BUSINESS REPORT

硕士生学术交流

汇报人：石佳妍

时间：2020年7月29日

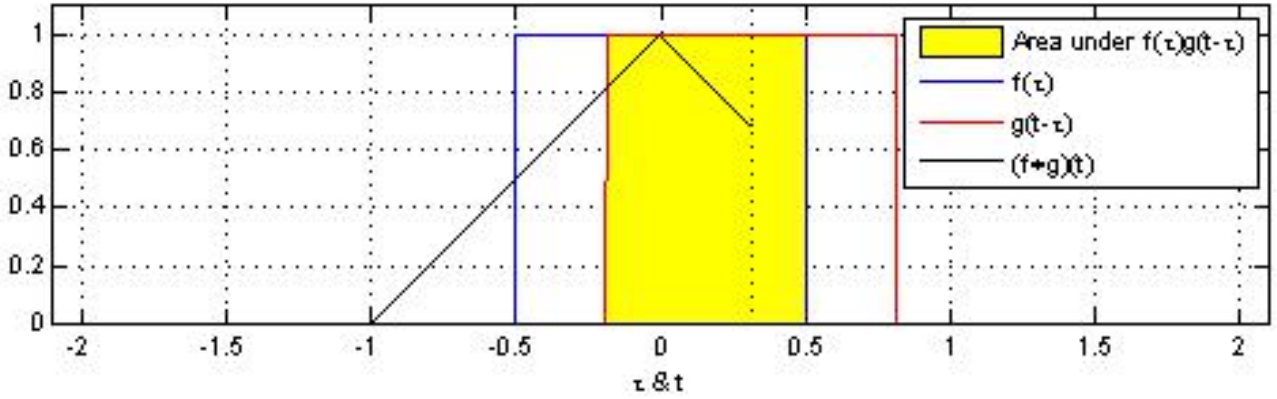
图卷积网络(GCN)

