Graph Convolutional Networks for Road Networks

Ju Hyeon Kim

Seoul National University

Jepsen, Tobias Skovgaard, Christian S. Jensen, and Thomas Dyhre Nielsen. "Graph convolutional networks for road networks." *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2019.

How Road Networks are Different?

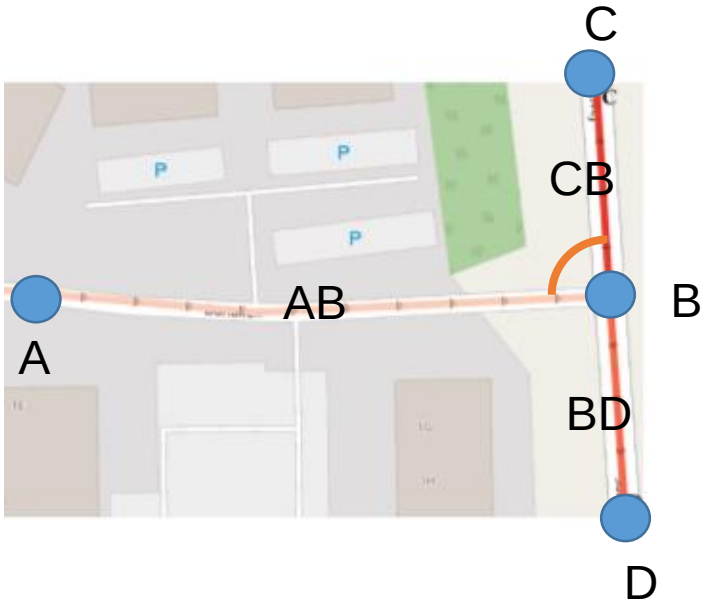
- GCN works well for social, citation, biological, etc…
- Road networks differ substantially from such tasks.
- Many implicit assumptions in GCN proposals do not hold.

How Road Networks are Different?

(1) Edge relational

not only node and edge attributes but also between-edge attributes.

e.g.) angle between two road segments affects transition time



(2) Low-density

Node degree: 2.2 (road) vs 492 (social) → Sensitive to abnormal neighbors.

How Road Networks are Different?

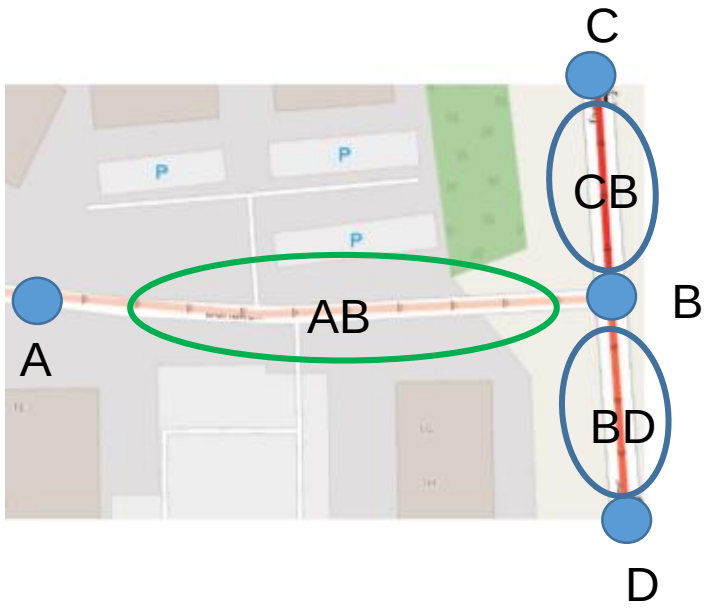
(3) Homophily is not always true

※ Homophily: adjacent roads are similar

AB, BD, CB are all adjacent at B.

