나도 너도 모르는 **Graph Neural Network**의 힘

2nd DLCAT

김 준 태

Who am i?

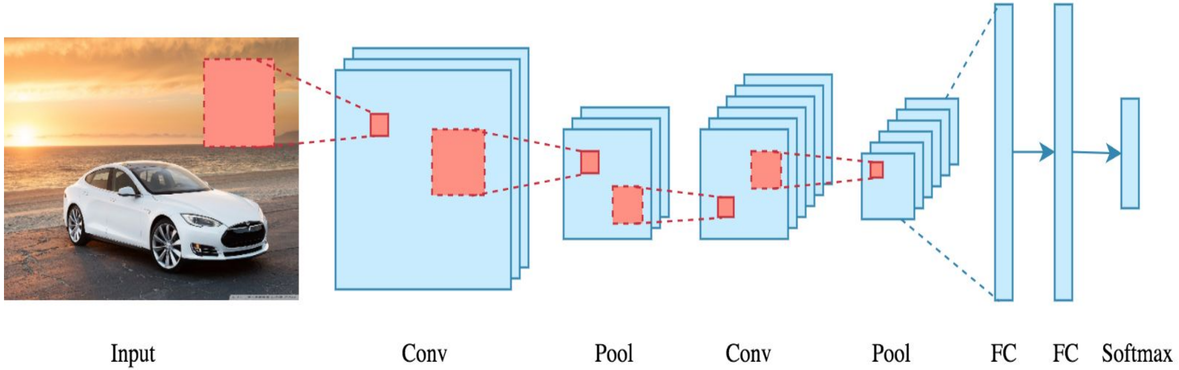
Juntae Kim

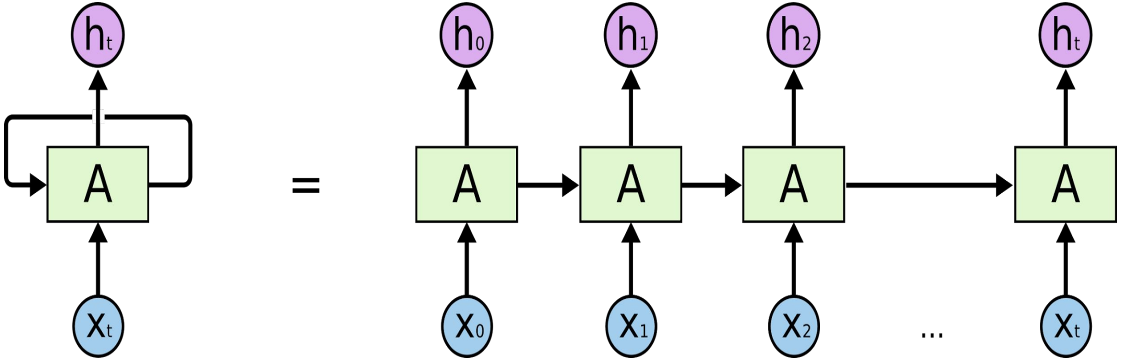
Korea University(Master Course):
DAVIAN Lab(prof Jaegul Choo)

Reinforcement Learning, Time Series, Anomaly Detection, VQA, etc…

Self driving car using GAT5 driving data

Previous





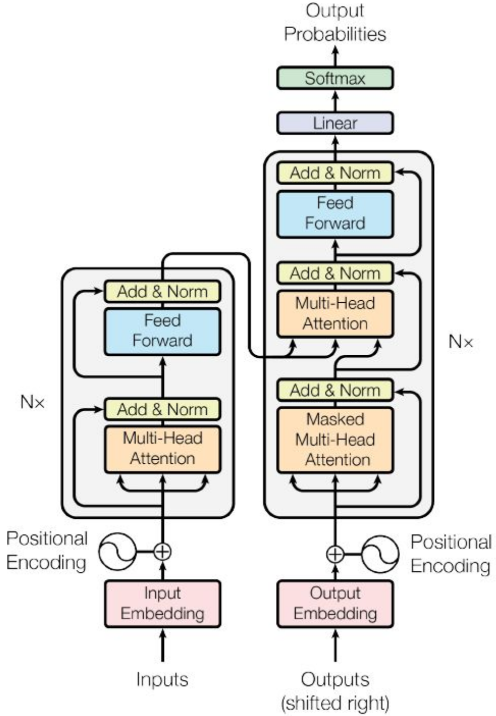
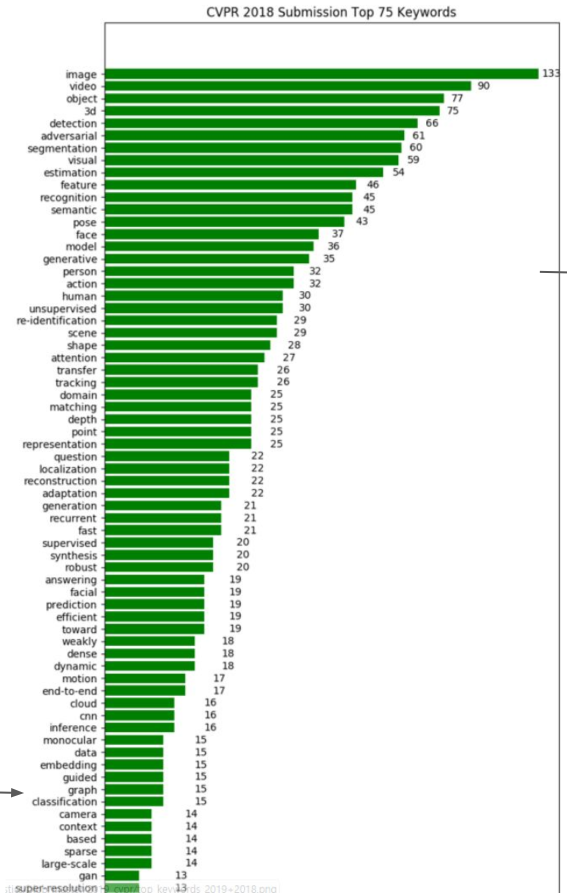
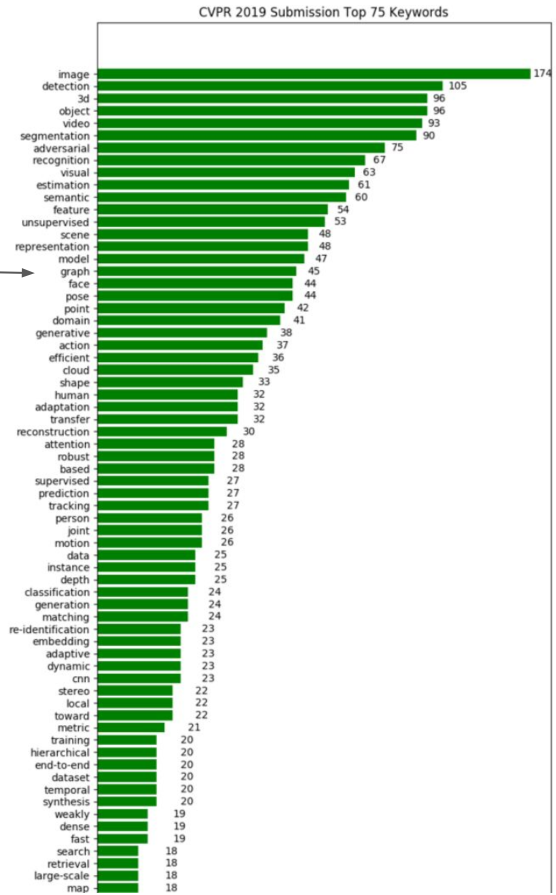


