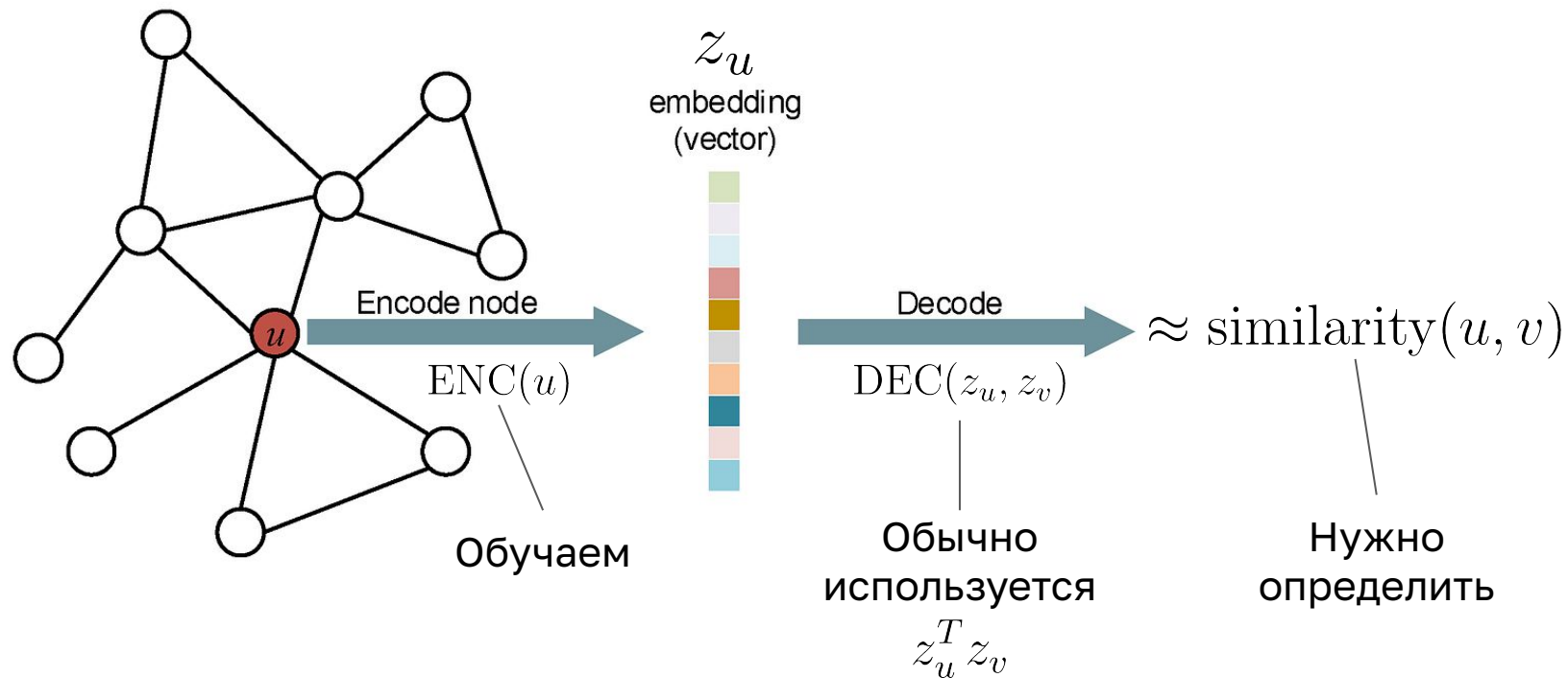


Анализ графовых данных и глубокое обучение

Азимов Рустам

В предыдущих сериях



Ограничения

- Рассмотренные методы для получения эмбедингов не позволяют работать с новыми вершинами
- Отсутствие общих параметров у эмбедингов вершин, всего $O(|V|d)$ параметров
- Не используют признаки вершин, рёбер, графов

Deep learning for graphs

Deep Graph Encoders

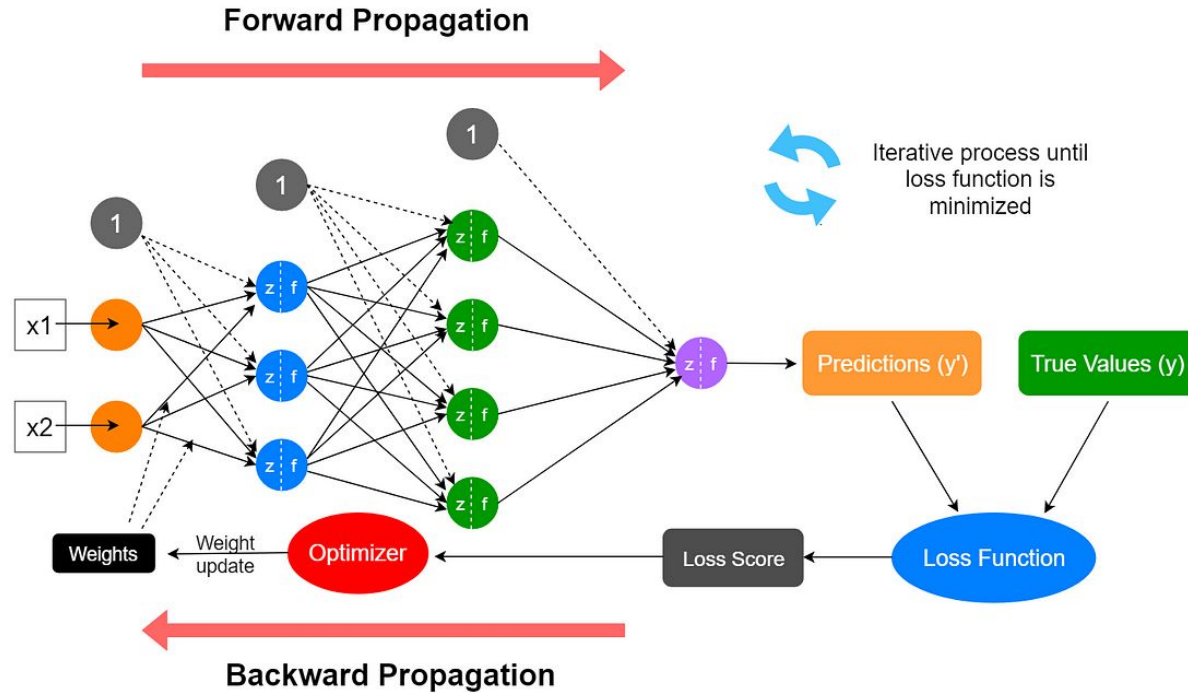
- Мы рассматривали

$$\text{ENC}(u) = \mathcal{Z}_u$$

- Основная идея теперь

$$\text{ENC}(u) = \text{MLP (основанный на структуре графа)}$$

Deep Learning

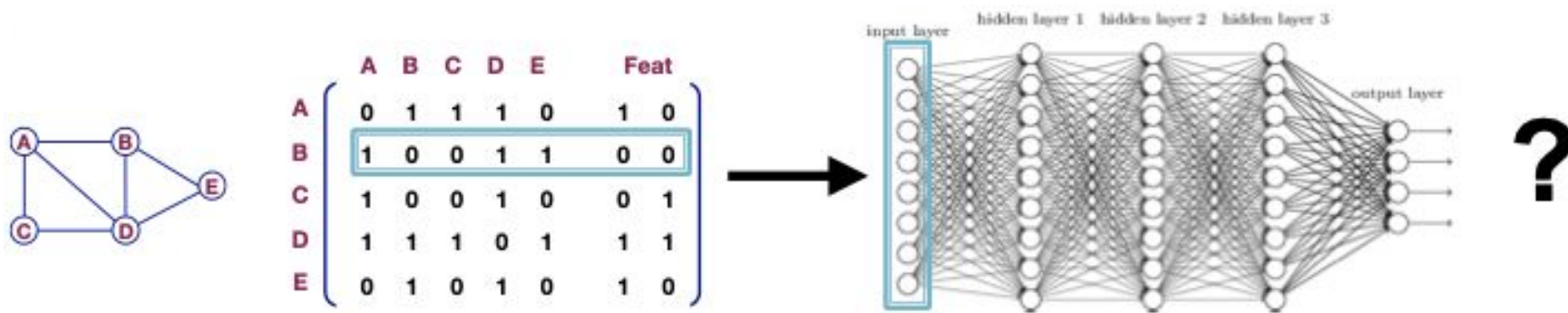


Графовые данные

- Пусть имеется граф G с множеством вершин V и матрицей смежности A
- $N(v)$ — соседи вершины v
- Будем рассматривать числовые признаки вершин $X \in \mathbb{R}^{|V| \times m}$
 - Информация о профиле в социальных сетях
 - Свойства молекул в биологических сетях
 - Если нету признаков, можно добавить one-hot кодирование вершин или вектор из единиц