目录

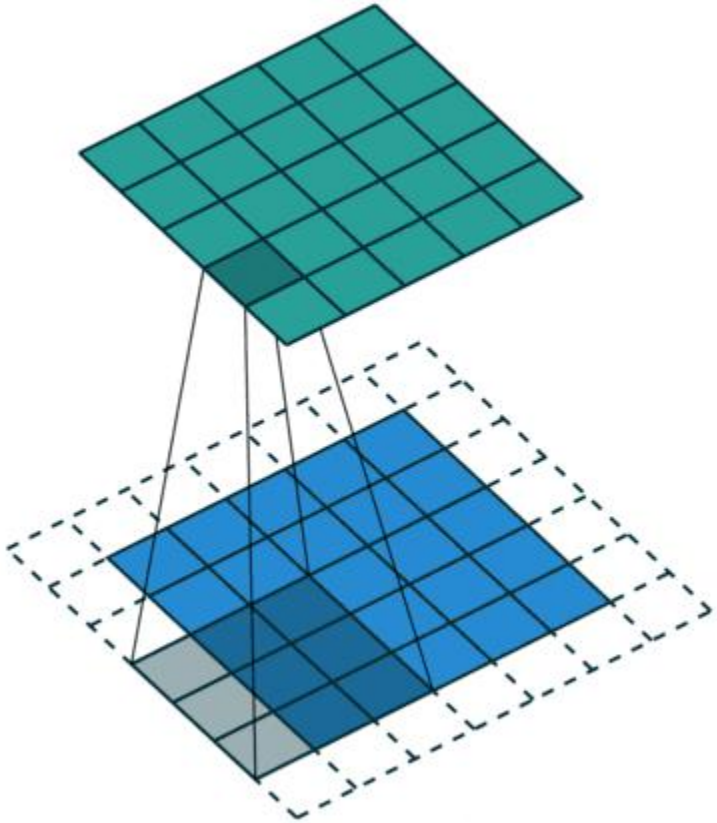
卷积介绍和作用

Convolution的数学定义： $$(f*g)(t)=\int_{\mathbb{R}} f(x)g(t-x)dx$$

一般称g为作用在f上的filter或kernel



一维的卷积示意图



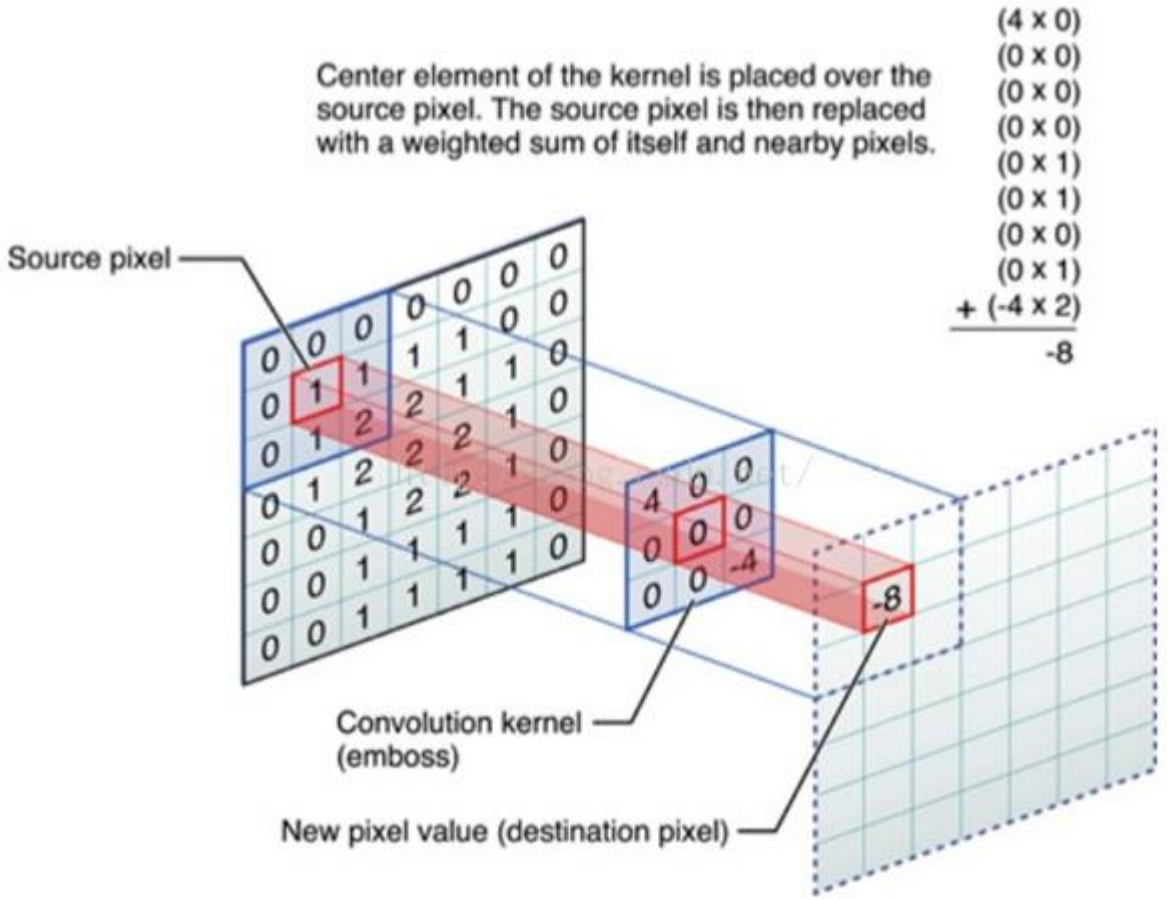
二维的卷积示意图

卷积介绍和作用

离散卷积本质就是一种加权求和

CNN中的卷积本质上是利用一个共享参数的过滤器(kernel)，通过计算中心像素点以及相邻像素点的加权和来构成feature map实现空间特征的提取

卷积核的参数通过优化求出才能实现特征提取的作用，GCN的理论很大一部分工作就是为了引入可以优化的卷积参数

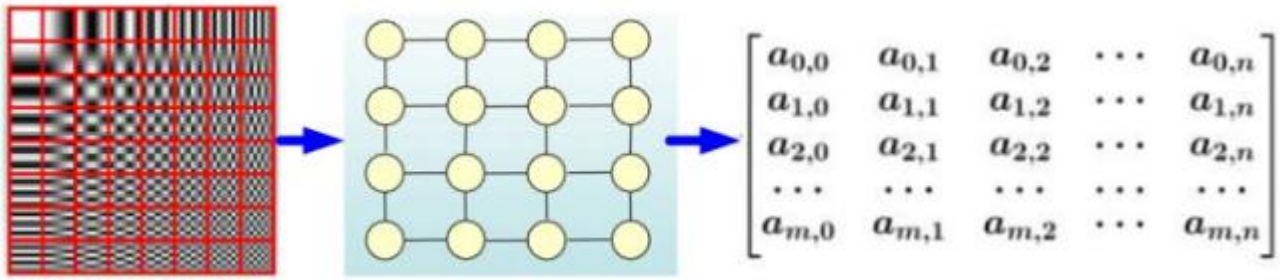


CNN中卷积提取feature map示意图

GCN的研究原因

➢ **GCN中的Graph指什么？**

Graph Convolutional Network中的Graph是指数学（图论）中的用顶点和边建立相应关系的拓扑图



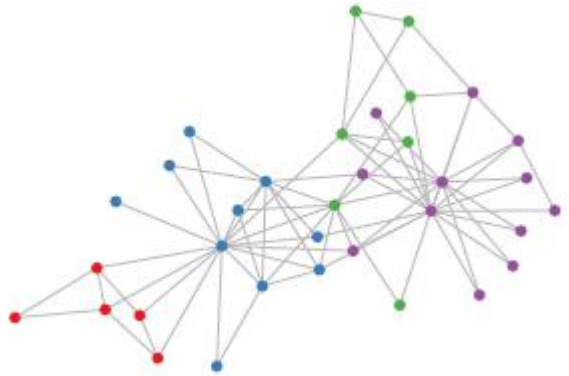
图像矩阵示意图（Euclidean Structure）

➢ **为什么研究GCN：**

(1)CNN无法处理Non Euclidean Structure的数据，传统的离散卷积在Non Euclidean Structure的数据上无法保持平移不变性

(2)需要在Non Euclidean Structure的数据结构（拓扑图）上有效地提取空间特征来进行机器学习

(3)拓扑连接是一种广义的数据结构，GCN有很大的应用空间



社交网络拓扑示意图（Non Euclidean Structure）

提取拓扑图空间特征的两种方式

(1)vertex domain

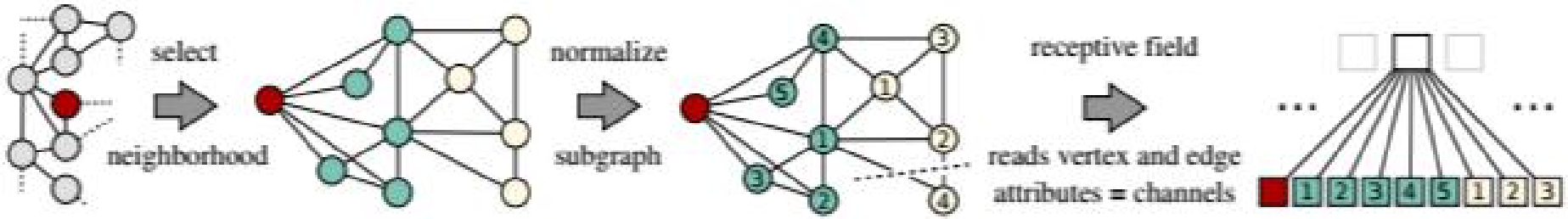
提取拓扑图上的空间特征，把每个顶点相邻的neighbors找出来

需要解决的问题：

- 寻找中心vertex的neighbors的条件(确定receptive field的方法)
- 处理包含不同数目neighbors的特征的方式

主要缺点：

- 每个顶点提取出来的neighbors不同，使得计算处理必须针对每个顶点
- 提取特征的效果可能没有卷积好



vertex domain提取空间特征示意图

提取拓扑图空间特征的两种方式

(2)spectral domain　GCN的理论基础

- 思路：借助图谱的理论来实现拓扑图上的卷积操作
- 研究进程

- 定义graph上的Fourier Transformation
- 定义graph上的convolution
- 与深度学习结合提出了Graph Convolutional Network

两个问题：

1. 什么是Spectral graph theory：

借助于图的拉普拉斯矩阵的特征值和特征向量来研究图的性质

（详细解读：https://link.zhihu.com/?target=https%3A//en.wikipedia.org/wiki/Spectral_graph_theory）