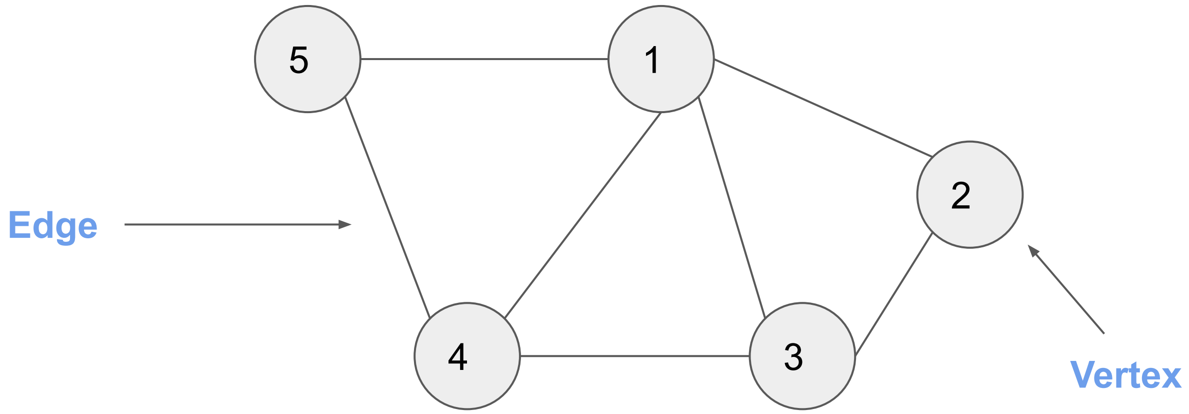
Figure 1: The Transformer - model architecture.

2019CVPR

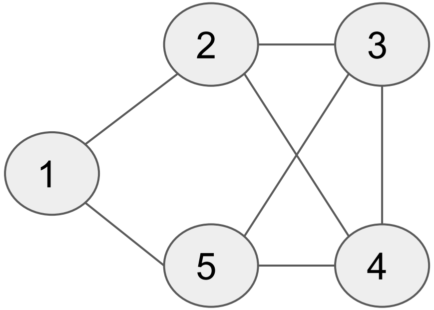

CVPR 2018 Submission Top 75 Keywords
image 133
video 90
object 77
3d 75
detection 66
adversarial 61
segmentation 60
visual 59
estimation 54
feature 46
recognition 45
semantic 45
pose 43
face 37
model 36
generative 35
person 32
action 32
human 30
unsupervised 30
re-identification 29
scene 29
shape 28
attention 27
transfer 26
tracking 26
domain 25
matching 25
depth 25
point 25
representation 25
question 22
localization 22
reconstruction 22
adaptation 22
generation 21
recurrent 21
fast 21
supervised 20
synthesis 20
robust 20
answering 19
facial 19
prediction 19
efficient 19
toward 19
weakly 18
dense 18
dynamic 18
motion 17
end-to-end 17
cloud 16
cnn 16
inference 16
monocular 15
data 15
embedding 15
guided 15
graph 15
classification 15
camera 14
context 14
based 14
sparse 14
large-scale 14
gan 13
super-resolution 13


CVPR 2019 Submission Top 75 Keywords
image 174
detection 105
3d 96
object 96
video 93
segmentation 90
adversarial 75
recognition 67
visual 63
estimation 61
semantic 60
feature 54
unsupervised 53
scene 48
representation 48
model 47
graph 45
face 44
pose 44
point 42
domain 41
generative 38
action 37
efficient 36
cloud 35
shape 33
human 32
adaptation 32
transfer 32
reconstruction 30
attention 28
robust 28
based 28
supervised 27
prediction 27
tracking 27
person 26
joint 26
motion 26
data 25
instance 25
depth 25
classification 24
generation 24
matching 24
re-identification 23
embedding 23
adaptive 23
dynamic 23
cnn 23
stereo 22
local 22
toward 22
metric 21
training 20
hierarchical 20
end-to-end 20
dataset 20
temporal 20
synthesis 20
weakly 19
dense 19
fast 19
search 18
retrieval 18
large-scale 18
map 18

What is a graph?