Наивный подход

- Добавить к признакам матрицу смежности и скормить нейронной сети

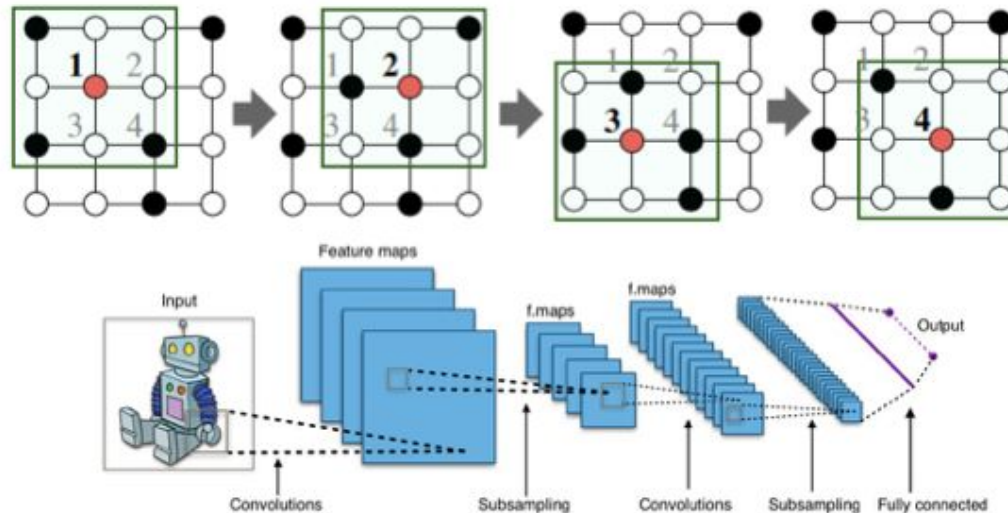


- Много параметров - $O(|V|)$
- Зависит от размера графа и порядка вершин

Сверточные сети

- Идея — обобщить классические сверточные сети от изображений/текстов (простые графы) до произвольных графов

CNN on an image:

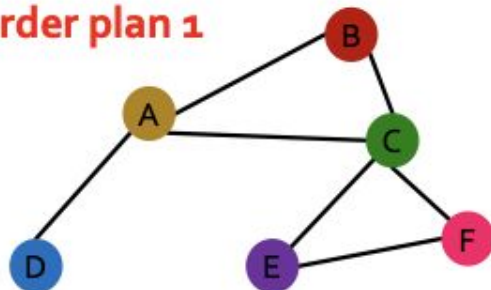


Сложность

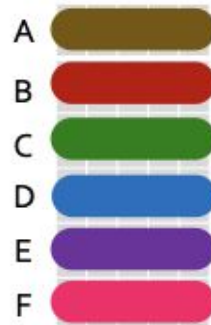
- Нету порядка на вершинах
- Нету фиксированного определения локальности или скользящего окна по вершинам графа
- Граф инвариантен к перестановкам вершин
- Граф и представления вершин должны быть одинаковыми, не важно какой порядок обхода мы выбрали