2. 为什么使用Spectral graph theory

图（Graph）上的热传播模型

一维的温度传播的模型：

$$\frac{d\phi_i}{dt} = k(\phi_{i+1} - \phi_i) - k(\phi_i - \phi_{i-1})$$

（k和单元的比热容、质量有关是个常数）

$$\frac{d\phi_i}{dt} - k[(\phi_{i+1} - \phi_i) - (\phi_i - \phi_{i-1})] = 0$$

$$\frac{\partial \phi}{\partial t} - k\frac{\partial^2 \phi}{\partial x^2} = 0$$

$$\frac{\partial \phi}{\partial t} - k\Delta\phi = 0$$

事实：

1、在欧氏空间中，某个点温度升高的速度正比于该点周围的温度分布，用拉普拉斯算子衡量。

2、拉普拉斯算子，是二阶导数对高维空间的推广

Δ这个符号代表的是对各个坐标二阶导数的加和，现在的主流写法也可以写作 ∇^2

图（Graph）上的热传播模型的推广

$$\frac{d\phi_i}{dt} = -k\sum_j A_{ij}(\phi_i - \phi_j)$$ A为邻接矩阵

$$\frac{d\phi_i}{dt} = -k[\phi_i \sum_j A_{ij} - \sum_j A_{ij}\phi_j]$$

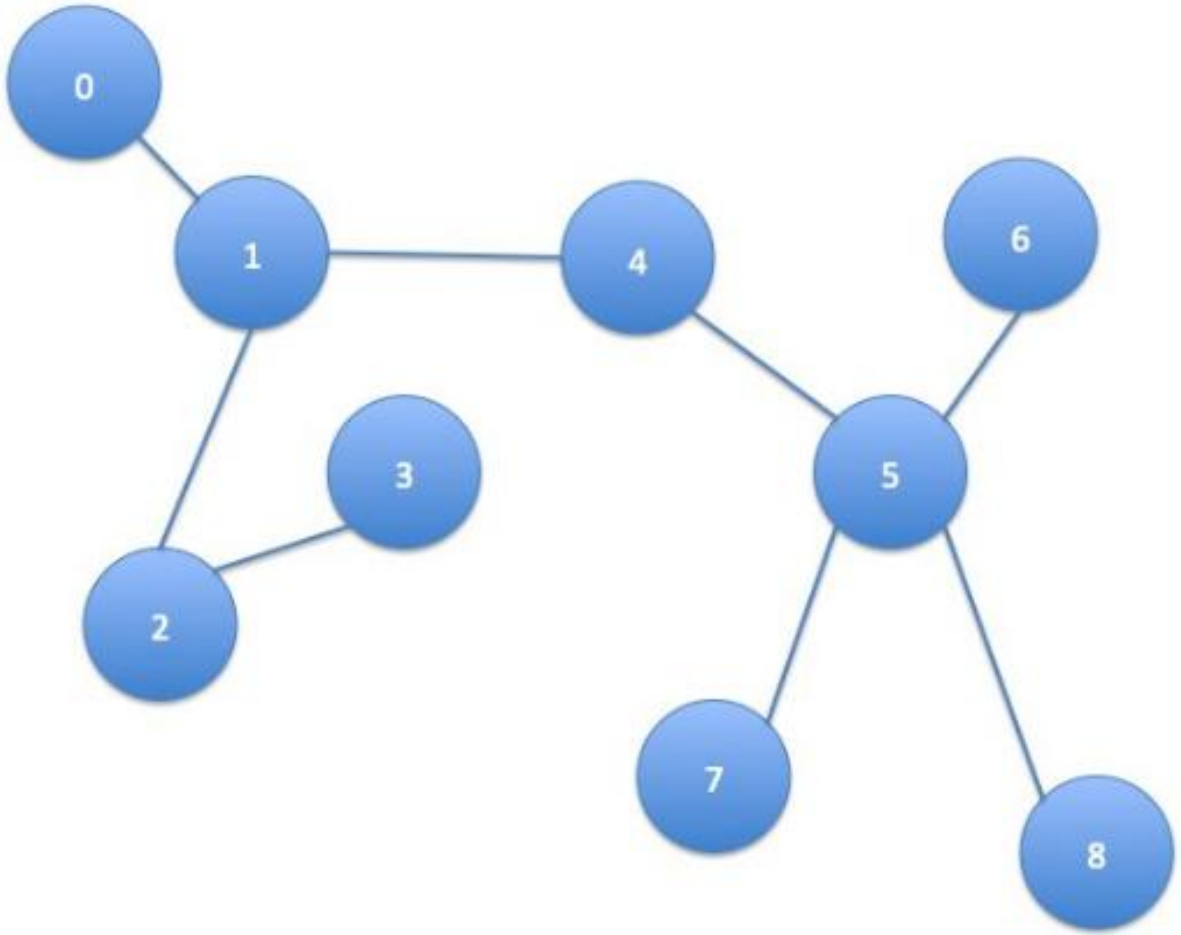
$$= -k[deg(i)\phi_i - \sum_j A_{ij}\phi_j]$$

$$\begin{bmatrix} \frac{d\phi_1}{dt} \\ \frac{d\phi_2}{dt} \\ \cdots \\ \frac{d\phi_n}{dt} \end{bmatrix} = -k \begin{bmatrix} deg(1) \times \phi_1 \\ deg(2) \times \phi_2 \\ \cdots \\ deg(n) \times \phi_n \end{bmatrix} + kA \begin{bmatrix} \phi_1 \\ \phi_2 \\ \cdots \\ \phi_n \end{bmatrix}$$

$$\boldsymbol{\phi} = [\phi_1, \phi_2, \ldots, \phi_n]^T \quad \frac{d\boldsymbol{\phi}}{dt} = -kD\boldsymbol{\phi} + kA\boldsymbol{\phi} = -k(D-A)\boldsymbol{\phi}$$