Graph structure



Adjacency matrix

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Node feature matrix

$$\begin{array}{r} \text{node 1} \\ \text{node 2} \\ \text{node 3} \\ \text{node 4} \\ \text{node 5} \end{array} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Graph Neural Network

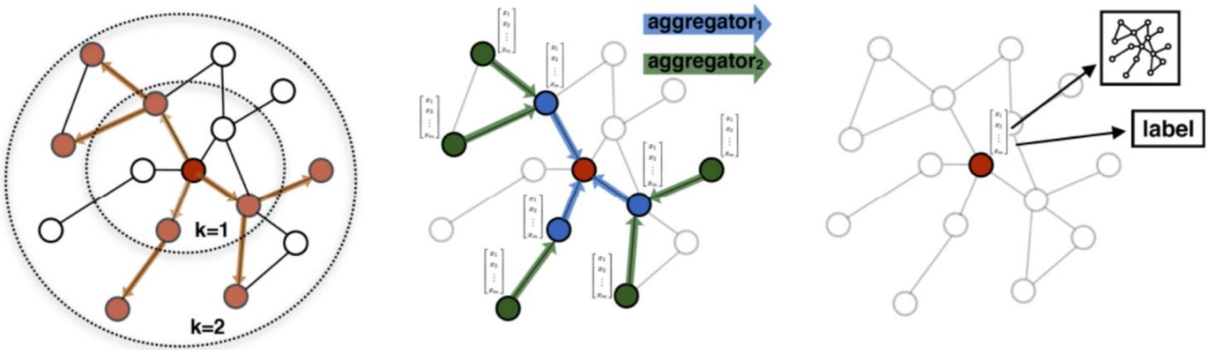
- (1) Node Classification (2) Graph Classification
- GNN은 graph structure 와 node features X_v 을 사용
- node representation vector h_v 를 학습
- entire graph vector h_G 를 학습

$$a_v^{(k)} = \mathrm{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right), \quad h_v^{(k)} = \mathrm{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$

- Neighborhood aggregation strategy
- GNN은 AGGREGATE 과 COMBINE 함수를 선택하는것이 중요!!

$$h_G = \mathrm{READOUT}\left(\{h_v^{(K)} \mid v \in G\}\right)$$

GraphSAGE



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

Aggregate Function

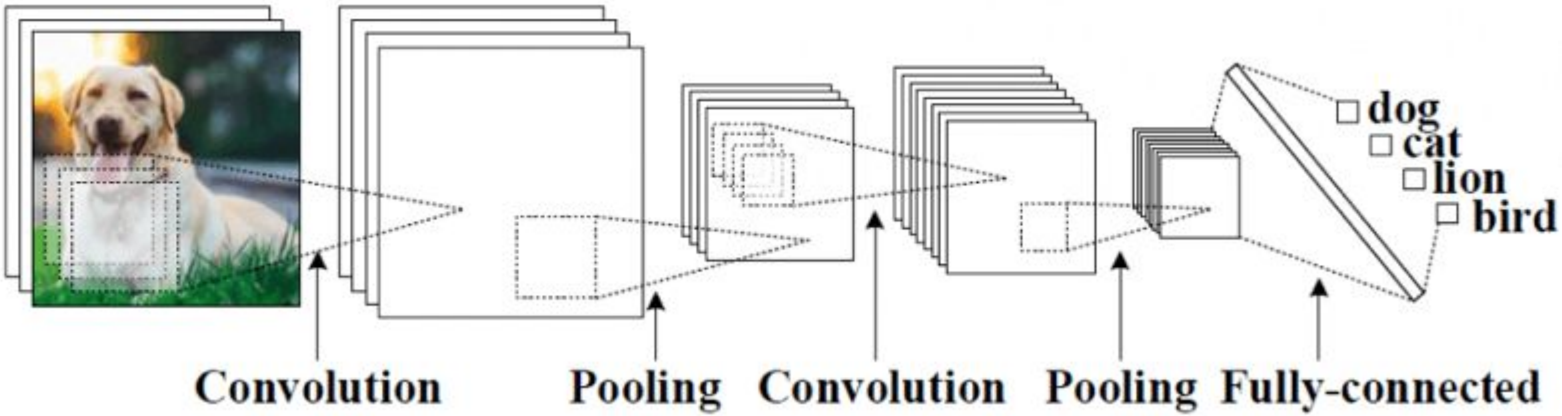
$$a_v^{(k)} = \mathrm{MAX}\left(\left\{\mathrm{ReLU}\left(W \cdot h_u^{(k-1)}\right),\ \forall u \in \mathcal{N}(v)\right\}\right)$$

