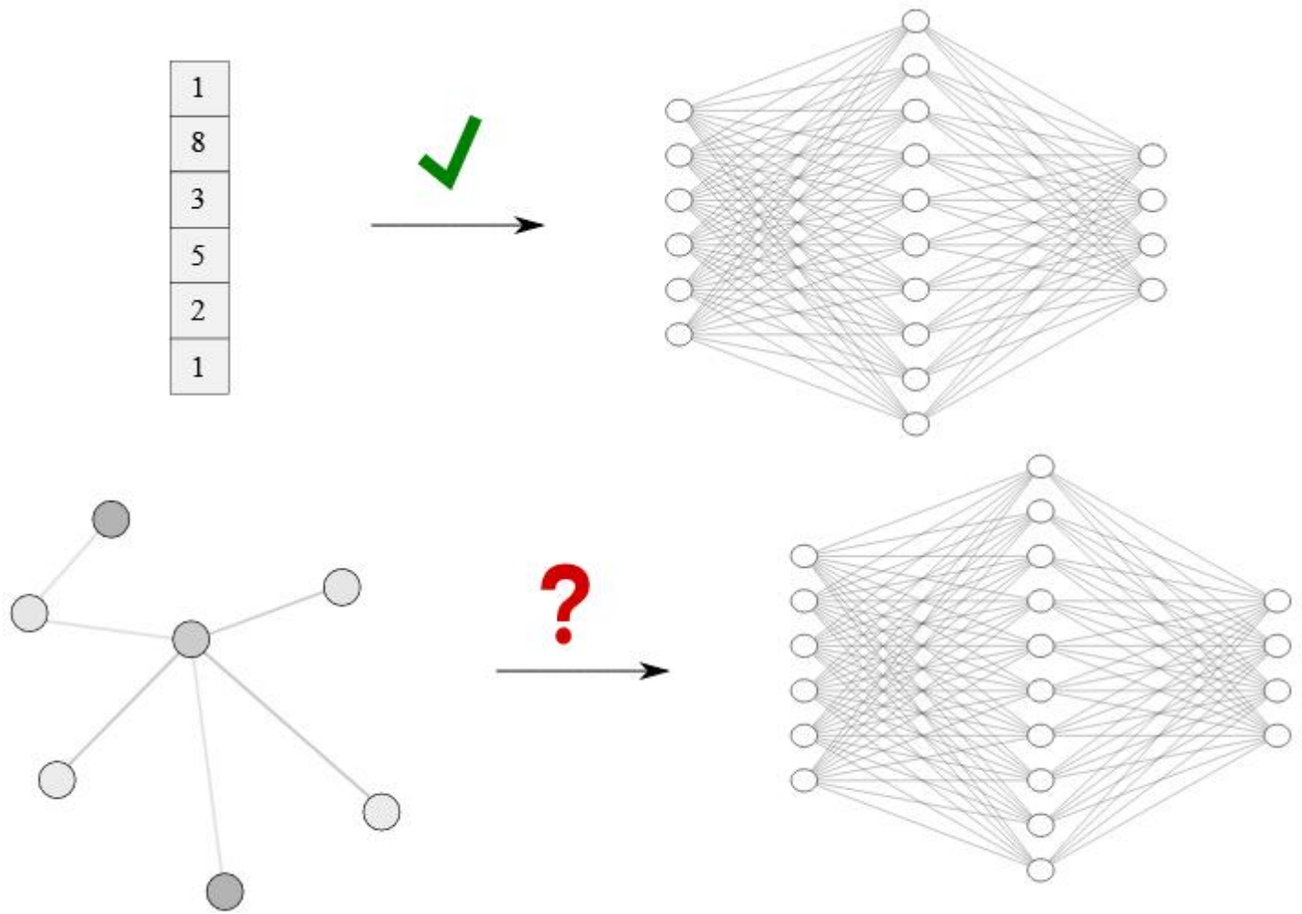
курс «Глубокое обучение»

Графовые нейронные сети

Александр Дьяконов

05 ноября 2022 года

Как подать граф в сеть?



Graph neural networks (GNNs) – нейронные сети, которые обрабатывают графы

есть и другие «более старые» техники (например, случайные блуждания)

Изображение – тоже граф

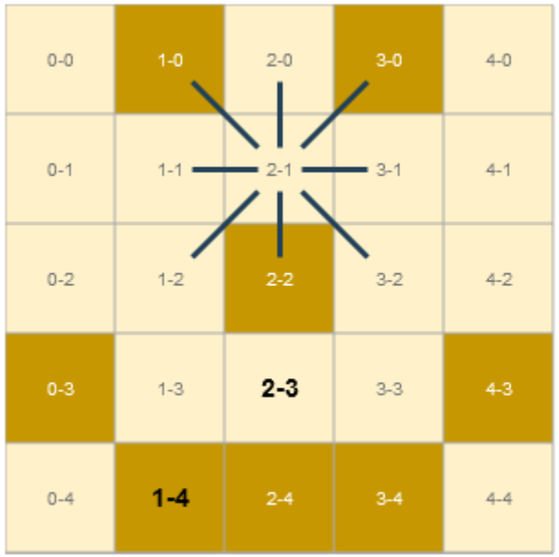
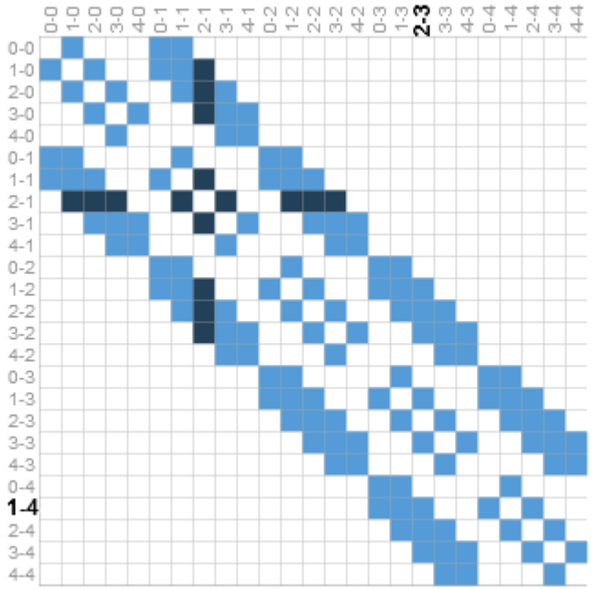
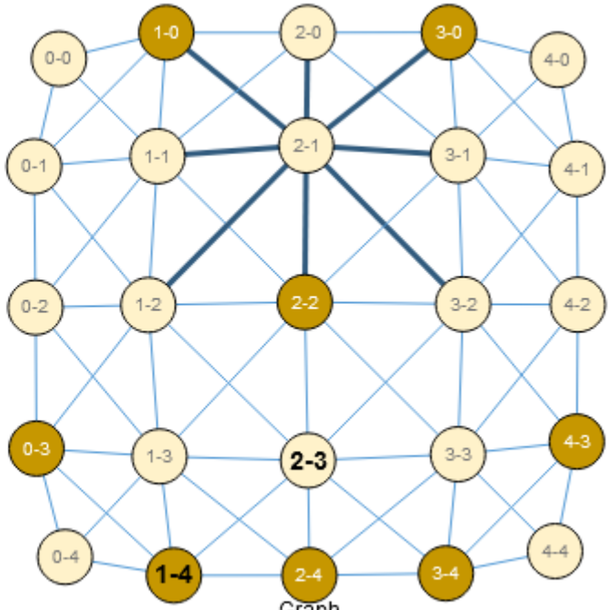


Image Pixels



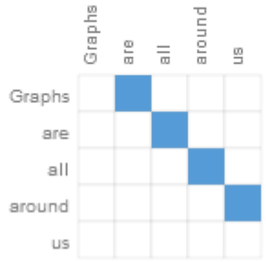
Adjacency Matrix



Graph

https://distill.pub/2021/gnn-intro/

Текст – тоже граф



Отличия произвольного графа от изображения

нет структуры (произвольные соединения, степени вершин и т.п.)

нет порядка вершин,

сложная топологическая структура

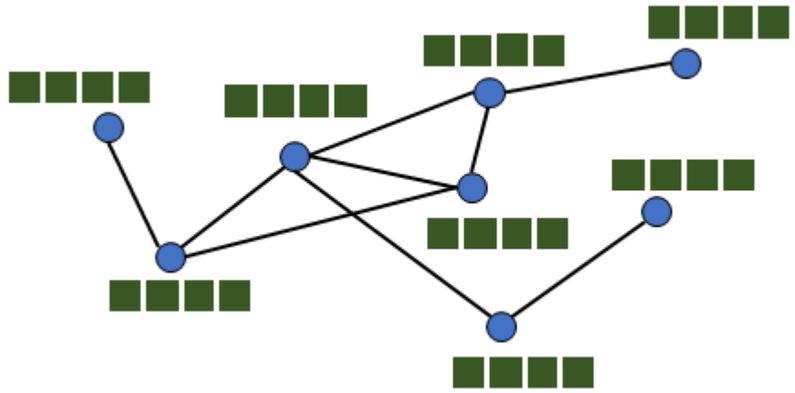
Почему нельзя просто подать матрицу смежности в НС?

Должна быть инвариантность относительно перенумерации вершин

«Equivariance»

здесь рассматриваем неориентированные графы

Обучение представлений вершин (Node representation learning)



Дальше будем говорить о Representation Learning
– получении представления вершин графа

эти представления можно использовать непосредственно
из них получить представления рёбер
агрегацией получить представление графа

GNN вычисляют такие представления в итеративном процессе

разные виды GNN по-разному
каждая итерация ~ слой

Нейронное распространение (Neural Message Passing)

скрытое состояние обновляется исходя из скрытых состояний соседей

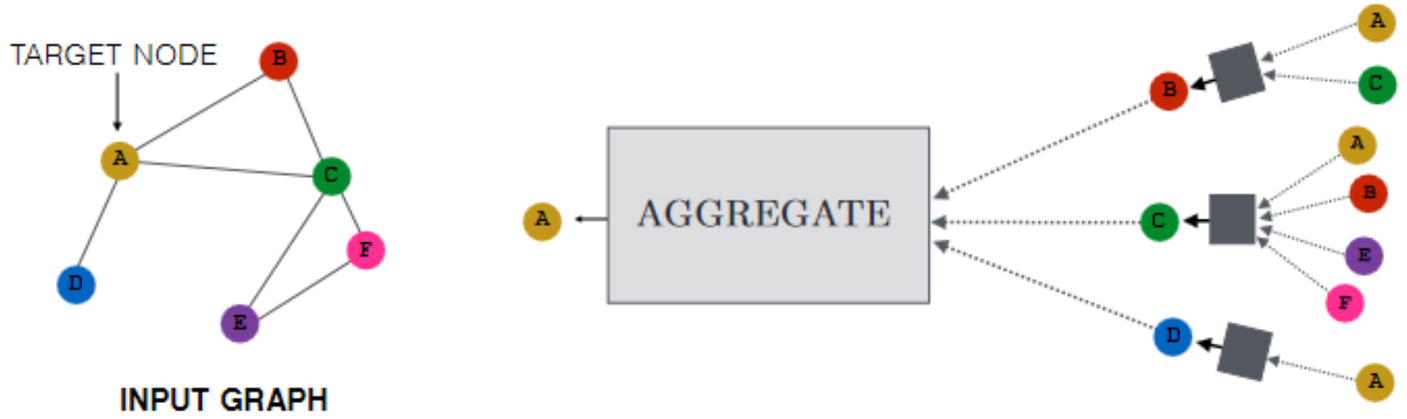


Figure 5.1: Overview of how a single node aggregates messages from its local neighborhood. The model aggregates messages from A's local graph neighbors (i.e., B, C, and D), and in turn, the messages coming from these neighbors are based on information aggregated from their respective neighborhoods, and so on. This visualization shows a two-layer version of a message-passing model. Notice that the computation graph of the GNN forms a tree structure by unfolding the neighborhood around the target node.