$$D = diag(deg(1), deg(2), \ldots, deg(n))$$

$$\frac{d\boldsymbol{\phi}}{dt} + kL\boldsymbol{\phi} = 0$$



$$\frac{d\boldsymbol{\phi}}{dt} + kL\boldsymbol{\phi} = 0$$ 刻画拓扑空间有限结点

$$\frac{\partial\phi}{\partial t} - k\Delta\phi = 0$$ 刻画欧氏空间的连续分布

拉普拉斯矩阵 L=D-A

推广到GCN

GCN的实质：

在一张Graph Network中特征（Feature）和消息（Message）中的流动和传播

该传播最原始的形态：状态的变化正比于相应空间（这里是Graph空间）拉普拉斯算子作用在当前的状态。

➢ 建模方面

- 牛顿冷却定律
- 引入核函数
- 引入神经网络

更非线性更能刻画复杂关系

➢ 解决方面

Fourier变换

把空域问题转化为频域问题

本质：是一种Message Passing，是一种Induction，卷积、傅立叶都是表象和解法。

拉普拉斯矩阵

常用的拉普拉斯矩阵实际有三种：

$$L = D - A$$ Combinatorial Laplacian

$$L^{sys} = D^{-1/2} L D^{-1/2}$$ Symmetric normalized Laplacian

$$L^{rw} = D^{-1} L$$ Random walk normalized Laplacian

为什么GCN用拉普拉斯矩阵

（1）拉普拉斯矩阵是对称矩阵，可以进行特征分解（谱分解），这就和GCN的spectral domain对应上了

（2）拉普拉斯矩阵只在中心顶点和一阶相连的顶点上（1-hop neighbor）有非0元素，其余之处均为0

（3）通过拉普拉斯算子与拉普拉斯矩阵进行类比

拉普拉斯矩阵的谱分解（特征分解）

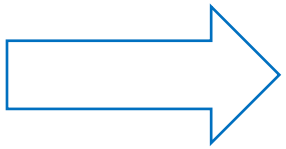
拉普拉斯矩阵是半正定对称矩阵，可以进行特征分解，且分解后有特殊的形式

$$L = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^{-1}$$

$U = (\vec{u_1}, \vec{u_2}, \cdots, \vec{u_n})$ 是列向量为单位特征向量的矩阵，也就说 $\vec{u_i}$ 是列向量

$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$ 是n个特征值构成的对角阵

$UU^T = E$



$$L = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^T$$

类比到Graph上的傅里叶变换及卷积

推广傅里叶变换

传统的傅里叶变换： $$F(\omega) = \mathcal{F}[f(t)] = \int f(t)e^{-i\omega t}dt$$

拉普拉斯算子满足： $$\Delta e^{-i\omega t} = \frac{\partial^2}{\partial t^2}e^{-i\omega t} = -\omega^2 e^{-i\omega t}$$

Graph上的傅里叶变换： $$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^{N} f(i)u_l^*(i)$$

- f: Graph上的N维向量，f(i)与Graph的顶点一一对应
- ul(i)：第l个特征向量的第i个分量

λ_l 下的f的Graph傅里叶变换就是与 λ_l 对应的特征向量ul进行内积运算

$$\begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix}$$

$$\Rightarrow \quad \hat{f} = U^T f \qquad (a)$$

类比到Graph上的傅里叶变换及卷积

推广傅里叶变换

Graph上的傅里叶逆变换

传统的傅里叶逆变换：$$\mathcal{F}^{-1}[F(\omega)] = \frac{1}{2\Pi}\int F(\omega)e^{i\omega t}d\omega$$

迁移到Graph上：$$f(i) = \sum_{l=1}^{N} \hat{f}(\lambda_l)u_l(i)$$

推广到矩阵形式：

$$\begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_2(1) & \dots & u_N(1) \\ u_1(2) & u_2(2) & \dots & u_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(N) & u_2(N) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} \Rightarrow \quad f = U\hat{f} \qquad (b)$$

推广卷积

卷积定理：函数卷积的傅里叶变换是函数傅里叶变换的乘积，即对于函数f(t)与h(t)两者的卷积是其函数傅里叶变换乘积的逆变换：

$$f * h = \mathcal{F}^{-1}\left[\hat{f}(\omega)\hat{h}(\omega)\right] = \frac{1}{2\Pi}\int \hat{f}(\omega)\hat{h}(\omega)e^{i\omega t}d\omega$$

类比到Graph：

f的傅里叶变换：$\hat{f} = U^T f$

卷积核 h 的傅里叶变换的对角矩阵形式：

$$\begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} \qquad \hat{h}(\lambda_l) = \sum_{i=1}^{N} h(i)u_l^*(i)$$

$\hat{f}$ 与 $\hat{h}(\lambda_l)$ 的乘积：

$$\begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} U^T f$$

逆变换(得到卷积)：

$$(f * h)_G = U \begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} U^T f \qquad (1)$$

很多论文中的Graph卷积公式为：

$$(f * h)_G = U((U^T h) \odot (U^T f)) \qquad (2)$$

⊙ 表示Hadamard product（哈达马积），对于两个维度相同的向量、矩阵、张量进行对应位置的逐元素乘积运算。

式(2)与式(1)是完全等价的

Deep Learning中的Graph Convolution

Deep learning 中的Convolution就是要设计含有trainable共享参数的kernel

第一代GCN

$$diag(\hat{h}(\lambda_l)) \Rightarrow diag(\theta_l)$$

$$y_{output} = \sigma\left(U g_\theta(\Lambda) U^T x\right) \qquad (3)$$

$$g_\theta(\Lambda) = \begin{pmatrix} \theta_1 & & \\ & \ddots & \\ & & \theta_n \end{pmatrix}$$

弊端：

（1）每一次前向传播，都要计算U，$diag(\theta_l)$，U^T 三者的矩阵乘积，特别是对于大规模的graph，计算的代价较高，也就是论文中 $\mathcal{O}(n^2)$ 的计算复杂度

（2）卷积核不具有spatial localization

（3）卷积核需要 n 个参数

Deep Learning中的Graph Convolution

第二代GCN $\hat{h}(\lambda_l) \Rightarrow \sum_{j=0}^{K} \alpha_j \lambda_l^j$

$$y_{output} = \sigma\left(U g_\theta(\Lambda) U^T x\right) \qquad (4)$$

$$g_\theta(\Lambda) = \begin{pmatrix} \sum_{j=0}^{K} \alpha_j \lambda_1^j & & \\ & \ddots & \\ & & \sum_{j=0}^{K} \alpha_j \lambda_n^j \end{pmatrix} = \sum_{j=0}^{K} \alpha_j \Lambda^j$$

$$U \sum_{j=0}^{K} \alpha_j \Lambda^j U^T = \sum_{j=0}^{K} \alpha_j U \Lambda^j U^T = \sum_{j=0}^{K} \alpha_j L^j$$