Considering driving speed, **BD and CB are similar but not AB.**

➔ **Volatile homophily**



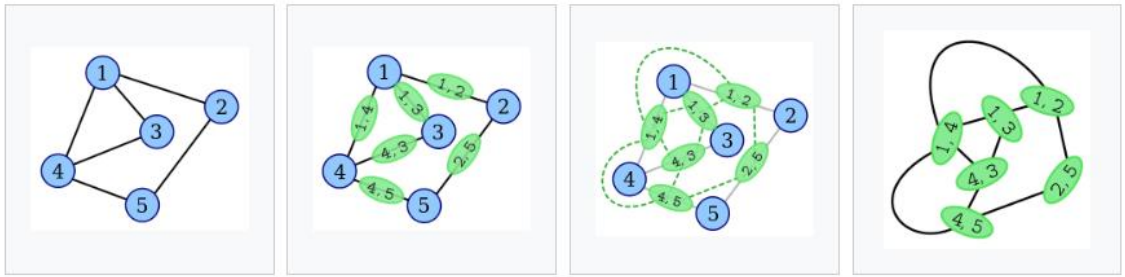
Main Contribution

- Traditional GCNs:
  - Do not support relational attributes between intersections and road segments, such as edge attributes and between-edge attributes.
  - They also rely on homophily assumption.
- Proposed Relational Fusion Network(RFN):
  - Consider both the relationships between intersections (nodes) and between road segments (edges) jointly.

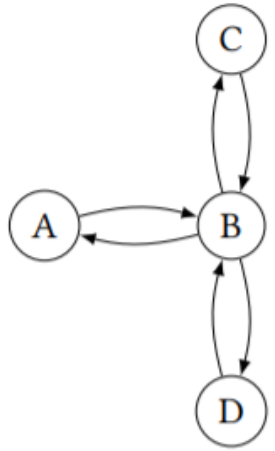
Modeling Road Networks

- Model a road network as an attributed directed graph $G = (V, E, A^V, A^E, A^B)$
  - V: set of nodes(intersections)
  - E: set of edges(roads)
  - A^V: attributes of intersections
  - A^E: attributes of roads
  - A^B: attributes of between-road segments.

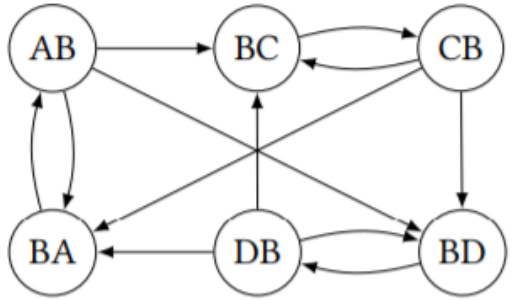
Primal and Dual Graph



Source: Wikipedia 'Line Graph'



(a) Primal Graph.



(b) Dual Graph.

Node : intersection
Edge : road segment

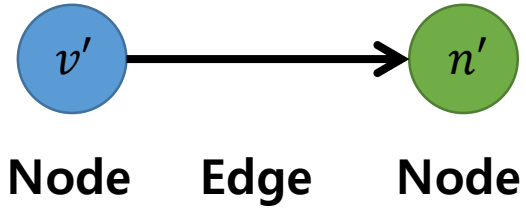
$$G^P = (V, E)$$

Node : road segment
Edge : *between-edge*

$$G^D = (E, B)$$

Relational Fusion

$$\mathbf{H}_{(v',n')}^{(R,k-1)} = \mathbf{H}_{v'}^{(V',k-1)} \oplus \mathbf{H}_{n'}^{(V',k-1)} \oplus \mathbf{H}_{(v',n')}^{(E',k-1)}$$



Representation: $\mathbf{H}_{v'}$ $\mathbf{H}_{(v',n')}$ $\mathbf{H}_{n'}$

FUSE

$\text{FUSE}(\mathbf{H}_{v'}, \mathbf{H}_{(v',n')}, \mathbf{H}_{n'})$

AdditiveFuse

$$\text{AdditiveFuse}^k(\mathbf{H}_{v'}^{(V',k-1)}, \mathbf{H}_{n'}^{(V',k-1)}, \mathbf{H}_{(v',n')}^{(E',k-1)}) = \sigma\left(\mathbf{H}_{(v',n')}^{(R,k-1)}\mathbf{W}^R + \mathbf{b}\right)$$

