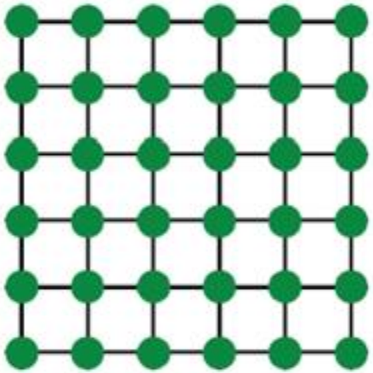
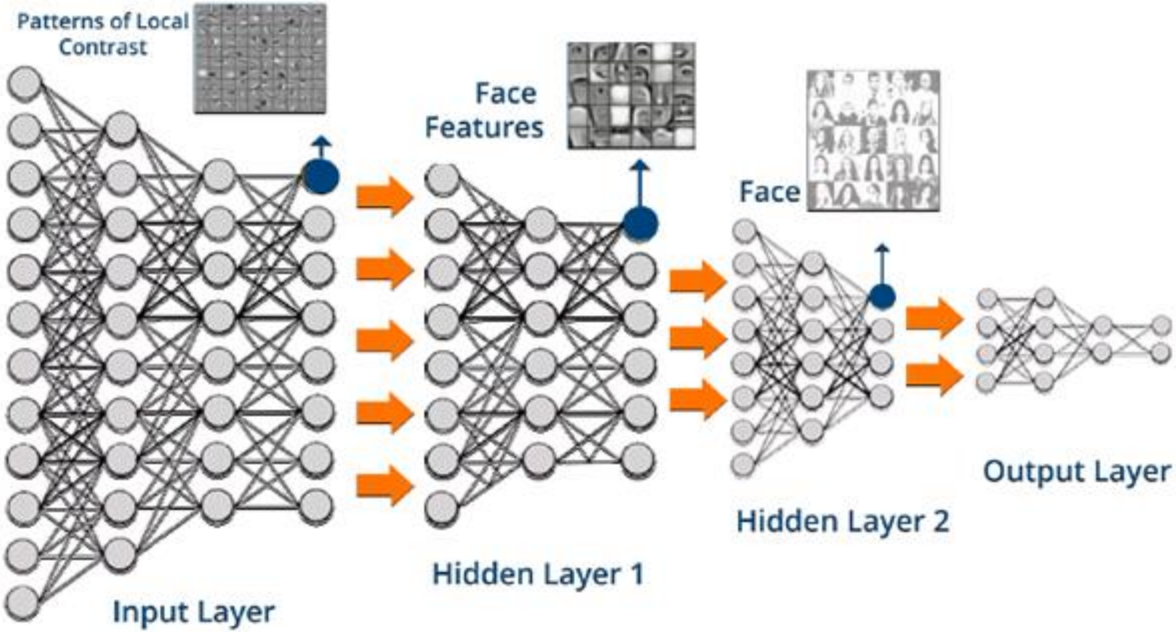
Graph Neural Network

Jure Leskovec, Michele Catasta, Stanford University

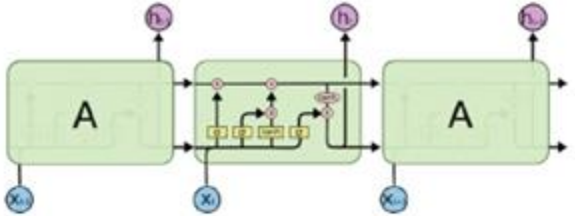
Modern DL Toolbox is Designed for Sequences & Grids



Images

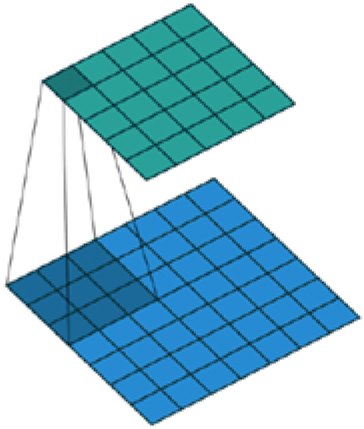


Text/Speech

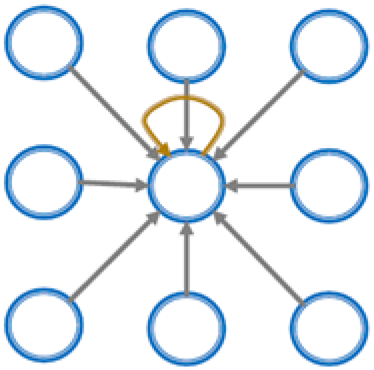


From Images to Graphs

Single CNN layer with 3x3 filter:



Image

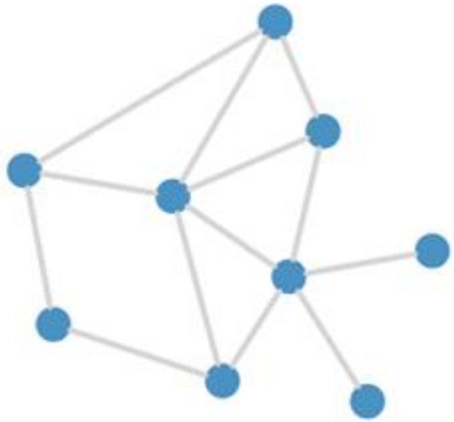


Graph

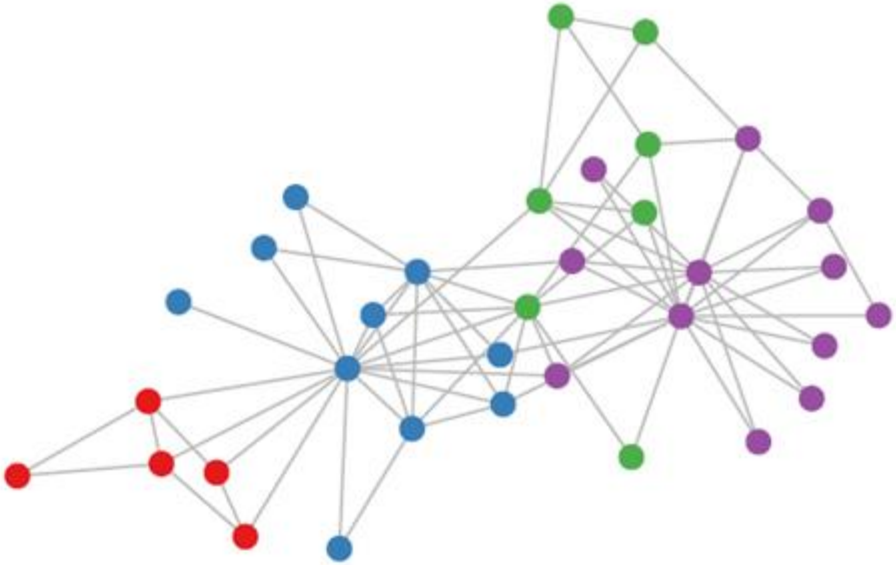
Transform information at the neighbors and combine it:

- Transform "messages" h_i from neighbors: $W_i\ h_i$
- Add them up: $\sum_i W_i\ h_i$

How about these graphs?



or this:



Biological networks, Medical networks, Social networks, Information networks, Knowledge graphs, Communication networks, Web graph

Problem Setup

- **Encoder:** Map a node to a low-dimensional vector:

d-dimensional embedding

$$\mathrm{ENC}(v) = \mathbf{z}_v$$

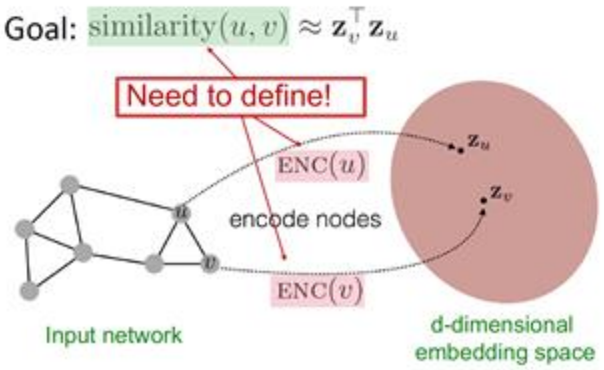
node in the input graph

- **Similarity function** defines how relationships in the input network map to relationships in the embedding space:

$$\mathrm{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

Similarity of *u* and *v* in the network

dot product between node embeddings



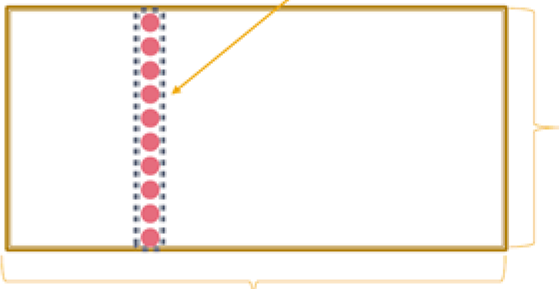
Shallow Encoding

- Simplest encoding approach: **encoder is just an embedding-lookup**

$$\mathrm{ENC}(v) = \mathbf{Z}\mathbf{v}$$

$\mathbf{Z} \in \mathbb{R}^{d\times|\mathcal{V}|}$ matrix, each column is a node embedding [what we learn!]

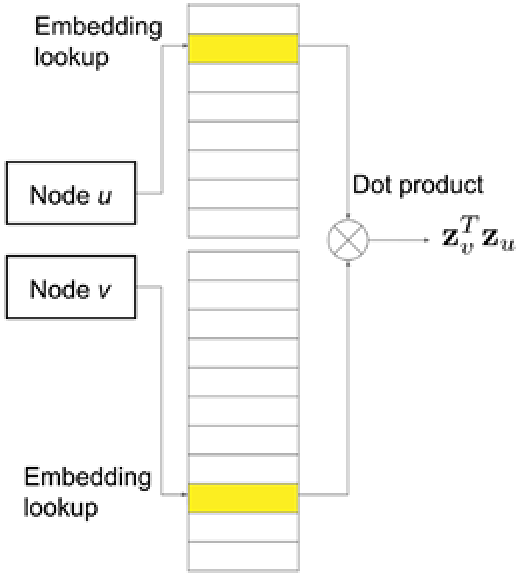
$\mathbf{v} \in \mathbb{I}^{|\mathcal{V}|}$ indicator vector, all zeroes except a one in column indicating node *v*

$\mathbf{Z} =$ 

Each node is assigned to a unique embedding vector

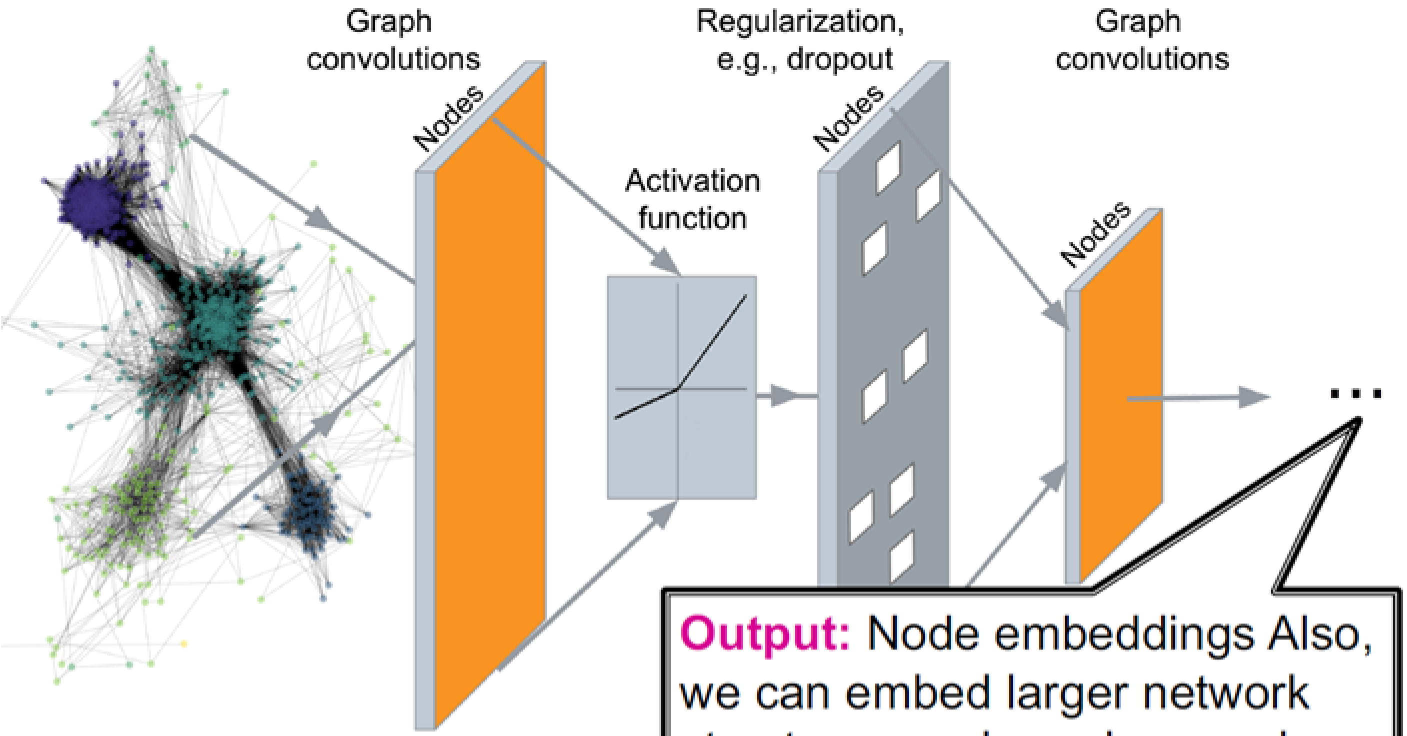
Many methods: DeepWalk, node2vec, TransE