Neural Message Passing

$$h_v^{(k)} = \mathrm{UPDATE}^{(k)}\left(h_v^{(k-1)}, \underbrace{\mathrm{AGG}^{(k)}(\{h_u^{(k-1)}\}_{u\in N(v)}}_{=m_{N(v)}^{(k)}}\right)$$

UPDATE, AGGREGATE – произвольные дифференцируемые функции

AGGREGATE – хорошо, если не зависит от порядка элементов!

делаем фиксированное число итераций

на k-й итерации учитываем информацию о k-соседстве

original GNN

$$h_v^{(k)} = \sigma\left(W_0^{(k)} h_v^{(k-1)} + W_1^{(k)} \sum_{u \in N(v)} h_u^{(k-1)} + b^{(k)} \right)$$

Merkwirth, Christian & Lengauer, Thomas «Automatic Generation of Complementary Descriptors with Molecular Graph Networks». Journal of chemical information and modeling, 2005. 45. 1159-68.
https://www.researchgate.net/publication/7583033_Automatic_Generation_of_Complementary_Descriptors_with_Molecular_Graph_Networks

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The Graph Neural Network Model," in IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61-80, Jan. 2009, doi: 10.1109/TNN.2008.2005605. https://ieeexplore.ieee.org/document/4700287

GraphSAGE= Graph Sample and Aggregate

$$h_v^{(k)} = \sigma\left(W^{(k)}\left[\mathrm{AGG}(\{h_u^{(k-1)}\}_{u\in N(v)}), h_v^{(k-1)}\right]\right)$$

в исходной работе сэмплировали соседей
(ещё была нормировка полученного состояния)

Для агрегации использовались:
1) усреднение – MEAN
2) LSTM-агрегация (перемешивали соседей)
3) Pooling (1 слой + пулинг)

$$\mathrm{AGG}(\{h_u^{(k-1)}\}_{u\in N(v)}) = \max[\{W_{\text{pool}} h_u^{(k-1)} + b\}_{u\in N(v)}]$$

+ unsupervised loss
чтобы у соседних вершин похожие представления, а у случайных – непохожие

W. L. Hamilton, R. Ying и J. Leskovec «Inductive Representation Learning onLarge Graphs» //
https://arxiv.org/pdf/1706.02216.pdf

GraphSAGE= Graph Sample and Aggregate

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : 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 $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ 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**

    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;

    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$

  end

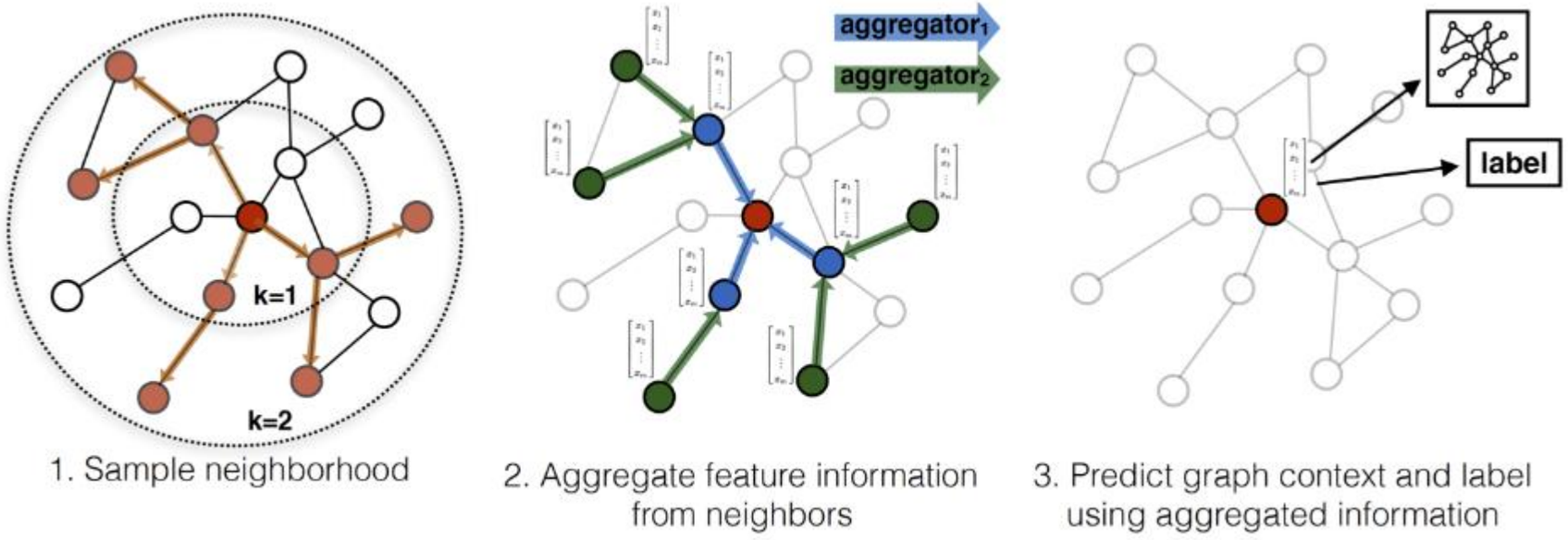
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$

end

$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

делается нормировка (для поддержания одинакового масштаба)

GraphSAGE= Graph Sample and Aggregate



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

PinSage

GraphSAGE + другое сэмплирование (для больших графов), ReLU

делаются случайные блуждания

среди topT посещённых сэмплируем

состояния соседей сначала пропускаются через отдельную сеть

R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. 2018a. Graph convolutional neural networks for web-scale recommender systems. In Proc. of SIGKDD. DOI: 10.1145/3219819.3219890 https://arxiv.org/abs/1806.01973

Algorithm 1: CONVOLVE

Input : Current embedding $\mathbf{z}_u$ for node u; set of neighbor embeddings $\{\mathbf{z}_v | v \in \mathcal{N}(u)\}$, set of neighbor weights $\boldsymbol{\alpha}$; symmetric vector function $\gamma(\cdot)$

Output: New embedding $\mathbf{z}_u^{\text{NEW}}$ for node u

$\mathbf{n}_u \leftarrow \gamma\left(\{\text{ReLU}\left(\mathbf{Q}\mathbf{h}_v + \mathbf{q}\right) \mid v \in \mathcal{N}(u)\}, \boldsymbol{\alpha}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \text{ReLU}\left(\mathbf{W} \cdot \text{CONCAT}(\mathbf{z}_u, \mathbf{n}_u) + \mathbf{w}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \mathbf{z}_u^{\text{NEW}} / \|\mathbf{z}_u^{\text{NEW}}\|_2$

Algorithm 2: MINIBATCH

Input : Set of nodes $\mathcal{M} \subset \mathcal{V}$; depth parameter K; neighborhood function $\mathcal{N}: \mathcal{V} \rightarrow 2^{\mathcal{V}}$

Output: Embeddings $\mathbf{z}_u, \forall u \in \mathcal{M}$

/* Sampling neighborhoods of minibatch nodes. */

$\mathcal{S}^{(K)} \leftarrow \mathcal{M}$;

for $k = K, ..., 1$ **do**

  $\mathcal{S}^{(k-1)} \leftarrow \mathcal{S}^{(k)}$;

  for $u \in \mathcal{S}^{(k)}$ **do**

    $\mathcal{S}^{(k-1)} \leftarrow \mathcal{S}^{(k-1)} \cup \mathcal{N}(u)$;

  end

end

/* Generating embeddings */

$\mathbf{h}_u^{(0)} \leftarrow \mathbf{x}_u, \forall u \in \mathcal{S}^{(0)}$;

for $k = 1, ..., K$ **do**

  for $u \in \mathcal{S}^{(k)}$ **do**

    $\mathcal{H} \leftarrow \left\{\mathbf{h}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\right\}$;

    $\mathbf{h}_u^{(k)} \leftarrow \text{CONVOLVE}^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathcal{H}\right)$

  end

end

for $u \in \mathcal{M}$ **do**

  $\mathbf{z}_u \leftarrow \mathbf{G}_2 \cdot \text{ReLU}\left(\mathbf{G}_1 \mathbf{h}_u^{(K)} + \mathbf{g}\right)$

end

Message Passing with Self-loops

«петля» – когда в формуле у нас окрестность фактически включает вершину и состояние вершины отдельно не фигурирует

$$h_v^{(k)} = \sigma\left(W^{(k)}\left[\mathrm{AGG}(\{h_u^{(k-1)}\}_{u\in N(v)\cup\{v\}})\right]\right)$$

George T. Cantwell, M. E. J. Newman «Message passing on networks with loops» // https://arxiv.org/abs/1907.08252

Generalized Neighborhood Aggregation

Neighborhood Normalization

$$m_{N(v)}^{(k)} = \frac{\sum_{u \in N(v)} h_v^{(k-1)}}{|N(v)|}$$

$$m_{N(v)}^{(k)} = \sum_{u \in N(v)} \frac{h_v^{(k-1)}}{\sqrt{|N(v)||N(u)|}}$$

Graph convolutional networks (GCNs)

$$h_v^{(k)} = \sigma\left(W^{(k)} \sum_{u \in N(v) \cup \{v\}} \frac{h_v^{(k-1)}}{\sqrt{|N(v)||N(u)|}} \right)$$

хорошая нормализация важна

она может терять некоторую важную информацию, например степень вершины

Thomas N Kipf, Max Welling «Semi-supervised classification with graph convolutional networks» // ICLR, 2017.

Матричная нотация при записи GNN

Рассмотрим такое преобразование состояний

$$h_v^{(k)} = \sigma\left(W_0^{(k)} h_v^{(k-1)} + \frac{W_1^{(k)}}{|N(v)|} \sum_{u \in N(v)} h_u^{(k-1)} + b^{(k)} \right)$$

тогда введя новые обозначения получаем

$$H^{(k)} = \begin{bmatrix} h_1^{(k)} \\ \dots \\ h_{|V|}^{(k)} \end{bmatrix}_{|V| \times r}$$

$$H^{(k)} = \sigma(H^{(k)} (W_0^{(k)})^{\mathrm{T}} + D^{-1} A H^{(k)} (W_1^{(k)})^{\mathrm{T}})$$

или с точностью до переобозначений

$$H^{(k)} = \sigma(H^{(k)} P_0^{(k)} + A' H^{(k)} P_1^{(k)})$$

$$D = \mathrm{diag}(|N(1)|, \dots, |N(|V|)|)$$

не всегда можно записать так красиво (не при всех агрегациях)

Generalized Neighborhood Aggregation

Set pooling

$$m_{N(v)}^{(k)} = f_\theta\left(\sum_{u \in N(v)} g_\varphi(h_u^{(k-1)})\right)$$

f и g – MLP = deep multi-layer perceptron

Показано, что это универсальный апроксиматор

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. 2017. Deep sets. In Proc. of NIPS, pages 3391–3401. https://arxiv.org/pdf/1703.06114.pdf

Generalized Neighborhood Aggregation

Janossy pooling

$$m_{N(v)}^{(k)} = f_\theta\left(\frac{1}{|\Pi|}\sum_{\pi\in\Pi} g_\varphi(h_{\pi_1}^{(k-1)},\ldots,h_{\pi_{|N(v)|}}^{(k-1)})\right)$$

суммируем по всем перестановкам

на практике сумма по нескольким случайным перестановкам или задаёмся каноническим порядком соседей

g_φ **– обрабатывает последовательности произвольной длины**

[8] Ryan L. Murphy «Janossy pooling: learning deep permutation-invariant functions for variable-size inputs» // https://arxiv.org/pdf/1811.01900.pdf

Generalized Neighborhood Aggregation

GIN (Graph Isomorphism Network)

$$h_v^{(k)} = f_\theta\left((1+\varepsilon^{(k)})h_v^{(k-1)} + \sum_{u\in N(v)} h_u^{(k-1)}\right)$$

K. Xu и др. «How Powerful are Graph Neural Networks?» // https://arxiv.org/pdf/1810.00826.pdf
– сравниваются различные агрегации

Neighborhood Attention: Graph Attention Network (GAT)