Инвариантность к перестановкам

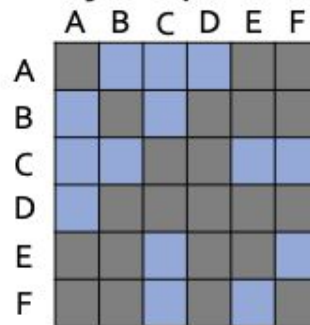
Order plan 1



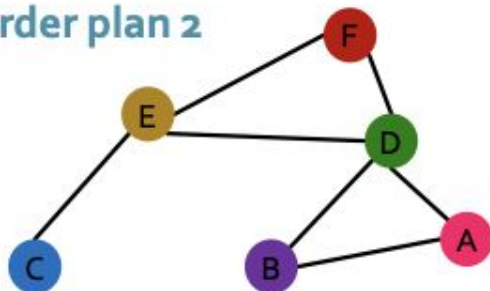
Node features X_1



Adjacency matrix A_1



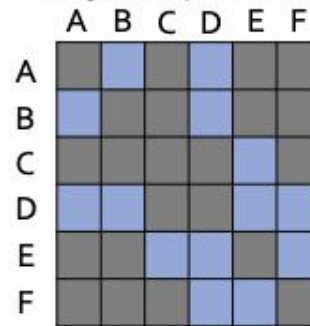
Order plan 2



Node features X_2

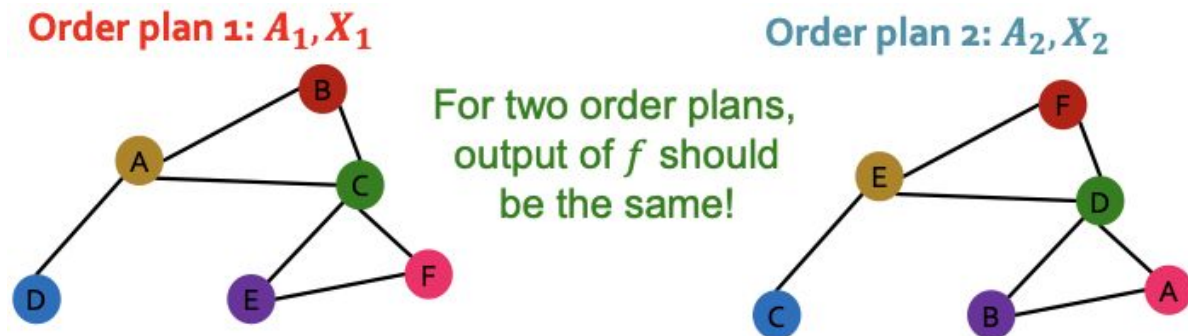


Adjacency matrix A_2

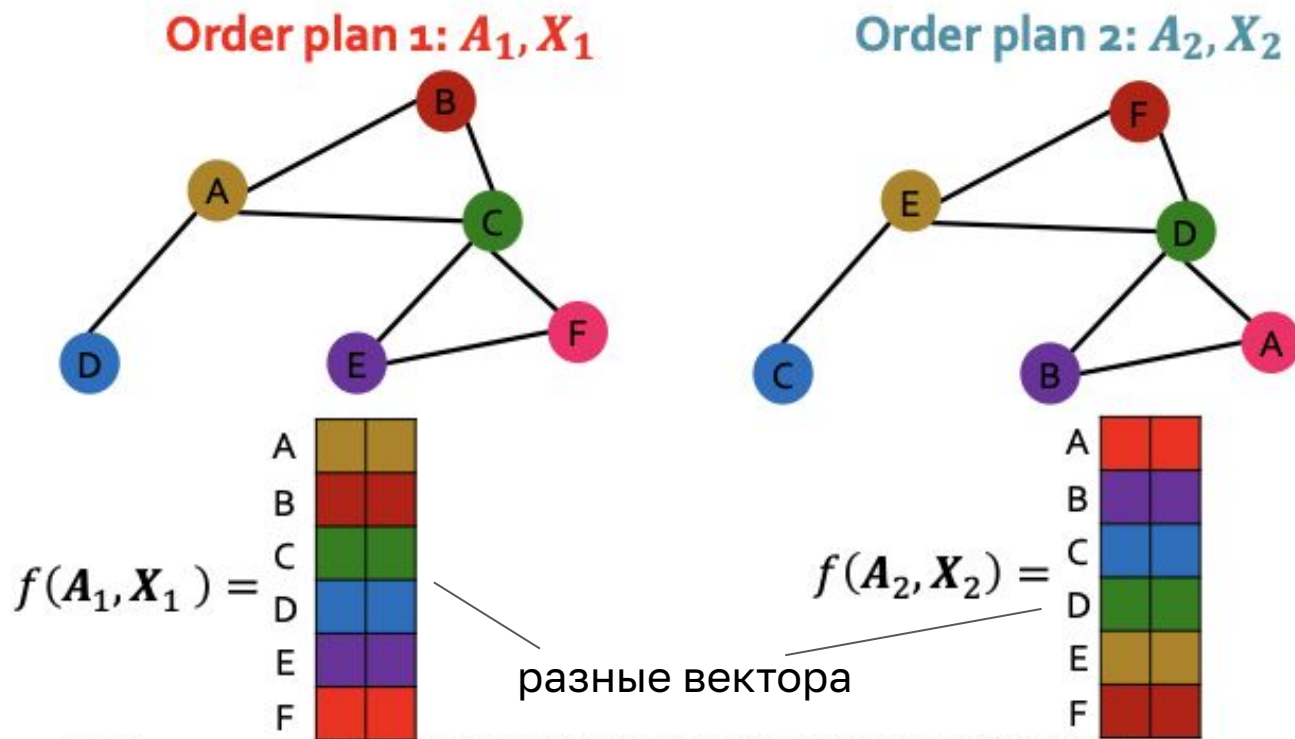


Инвариантность к перестановкам

- Пусть мы обучаем функцию f которая отображает граф $G = (A, X)$ в вектор \mathbb{R}^d
- Тогда f инвариантна к перестановкам, если $f(A, X) = f(PAP^T, PX)$ для любой перестановки P
- Например, $f(A_1, X_1) = f(A_2, X_2)$



Эмбединги для каждой вершины

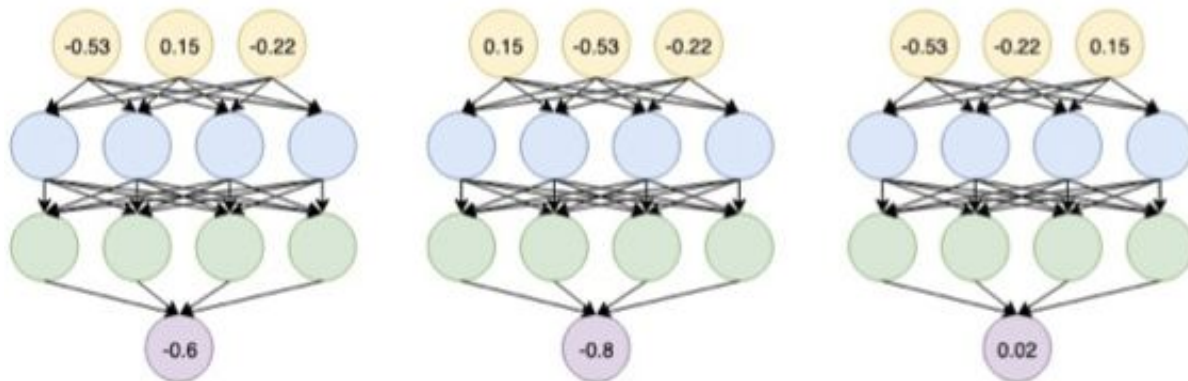


Эквивариантность к перестановкам

- Пусть мы обучаем функцию f , которая отображает граф $G = (A, X)$ в матрицу $\mathbb{R}^{|V| \times d}$
- Тогда f эквивариантна к перестановкам, если $Pf(A, X) = f(PAP^T, PX)$ для любой перестановки P
- Например, GNN состоят из инвариантных и эквивариантных к перестановкам вершин функций

Цель

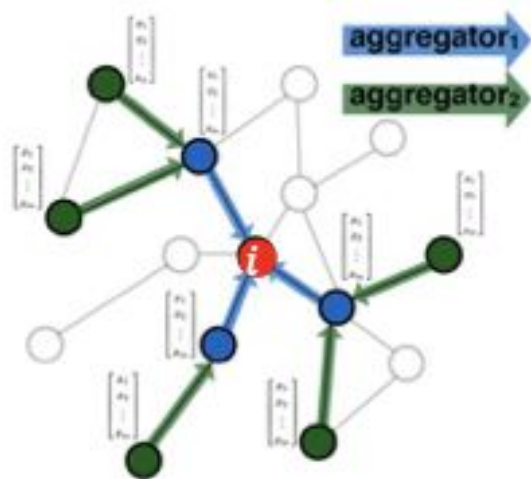
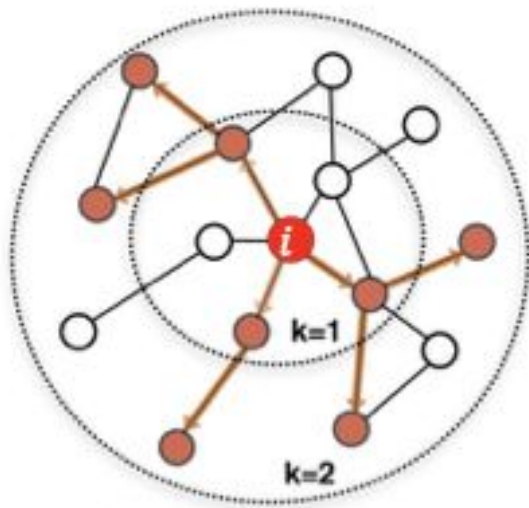
- Спроектировать нейронные сети, которые будут инвариантны/эквивариантны к перестановкам вершин
- Классические нейронные сети не являются таковыми, поэтому наивный подход не работает



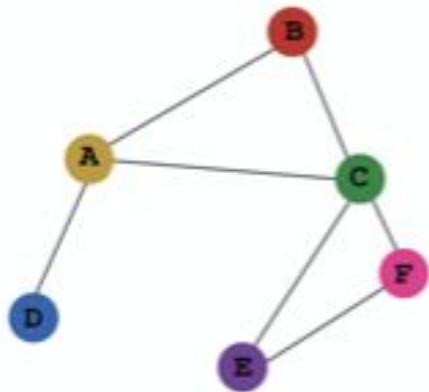
Graph Convolutional Networks

GCN

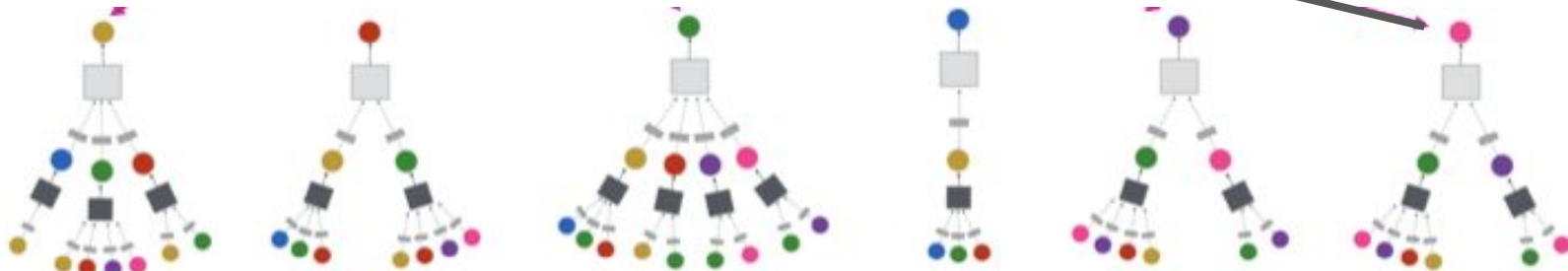
- Граф вычислений определяется соседями вершин
- Через них информация распространяется и агрегируется