Limitations of Shallow Encoding



- **O(|V|) parameters are needed:**
  - No sharing of parameters between nodes
  - Every node has its own unique embedding
- **Inherently "transductive":**
  - Cannot generate embeddings for nodes that are not seen during training
- **Do not incorporate node features:**
  - Many graphs have features that we can and should leverage

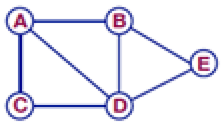
Deep Graph Encoders



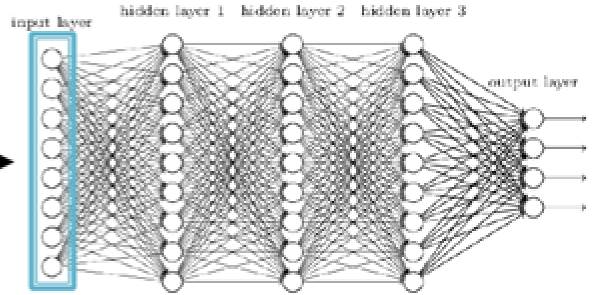
Output: Node embeddings Also, we can embed larger network structures, subgraphs, graphs

A Naive Approach

- Join adjacency matrix and features
- Feed them into a deep neural net:



$$
\begin{array}{c|ccccc|cc}
 & A & B & C & D & E & \multicolumn{2}{c}{Feat} \\
\hline
A & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
B & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
C & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
D & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
E & 0 & 1 & 0 & 1 & 0 & 1 & 0
\end{array}
\longrightarrow
$$



?

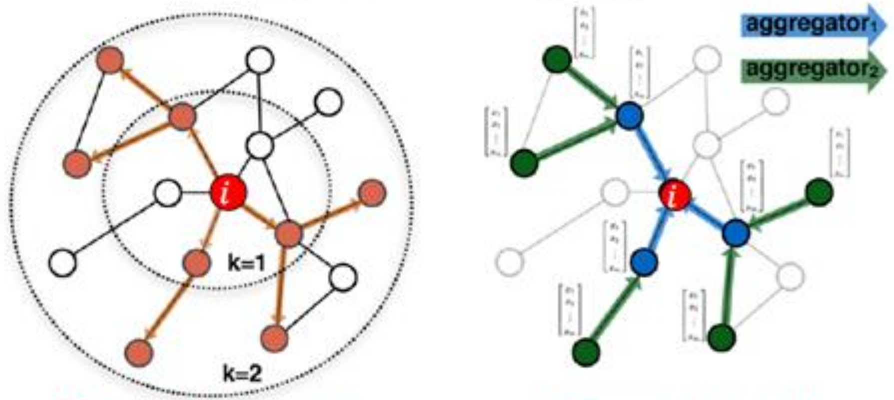
- Issues with this idea:
  - $O(N)$ parameters
  - Not applicable to graphs of different sizes
  - Not invariant to node ordering

Setup

- Assume we have a graph G:
  - V is the **vertex set**
  - $\boldsymbol{A}$ is the **adjacency matrix** (assume binary)
  - $\boldsymbol{X} \in \mathbb{R}^{m \times |V|}$ is a matrix of **node features**
  - Node features:
    - Social networks: User profile, User image
    - Biological networks: Gene expression profiles, gene functional information
    - No features:
      - Indicator vectors (one-hot encoding of a node)
      - Vector of constant 1: [1, 1, …, 1]

Message Passing of Nodes

Idea: Node's neighborhood defines a computation graph



Determine node computation graph

Propagate and transform information

Learn how to propagate information across the graph to compute node features

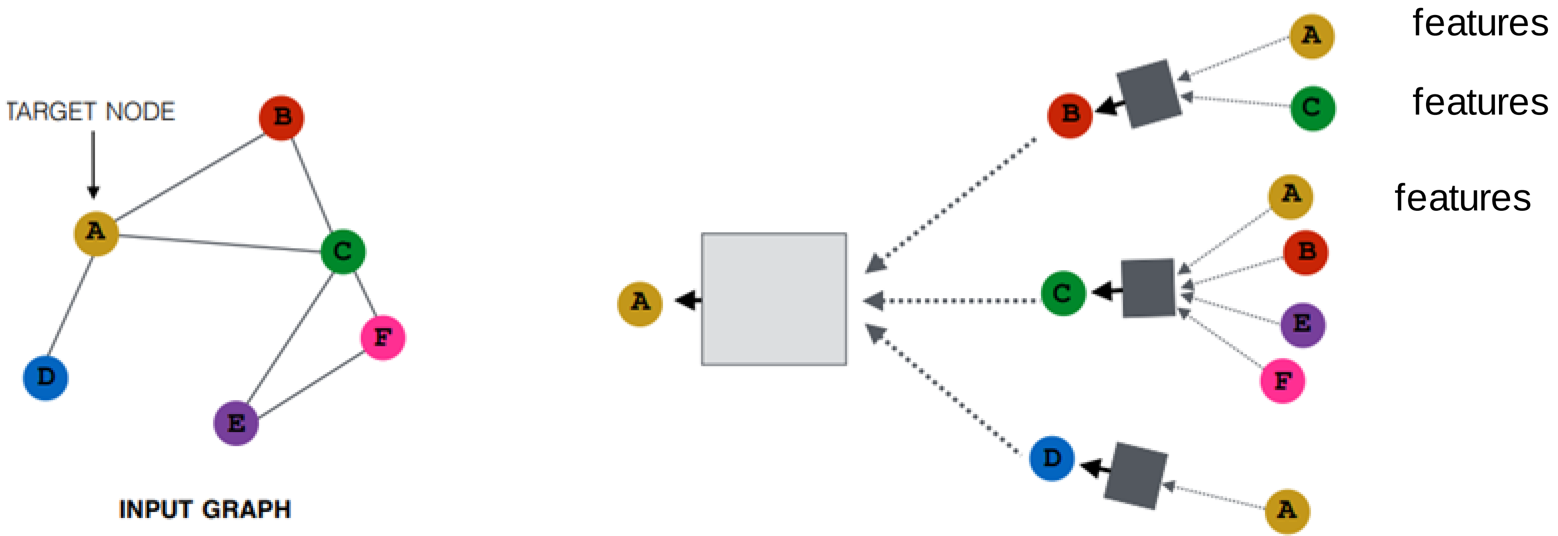
Advantages:

Capturing the Structure

Utilizing feature information

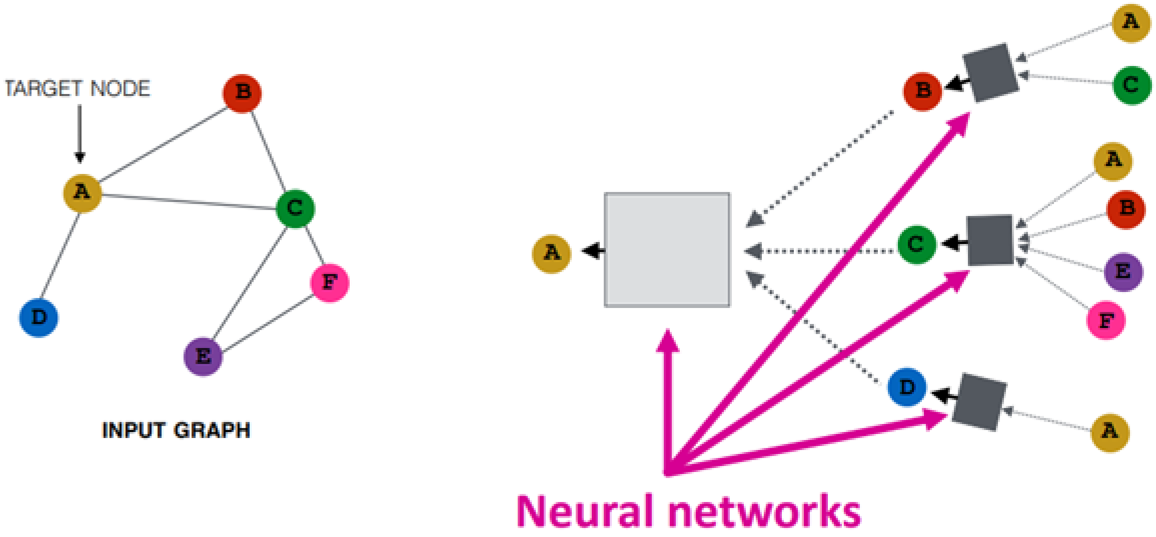
Aggregate Neighbors

- **Key idea:** Generate node embeddings based on **local network neighborhoods**



Shared Params of Neural Networks

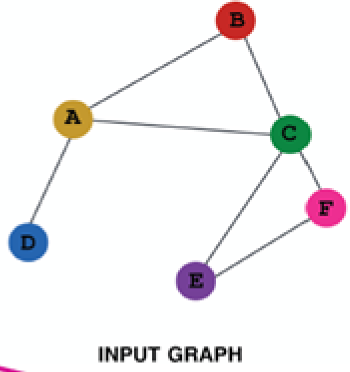
- **Intuition:** Nodes aggregate information from their neighbors using neural networks