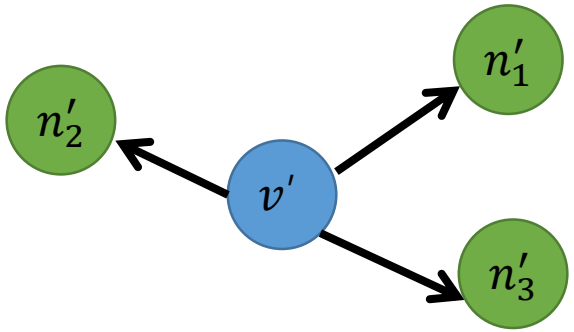
InteractionalFuse

$$\text{InteractionalFuse}^k(\mathbf{H}_{v'}^{(V',k-1)}, \mathbf{H}_{n'}^{(V',k-1)}, \mathbf{H}_{(v',n')}^{(E',k-1)}) = \sigma\left((\mathbf{H}^{(R,k-1}\mathbf{W}^I \odot \mathbf{H}^{(R,k-1)})\mathbf{W}^R\right) + \mathbf{b}, \quad (4)$$

trainable interaction weight matrix

InteractionalFuse offers improved modeling capacity over AdditiveFuse, enabling it to better address the challenge of volatile homophily.

Relational Fusion



$$F_{v'} = \left\{ \text{FUSE}\left(\mathbf{H}_{v'}, \mathbf{H}_{(v',n')}, \mathbf{H}_{n'}\right) | n' \in N(v') \cup \{v'\} \right\}$$

$$\mathbf{H}_{v',new} = \text{AGGREGATE}(F_{v'})$$

Attentional Aggregator

$$\text{AGGREGATE}(F_v) = \sum_{\mathbf{Z}_n \in F_v} A(v,n)\mathbf{f}_n$$

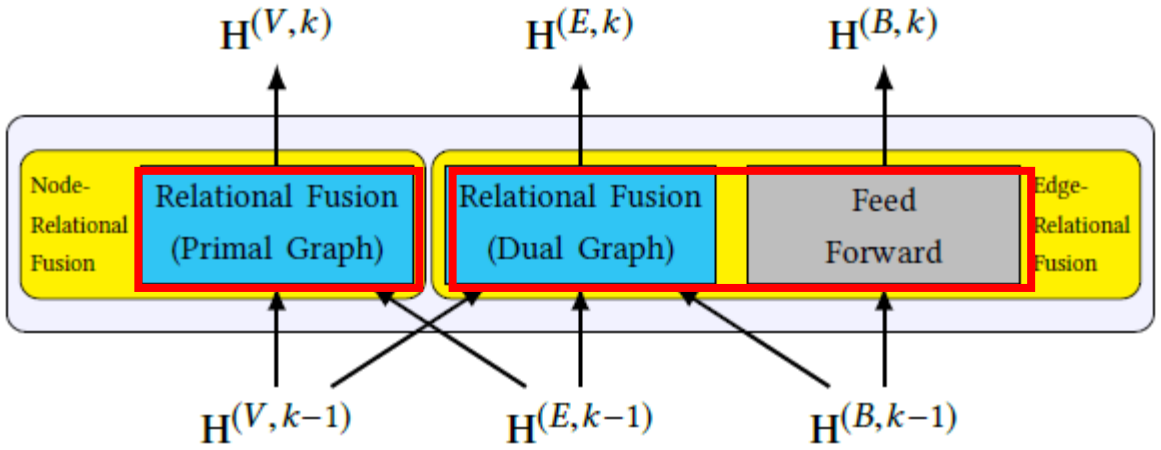
$$A(v',n') = \frac{\exp\left(C(\mathbf{H}_{(v',n')}^{(R,k-1)})\right)}{\sum_{n'' \in N(v')} \exp\left(C(\mathbf{H}_{(v',n'')}^{(R,k-1)})\right)}$$

$$C(\mathbf{H}_{(v',n')}^{(R,k-1)}) = \sigma(\mathbf{H}_{(v',n')}^{(R,k-1)}\mathbf{W}^C)$$

Non-Attentional Aggregator

Forward Propagation

Apply Relational Fusion to both primal and dual graph



(b) Relational Fusion Layer

$$\mathbf{H}^{(V,k)} \leftarrow \textsc{RelationalFusion}^k(G^P, \mathbf{H}^{(V,k-1)}, \mathbf{H}^{(E,k-1)})$$
$$\mathbf{H}^{(B',k-1)} \leftarrow \textsc{Join}(\mathbf{H}^{(V,k-1)}, \mathbf{H}^{(B,k-1)})$$
$$\mathbf{H}^{(E,k)} \leftarrow \textsc{RelationalFusion}^k(G^D, \mathbf{H}^{(E,k-1)}, \mathbf{H}^{(B',k-1)})$$
$$\mathbf{H}^{(B,k)} \leftarrow \textsc{FeedForward}^k(\mathbf{H}^{(B,k-1)})$$