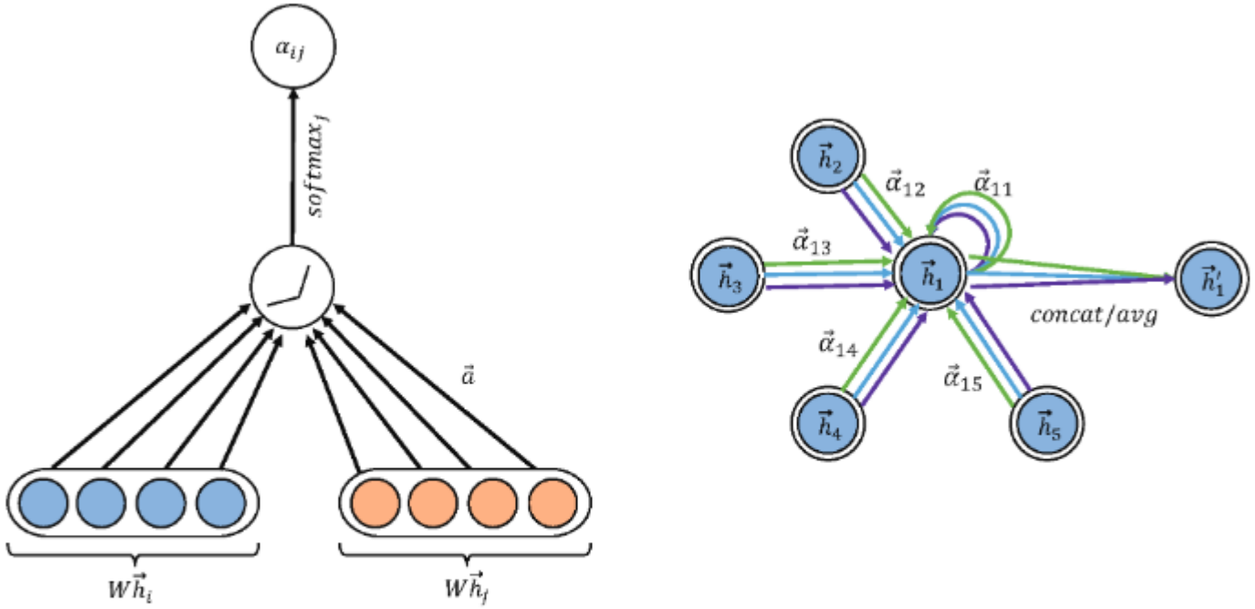


Figure 7.1: The illustration of the GAT model. **Left:** The attention mechanism employed in the model. **Right:** An illustration of multihead attention (with three heads denoted by different colors) by node 1 on its neighborhood.

Есть ещё Gated Attention Network (GaAN) – там multi-head attention

J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung «GaAN: Gated attention networks for learning on large and spatiotemporal graphs» // Proc. of UAI. 2018b

Column Networks

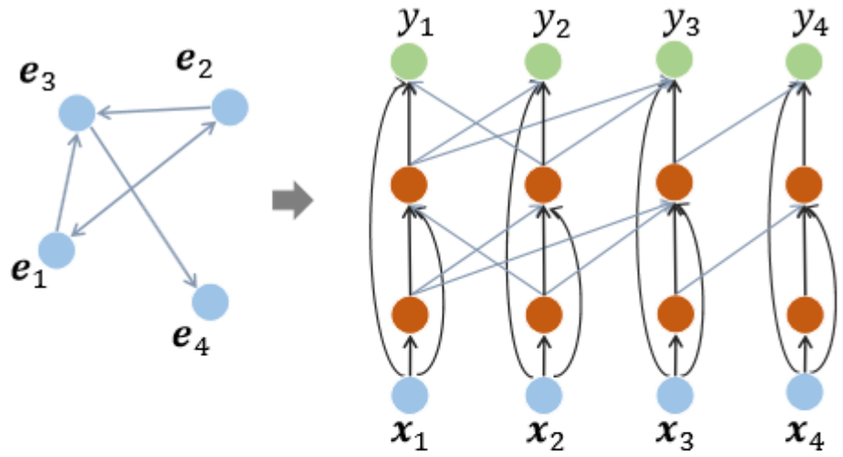
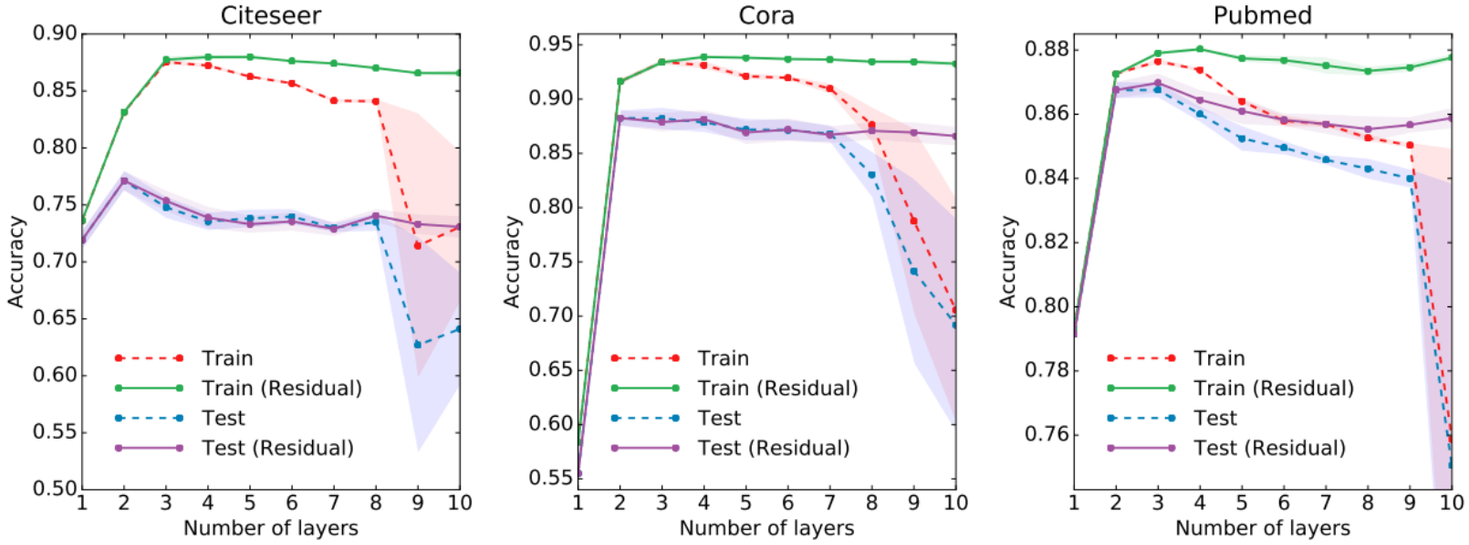


Figure 1: Collective classification with Stacked Learning (SL). (**Left**): A graph with 4 entities connected by unidirectional and bidirectional links, (**Right**): SL model for the graph with three steps where x_i is the feature vector of entity e_i. The bidirectional link between e_1 and e_2 is modeled as two unidirectional links from e_1 to e_2 and vice versa.

тут прокидываются состояния

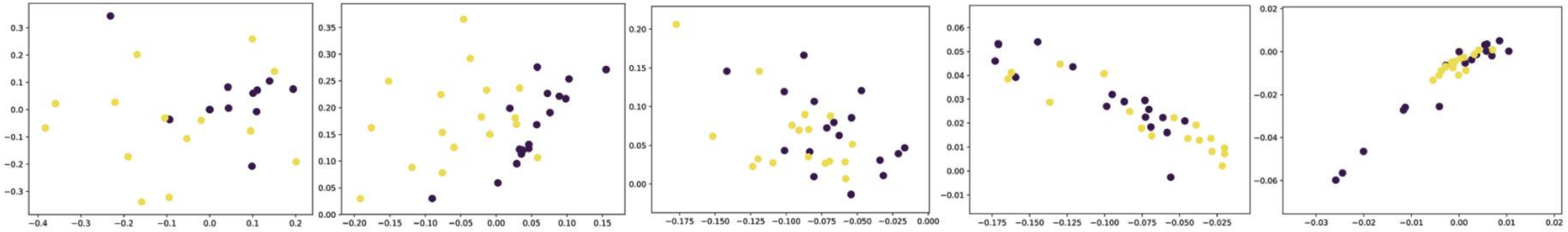
T. Pham, T. Tran, D. Phung, and S. Venkatesh «Column networks for collective classification» // 2017 In Proc. of AAAI. https://arxiv.org/pdf/1609.04508.pdf

Проблема глубоких сетей



Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

Проблема глубоких сетей



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

Figure 2: Vertex embeddings of Zachary's karate club network with GCNs with 1,2,3,4,5 layers.

Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. AAAI 2018

Проблема адаптации

основная проблема в адаптации – «over-smoothing»

представления соседних вершин становятся похожими

для борьбы с over-smoothing

- используют слои, которые не агрегирую, а обрабатывают признаки
- используют прокидывание связей / конкатенация
- используют архитектуры, в которых есть эффект памяти (например GRU)
- приёмы с нормировками (поддерживают равной константе сумму квадратов попарных расстояний между представлениями)

PairNorm: Tackling Oversmoothing in GNNs. ICLR 2020

- Используют аугментацию, например DropEdge
- noise regularization (Godwin)

DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. ICLR 2020

Simple and Deep Graph Convolutional Networks. ICML 2020

Towards Deeper Graph Neural Networks. KDD'2020

Godwin et al. Very Deep Graph Neural Networks Via Noise Regularisation. arXiv:2106.07971

GGNN (GATED GRAPH NEURAL NETWORKS)

$$
\begin{aligned}
\mathbf{a}_v^t &= \mathbf{A}_v^T \left[\mathbf{h}_1^{t-1} \ldots \mathbf{h}_N^{t-1}\right]^T + \mathbf{b} \\
\mathbf{z}_v^t &= \sigma\left(\mathbf{W}^z \mathbf{a}_v^t + \mathbf{U}^z \mathbf{h}_v^{t-1}\right) \\
\mathbf{r}_v^t &= \sigma\left(\mathbf{W}^r \mathbf{a}_v^t + \mathbf{U}^r \mathbf{h}_v^{t-1}\right) \\
\widetilde{\mathbf{h}}_v^t &= \tanh\left(\mathbf{W} \mathbf{a}_v^t + \mathbf{U}\left(\mathbf{r}_v^t \odot \mathbf{h}_v^{t-1}\right)\right) \\
\mathbf{h}_v^t &= \left(1 - \mathbf{z}_v^t\right) \odot \mathbf{h}_v^{t-1} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}}_v^t.
\end{aligned}
$$

Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. 2016. «Gated graph sequence neural networks» // In Proc. of ICLR. 22, 33, 59, 75, 91

TREE LSTM

$$\widetilde{\mathbf{h}}_v^{t-1} = \sum_{k \in N_v} \mathbf{h}_k^{t-1}$$

$$\mathbf{i}_v^t = \sigma\left(\mathbf{W}^i \mathbf{x}_v^t + \mathbf{U}^i \widetilde{\mathbf{h}}_v^{t-1} + \mathbf{b}^i\right)$$

$$\mathbf{f}_{vk}^t = \sigma\left(\mathbf{W}^f \mathbf{x}_v^t + \mathbf{U}^f \mathbf{h}_k^{t-1} + \mathbf{b}^f\right)$$

$$\mathbf{o}_v^t = \sigma\left(\mathbf{W}^o \mathbf{x}_v^t + \mathbf{U}^o \widetilde{\mathbf{h}}_v^{t-1} + \mathbf{b}^o\right)$$

$$\mathbf{u}_v^t = \tanh\left(\mathbf{W}^u \mathbf{x}_v^t + \mathbf{U}^u \widetilde{\mathbf{h}}_v^{t-1} + \mathbf{b}^u\right)$$

$$\mathbf{c}_v^t = \mathbf{i}_v^t \odot \mathbf{u}_v^t + \sum_{k \in N_v} \mathbf{f}_{vk}^t \odot \mathbf{c}_k^{t-1}$$

$$\mathbf{h}_v^t = \mathbf{o}_v^t \odot \tanh(\mathbf{c}_v^t),$$

K. S. Tai, R. Socher, and C. D. Manning «Improved semantic representations from tree-structured long short-term memory networks» // Proc. of IJCNLP, pages 1556–1566. DOI: 10.3115/v1/p15-1150 34, 78, 81