$L^2 = U\Lambda U^T U\Lambda U^T = U\Lambda^2 U^T$ 且 $U^T U = E$

优点：

（1）卷积核只有 K个参数，一般K远小于 n，参数的复杂度被大大降低了

（2）不需要特征分解，直接使用拉普拉斯矩阵L进行变换

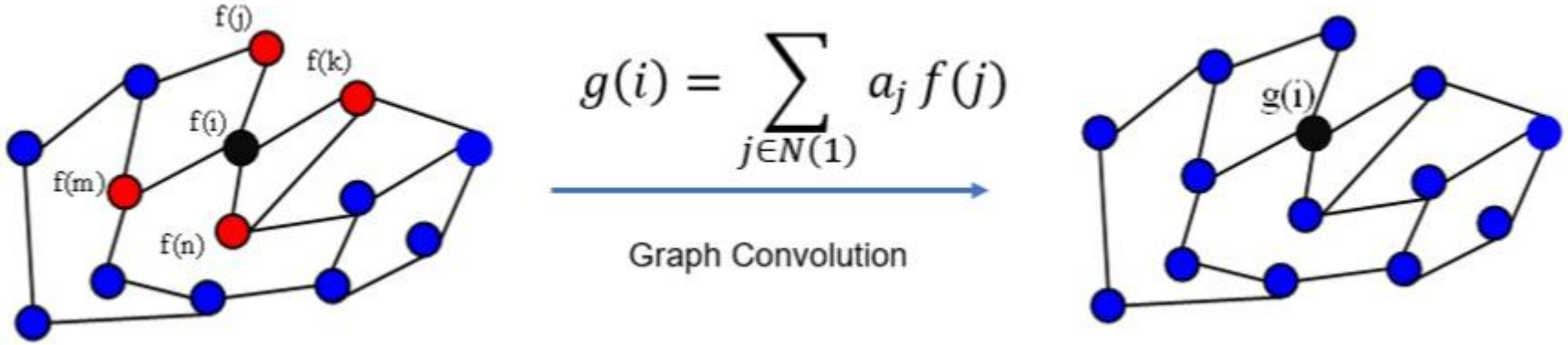
（3）卷积核具有很好的spatial localization，K就是卷积核的receptive field，也就是说每次卷积会将中心顶点K-hop neighbor上的feature进行加权求和，权系数就是α_k

(4)式变为 $$y_{output} = \sigma\left(\sum_{j=0}^{K-1} \alpha_j L^j x\right) \qquad (5)$$