$$G^P = (V, E) \qquad G^D = (E, B)$$

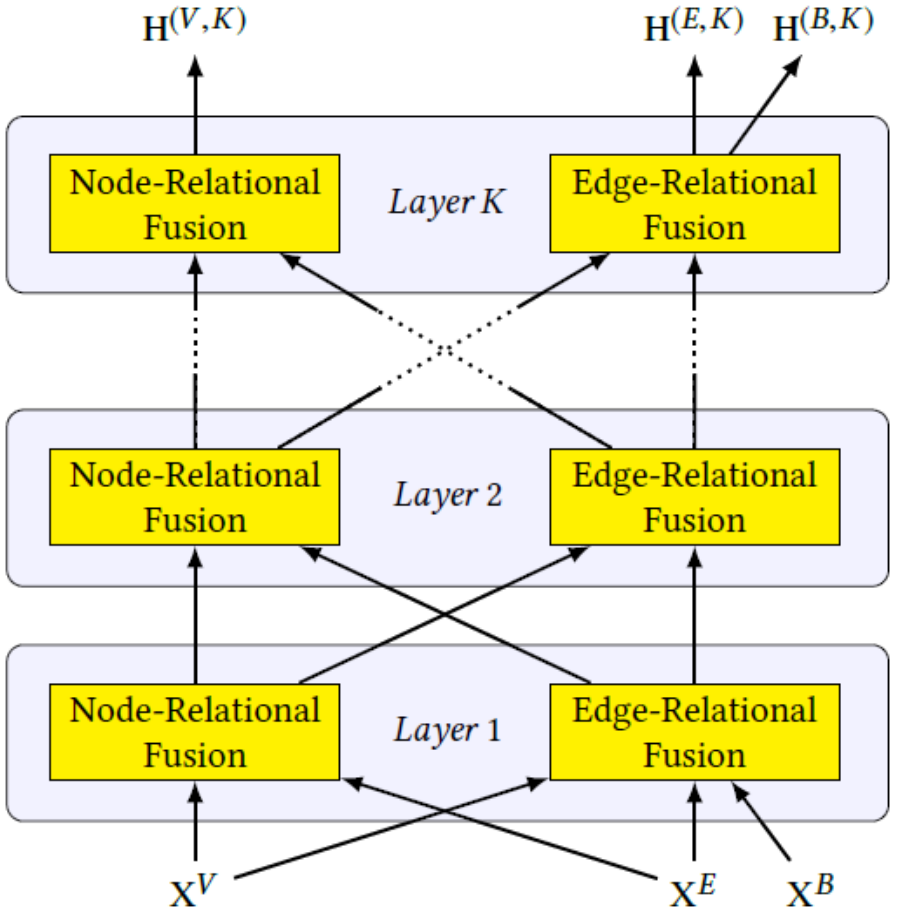
Note 1) Vertex representations $\mathbf{H}^V$ are concatenated to between-edge representations $\mathbf{H}^B$.

function $\textsc{Join}(\mathbf{H}^V, \mathbf{H}^E)$
  for all $((u, v), (v, w)) \in B$ **do**
    $\mathbf{H}^{B'}_{((u,v),(v,w))} \leftarrow \mathbf{H}^{B}_{((u,v),(v,w))} \oplus \mathbf{H}^V_v$
  return $\mathbf{H}^{B'}$

Note 2) Between-edge representations are transformed using a single feed-forward operator.

Relational Fusion Network

Stack Relational Fusion Layer



(a) Relational Fusion Network

Dataset

- Road Network Data:
  - Aalborg, Denmark
- Attributes:
  - Node : zone type (city / rural)
  - Edge : road type, length
  - Between-Edge : turning angle (right, left, U-turn)
- Target Task:
  - Driving speed estimation
  - Speed limit classification

Road Network Characteristics	
No. of Nodes	16 294
No. of Edges	35 947
No. of Between-Edges	94 718
No. of Node Features	3
No. of Edge Features	16
No. of Between-Edge Features	5

Algorithms

- Grouping Estimator
  - Group all road depending on road category and put out the mean value.
- MLP
  - Regular multi-layer perceptron that performs predictions independent of adjacent road segments.
- GraphSAGE (Max-Pooling variant)
- GAT

Note: GCNs are applied on dual graph

- 4 variants of RFN
  - RFN-Attentional+Interactional
  - RFN-NonAttentional+Interactional
  - RFN-Attentional+Additive
  - RFN-NonAttentional+Additive

Experimental Results

Table 2: Algorithm performance on Driving Speed Estimation (DSE) and Speed Limit Classification (SLC).