GRAPH LSTM

$$\mathbf{i}_v^t = \sigma\left(\mathbf{W}^i \mathbf{x}_v^t + \sum_{k \in N_v} \mathbf{U}^i_{m(v,k)} \mathbf{h}_k^{t-1} + \mathbf{b}^i\right)$$

$$\mathbf{f}_{vk}^t = \sigma\left(\mathbf{W}^f \mathbf{x}_v^t + \mathbf{U}^f_{m(v,k)} \mathbf{h}_k^{t-1} + \mathbf{b}^f\right)$$

$$\mathbf{o}_v^t = \sigma\left(\mathbf{W}^o \mathbf{x}_v^t + \sum_{k \in N_v} \mathbf{U}^o_{m(v,k)} \mathbf{h}_k^{t-1} + \mathbf{b}^o\right)$$

$$\mathbf{u}_v^t = \tanh\left(\mathbf{W}^u \mathbf{x}_v^t + \sum_{k \in N_v} \mathbf{U}^u_{m(v,k)} \mathbf{h}_k^{t-1} + \mathbf{b}^u\right)$$

$$\mathbf{c}_v^t = \mathbf{i}_v^t \odot \mathbf{u}_v^t + \sum_{k \in N_v} \mathbf{f}_{vk}^t \odot \mathbf{c}_k^{t-1}$$

$$\mathbf{h}_v^t = \mathbf{o}_v^t \odot \tanh(\mathbf{c}_v^t),$$

V. Zayats and M. Ostendorf «Conversation modeling on reddit using a graph-structured LSTM» // 2018, TACL, 6:121–132. DOI: 10.1162/tacl_a_00009

HIGHWAY GCN

$$\mathrm{T}(\mathbf{h}^t) = \sigma\left(\mathbf{W}^t\mathbf{h}^t + \mathbf{b}^t\right)$$
$$\mathbf{h}^{t+1} = \mathbf{h}^{t+1} \odot \mathrm{T}\left(\mathbf{h}^t\right) + \mathbf{h}^t \odot \left(1 - \mathrm{T}\left(\mathbf{h}^t\right)\right)$$

A. Rahimi, T. Cohn, and T. Baldwin. 2018. Semi-supervised user geolocation via graph convolutional networks. In Proc. of ACL, 1:2009–2019. DOI: 10.18653/v1/p18-1187

Jumping Knowledge Connections

в качестве представления вершины можно использовать
не последнее скрытое состояние,
а некоторую агрегацию всех состояний
(по всем итерациям):

$$z_v = f_{\text{JK}}\left([h_v^{(0)}, h_v^{(1)}, \ldots, h_v^{(k)}]\right)$$

– здесь конкатенация

K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In Proc. of ICML, pages 5449–5458.
https://arxiv.org/pdf/1806.03536.pdf

JUMP KNOWLEDGE NETWORK (Jumping Knowledge Connections)

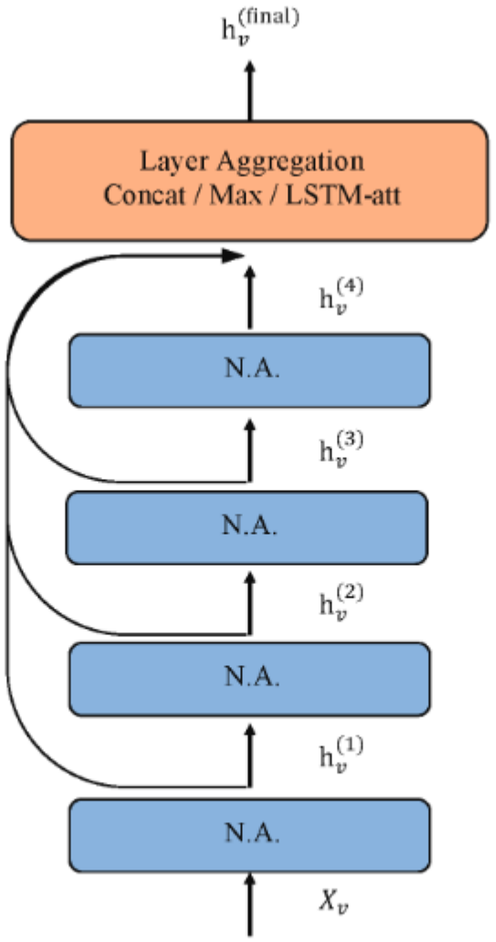


Figure 8.1: The illustration of Jump Knowledge Network. N.A. stands for neighborhood aggregation.

DEEPGCNS

Идея DenseNet + dilated convolutions

G. Li, M. Muller, A. Thabet, and B. Ghanem «DeepGCNs: Can GCNs go as deep as CNNs?» // In Proc. of ICCV 2019

Dilated k-NN

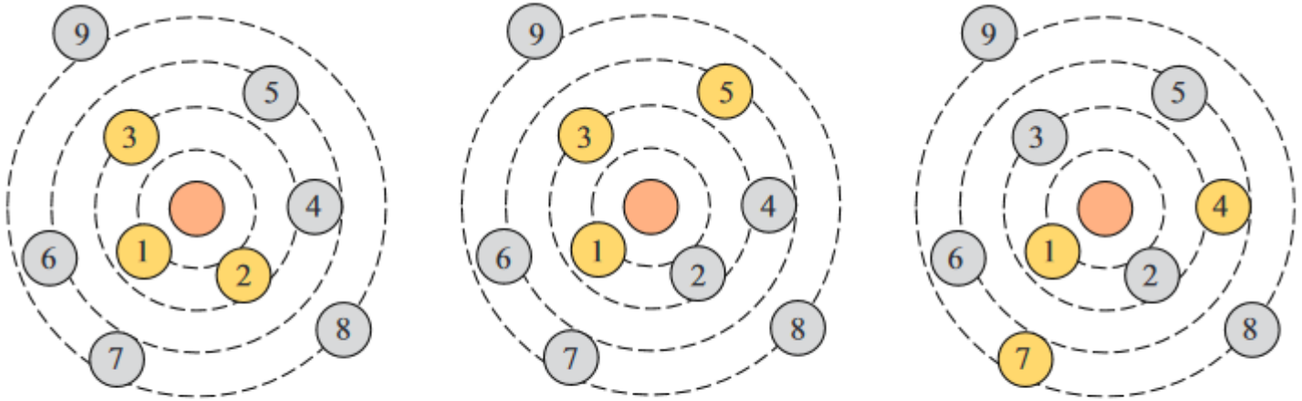


Figure 8.3: An example of the dilated convolution. The dilation rate is 1, 2, 3 for figures from left to right.

DEEPGCNS

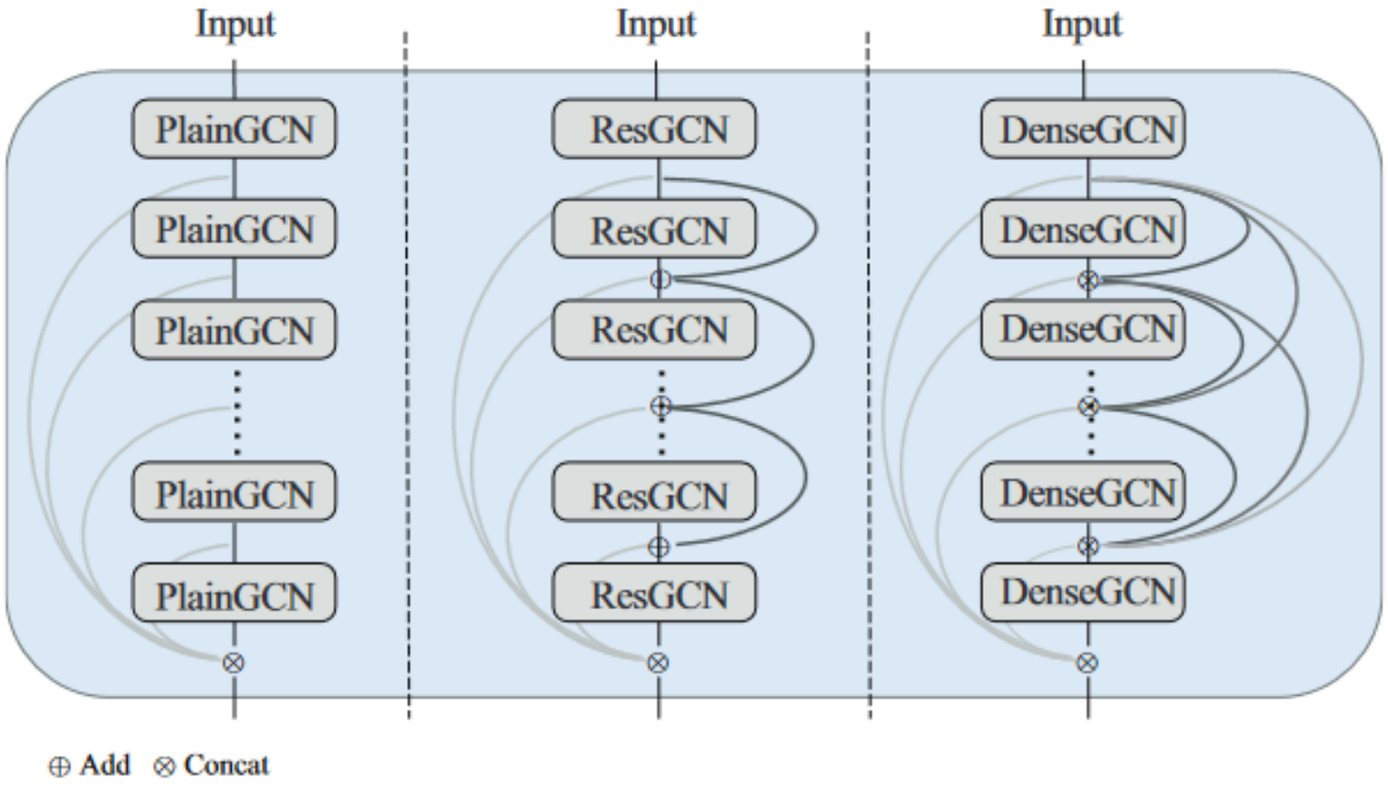


Figure 8.2: DeepGCN blocks (PlainGCN, ResGCN, DenseGCN) proposed in Li et al. [2019].

Graph Pooling

если теперь надо получить представление графа по представлениям вершин

$$z_G = \frac{1}{f(|V|)} \sum_{u \in V} z_u$$

Readout-слои:

$$z_G = \left[\frac{1}{f(|V|)} \sum_{u \in V} z_u, \quad \max\{z_u\}_{u \in V} \right]$$

Cangea, C., Velickovic, P., Jovanovic, N., Kipf, T., and Lio,P. «Towards sparse hierarchical graph classifiers» // arXiv:1811.01287, 2018

Graph Pooling

другой подход – комбинация LSTM+attention

$$\mathbf{q}_t = \mathrm{LSTM}(\mathbf{o}_{t-1}, \mathbf{q}_{t-1}),$$
$$e_{v,t} = f_a(\mathbf{z}_v, \mathbf{q}_t), \forall v \in \mathcal{V},$$
$$a_{v,t} = \frac{\exp(e_{v,i})}{\sum_{u \in \mathcal{V}} \exp(e_{u,t})}, \forall v \in \mathcal{V},$$
$$\mathbf{o}_t = \sum_{v \in \mathcal{V}} a_{v,t} \mathbf{z}_v.$$
$$\mathbf{z}_{\mathcal{G}} = \mathbf{o}_1 \oplus \mathbf{o}_2 \oplus \ldots \oplus \mathbf{o}_T$$

Graph Pooling

SortPool

каким-то образом сортируются вершины (например по последнему элементу в векторе-представлении), конкатенируются их представления

и → в обычную НС

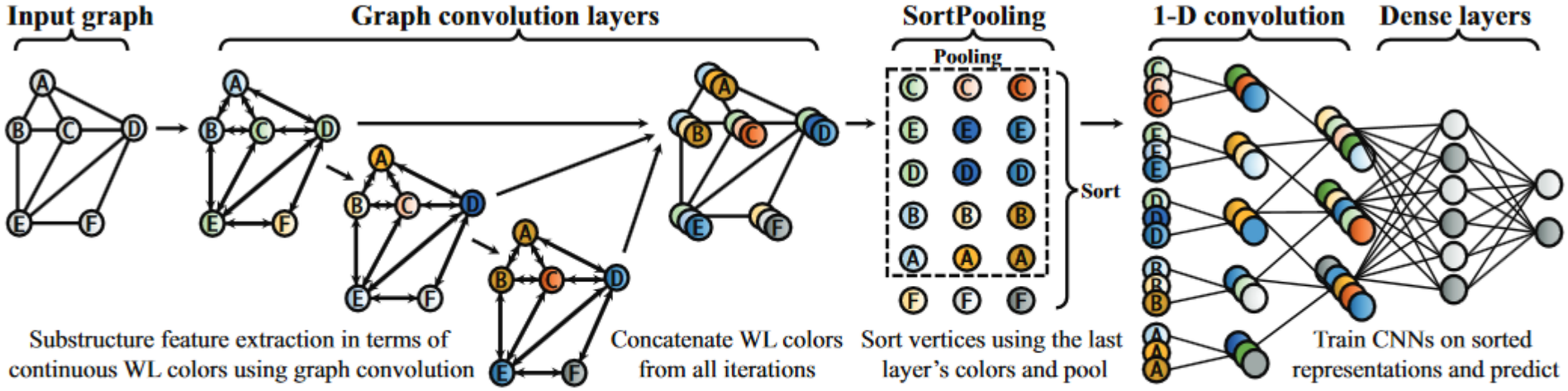


Figure 2: The overall structure of DGCNN. An input graph of arbitrary structure is first passed through multiple graph convolution layers where node information is propagated between neighbors. Then the vertex features are sorted and pooled with a SortPooling layer, and passed to traditional CNN structures to learn a predictive model. Features are visualized as colors.