MAX: element-wise max-pooling

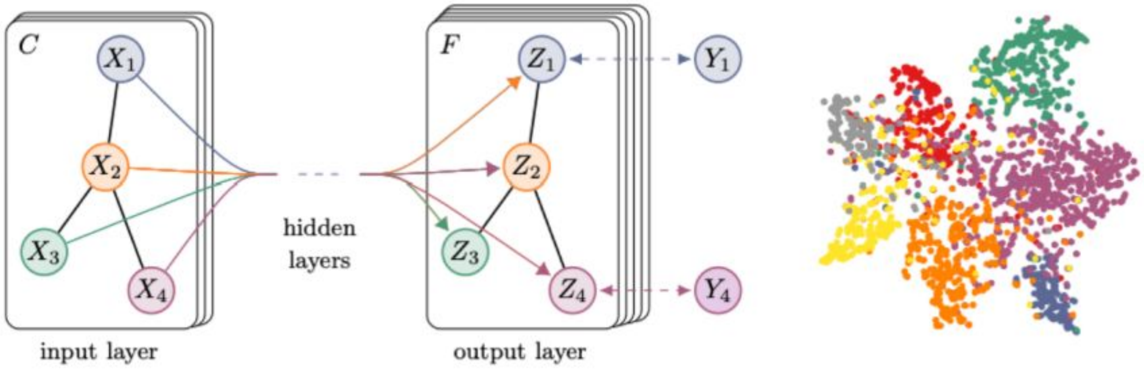
Combine Function

$$\text{linear mapping } W \cdot \left[h_v^{(k-1)}, a_v^{(k)}\right]$$

Convolution



Graph Convolutional Network



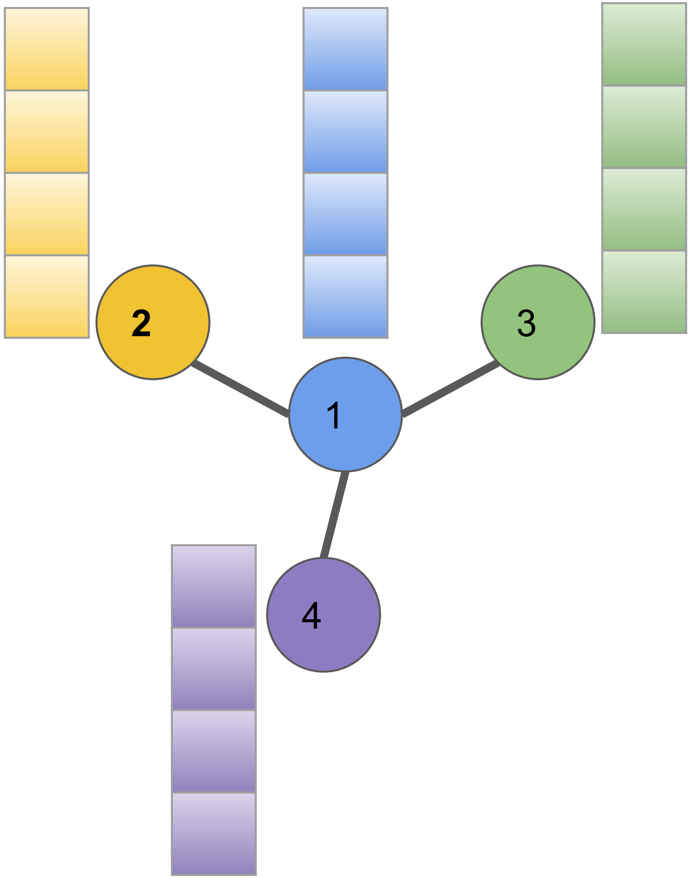
(a) Graph Convolutional Network (b) Hidden layer activations

Aggregate & Combine Function:

$$h_v^{(k)} = \mathrm{ReLU}\left(W \cdot \mathrm{MEAN}\left\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v) \cup \{v\}\right\}\right)$$

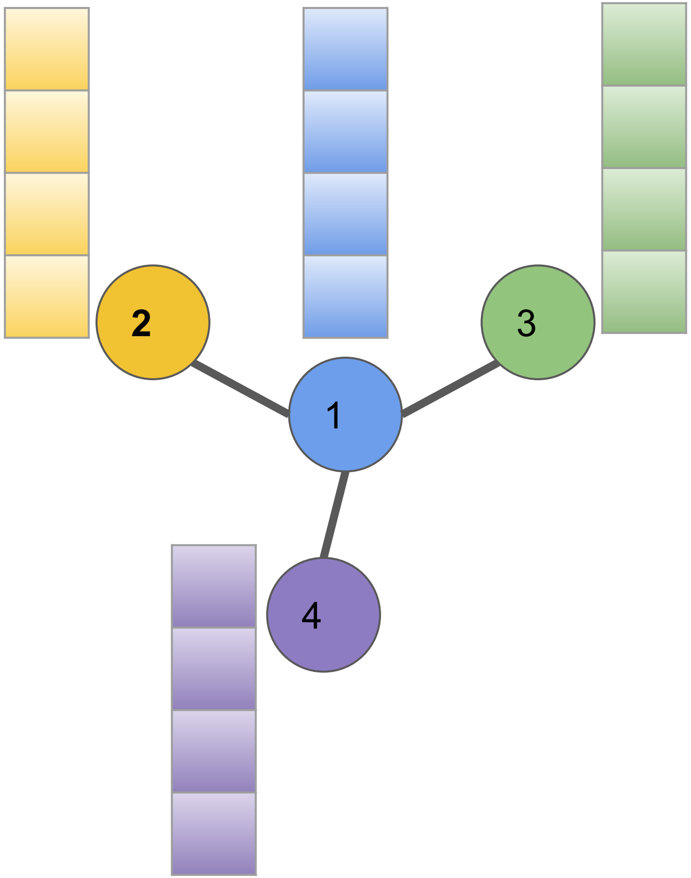
MEAN: element-wise mean-pooling

Graph Convolutional Network



$$H_1^{(l+1)} = \sigma(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + H_4^{(l)}W^{(l)})$$

Graph Convolutional Network

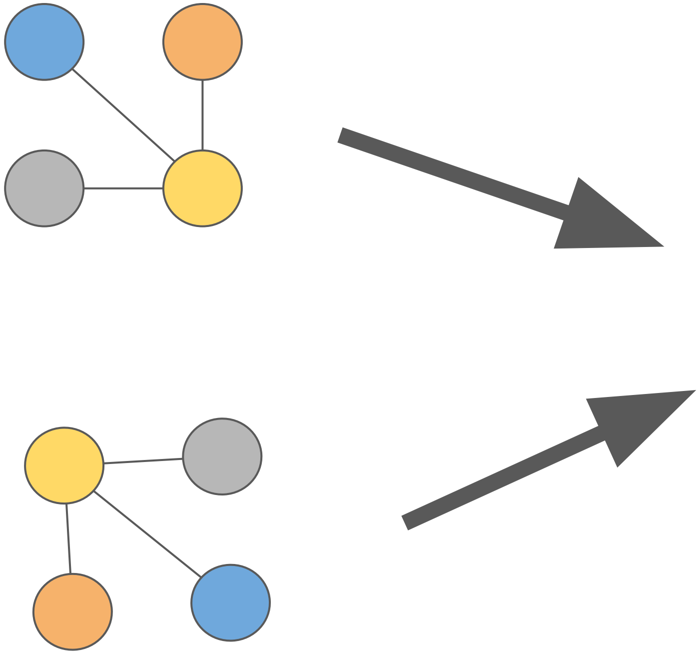


$$H_1^{(l+1)} = \sigma(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + H_4^{(l)}W^{(l)})$$

$$H_1^{(l+1)} = \sigma\left(\sum H_j^{l} W^{(l)} + b^{(l)}\right)$$

Graph Convolutional Network

ReadOut - Permutation Invariance



$$z_G = \tau\left(\sum_{i \in G} MLP\left(H_i^{(L)}\right)\right)$$

GCN code

```python
class GCNLayer(nn.Module):

    def __init__(self, in_dim, out_dim, n_atom, act=None, bn=False):
        super(GCNLayer, self).__init__()

        self.use_bn = bn
        self.linear = nn.Linear(in_dim, out_dim)
        nn.init.xavier_uniform_(self.linear.weight)
        self.bn = nn.BatchNorm1d(n_atom)
        self.activation = act

    def forward(self, x, adj):
        out = self.linear(x)
        out = torch.matmul(adj, out)
        if self.use_bn:
            out = self.bn(out)
        if self.activation != None:
            out = self.activation(out)
        return out, adj
```

