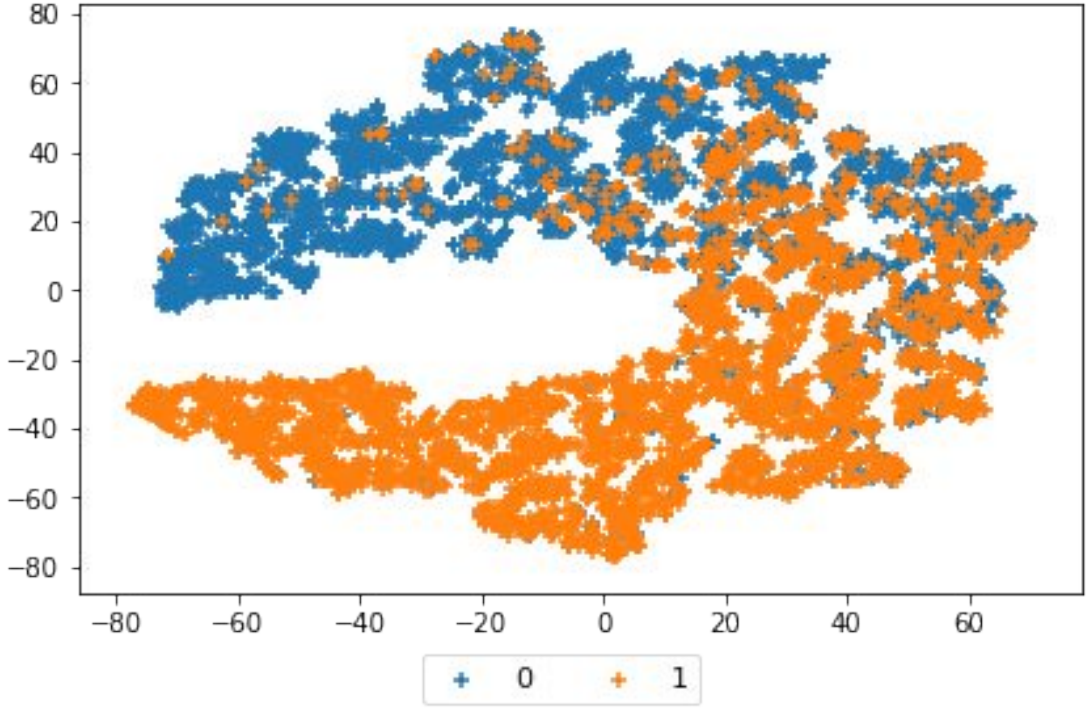


Training data

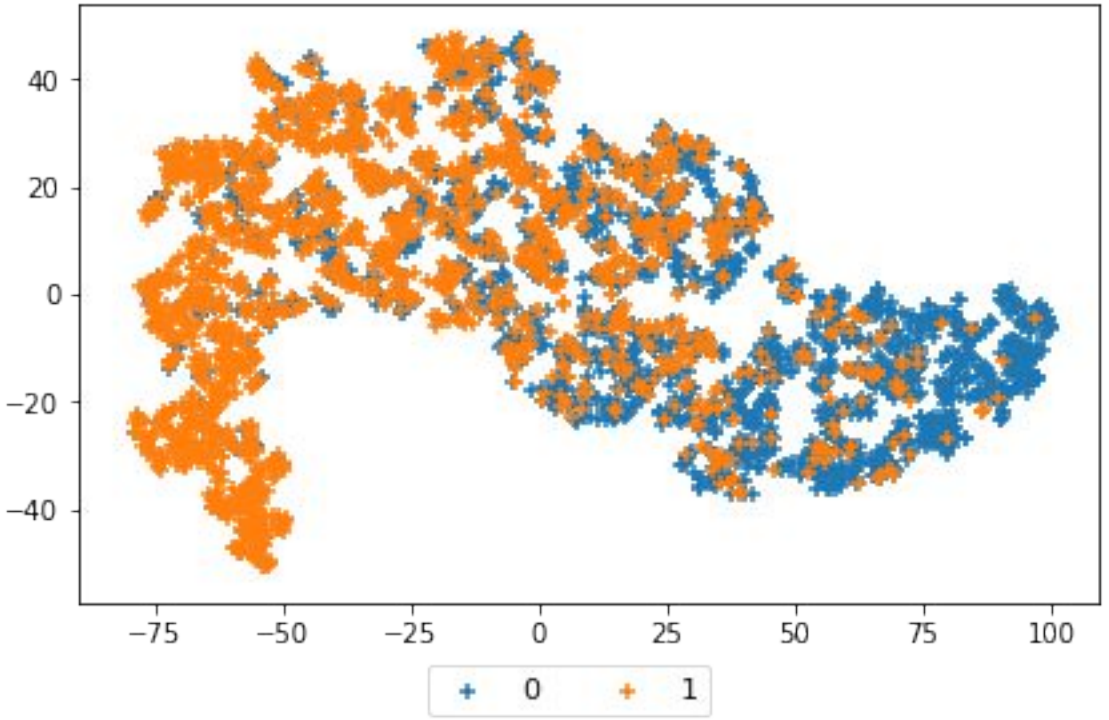


Test data

TSNE for MR

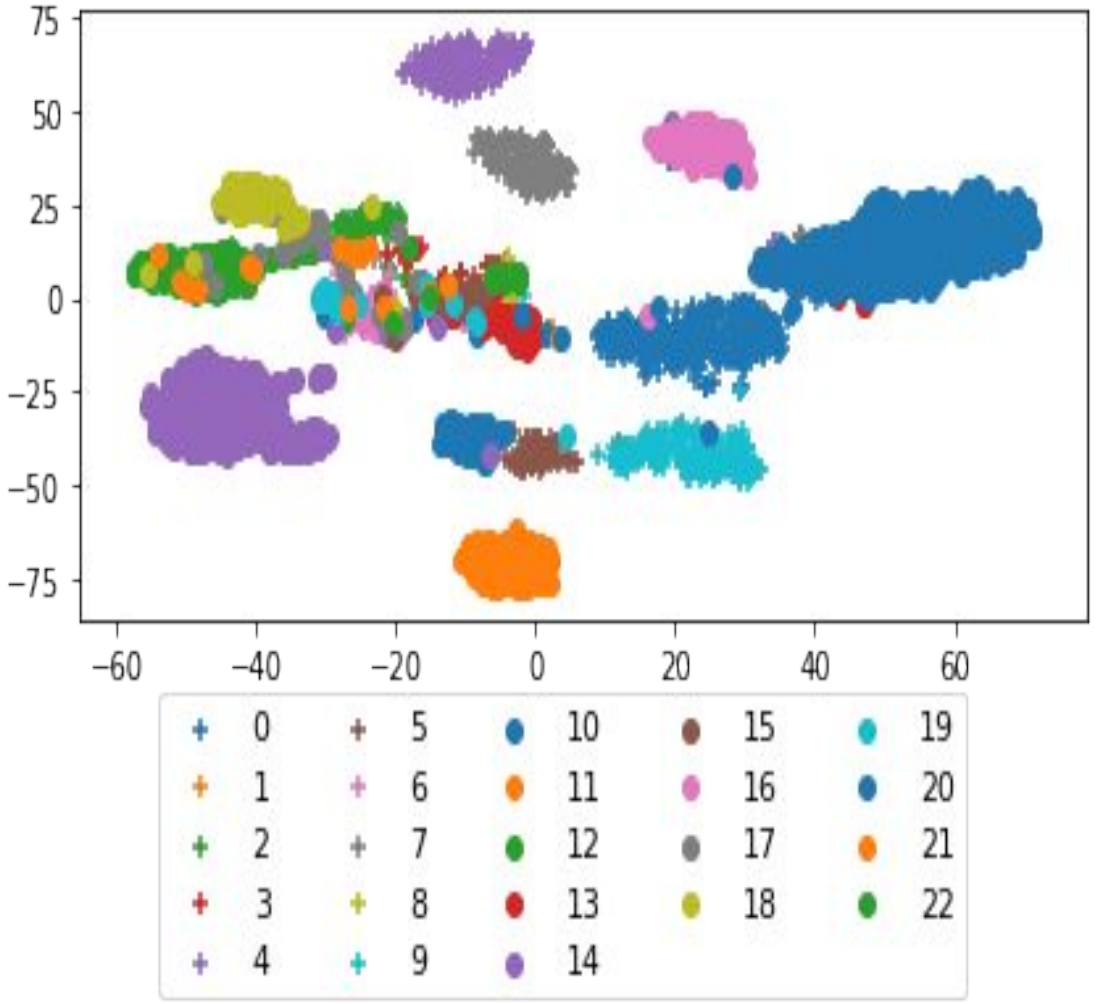


Training data

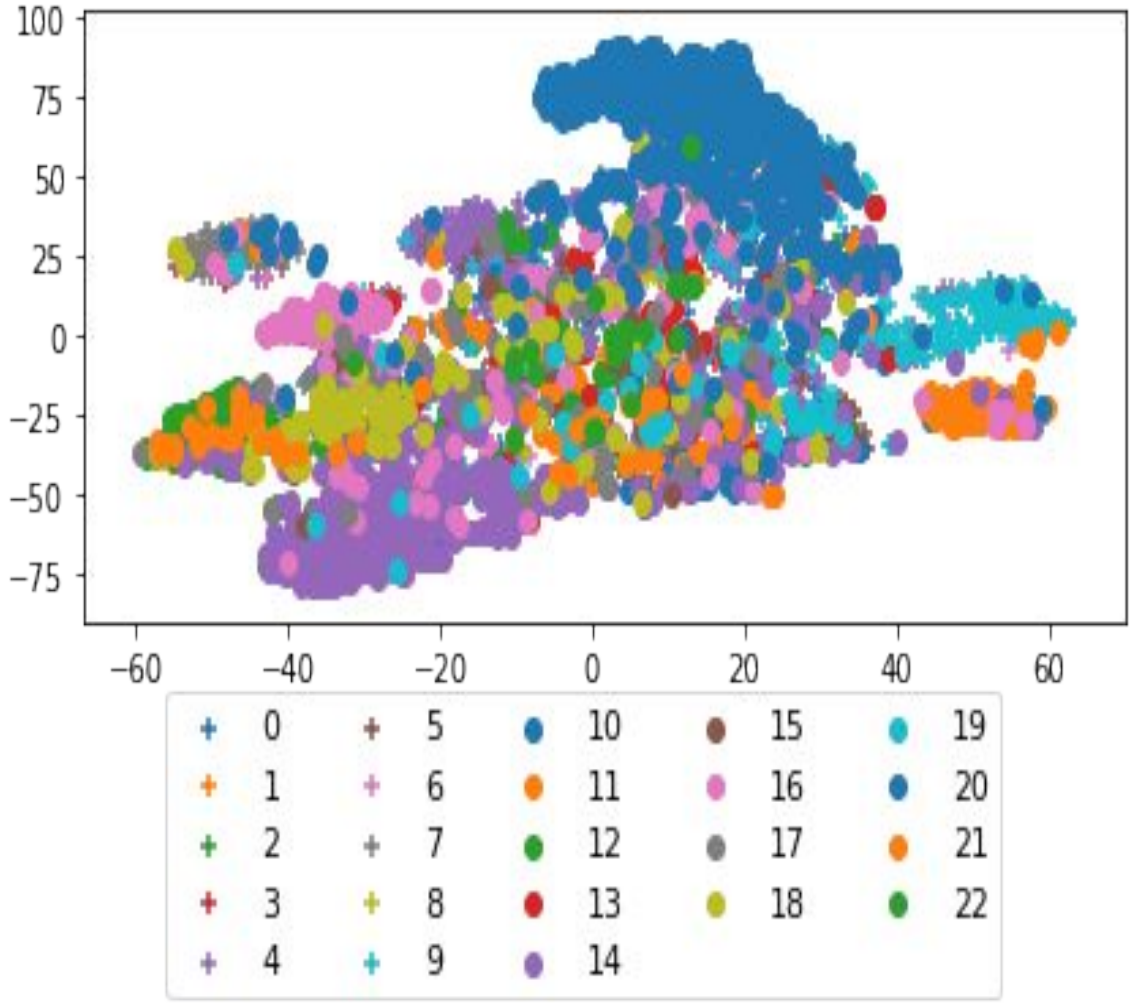


Test data

TSNE for ohsumed



Training data



Test data

Results on other datasets:

Dataset	Training	Validation	Test	SOTA
R8	97.08	97.45		97.08
R52	82.088	84.3	82.28	93.36
MR	91.69	78.31	76.45	77.75
Ohsumed	87.99	62.99	58.42	68.36

Problem still exist:

- Sparse matrix operation(especially backward).
- Above implementation does not able to deal with features(pretrained word embedding), as it creates problem while training in Sparse attention layer (i.e. it just using structural informations only).
- Slow training

Conclusions:

- Attention over weighted aggregation can provide better or parallel results as GCN.
- Better implementation of attention layer will help to increase performance.

References:

[1]Graph Convolutional Networks for Text Classification(Liang Yao, Chengsheng Mao, Yuan Luo. AAAI-19)

[2]Semi-Supervised Classification with Graph Convolutional Networks (Thomas N. Kipf, Max Welling, ICLR 2017)

[3]Graph Attention Networks (Veličković *et al.*, ICLR 2018)

[4]Text level Graph Neural Network (Huang et al. 2019)

[5]Simplifying GCN(Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. ICML 2019.)

[6]Inductive Representation Learning on Large Graphs(W.L. Hamilton, R. Ying, and J. Leskovec *arXiv:1706.02216 [cs.SI]*, 2017)

[7]Neural Machine Translation by Jointly Learning to Align and Translate(Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio,ICLR 2015)

[8]Attention Is All You Need(Vaswani et. al NIPS 2017)

Q/A

Thank you