M. Zhang, Z. Cui, M. Neumann, Y. Chen «An End-to-End Deep Learning Architecture for Graph Classification»

Graph Pooling

SAGPool

получаем оценки вершин, оставляем top-вершины, продолжаем, пока не останется одна

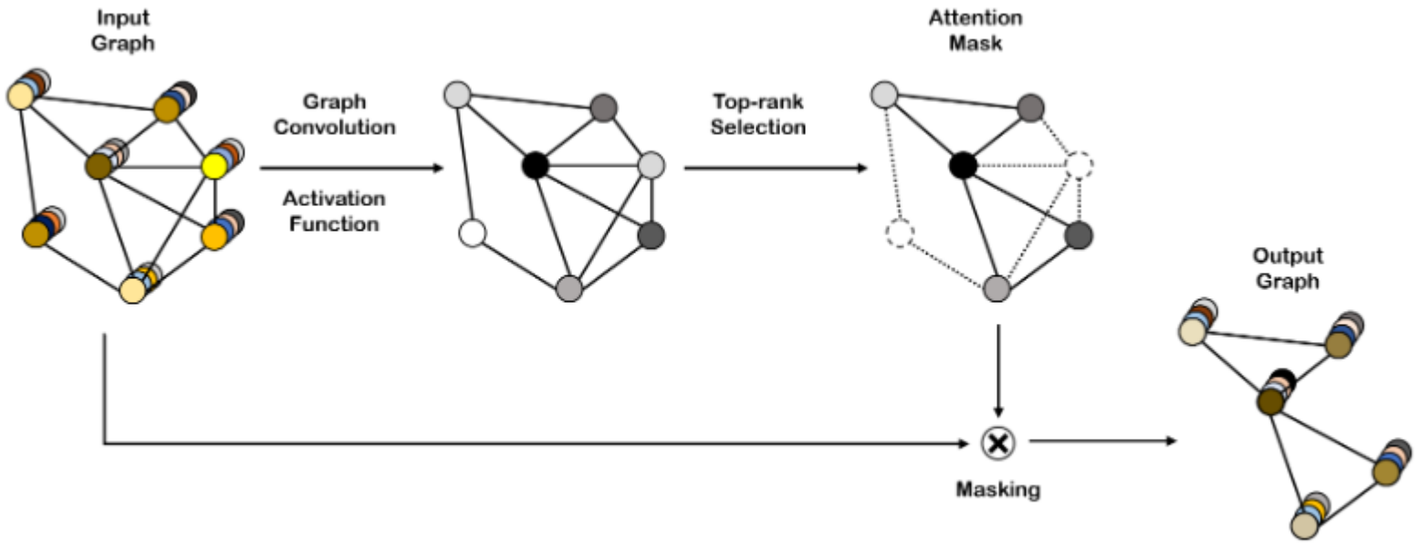


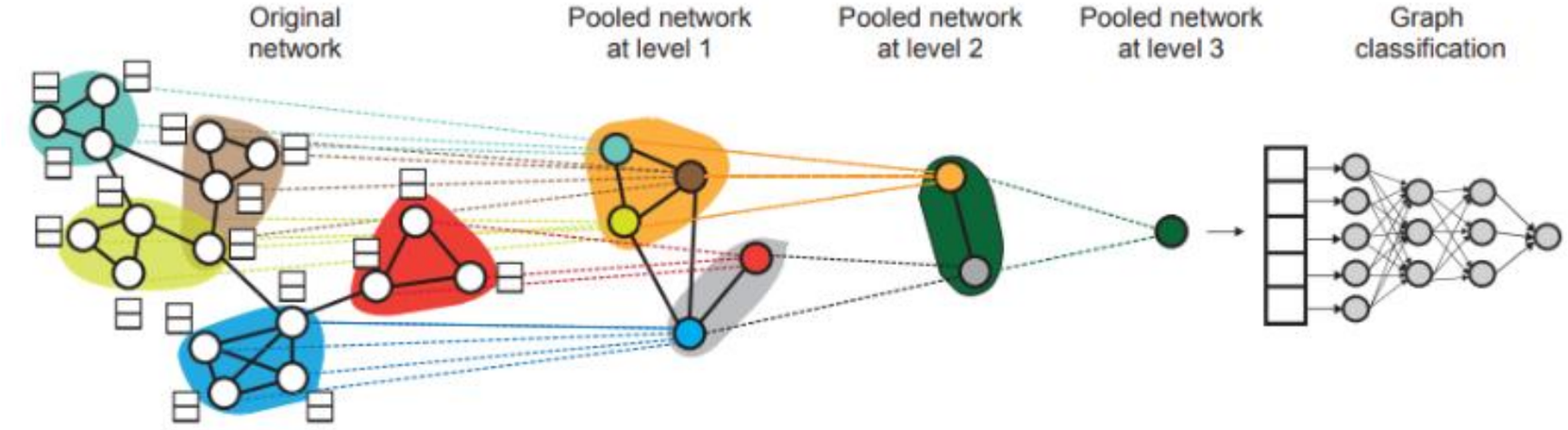
Figure 1. An illustration of the SAGPool layer.

J. Lee, I. Lee, J. Kang «Self-Attention Graph Pooling»

Graph Pooling

ещё подход – кластеризация вершин графа
сначала агрегация в кластере
потом агрегация по кластерам

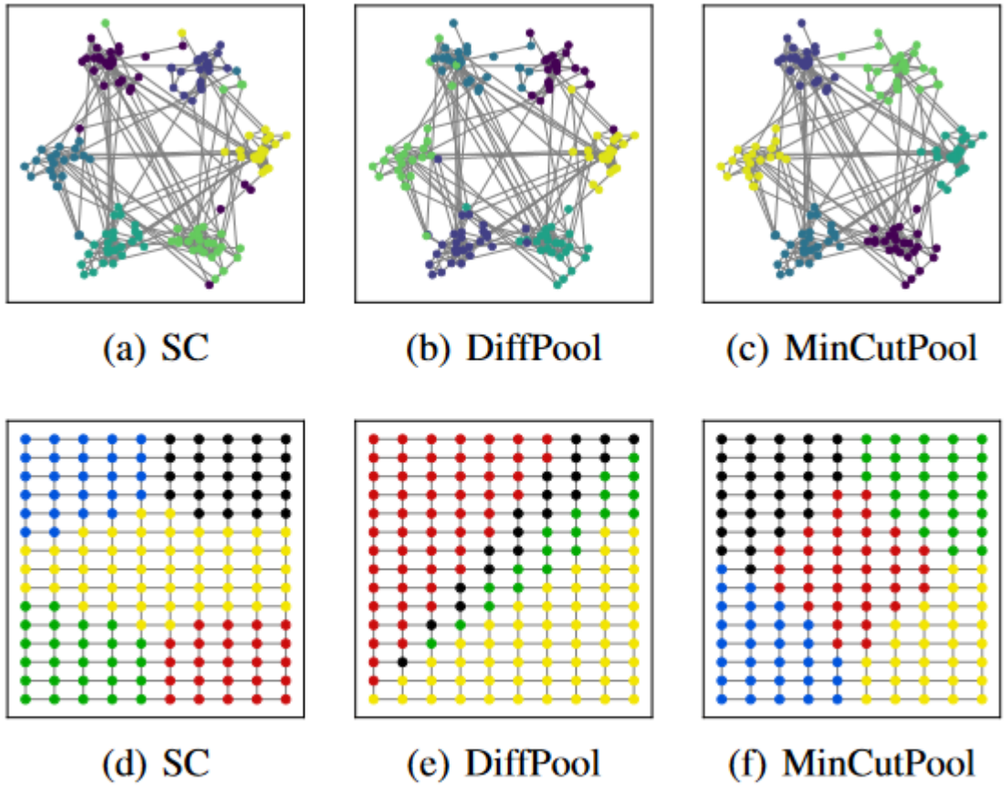
DiffPool [16] – применение «hierarchical global pooling»
иерархическая модель, на каждом уровне выделение сообществ



R. Ying и др. «Hierarchical Graph Representation Learning with Differentiable Pooling» // https://arxiv.org/pdf/1806.08804.pdf

Graph Pooling

MinCutPool – кластеризация спектральным методом



(a) SC (b) DiffPool (c) MinCutPool

(d) SC (e) DiffPool (f) MinCutPool

Figure 3. Node clustering on a community network (K=6) and on a grid graph (K=5).

[Filippo Maria Bianchi «Spectral Clustering with Graph Neural Networks for Graph Pooling» // https://arxiv.org/pdf/1907.00481.pdf

Generalized Message Passing

$$\mathbf{h}_{(u,v)}^{(k)} = \text{UPDATE}_{\text{edge}}(\mathbf{h}_{(u,v)}^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{h}_v^{(k-1)}, \mathbf{h}_{\mathcal{G}}^{(k-1)})$$

$$\mathbf{m}_{\mathcal{N}(u)} = \text{AGGREGATE}_{\text{node}}(\{\mathbf{h}_{(u,v)}^{(k)} \forall v \in \mathcal{N}(u)\})$$

$$\mathbf{h}_u^{(k)} = \text{UPDATE}_{\text{node}}(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}, \mathbf{h}_{\mathcal{G}}^{(k-1)})$$

$$\mathbf{h}_{\mathcal{G}}^{(k)} = \text{UPDATE}_{\text{graph}}(\mathbf{h}_{\mathcal{G}}^{(k-1)}, \{\mathbf{h}_u^{(k)}, \forall u \in \mathcal{V}\}, \{\mathbf{h}_{(u,v)}^{(k)} \forall (u,v) \in \mathcal{E}\})$$

не только на уровне вершин,
но и рёбер

Использование GNN

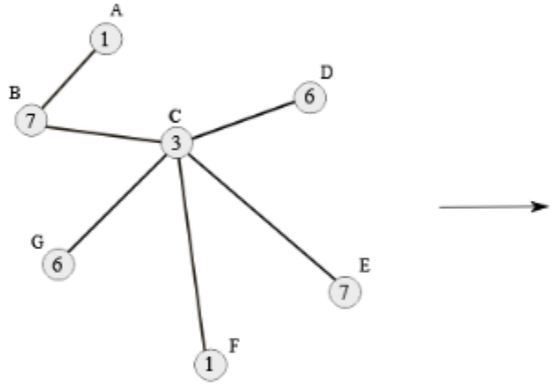
node-level predictions / node classification