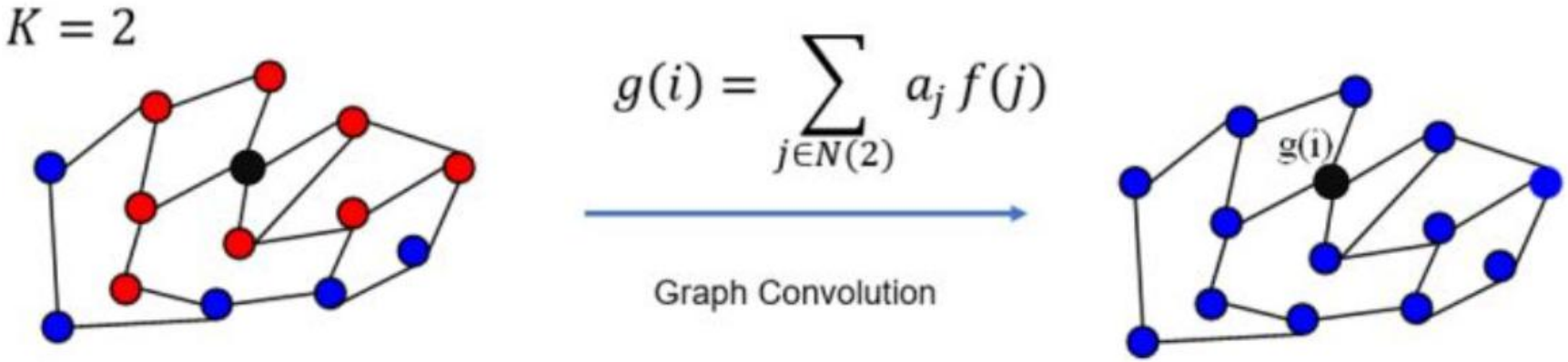
Deep Learning中的Graph Convolution

K=1就是对每个顶点上一阶neighbor的feature进行加权求和

$K = 1$



$K = 2$



应用

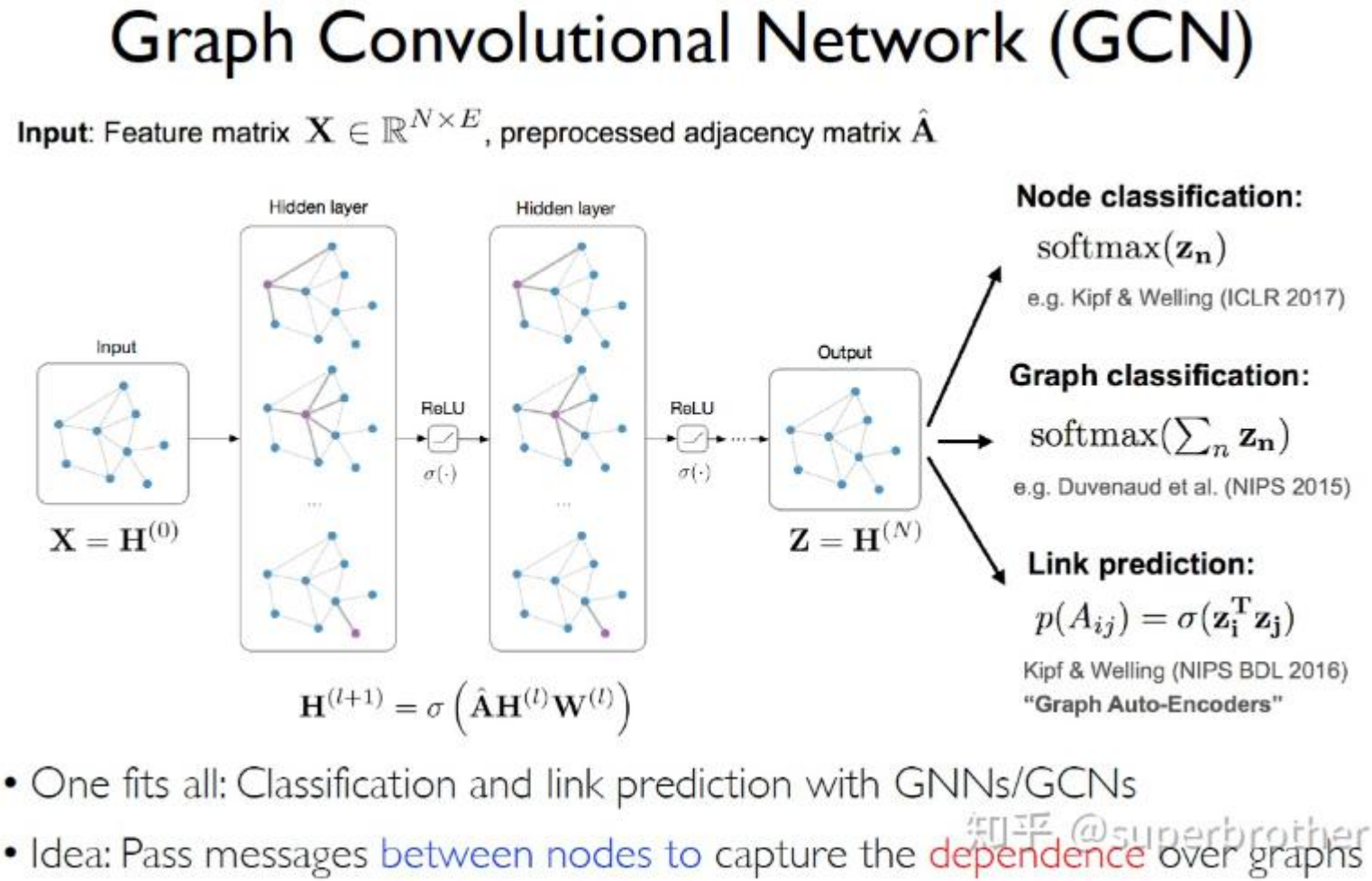
1.进行社交网络相关的研究

处理node-wise的问题：顶点分类

处理子图相关的问题：社区挖掘

处理整个图相关的问题：图分类

处理顶点之间是否有边相连的问题

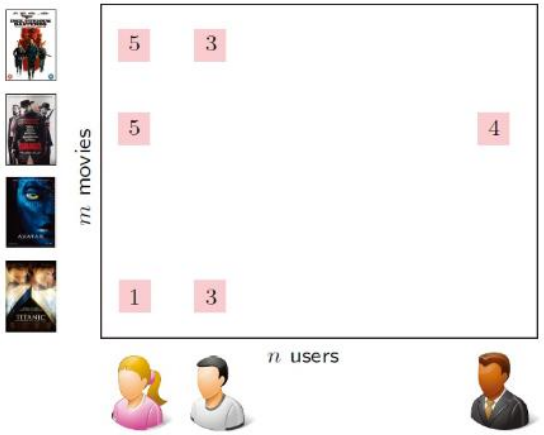


GCN在社交网络中的应用示意图

应用

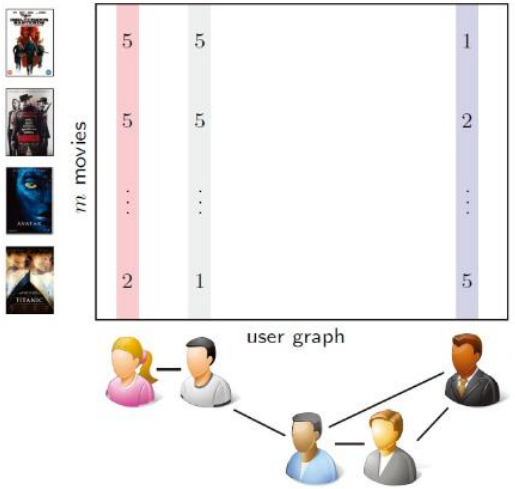
2.推荐系统与社交网络的结合

Matrix completion: 'Netflix challenge'



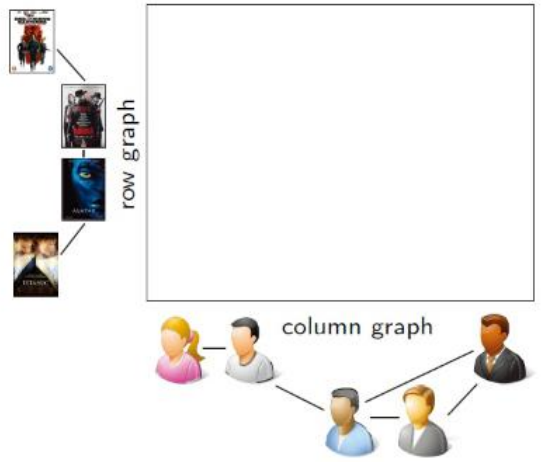
$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \|\mathbf{X}\|_* + \mu\|\boldsymbol{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2$$

Geometric matrix completion



$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \mu\|\boldsymbol{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2 + \mu_{\mathrm{c}}\underbrace{\mathrm{tr}(\mathbf{X}\boldsymbol{\Delta}_{\mathrm{c}}\mathbf{X}^\top)}_{\|\mathbf{X}\|_{\mathcal{G}_{\mathrm{c}}}^2}$$

Multi-graph convolution



Multi-graph spectral convolution

$$\mathbf{X}\star\mathbf{G} = \boldsymbol{\Phi}_{\mathrm{r}}(\hat{\mathbf{X}}\circ\hat{\mathbf{G}})\boldsymbol{\Phi}_{\mathrm{c}}^\top$$