Node feature matrix와 **adjacency matrix**의 **list**를 받아 **graph convolution** 연산을 수행

GCN code

```python
class GCNBlock(nn.Module):

    def __init__(self, n_layer, in_dim, hidden_dim, out_dim, n_atom, bn=True, sc='gsc'):
        super(GCNBlock, self).__init__()

        self.layers = nn.ModuleList()
        for i  in range(n_layer):
            self.layers.append(GCNLayer(in_dim if i==0 else hidden_dim,
                                        out_dim if i==n_layer-1 else hidden_dim,
                                        n_atom,
                                        nn.ReLU() if i!=n_layer-1 else None,
                                        bn))
        self.relu = nn.ReLU()

    def forward(self, x, adj):
        for i, layer in enumerate(self.layers):
            out, adj = layer((x if i==0 else out), adj)

        out = self.relu(out)
        return out, adj
```

GCN code

```python
class ReadOut(nn.Module):

    def __init__(self, in_dim, out_dim, act=None):
        super(ReadOut, self).__init__()

        self.in_dim = in_dim
        self.out_dim= out_dim

        self.linear = nn.Linear(self.in_dim,
                                self.out_dim)
        nn.init.xavier_uniform_(self.linear.weight)
        self.activation = act

    def forward(self, x):
        out = self.linear(x)
        out = torch.sum(out, 1)
        if self.activation != None:
            out = self.activation(out)
        return out
```

GCN code

```
class Predictor(nn.Module):

    def __init__(self, in_dim, out_dim, act=None):
        super(Predictor, self).__init__()

        self.in_dim = in_dim
        self.out_dim = out_dim

        self.linear = nn.Linear(self.in_dim,
                                self.out_dim)
        nn.init.xavier_uniform_(self.linear.weight)
        self.activation = act

    def forward(self, x):
        out = self.linear(x)
        if self.activation != None:
            out = self.activation(out)
        return out
```

GCN code

```
class GCNNet(nn.Module):

    def __init__(self, args):
        super(GCNNet, self).__init__()

        self.blocks = nn.ModuleList()
        for i in range(args.n_block):
            self.blocks.append(GCNBlock(args.n_layer,
                                        args.in_dim if i==0 else args.hidden_dim,
                                        args.hidden_dim,
                                        args.hidden_dim,
                                        args.n_atom,
                                        args.bn
                                        ))
        self.readout = ReadOut(args.hidden_dim,
                               args.pred_dim1,
                               act=nn.ReLU())
        self.pred1 = Predictor(args.pred_dim1,
                               args.pred_dim2,
                               act=nn.ReLU())
        self.pred2 = Predictor(args.pred_dim2,
                               args.pred_dim3,
                               act=nn.Tanh())
        self.pred3 = Predictor(args.pred_dim3,
                               args.out_dim)

    def forward(self, x, adj):
        for i, block in enumerate(self.blocks):
            out, adj = block((x if i==0 else out), adj)
        out = self.readout(out)
        out = self.pred1(out)
        out = self.pred2(out)
        out = self.pred3(out)
        return out
```