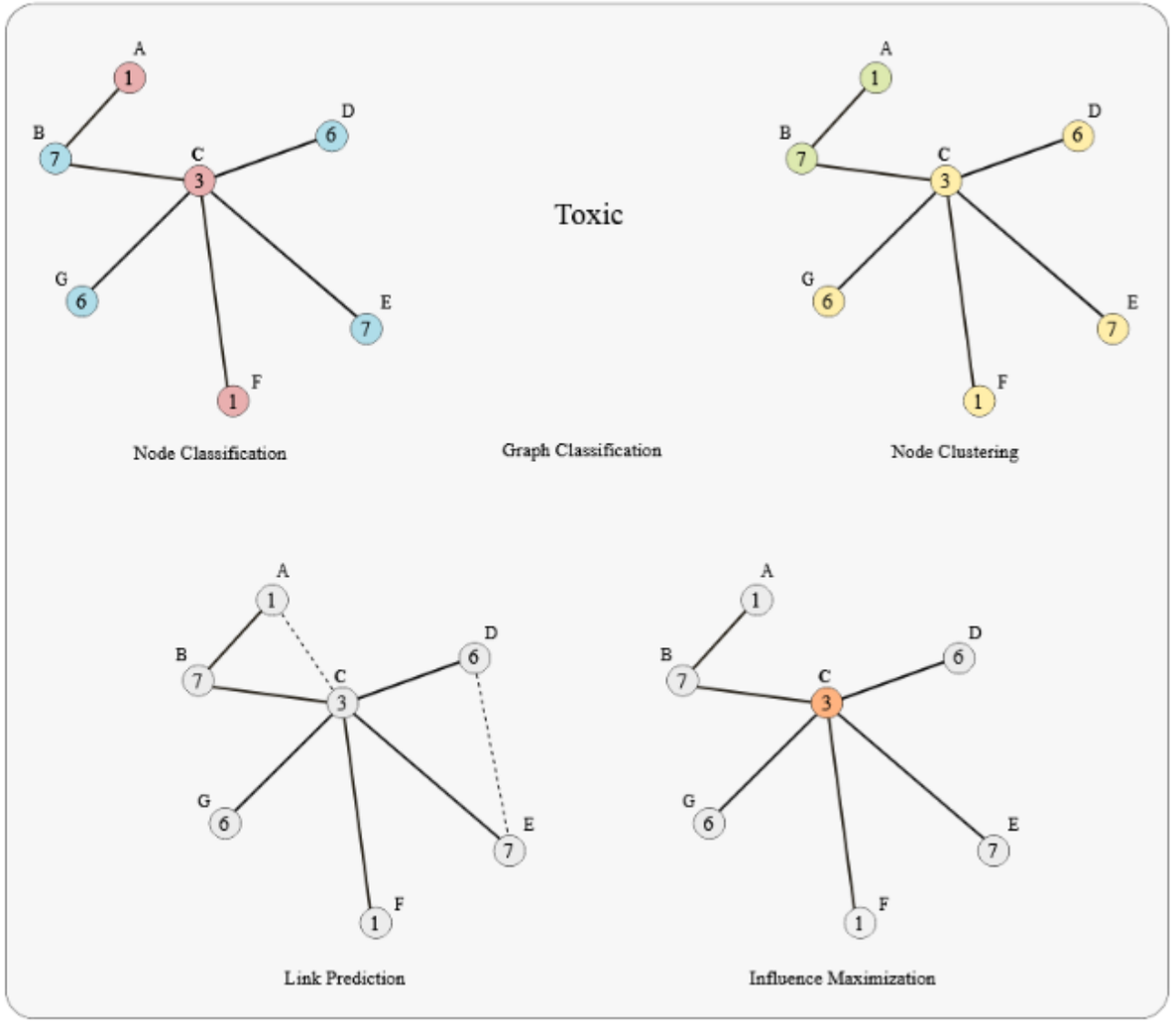
edge-level predictions (link predictions)

graph-level predictions / graph classification

relation prediction / community detection

Задачи на графах





https://distill.pub/2021/understanding-gnns/

Использование GNN

node classification

смотрим на итоговое представление вершины

$$L = \sum_{u \in \text{train}} -\log(\text{softmax}(z_u, y_u))$$

$$L = \sum_{u \in \text{train}} \| MLP(z_u) - y_u \|^2$$

Node Clustering

кластеризация представлений

edge-level predictions (link predictions)

$$L = \sum_{(u,v) \in \text{train}} \text{CE}(y_{u,v}, f(z_u, z_v))$$

регрессия на графе (предсказание биологических свойств)

НС над итоговым представлением графа

$$L = \sum_{G \in \text{train}} \| MLP(z_G) - y_G \|^2$$

Deep Graph Infomax (DGI)

в задаче без разметки:

$$\mathcal{L} = -\sum_{u \in \mathcal{V}_{\text{train}}} \mathbb{E}_{\mathcal{G}} \log(D(\mathbf{z}_u, \mathbf{z}_{\mathcal{G}})) + \gamma \mathbb{E}_{\tilde{\mathcal{G}}} \log(1 - D(\tilde{\mathbf{z}}_u, \mathbf{z}_{\mathcal{G}}))$$

здесь строится дискриминатор,
который определяет,
согласуются ли представления вершин и графа

«z с волной» – представление вершины на базе «испорченного» графа

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, R Devon Hjelm «Deep Graph Infomax» // https://arxiv.org/abs/1809.10341

Аугментации графа

Если граф очень плотный, соседей можно сэмлировать

есть DropNode…

Если граф большой, то сэмплировать подграфы

Если граф очень разреженный, можно добавлять виртуальные вершины

$$A \to A + A^2$$

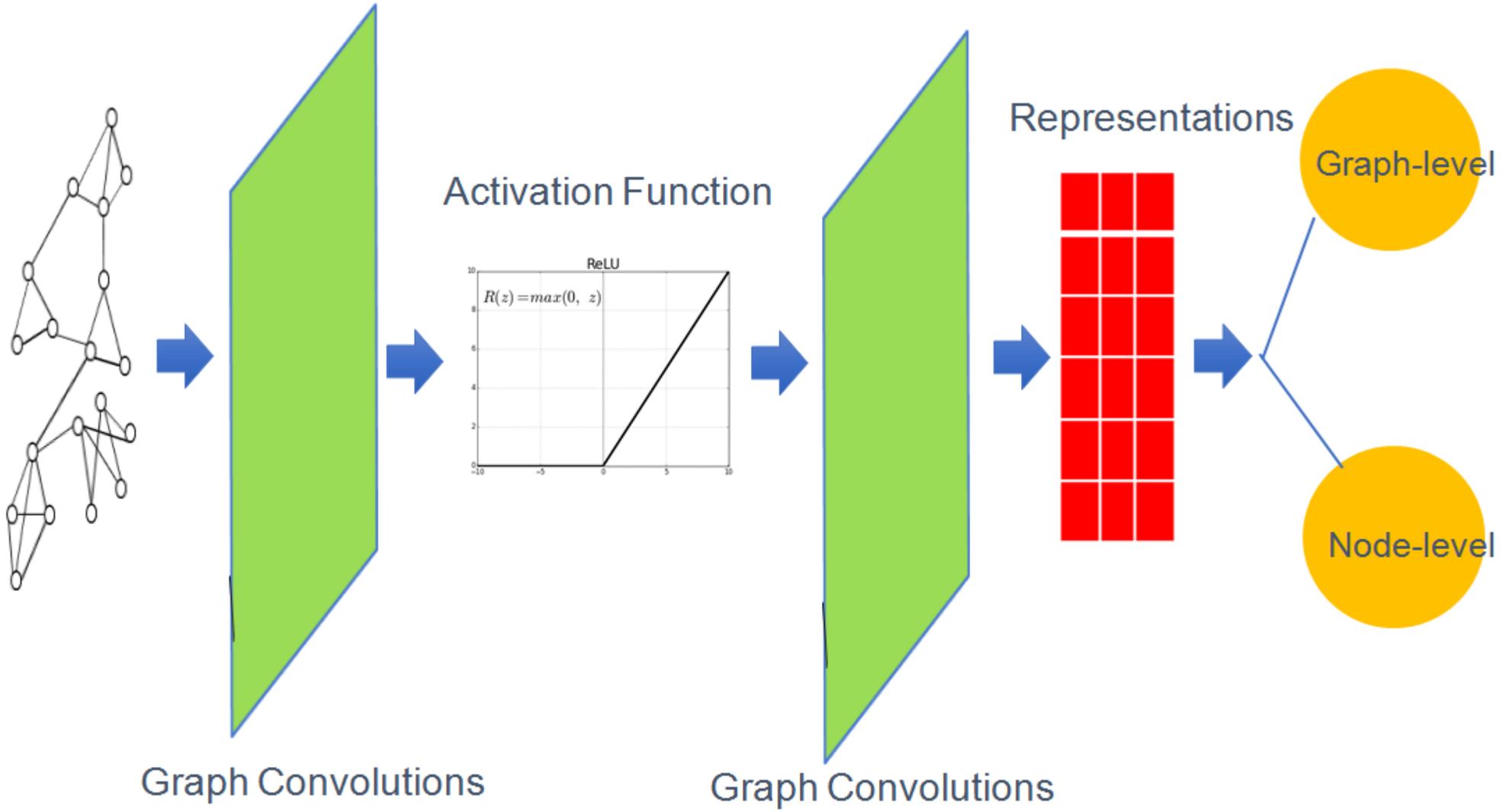
FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling (ICLR'18)

Adaptive Sampling Towards Fast Graph Representation Learning (NeurIPS'18)

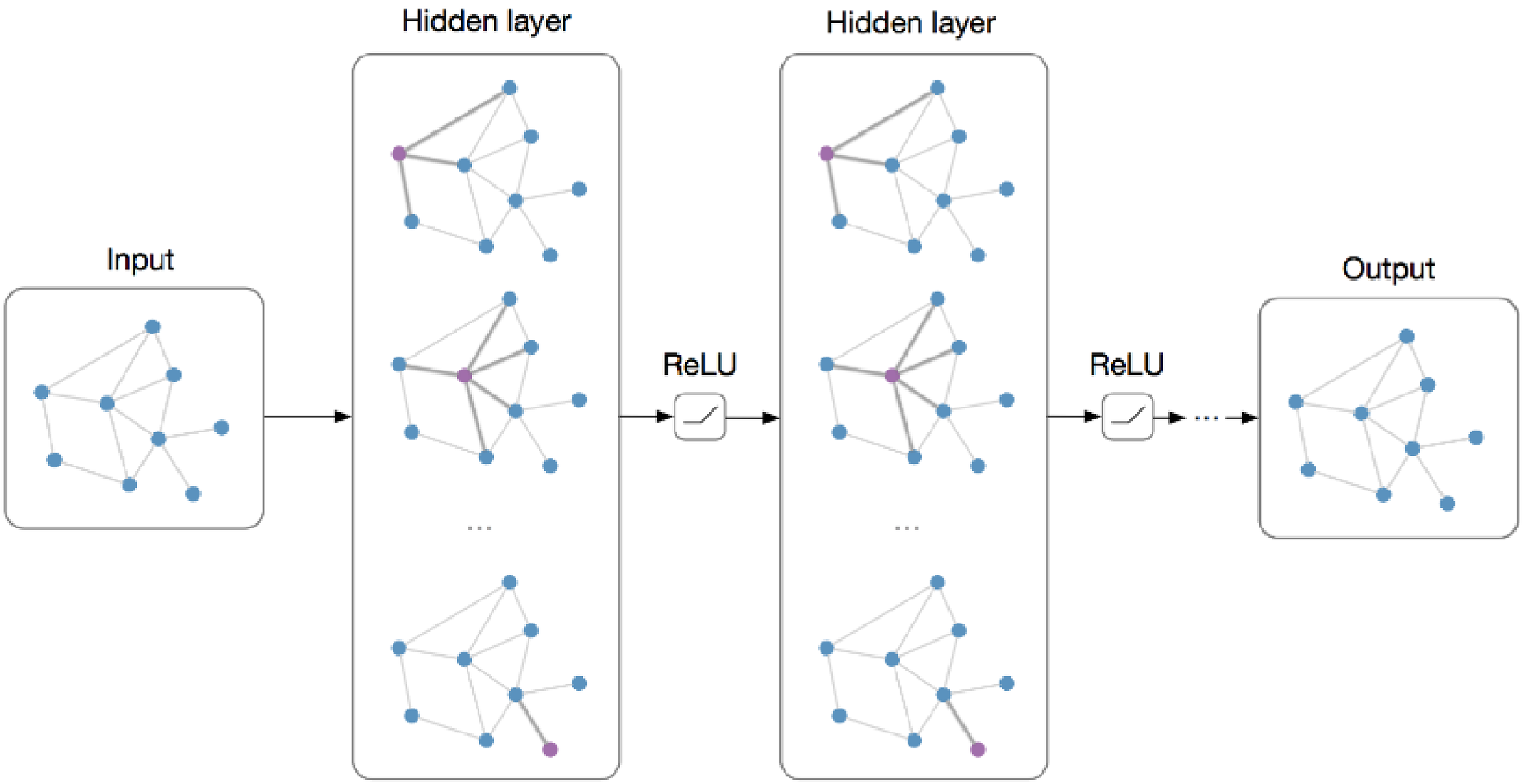
Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks (KDD'19)