Графы вычислений

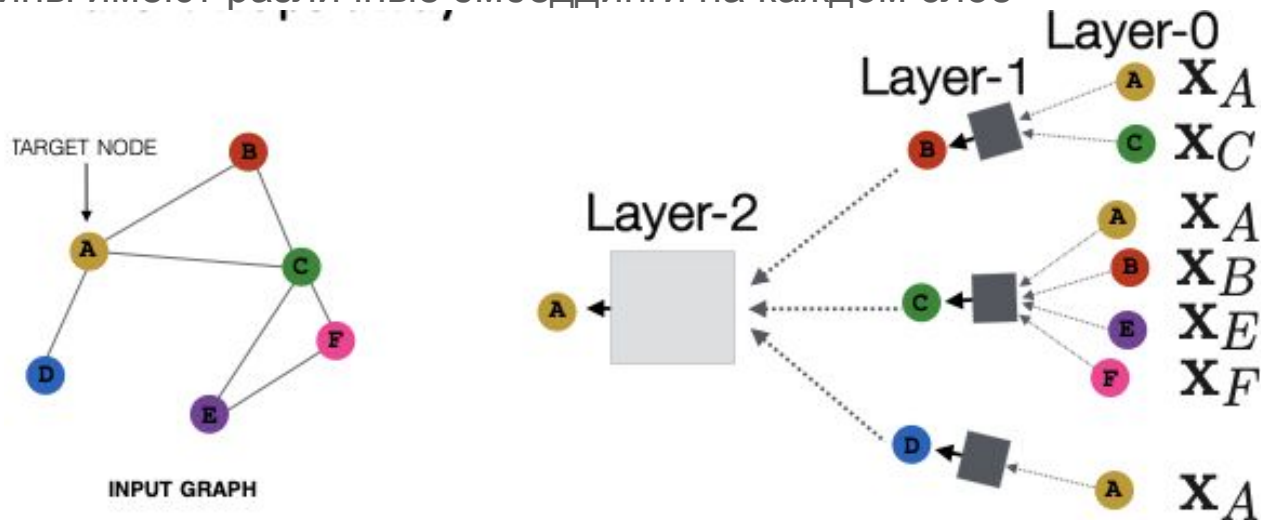


граф вычислений для
вершины F

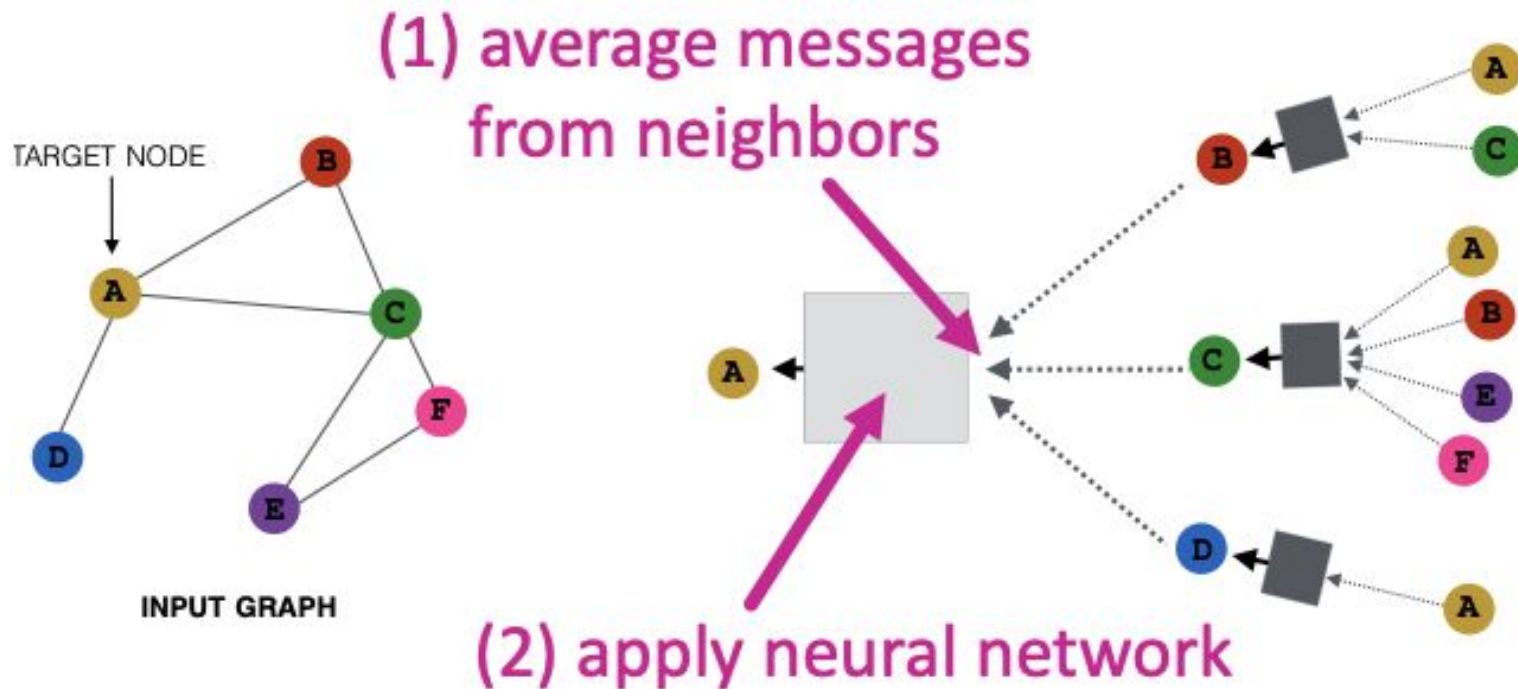


Глубина графа вычислений

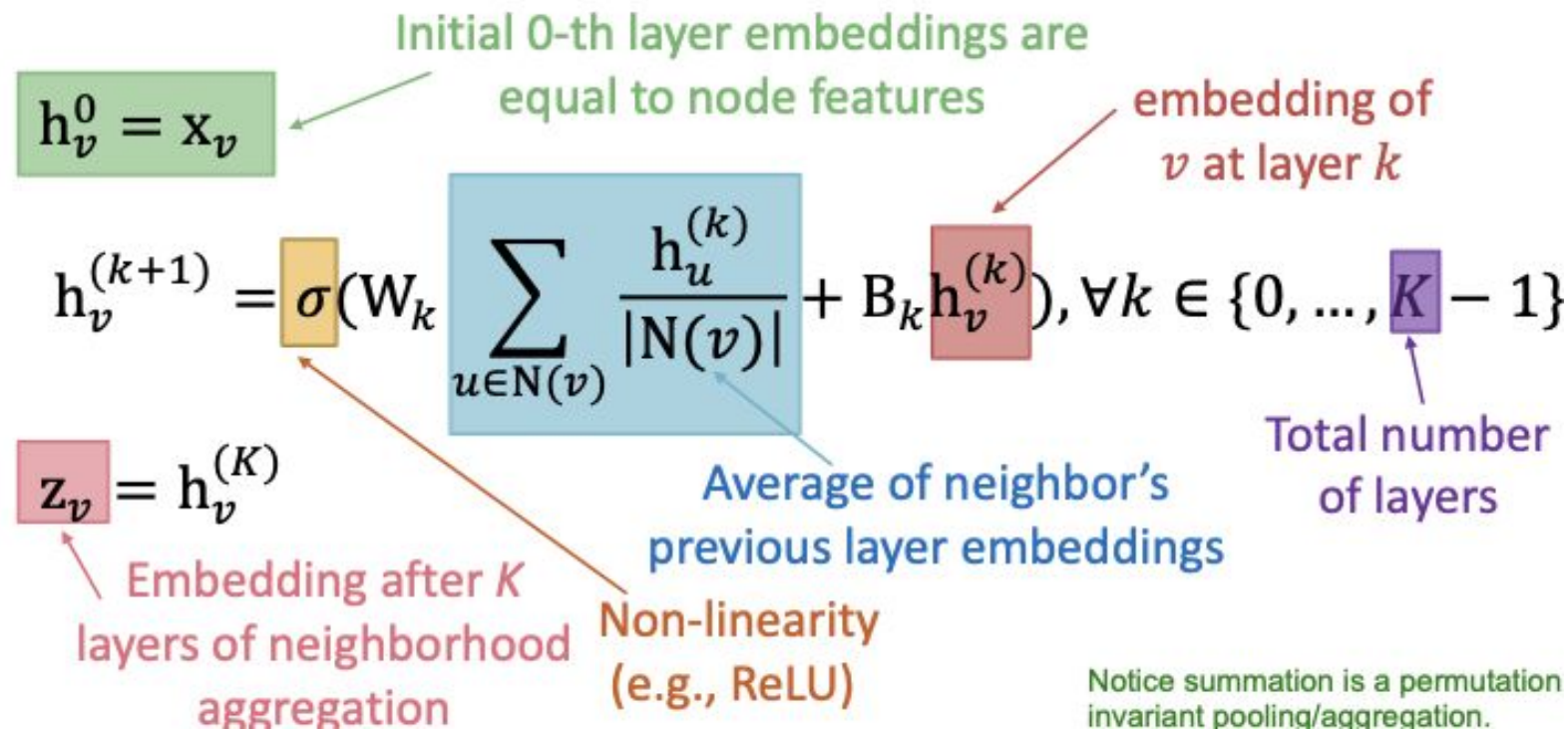
- Глубина может быть любой, от нее зависит насколько далекое соседство рассматривается для распространения информации
- Вершины имеют различные эмбединги на каждом слое



Базовый подход

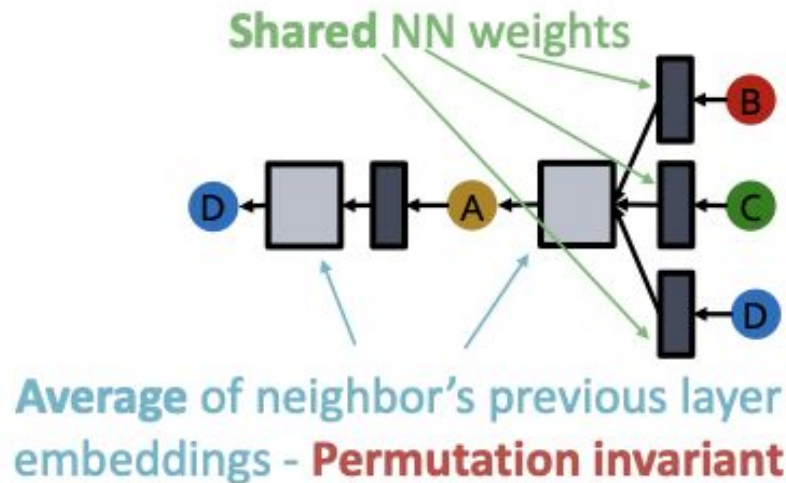
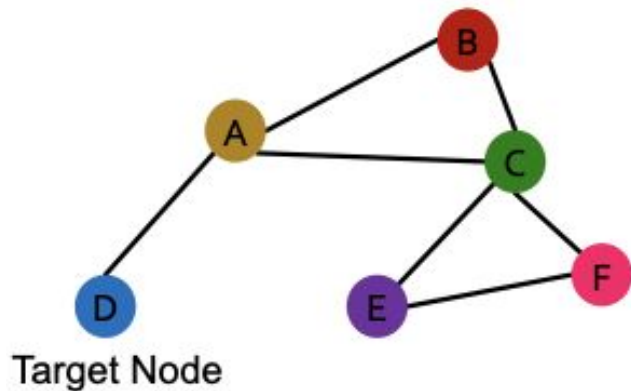


Базовый подход



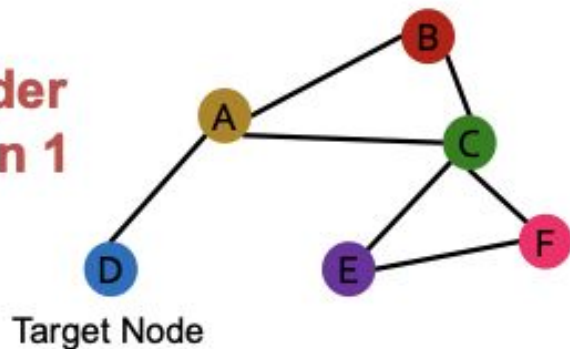
Свойства GCN для одной вершины

- GCN, вычисляющая эмбединг одной вершины, инвариантна относительно перестановок вершин

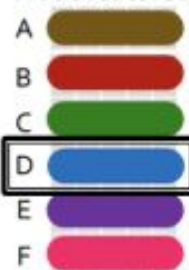


Эквивариантность GCN