GraphAttention

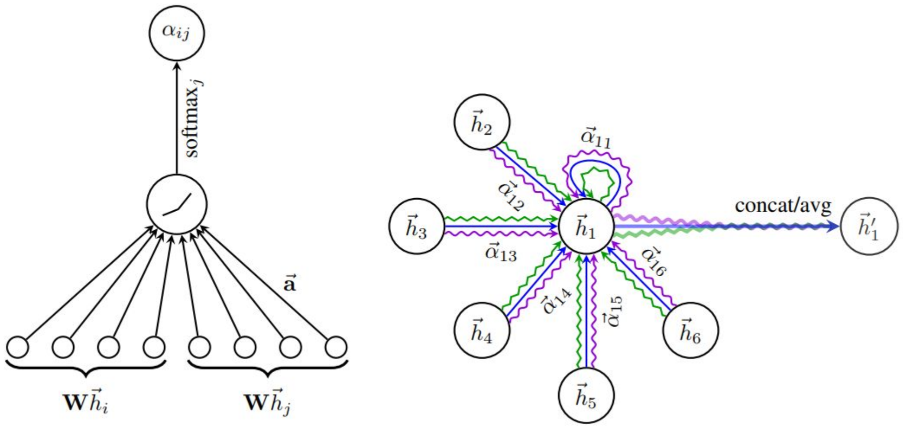


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

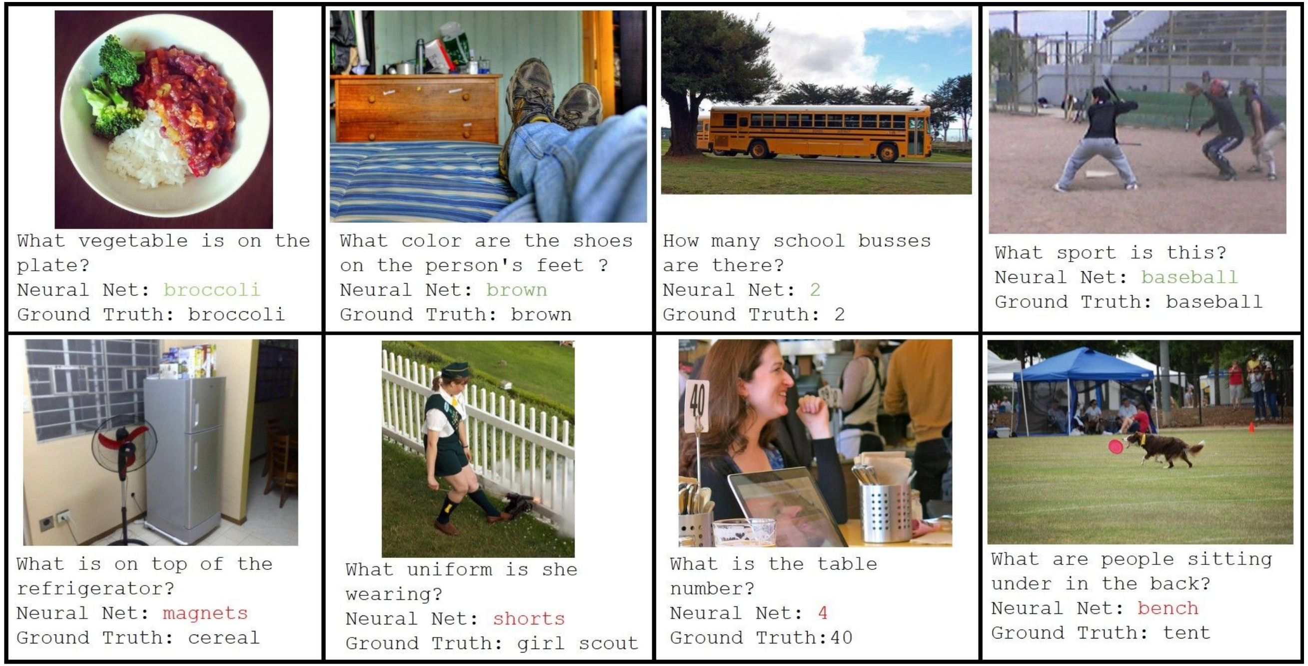
Attention Coefficients

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

Combine function

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

VQA

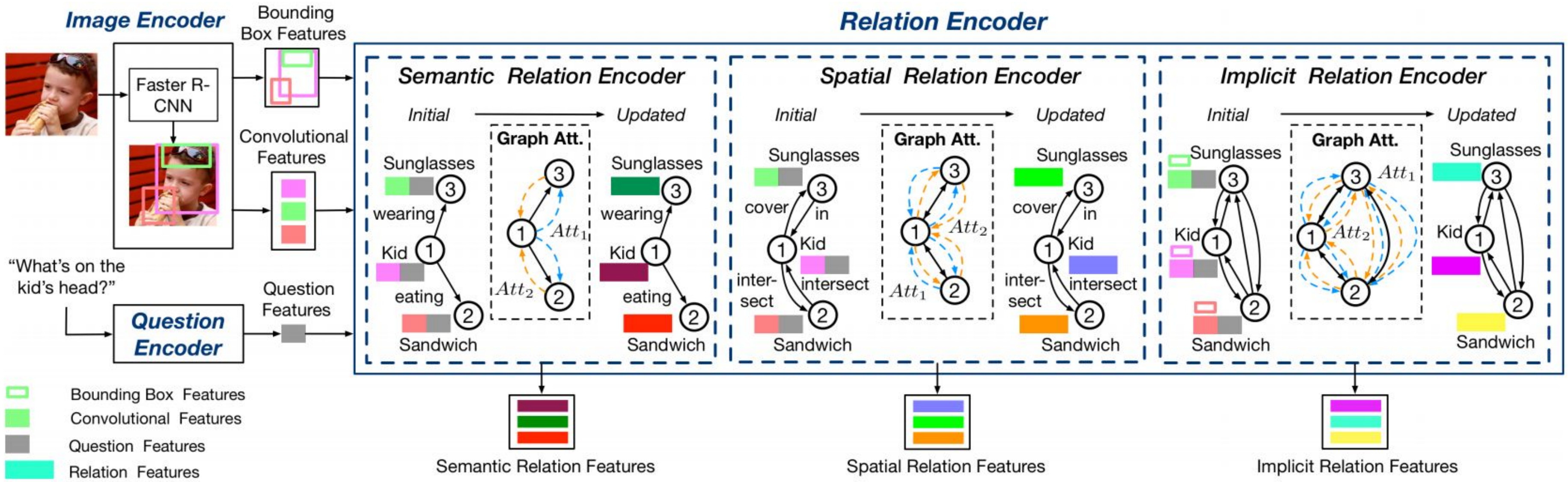


VQA aims to train a model that can achieve comprehensive and semantically-aligned understanding of multimodal input.

VQA

Want to solve:

ReGAT



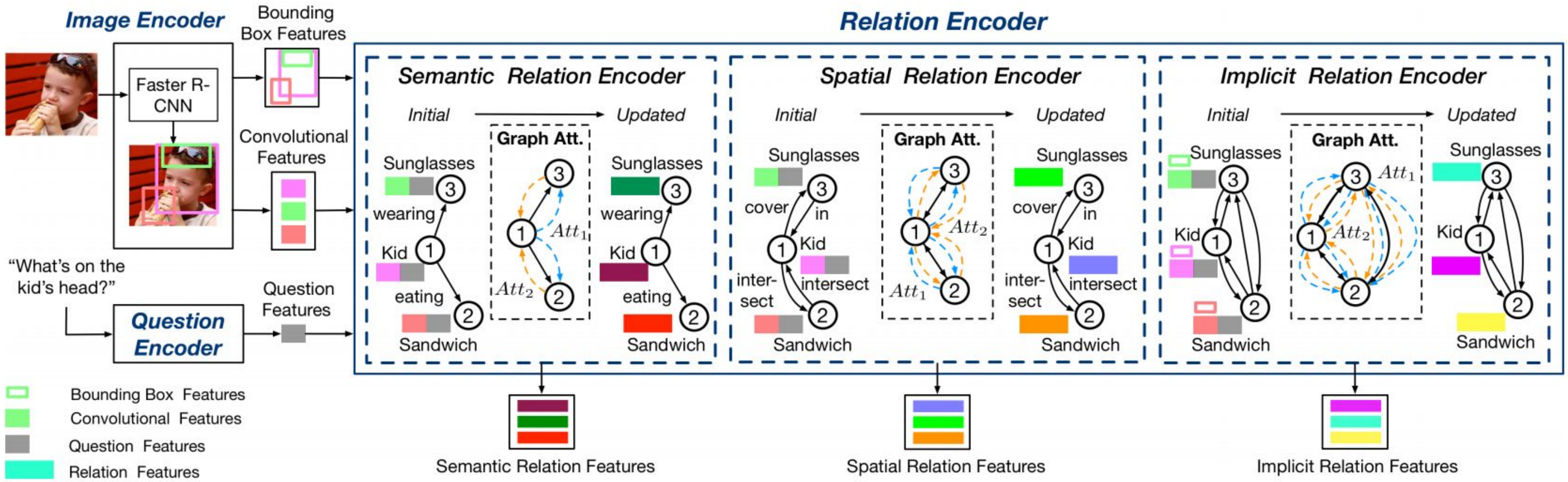
Explicit(명시적) relations

- GAT allows for assigning different importance to nodes of the same neighborhood.