GraphSAINT: Graph Sampling Based Inductive Learning Method (ICLR'20)

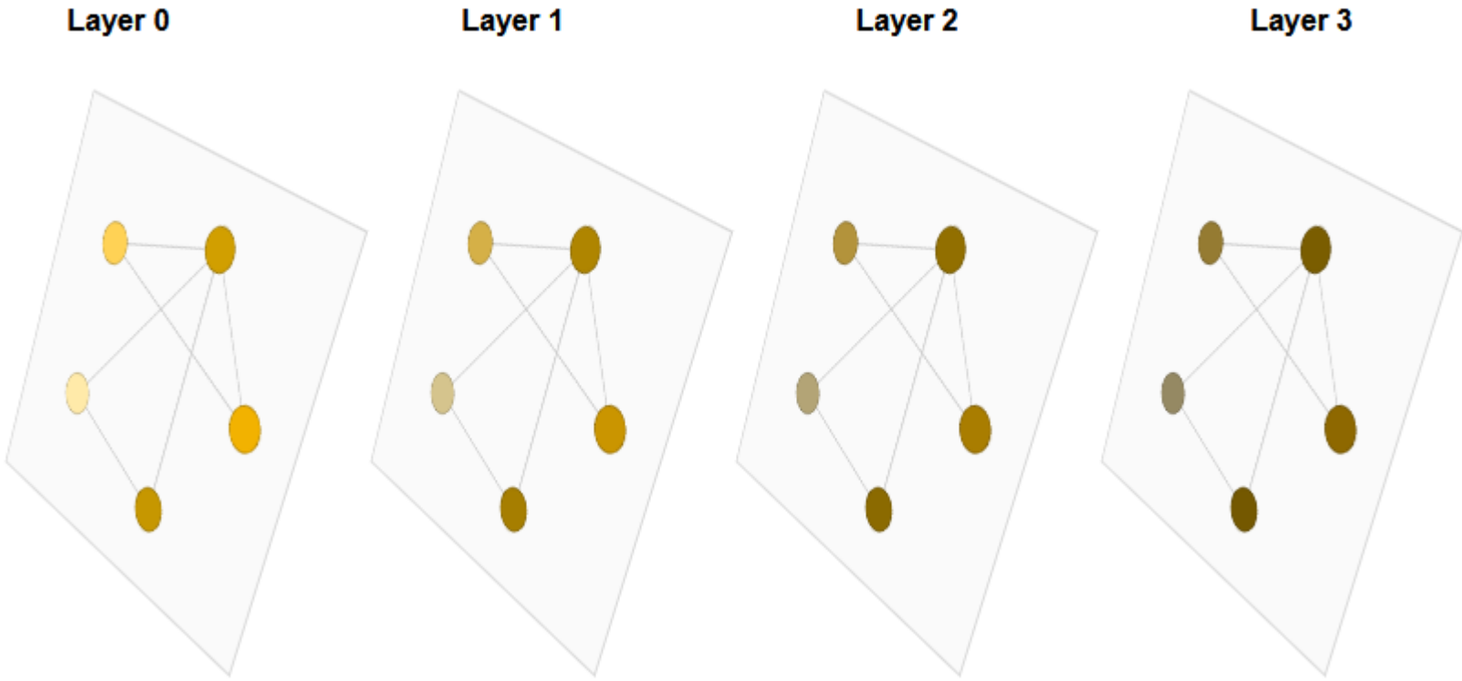
Graph Neural Networks



Graph Neural Networks

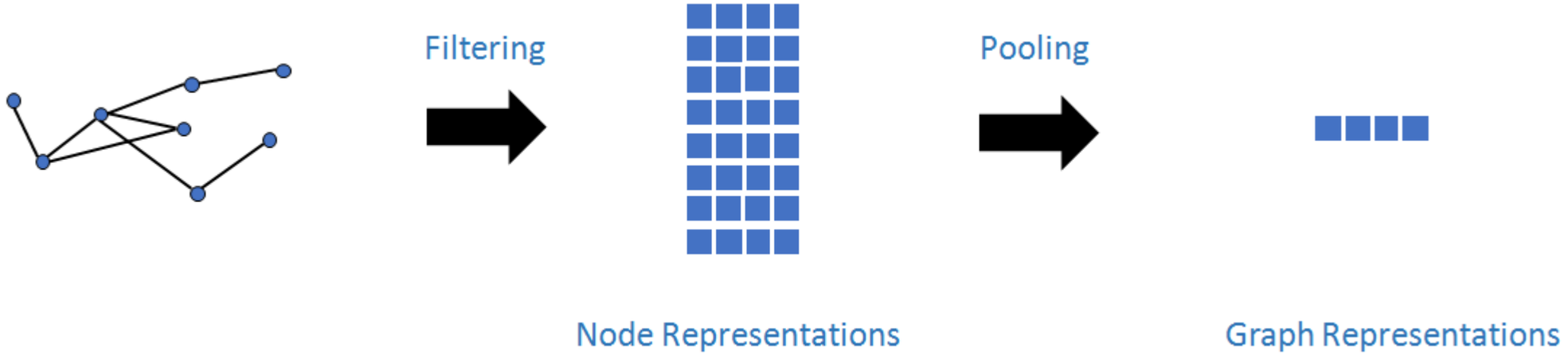


Graph Neural Networks

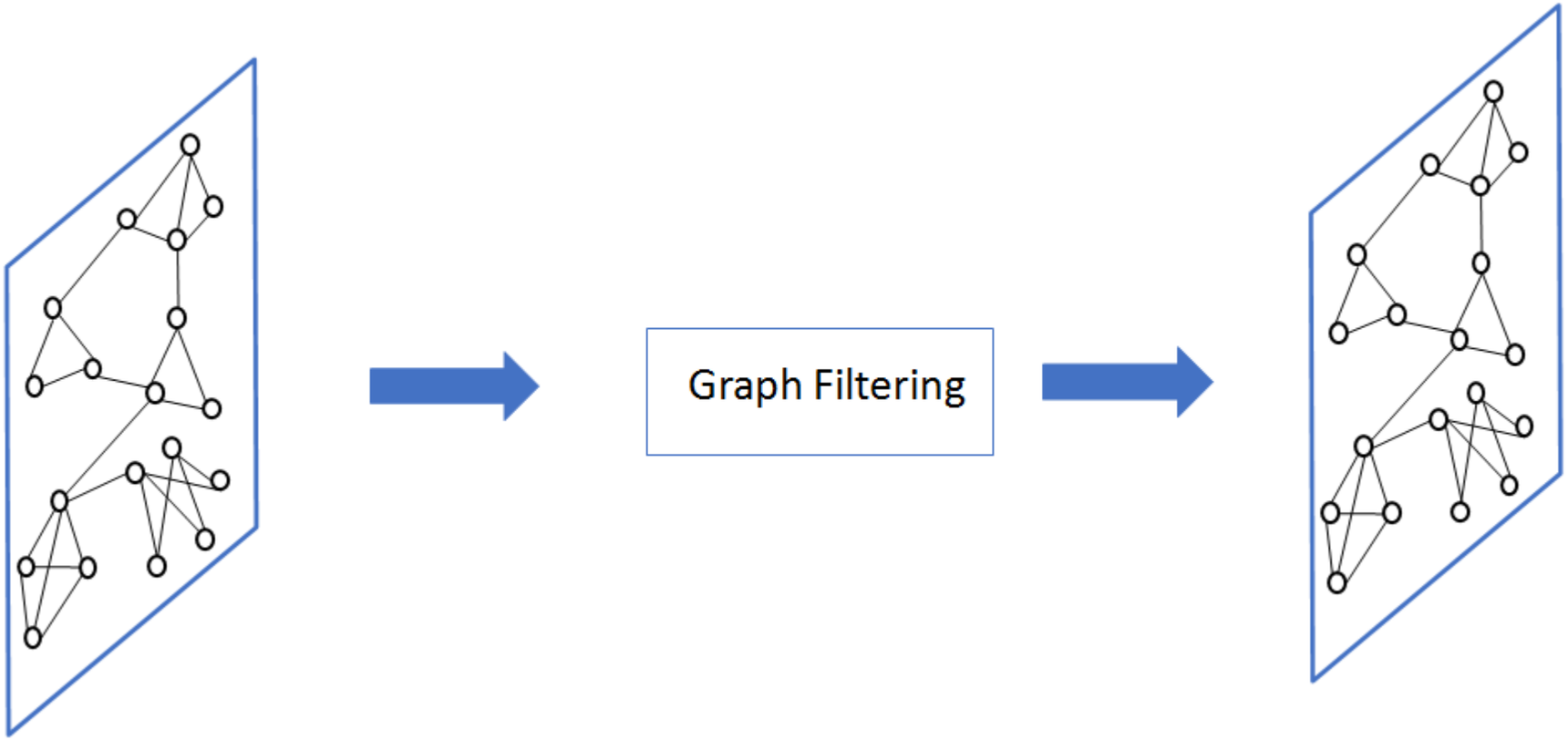


https://distill.pub/2021/gnn-intro/

Операции над графами



Graph Filtering

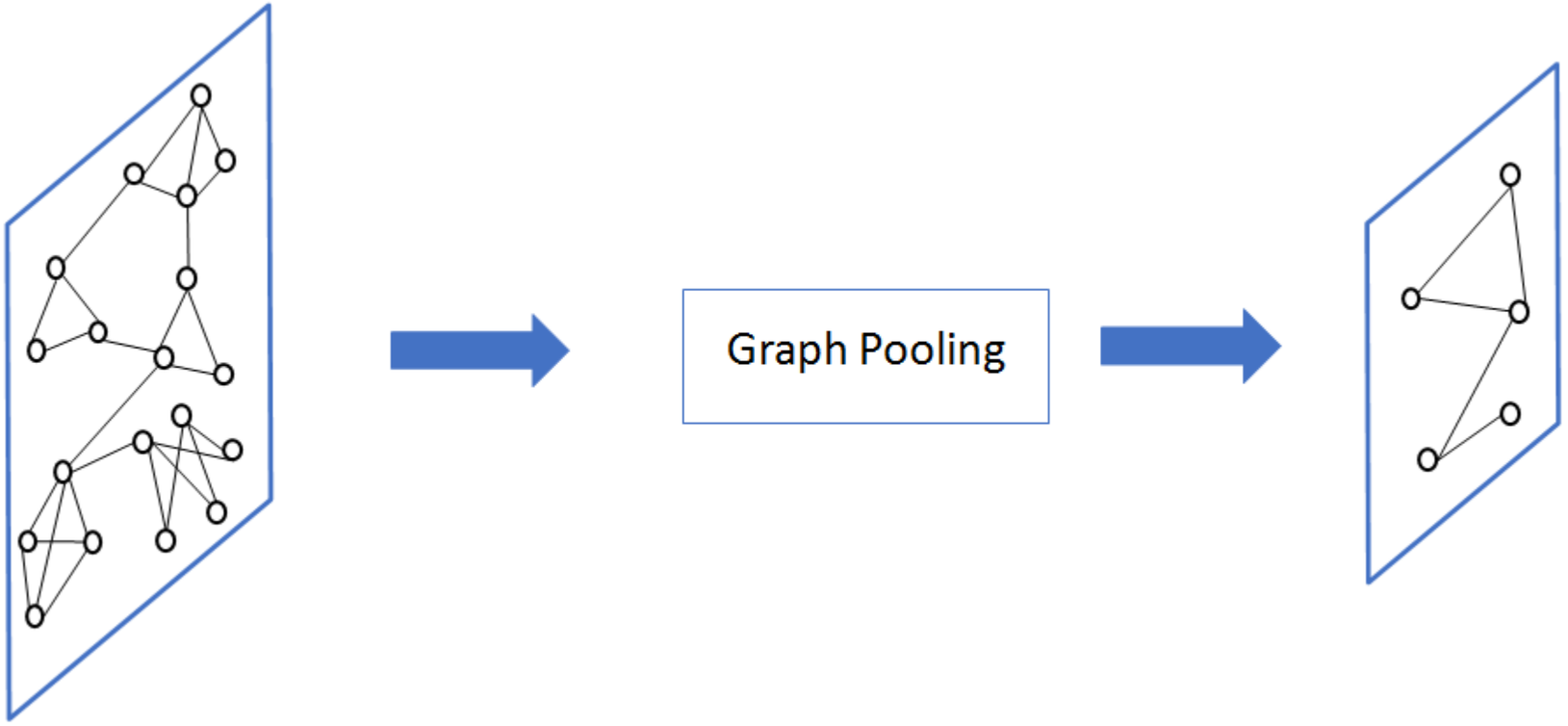


$$X \in \mathbb{R}^{n \times d} \rightarrow X' \in \mathbb{R}^{n \times k}$$

Schuster, Nakajima «Japanese and Korea voice search», 2012

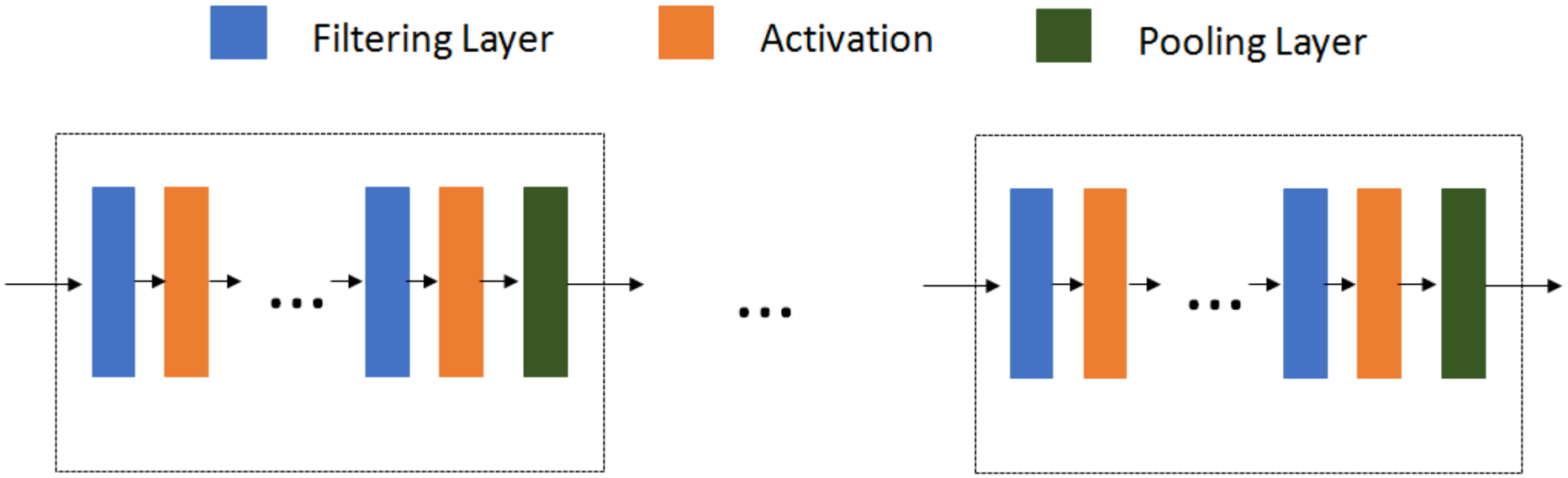
https://static.googleusercontent.com/media/research.google.com/ja//pubs/archive/37842.pdf

Graph Pooling

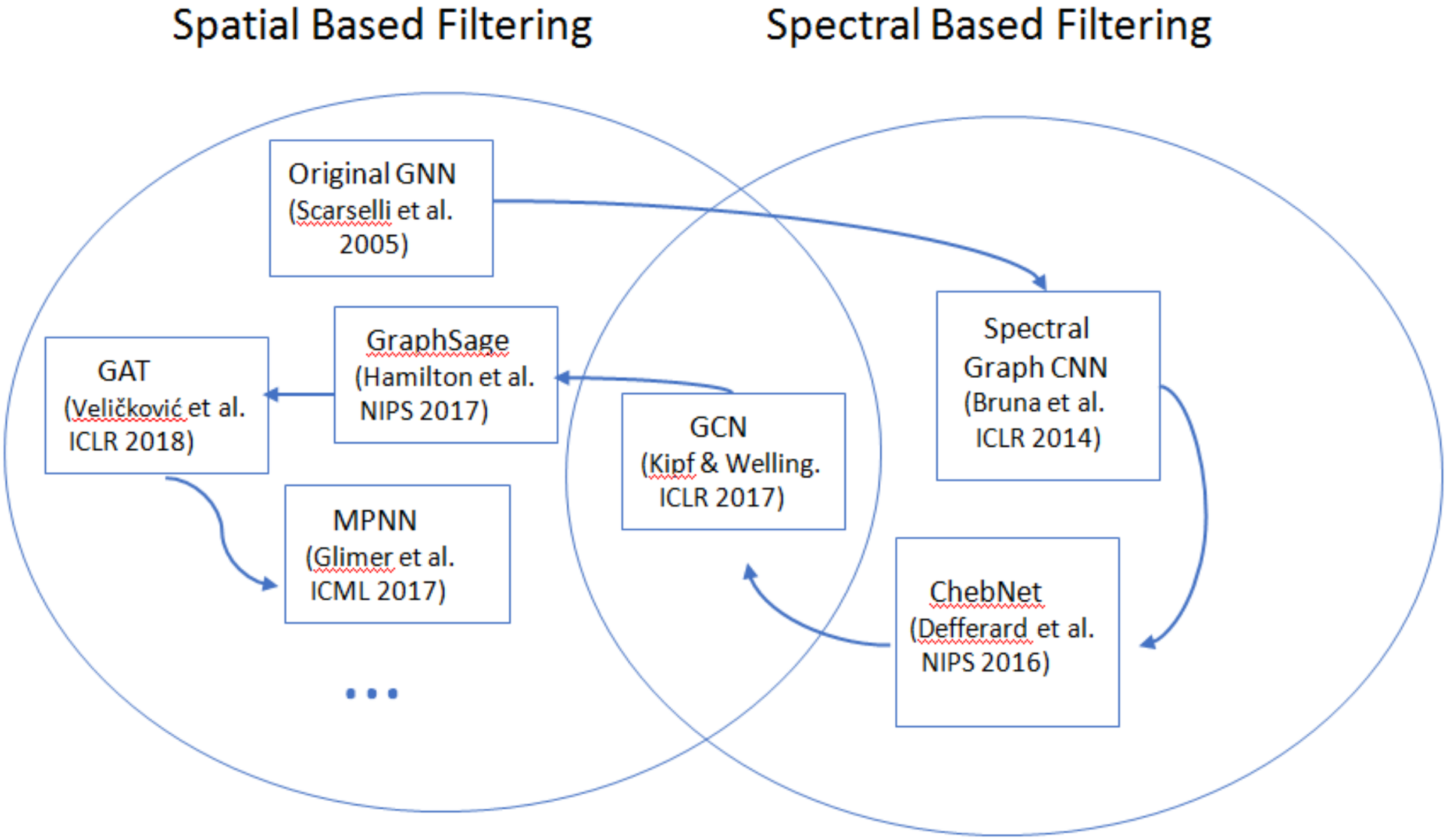


$$(V,E), X \in \mathbb{R}^{n\times d} \rightarrow (V',E'), X' \in \mathbb{R}^{n'\times k}, n' \le n$$

General GNN Framework



Two Types of Graph Filtering Operation



Генеративные графовые модели

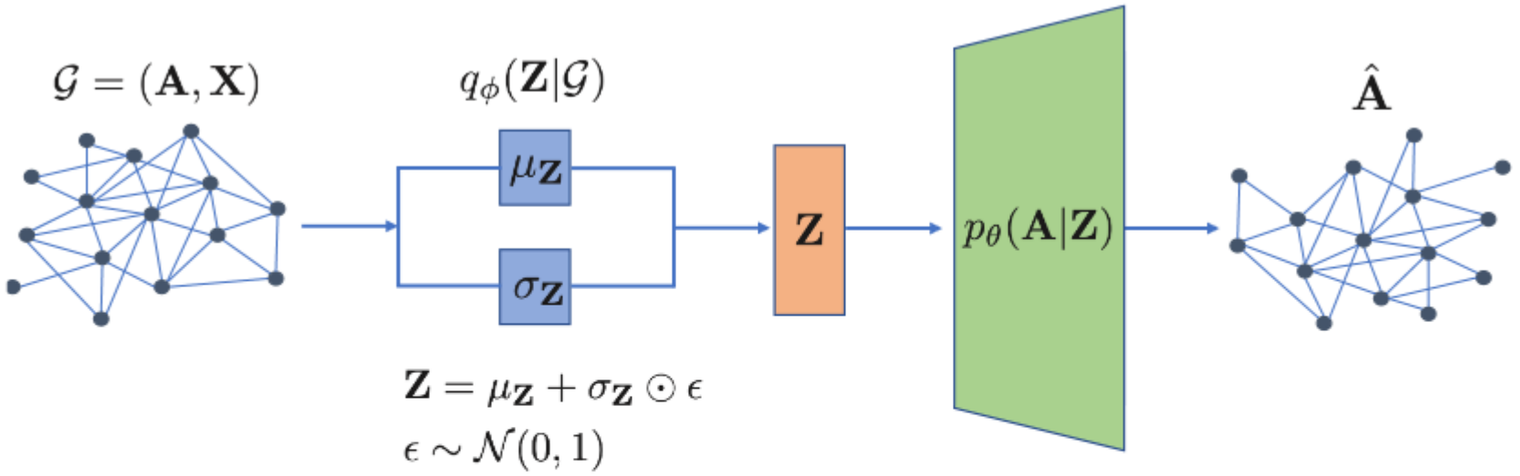


Figure 9.1: Illustration of a standard VAE model applied to the graph setting. An *encoder* neural network maps the input graph $\mathcal{G} = (\mathbf{A}, \mathbf{X})$ to a *posterior distribution* $q_\phi(\mathbf{Z}|\mathcal{G})$ over latent variables $\mathbf{Z}$. Given a sample from this posterior, the *decoder* model $p_\theta(\mathbf{A}|\mathbf{Z})$ attempts to reconstruct the adjacency matrix.

промежуточные представления м.б. представлениями вершин |V|×k
или даже всего графа GraphVAE [Simonovsky and Komodakis, 2018]

Генеративные графовые модели

Когда по латентному представлению пытаемся генерировать граф, есть две тонкости

1) сколько будет вершин

выбираем некоторое максимальное число N

при обучении ненужные маскируем

2) мы не знаем соответствия порядка порождаемых вершин и вершин реального графа

- применить алгоритм матчинга и потом считать ошибку
- задать какой-то канонический порядок, например, по убыванию степени вершины

(можно несколько канонических и усреднить ошибки)

Ссылки

Представления вершин, графов и графовые сети

Hamilton «Graph Representation Learning»

https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book.pdf

Understanding Convolutions on Graphs

https://distill.pub/2021/understanding-gnns/

A Gentle Introduction to Graph Neural Networks

https://distill.pub/2021/gnn-intro/

Zhiyuan Liu, Jie Zhou «Introduction to Graph Neural Networks» // 2020

Библиотеки

https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html

https://stellargraph.readthedocs.io/en/stable/api.html#module-stellargraph.layer

Graph Neural Networks: Models and Applications

http://cse.msu.edu/~mayao4/tutorials/aaai2021/