Algorithm	*DSE*	*SLC*
Grouping Predictor	11.026	0.356
MLP	10.160 ± 0.119	0.443 ± 0.027
GraphSAGE	8.960 ± 0.115	0.432 ± 0.014
GAT	9.548 ± 0.151	0.442 ± 0.018
RFN-Attentional+Interactional	**6.797 ± 0.124**	**0.535 ± 0.014**
RFN-NonAttentional+Interactional	6.911 ± 0.080	0.507 ± 0.012
RFN-Attentional+Additive	7.440 ± 0.133	0.518 ± 0.022
RFN-NonAttentional+Additive	7.685 ± 0.189	0.500 ± 0.011

Lower is better

Higher is better

Case Study: Capturing Volatile Homophily

- RFN is capable of making a much sharper difference on adjacent roads, while GAT and GraphSAGE shows "smooth" homophily.

Segment	*Ground Truth*	*RFN-AA-I*	*GAT*	*GraphSAGE*
AB	31.92	22.33	26.87	27.15
BD	17.74	20.97	20.88	25.44
CB	24.11	17.39	27.25	27.04
DE	51.44	56.87	47.47	45.44
DF	41.96	56.02	43.99	45.19
Score	0	7.80	3.46	4.92

Conclusion

- State-of-the-art GCNs are not appropriate for the road network setting (cannot support relational attribute, volatile homophily).
- Propose Relational Fusion Network(RFN) that consider both the relationships between intersections (nodes) and between road segments (edges) jointly.
- Experimental result demonstrates effectiveness of RFN.
- Other applications?

Reproduction Result

- Code is uploaded on my github page:
  - https://github.com/juhyeonkim95/GCN2020FinalProject
- Original code:
  - https://github.com/TobiasSkovgaardJepsen/relational-fusion-networks
- What is added?
  - Road network generation on real cities (Korean cities).
  - Determining node/edge/between-edge attributes.
  - Implementation of baseline models (GraphSAGE, GAT).
  - Miscellaneous things on establishing training/testing environment.

Reproduction Result

- Details are written in the report. Only summary on PPT.
- Use osmnx library to construct road network.
- Tested on 12 Korean cities (8 for training, 4 for testing).

Cheongju **Goyang** **Daejeon** **Daegu**

Reproduction Result

- Input Attributes:
  - Node : in-out degree
  - Edge : road category, length
  - Between-Edge : Turning angle
- Target data:
  - I couldn’t obtain historical driving speed data.
  - Instead used driving speed data from osmnx default data.
  - But speed values are limited to few types + not time-varying..
    - seems to be related to road category(대로, 로, 길)

Road network of Suwon. Different color means different speed.