应用

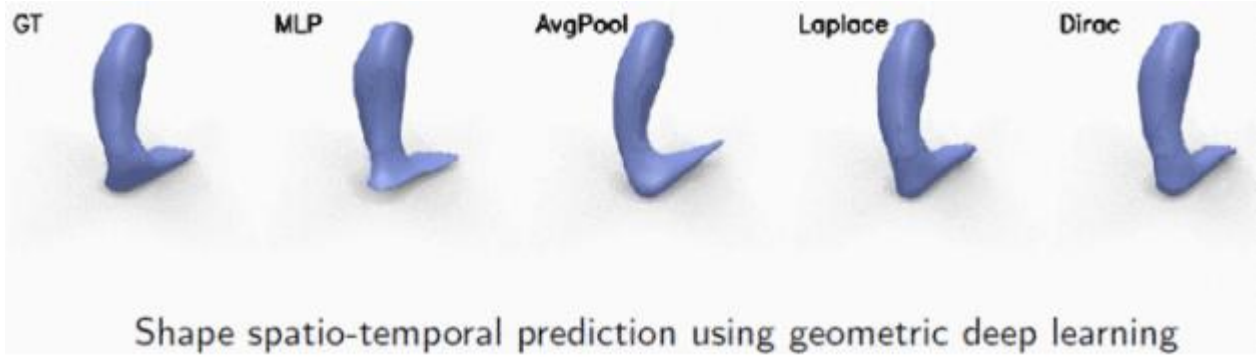
3.CV方向

GCN可以按照三维流形的方向进行convolution

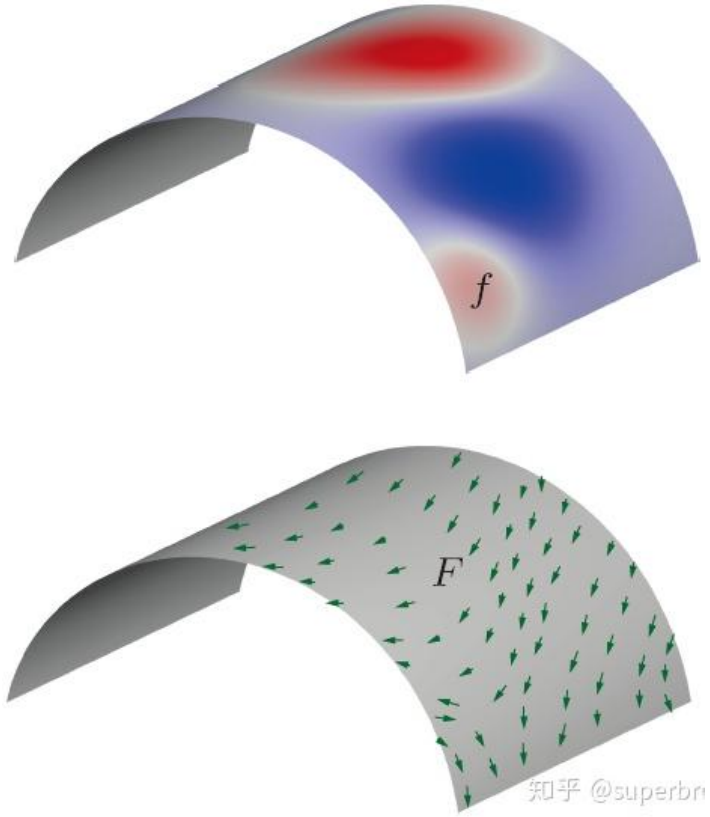
用于已知部分散点进行三维形状的补全：

Shape completion from partial point cloud using a geometric autoencoder

对三维形状时空变化的预测：



Shape spatio-temporal prediction using geometric deep learning

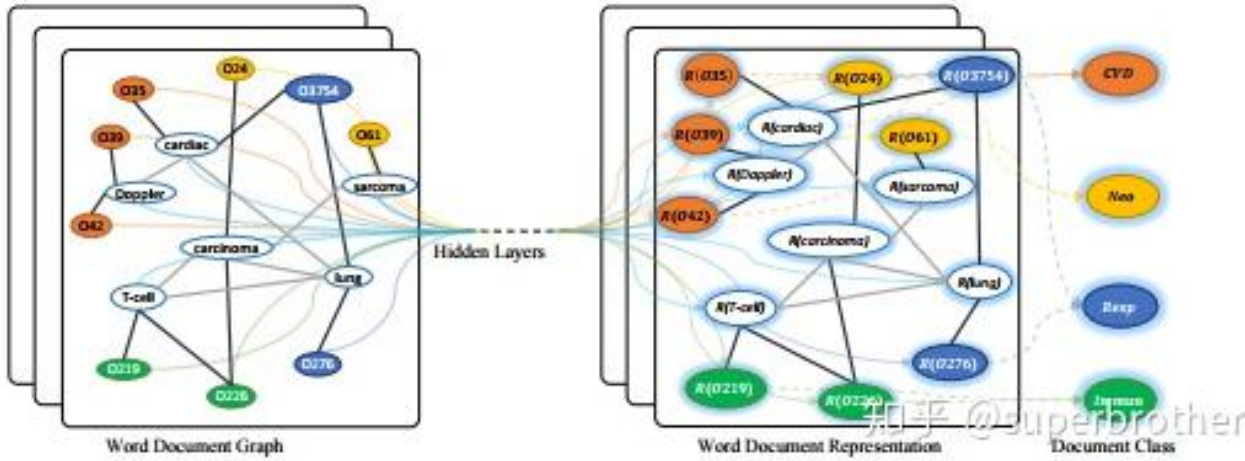


三维流形

应用

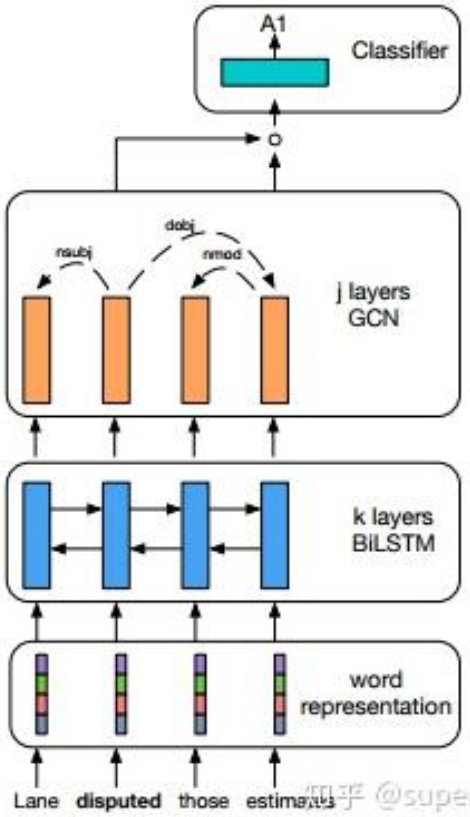
4.NLP方向

Text Classification



文本GCN模型示意图

Semantic Role Labeling



利用GCN+LSTM预测语料的label

应用

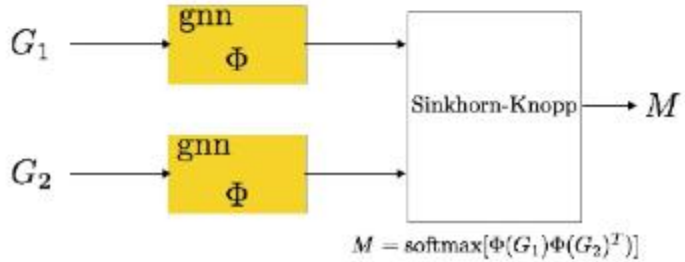
5.求解Quadratic Assignment Problem等经典数学问题

Siamese GNNs for QAP

- Goal: learn data-driven relaxations using GNNs trained on certain random graph families
- Data generation:

$$\mathbf{W}_1 \sim \mu \ , \ \mathbf{W}_2 \,|\, \mathbf{W}_1 = \mathbf{P}\mathbf{W}_1\mathbf{P}^\top + \mathbf{N}$$