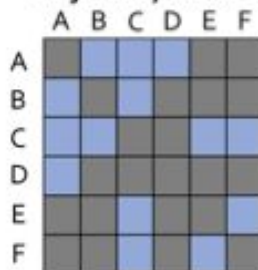
Order
plan 1



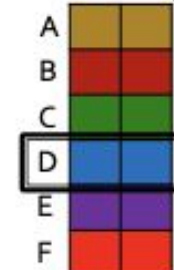
Node feature X_1



Adjacency matrix A_1

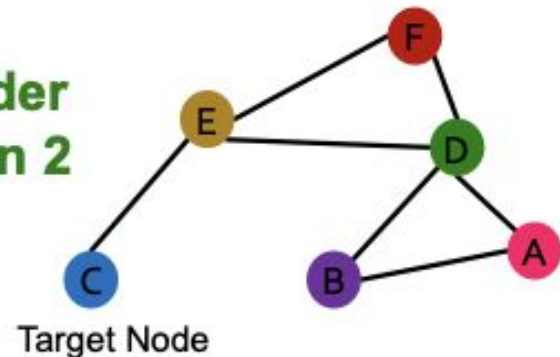


Embeddings H_1

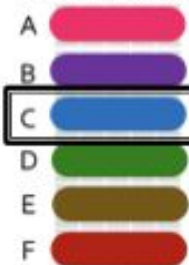


Permute the input, the output also permutes
accordingly - permutation equivariant

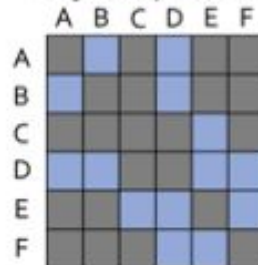
Order
plan 2



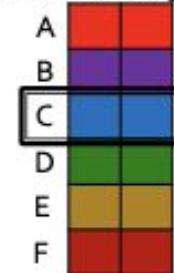
Node feature X_2



Adjacency matrix A_2



Embeddings H_2



Обучение

Trainable weight matrices
(i.e., what we learn)

$$\begin{aligned} h_v^{(0)} &= x_v \\ h_v^{(k+1)} &= \sigma \left(W_k \sum_{u \in N(v)} \frac{h_u^{(k)}}{|N(v)|} + B_k h_v^{(k)} \right), \forall k \in \{0..K-1\} \\ z_v &= h_v^{(K)} \end{aligned}$$