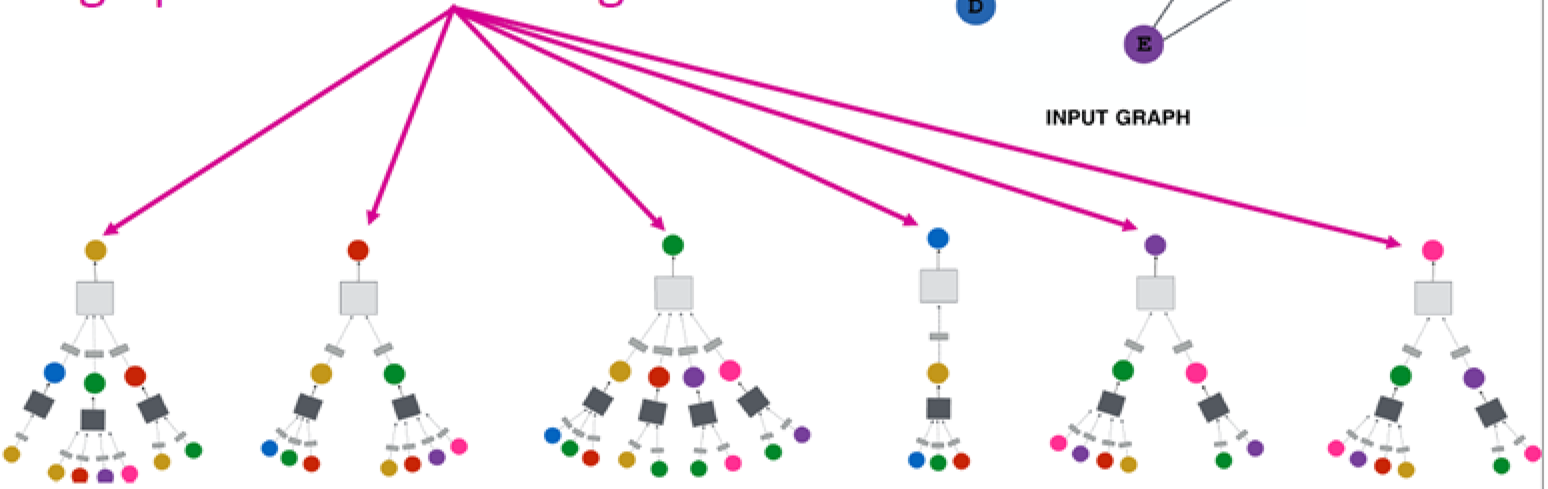
Order invariant

Aggregate Neighbors

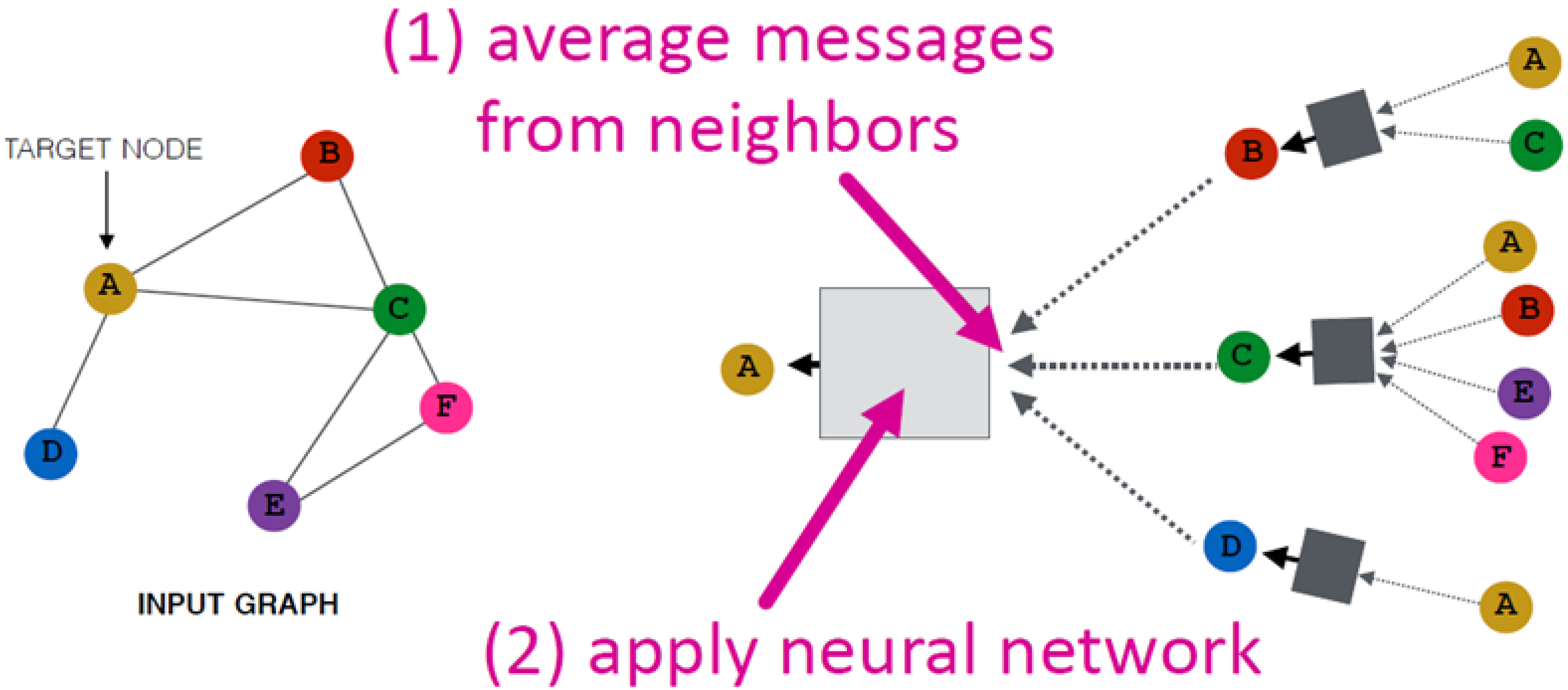
- **Intuition:** Network neighborhood defines a computation graph



Every node defines a computation graph based on its neighborhood!



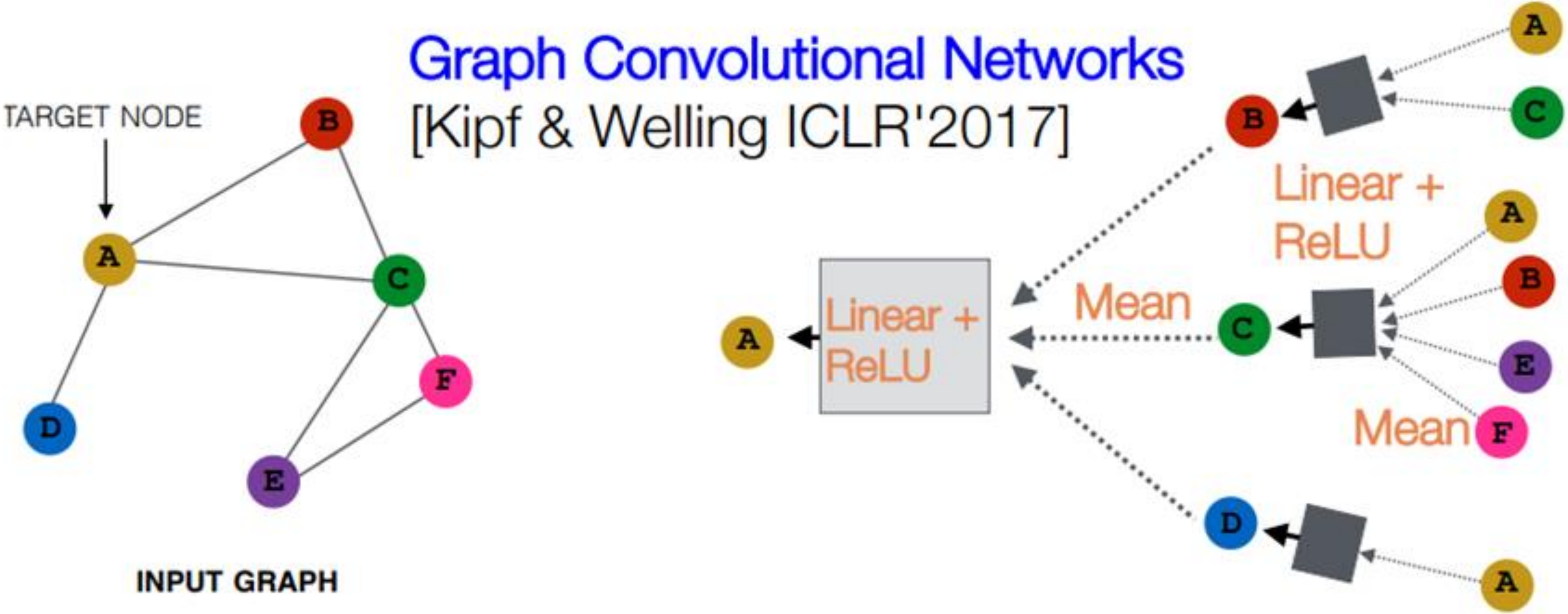
- **Basic approach:** Average information from neighbors and apply a neural network



- Many model variants have been proposed with difference choice of neural networks.