  - μ : Erdos-Renyi, Random-Regular
  - $\mathbf{P}$: uniform permuation.
  - $\mathbf{N}$: Erdos-Renyi noise of varying intensity.
- Approach: Map from Quadratic to Linear Assignment using a GNN siamese embedding architecture:



应用

6.对于多智能体相关的研究

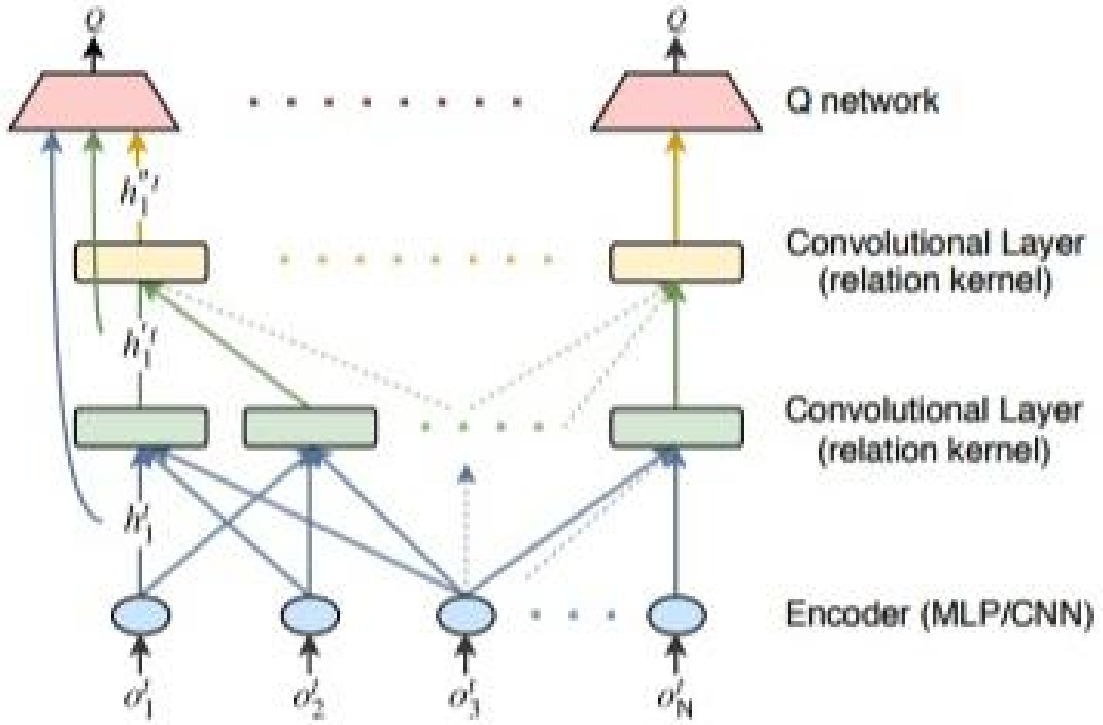
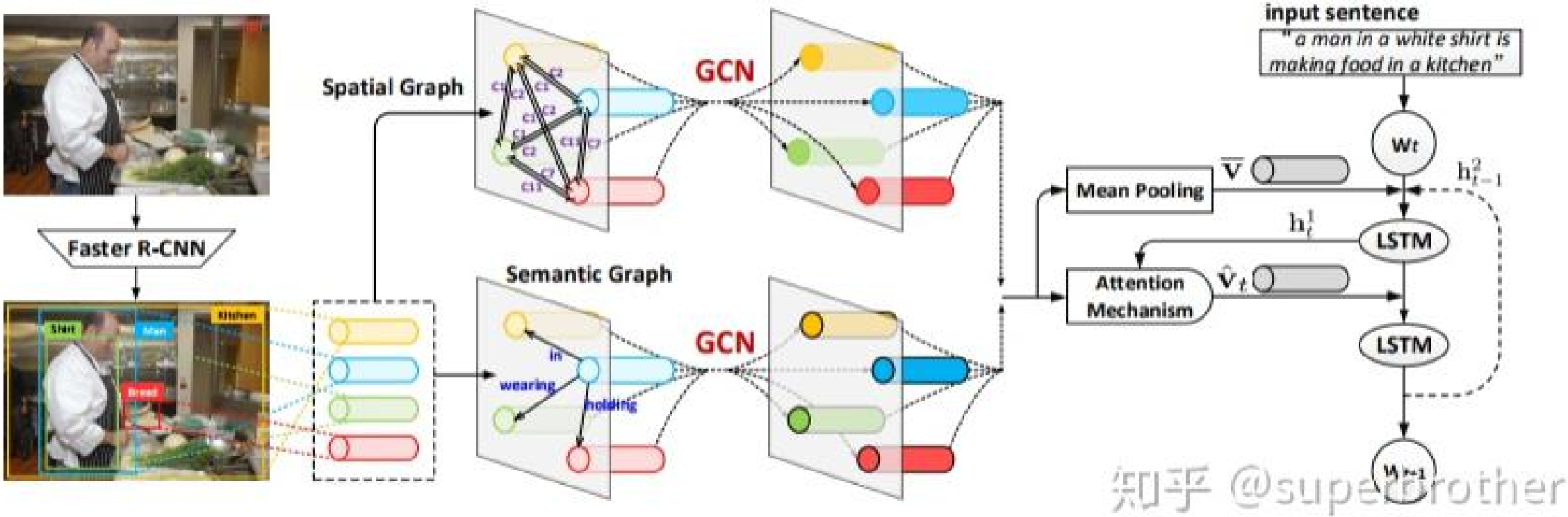


Figure 1: DGN architecture. All agents share weights and gradients are accumulated to update the weights.

应用

7.Image Captioning



参考资料

https://www.zhihu.com/question/54504471/answer/332657604 如何理解 Graph Convolutional Network（GCN）？

https://www.zhihu.com/question/54504471/answer/630639025 如何理解 Graph Convolutional Network（GCN）？

https://www.zhihu.com/question/305395488/answer/554847680 graph convolutional network有什么比较好的应用task？

感谢您的聆听

THANK YOU FOR LISTENING

汇报人：石佳妍　　时间：2020年7月29日