Reproduction Result

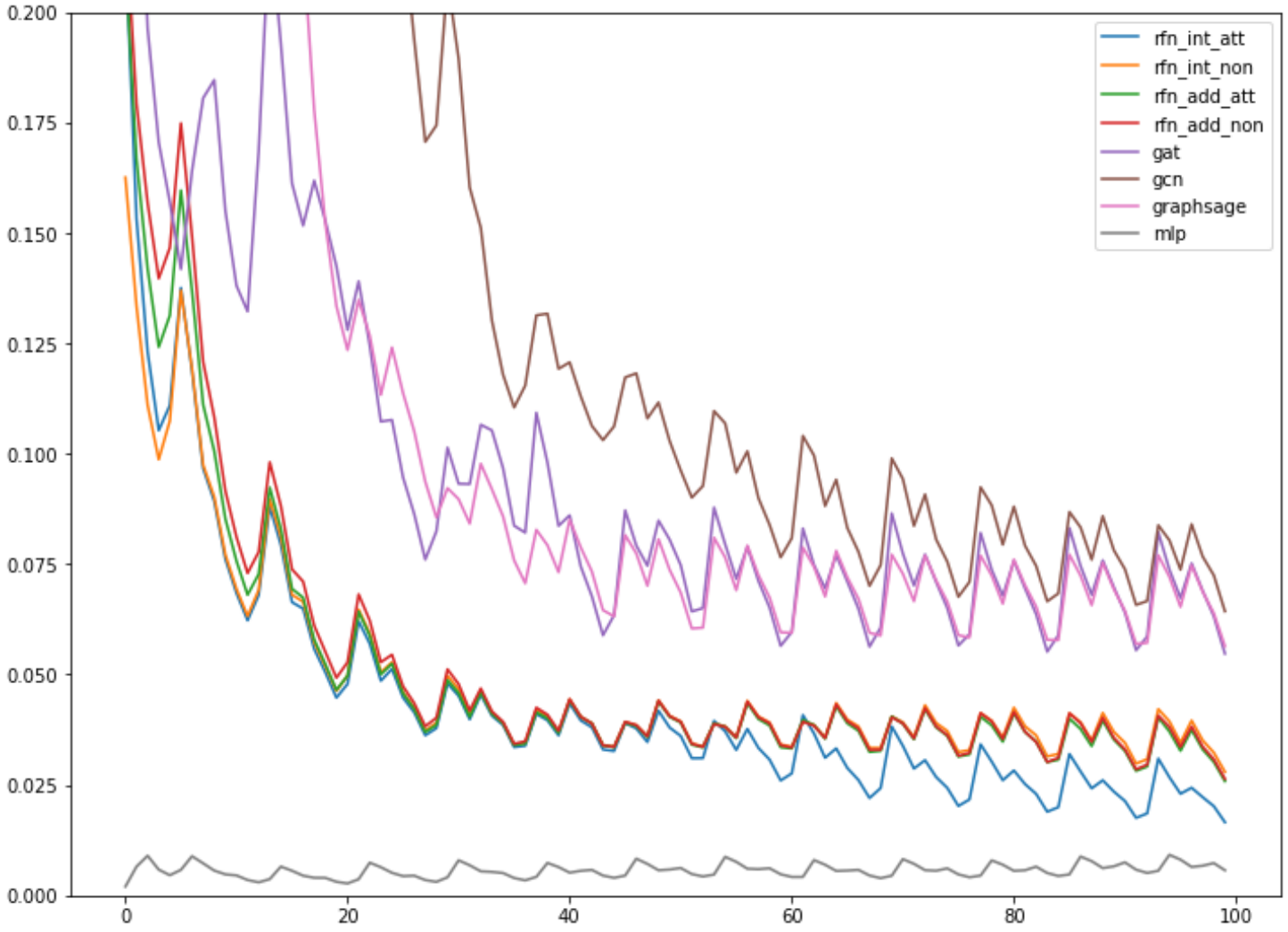
- Baseline Algorithms:
  - MLP / GCN / GraphSAGE / GAT
- 4 RFN variations:
  - Fuse : interactional / additional
  - Aggregator : attentional / non-attentional
- MXNet / deep graph libray with Pytorch was used.

Reproduction Result

Train error

Y axis: MSE loss
X axis: iteration

- Trained for 100 iteration.
- Train error per iteration is on right graph
- RFN is better than GCNs
- MLP achieves near zero error
  - Why? → mini-batch training + defect on data(speed depends on road category). But gave worse result on test data set. Seems to be overfitted.



Reproduction Result

- 4 variations of RFN gave better result than baseline models.
- RFN with interactional fuse + attentional aggregator gave was best.

Table 2: Mean square loss of driving speed estimation for each algorithm/city.

Algorithm/city	Daegu	Suwon	Ulsan	Yongin	average
RFN_Int_Att	**0.01380**	**0.00647**	**0.01754**	**0.00821**	**0.01150**
RFN_Int_Non	0.01725	0.01252	0.01902	0.02812	0.01923
RFN_Add_Att	0.01598	0.01097	0.01968	0.02828	0.01873
RFN_Add_Non	0.01582	0.01013	0.01954	0.02983	0.01883
GAT[4]	0.04189	0.02764	0.03788	0.05231	0.03993
GCN[3]	0.04681	0.03600	0.04592	0.06588	0.04865
GraphSAGE[1]	0.04487	0.03236	0.03112	0.04475	0.03827
MLP	0.02547	0.01225	0.02236	0.01405	0.01853

Reproduction Result

- Conclusion:
  - although limited data was used, it was possible to verify the effectiveness of RFN over traditional graph convolution networks.
  - It was a great experience to be able to actually implement the various GCN models I learned in class
  - It will be a great help in my future research.

감사합니다.