Scarselli et al., 2009b; Battaglia et al., 2016; Defferrard et al., 2016; Duvenaud et al., 2015; Hamilton et al., 2017a; Kearnes et al., 2016; Kipf & Welling, 2017; Lei et al., 2017; Li et al., 2016; Velickovic et al., 2018; Verma & Zhang, 2018; Ying et al., 2018; Zhang et al., 2018

Graph Convolutional Networks
[Kipf & Welling ICLR'2017]



- **Basic approach:** Average neighbor messages and apply a neural network

Initial 0-th layer embeddings are equal to node features

$$\boxed{\mathbf{h}_v^0 = \mathbf{x}_v}$$

Previous layer embedding of *v*

$$\mathbf{h}_v^k = \boxed{\sigma}\left(\mathbf{W}_k \boxed{\sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}} + \mathbf{B}_k \boxed{\mathbf{h}_v^{k-1}}\right), \quad \forall k \in \{1, \ldots, K\}$$

Average of neighbor's previous layer embeddings

$$\boxed{\mathbf{z}_v} = \mathbf{h}_v^K$$

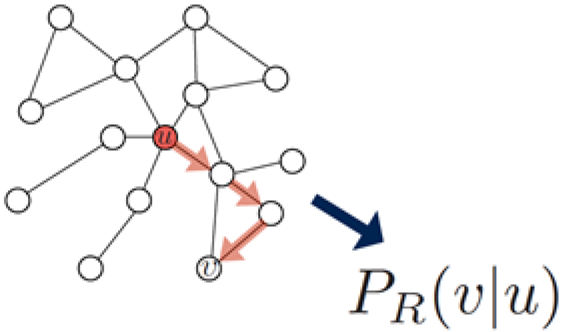
Embedding after K layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

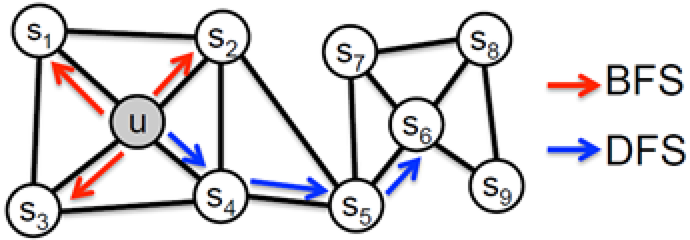
Unsupervised Training

Similar nodes should have similar embeddings

Can define a loss function based on results from node2vec, deepwalk, struc2vec



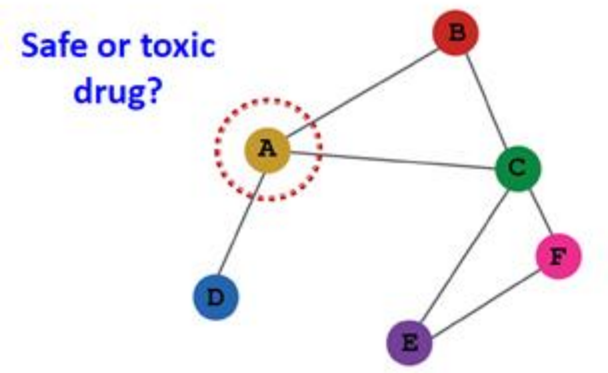
Random Walk



Node2vec: biased random walk

Supervised Training

Node Classification



E.g., a drug-drug interaction network

Directly train the model for a supervised task (e.g., **node classification**)

$$\mathcal{L} = \sum_{v \in V} \boxed{y_v} \log(\sigma(\boxed{\mathbf{z}_v^\top} \boxed{\boldsymbol{\theta}})) + (1 - \boxed{y_v}) \log(1 - \sigma(\boxed{\mathbf{z}_v^\top} \boxed{\boldsymbol{\theta}}))$$