Final node embedding

- Оптимизируем любую loss-функцию от полученных эмбедингов, например с помощью SGD

Агрегация в матричной форме

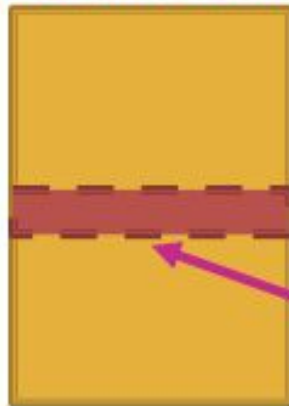
- Let $H^{(k)} = [h_1^{(k)} \dots h_{|V|}^{(k)}]^T$
- Then: $\sum_{u \in N_v} h_u^{(k)} = A_{v,:} H^{(k)}$
- Let D be diagonal matrix where
 $D_{v,v} = \text{Deg}(v) = |N(v)|$
 - The inverse of D : D^{-1} is also diagonal:
 $D_{v,v}^{-1} = 1/|N(v)|$
- **Therefore,**

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



$$H^{(k+1)} = D^{-1} A H^{(k)}$$

Matrix of hidden embeddings $H^{(k-1)}$



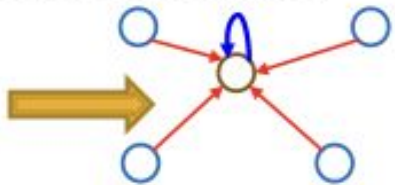
$h_i^{(k-1)}$

GCN в матричной форме

Re-writing update function in matrix form:

$$H^{(k+1)} = \sigma(\tilde{A}H^{(k)}W_k^T + H^{(k)}B_k^T)$$

where $\tilde{A} = D^{-1}A$


$$H^{(k)} = [h_1^{(k)} \dots h_{|V|}^{(k)}]^T$$

- Red: neighborhood aggregation
- Blue: self transformation
- Можно использовать реализации умножения разреженных матриц (\tilde{A} - разреженная матрица)
- Другие GNN с более сложными агрегациями не всегда могут быть записаны в такой простой матричной форме

Процесс обучения

- Полученные эмбединги используются для предсказания
- Обучение с учителем: \mathbf{y} — метки в тренировочном наборе

$$\min_{\Theta} \mathcal{L}(\mathbf{y}, f_{\Theta}(\mathbf{z}_v))$$

- Обучение без учителя: нету меток, используем структуру графа

Пример обучения без учителя

- Схожие вершины должны иметь близкие эмбединги

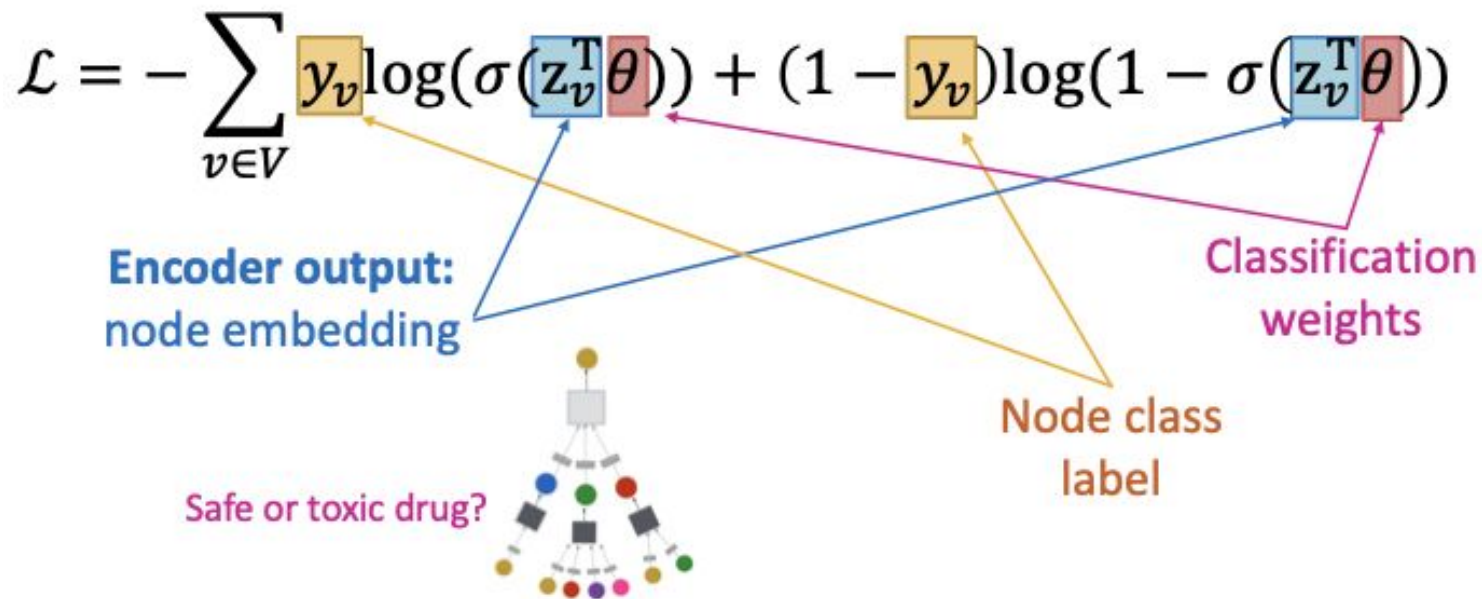
$$\min_{\Theta} \mathcal{L} = \sum_{z_u, z_v} \text{CE}(y_{u,v}, \text{DEC}(z_u, z_v))$$

- where $y_{u,v} = 1$ when node u and v are **similar**
- $z_u = f_{\Theta}(u)$ and $\text{DEC}(\cdot, \cdot)$ is the dot product

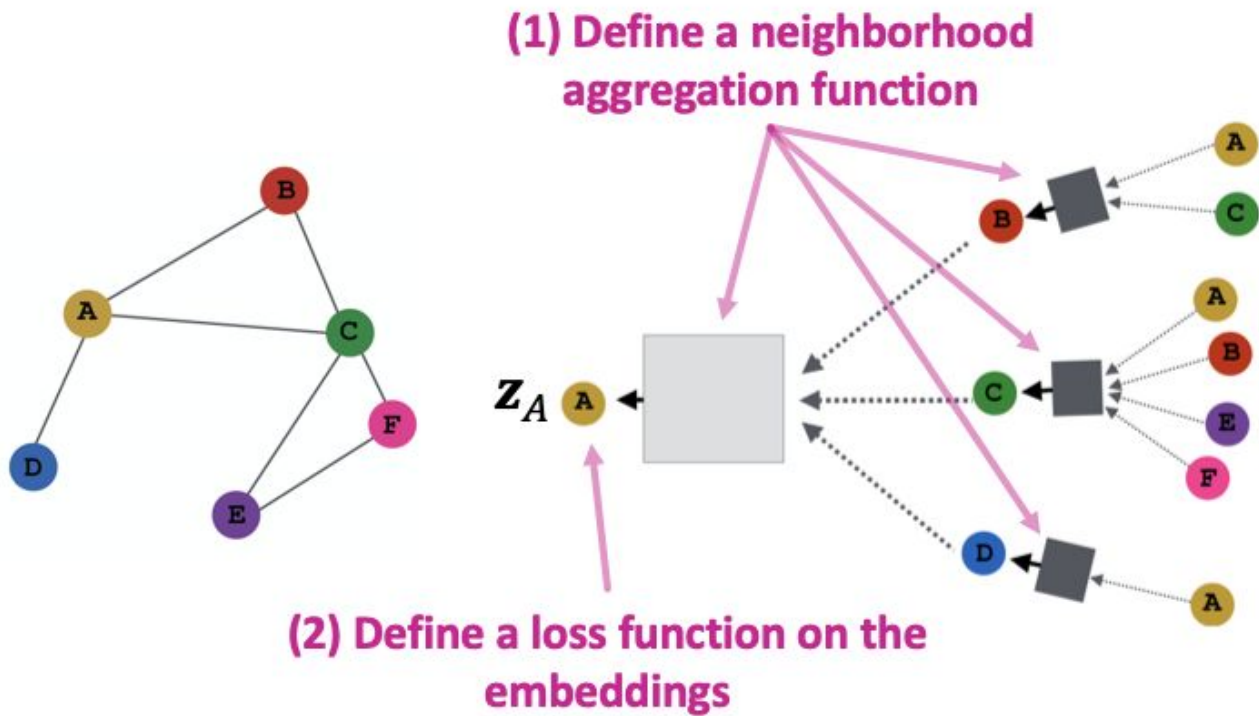
- Схожесть вершин определяем самостоятельно
 - Случайные блуждания (DeepWalk, node2vec, ...)