Implicit(암시적) relations

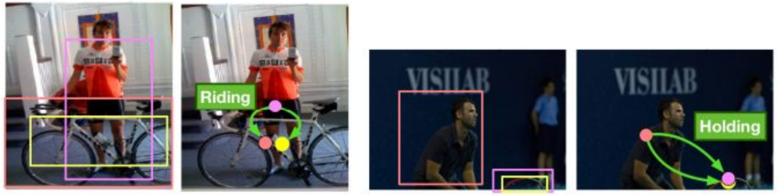
- Adaptive to each question by filtering out question-irrelevant relations, instead of treating all the relations equally as in

ReGAT



- Visual feature vector, $v_i \in R^{dv}$, set of objects $\mathcal{V} = \{v_i\}_{i=1}^{K}$ extracted from Fast R-CNN (K=36, dv=2048)
- Bounding-box feature vector $b_i \in \mathbb{R}^{db}$, $b_i = [x, y, w, h]$ to 4-dimension
- Question embedding $q \in \mathbb{R}^{dq}$, $d_q = 1024$ with self-attention, Bi-GRU

ReGAT



Q: Is this the typical fashion for riding this bike?
A: Yes

Q: What is he holding?
A: Tennis Racket

(a) Semantic Relation

Q: What's the clock attached to?
A: Pole

Q: Are his feet touching the skateboard?
A: No

(b) Spatial Relation

Q: Where is the vase?
A: On the table

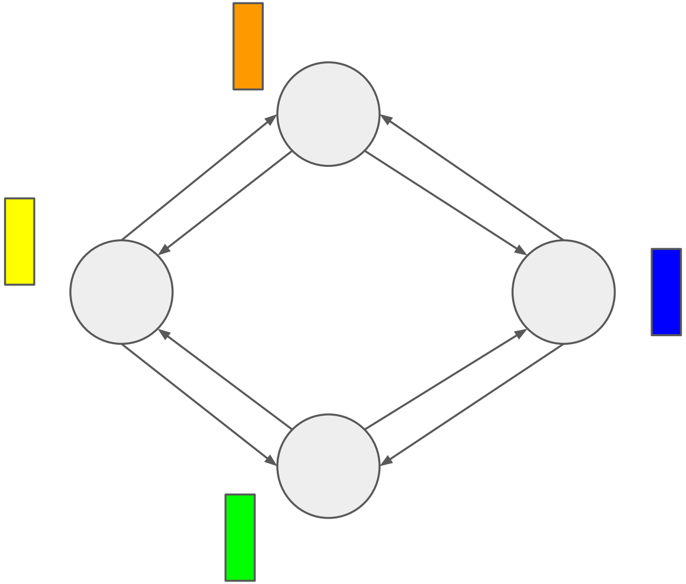
Q: Should the people be walking according to the light?
A:No

(c) Implicit Relation

ReGAT(Graph Construction)

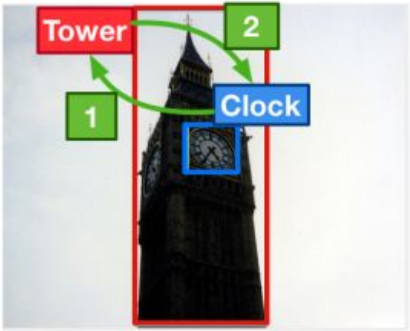
Implicit Graph

- Each object in the image as on vertex, we can construct a fully-connected undirected graph (V, E), E is the set of K(K-1) edges

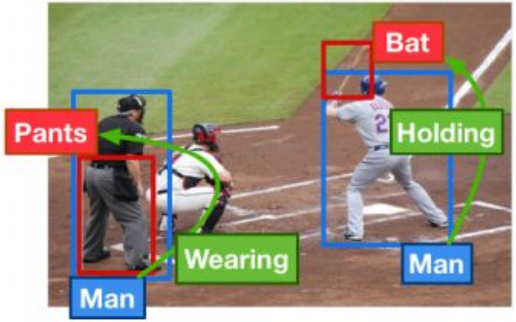


ReGAT(Graph Construction)

Explicit Graph



(a) Spatial Relation



(b) Semantic Relation

- Spatial Relation

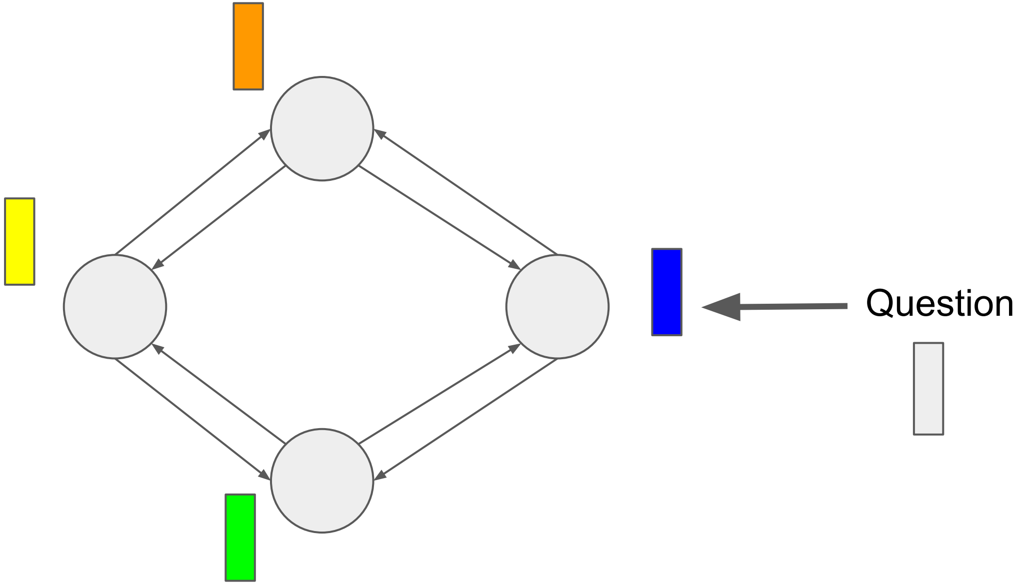
$$spa_i = < object_i - predicate - object_j >$$