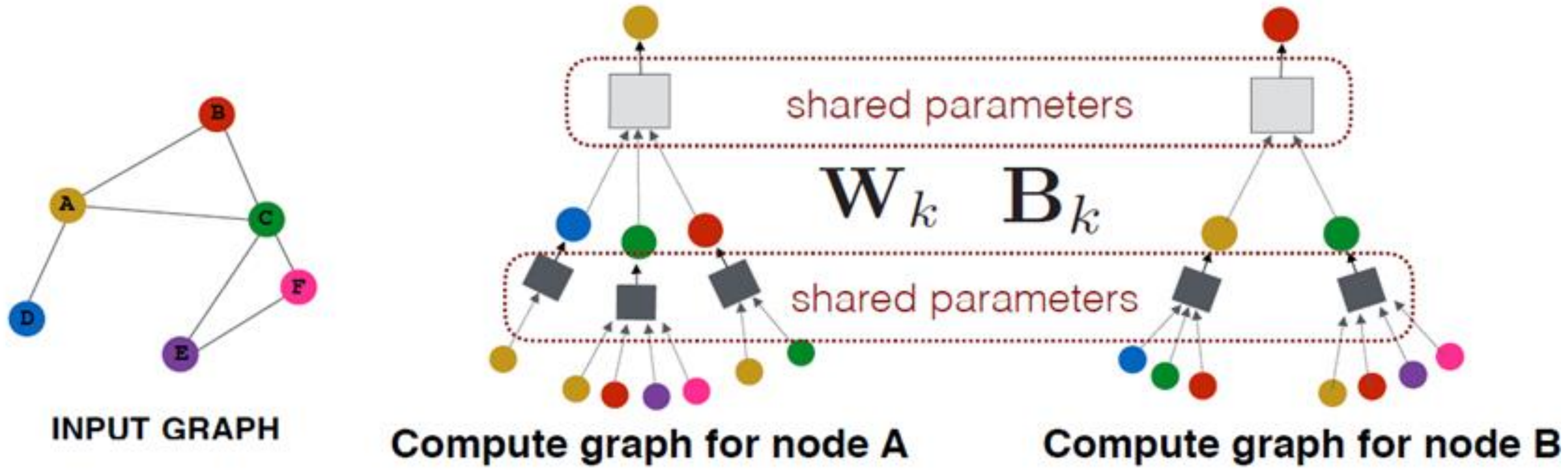
Encoder output: node embedding

Classification weights

Node class label

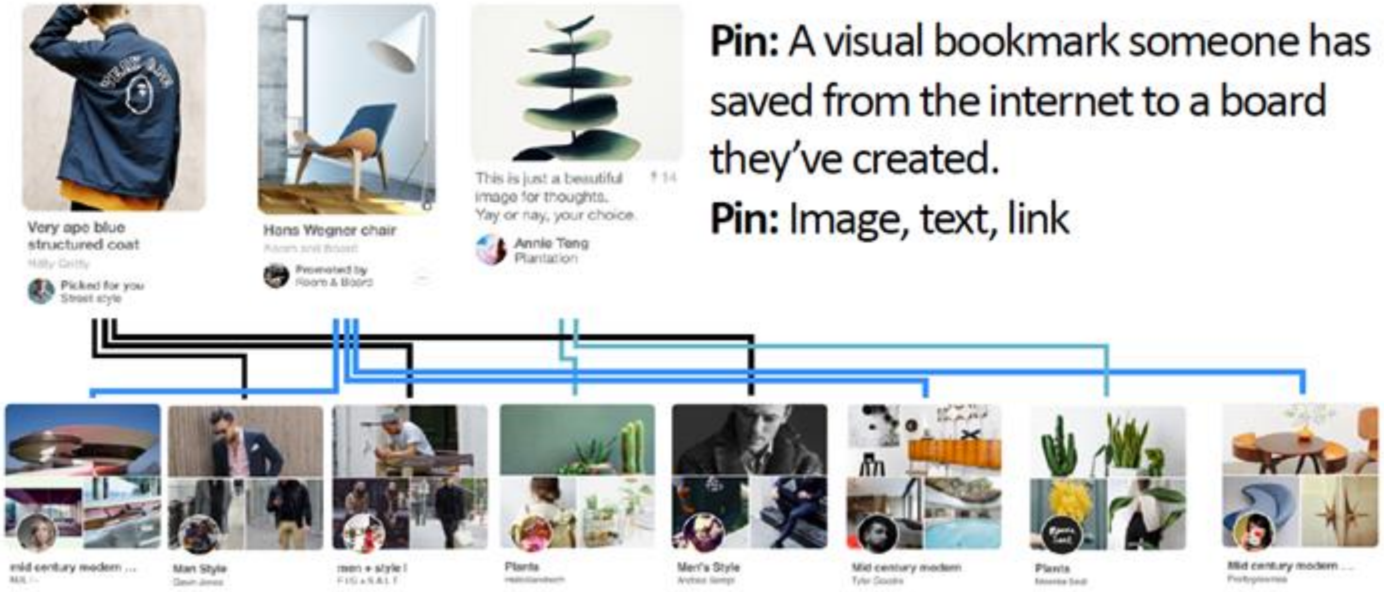
Inductive Capability

- The same aggregation parameters are shared for all nodes:
  - The number of model parameters is sublinear in $|V|$ and we can **generalize to unseen nodes**!



Application: Pinterest

Human curated collection of pins



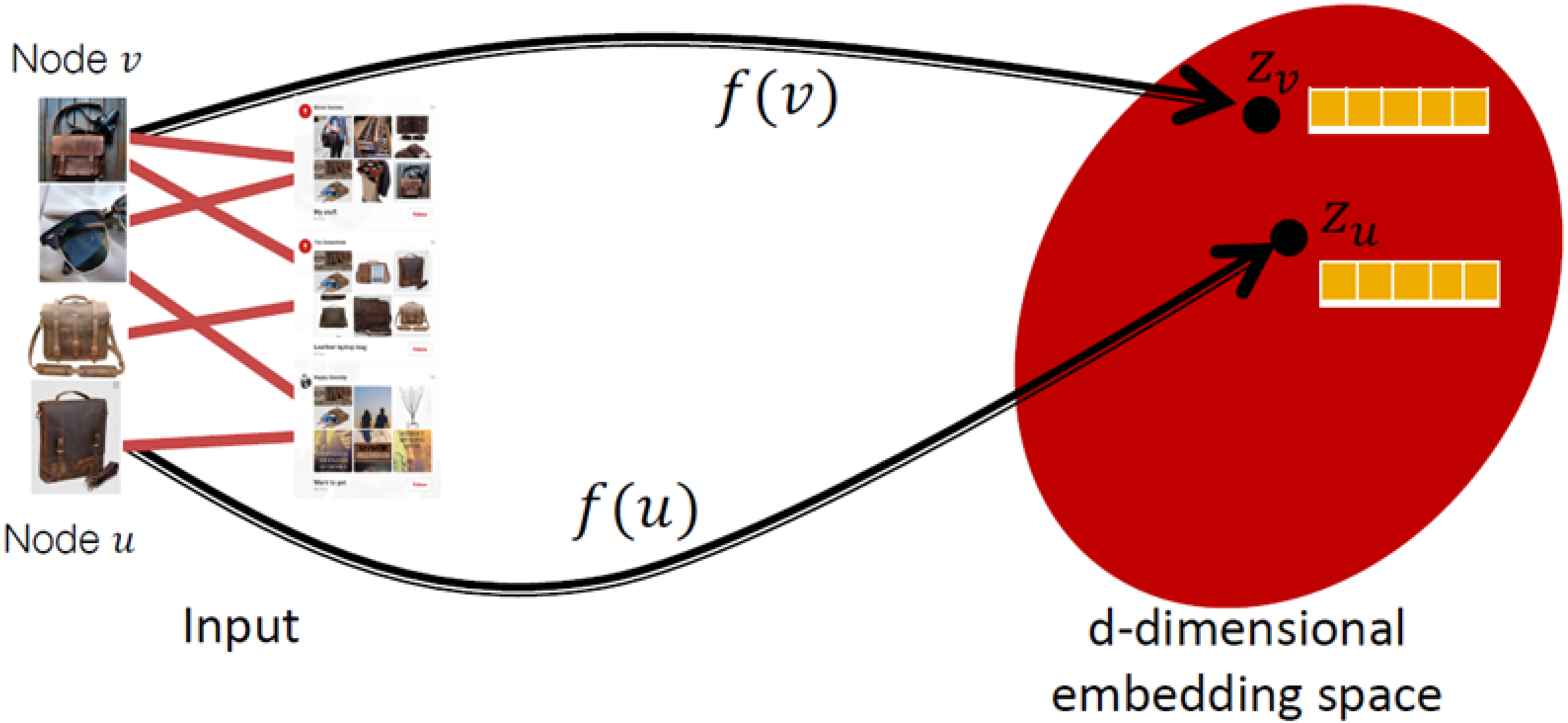
Pin: A visual bookmark someone has saved from the internet to a board they've created.