Пример обучения с учителем

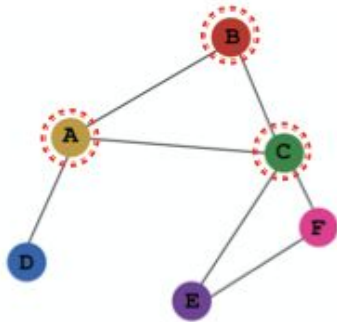
- Классификация вершин



Архитектура

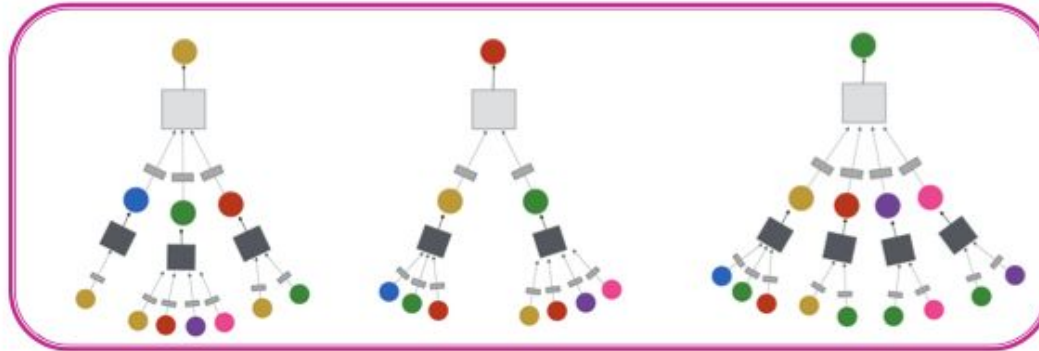


Архитектура

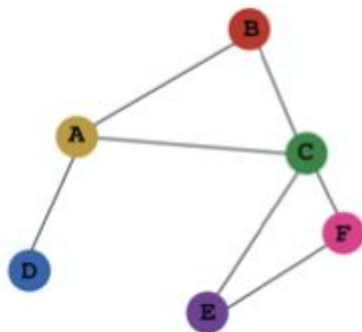


INPUT GRAPH

(3) Train on a set of nodes, i.e.,
a batch of compute graphs



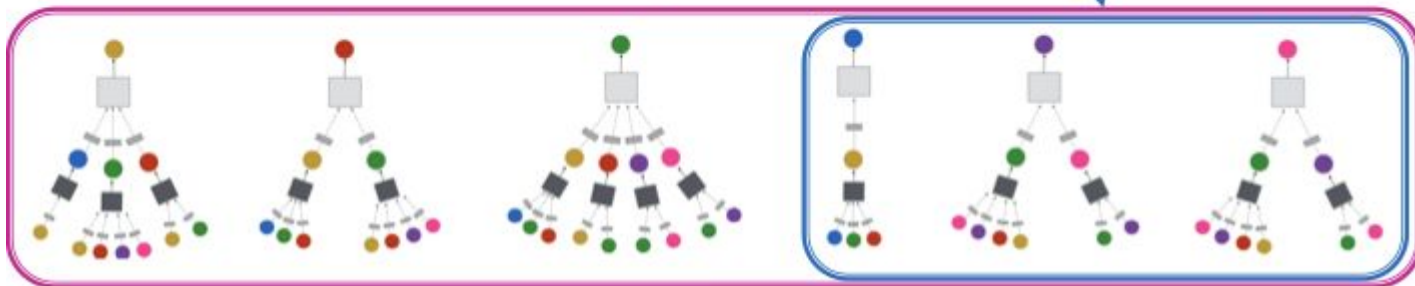
Архитектура



INPUT GRAPH

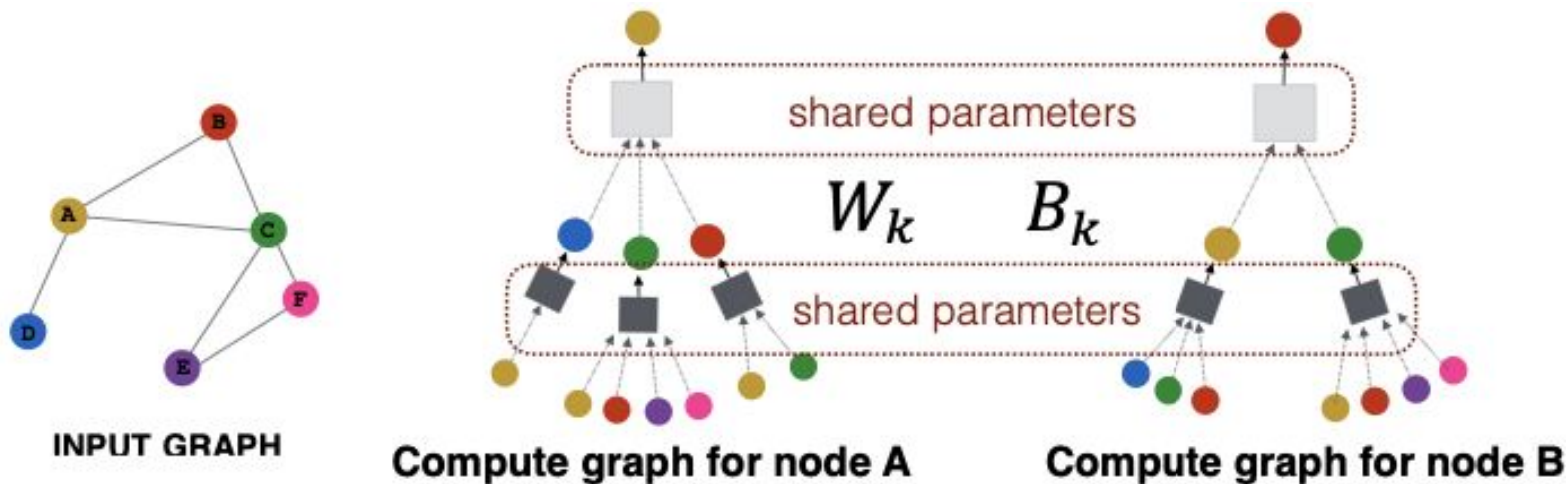
**(4) Generate embeddings
for nodes as needed**