- Semantic Relation

$$< subject - predicate - object >$$

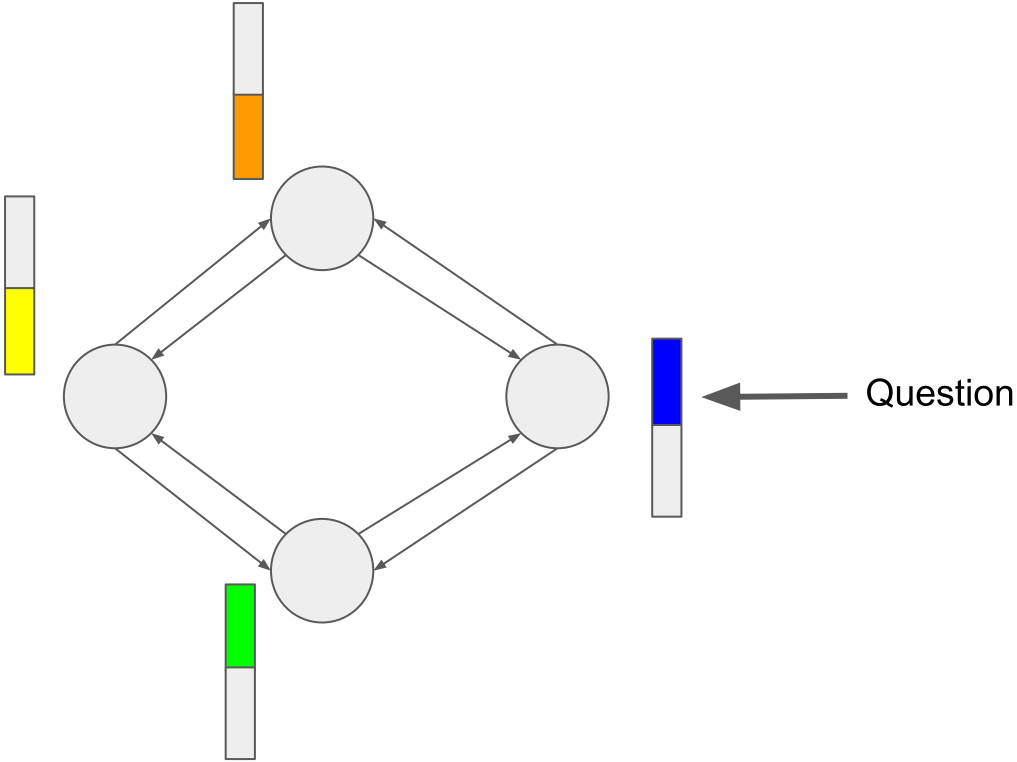
ReGAT(Relation)

Implicit Relation



ReGAT(Relation)

Implicit Relation



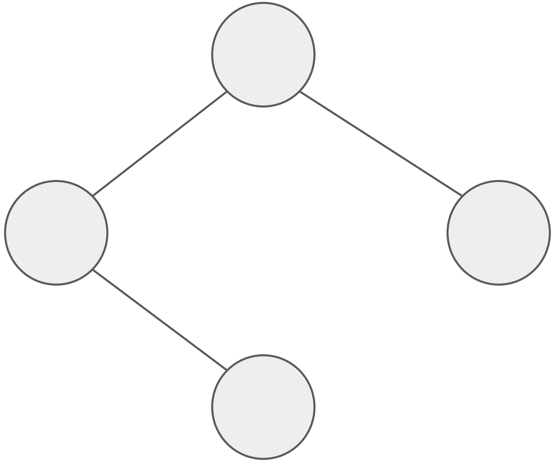
$$\boldsymbol{v}_i' = [\boldsymbol{v}_i || \boldsymbol{q}] \quad \text{for } i = 1, \dots, K\,.$$

$$\boldsymbol{v}_i^\star = \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot \mathbf{W}\boldsymbol{v}_j'\Big)\,.$$

$$\boldsymbol{v}_i^\star = \|_{m=1}^{M} \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^m \cdot \mathbf{W}^m \boldsymbol{v}_j'\Big)$$

ReGAT(Relation)

Explicit Relation



$$v_i^{\star} = \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot (\mathbf{W}_{dir(i,j)} \boldsymbol{v}'_j + \boldsymbol{b}_{lab(i,j)})\Big), \quad (8)$$

$$\alpha_{ij} = \frac{\exp((\mathbf{U}\boldsymbol{v}'_i)^\top \cdot \mathbf{V}_{dir(i,j)} \boldsymbol{v}'_j + \boldsymbol{c}_{lab(i,j)})}{\sum_{j \in \mathcal{N}_i} \exp((\mathbf{U}\boldsymbol{v}'_i)^\top \cdot \mathbf{V}_{dir(i,j)} \boldsymbol{v}'_j + \boldsymbol{c}_{lab(i,j)})},$$

Multimodal Fusion

$$\mathbf{J} = f(\boldsymbol{v}^{\star}, \boldsymbol{q}; \Theta) \tag{9}$$

where f is a multi-modal fusion method and Θ are trainable parameters of the fusion module.

- Bottom-up Top-down, CVPR’18
- Multimodal Tucker Fusion, ICCV’17
- Bilinear Attention Network, NIPS’18

$$\begin{aligned} Pr(a = a_i) = \alpha Pr_{sem}(a = a_i) + \beta Pr_{spa}(a = a_i) \\ + (1 - \alpha - \beta) Pr_{imp}(a = a_i)\,, \end{aligned} \tag{10}$$

where α and β are trade-off hyper-parameters ($0 \leq \alpha + \beta \leq 1, 0 \leq \alpha, \beta \leq 1$). $Pr_{sem}(a = a_i)$, $Pr_{spa}(a = a_i)$ and $Pr_{imp}(a = a_i)$ denote the predicted probability for answer a_i, from the model trained with semantic, spatial and implicit relations, respectively.

Result

물 들어올 때 노 젓자!!!

감사합니다