Pin: Image, text, link

Board: A collection of ideas (pins having something in common)

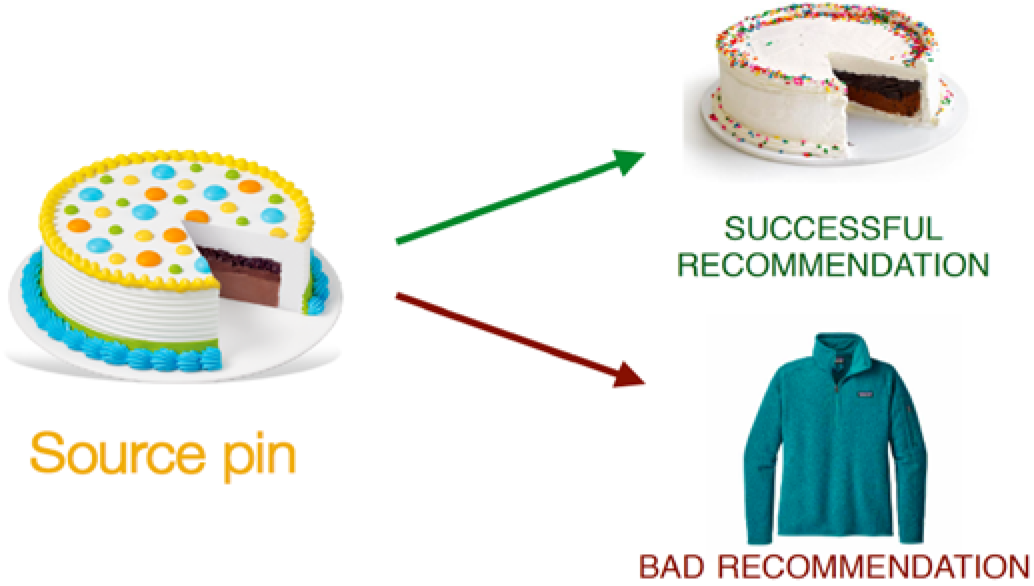
- **PinSage** graph convolutional network:
  - **Goal:** Generate embeddings for nodes (e.g., Pins/images) in a web-scale Pinterest graph containing billions of objects
  - **Key Idea:** Borrow information from nearby nodes
    - E.g., bed rail Pin might look like a garden fence, but gates and beds are rarely adjacent in the graph


    - Pin embeddings are essential to various tasks like recommendation of Pins, classification, clustering, ranking
      - Services like "Related Pins", "Search", "Shopping", "Ads"



Goal: Map nodes to d-dimensional embeddings such that nodes that are related are embedded close together

Task: Recommend related pins to users



Task: Learn node embeddings z_i such that $d(z_{cake1}, z_{cake2}) < d(z_{cake1}, z_{sweater})$

- **Challenges:**
  - **Massive size:** **3** billion nodes, 20 billion edges
  - **Heterogeneous data:** Rich image and text features

Goal: **Identify target pin among 3B pins**

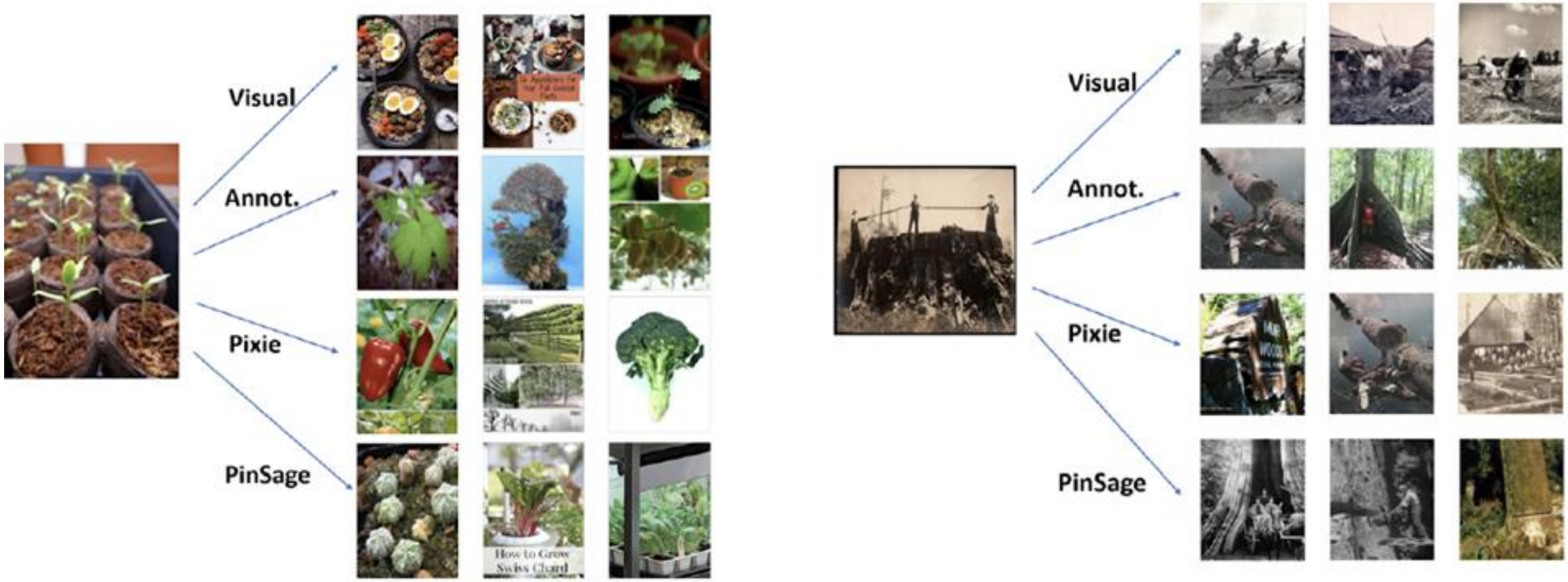
- **Issue: Need to learn with resolution of 100 vs. 3B**
- **Idea:** Use harder and harder negative samples
- Include more and more hard negative samples for each epoch

Source pin

Positive

Easy negative

Hard negative



Pixie: Random Walk (Dksombatchai et al., 2018)

Further Reading

PyTorch Geometric

Tutorials and overviews:
- Relational inductive biases and graph networks (Battaglia et al., 2018)
- Representation learning on graphs: Methods and applications (Hamilton et al., 2017)

Attention-based neighborhood aggregation:
- Graph attention networks (Hoshen, 2017; Velickovic et al., 2018; Liu et al., 2018)

Embedding entire graphs:
- Graph neural nets with edge embeddings (Battaglia et al., 2016; Gilmer et. al., 2017)
- Embedding entire graphs (Duvenaud et al., 2015; Dai et al., 2016; Li et al., 2018) and graph pooling (Ying et al., 2018, Zhang et al., 2018)
- Graph generation and relational inference (You et al., 2018; Kipf et al., 2018)
- How powerful are graph neural networks(Xu et al., 2017)

Embedding nodes:
- Varying neighborhood: Jumping knowledge networks (Xu et al., 2018), GeniePath (Liu et al., 2018)
- Position-aware GNN (You et al. 2019)

Spectral approaches to graph neural networks:
- Spectral graph CNN & ChebNet (Bruna et al., 2015; Defferrard et al., 2016)
- Geometric deep learning (Bronstein et al., 2017; Monti et al., 2017)

Other GNN techniques:
- Pre-training Graph Neural Networks (Hu et al., 2019)
- GNNExplainer: Generating Explanations for Graph Neural Networks (Ying et al., 2019)