**Even for nodes we never
trained on!**

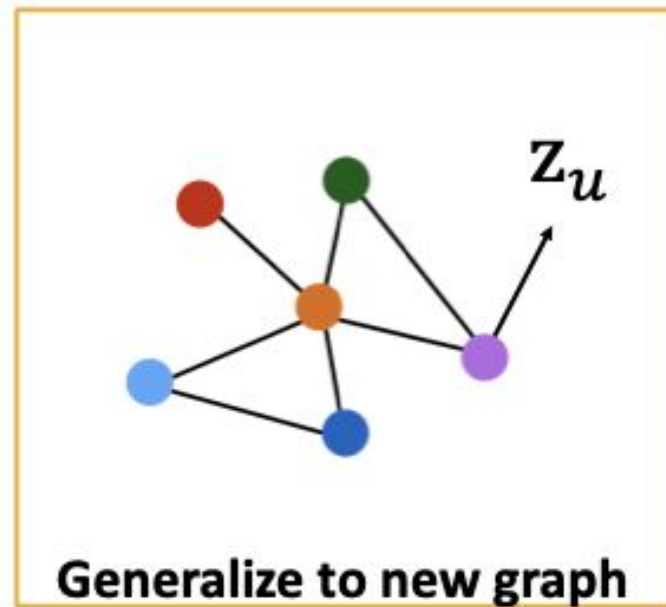
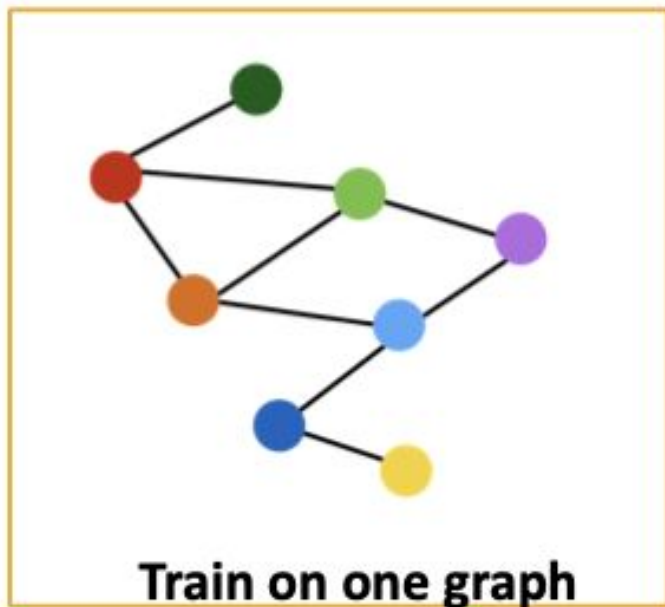


Общие параметры

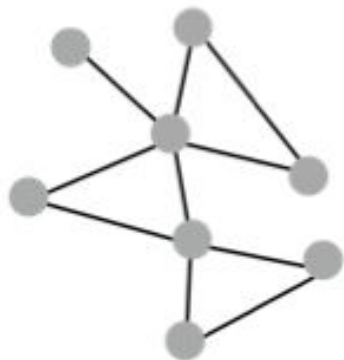
- Количество параметров сублинейно от $|V|$
- Модель может получать эмбединги и для новых вершин



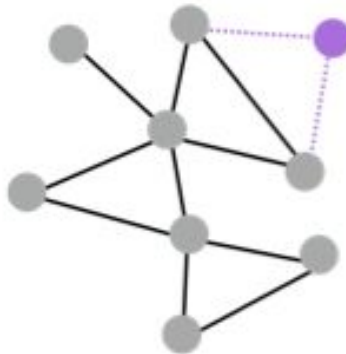
Использование на новых графах



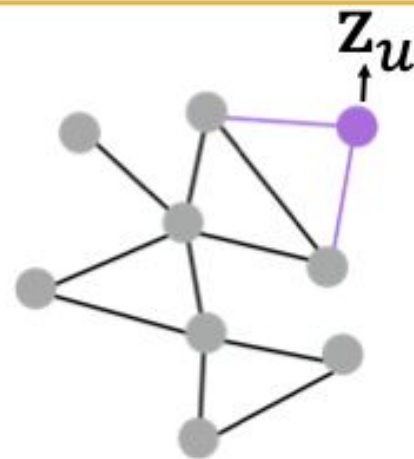
Добавление новых вершин



Train with snapshot



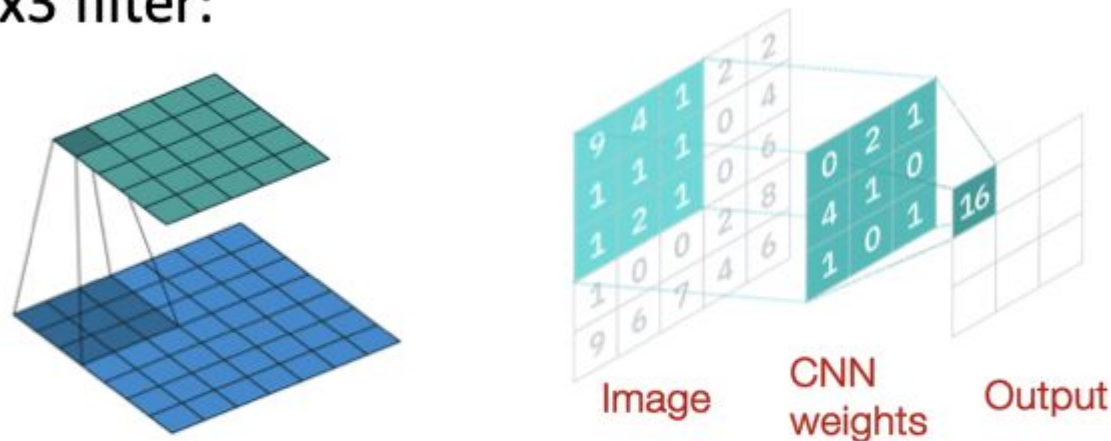
New node arrives



**Generate embedding
for new node**

CNN

Convolutional neural network (CNN) layer with 3x3 filter:

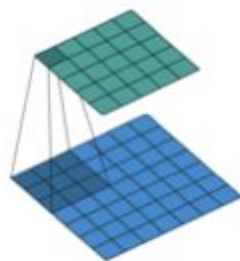


$$\text{CNN formulation: } h_v^{(l+1)} = \sigma(\sum_{u \in N(v) \cup \{v\}} W_l^u h_u^{(l)}), \quad \forall l \in \{0, \dots, L-1\}$$

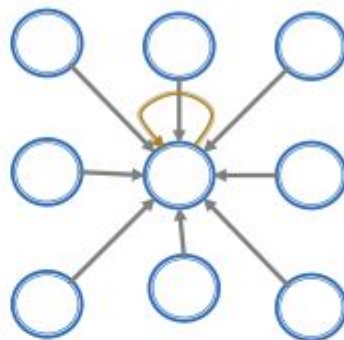
$N(v)$ represents the 8 neighbor pixels of v .

GNN vs. CNN

Convolutional neural network (CNN) layer with 3x3 filter:



Image



Graph

- GNN formulation: $h_v^{(l+1)} = \sigma(\mathbf{W}_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)}), \forall l \in \{0, \dots, L-1\}$
- CNN formulation: (previous slide) $h_v^{(l+1)} = \sigma(\sum_{u \in N(v) \cup \{v\}} W_l^u h_u^{(l)}), \forall l \in \{0, \dots, L-1\}$
if we rewrite: $h_v^{(l+1)} = \sigma(\sum_{u \in N(v)} \mathbf{W}_l^u h_u^{(l)} + B_l h_v^{(l)}), \forall l \in \{0, \dots, L-1\}$

Процесс обучения

- В CNN фиксируются соседи, порядок вершин
- CNN не инвариантна/эквивариантна относительно перестановок вершин
- GNN можно рассмотреть как обобщение CNN от изображений до любых графов

Заклучение

