

Graph Neural Networks

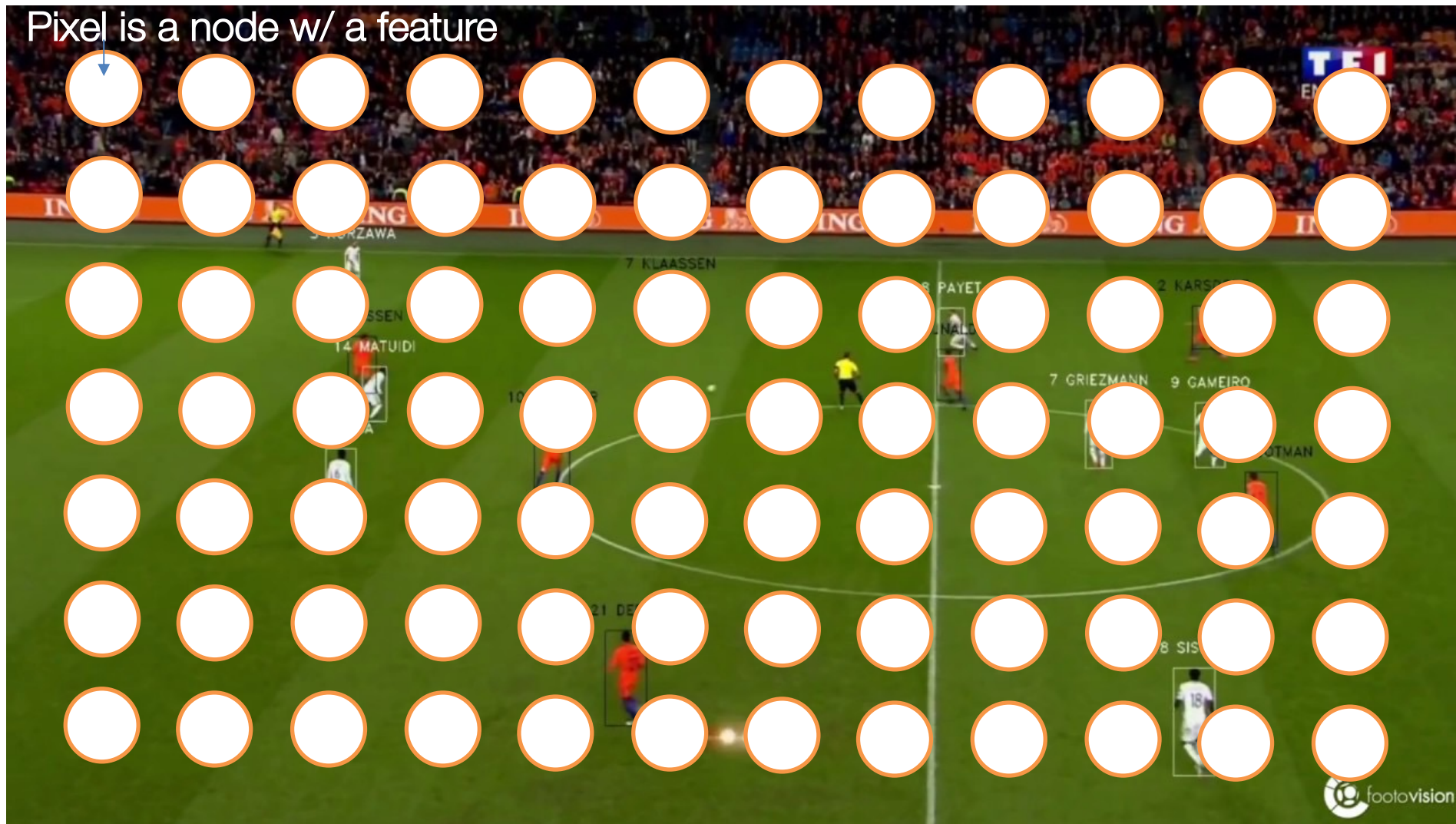
Based on material from:

- Hamilton et al. 2017. [Representation Learning on Graphs: Methods and Applications](#). *IEEE Data Engineering Bulletin on Graph Systems*.
- Scarselli et al. 2005. [The Graph Neural Network Model](#). *IEEE Transactions on Neural Networks*.
- <http://snap.stanford.edu/proj/embeddings-www>

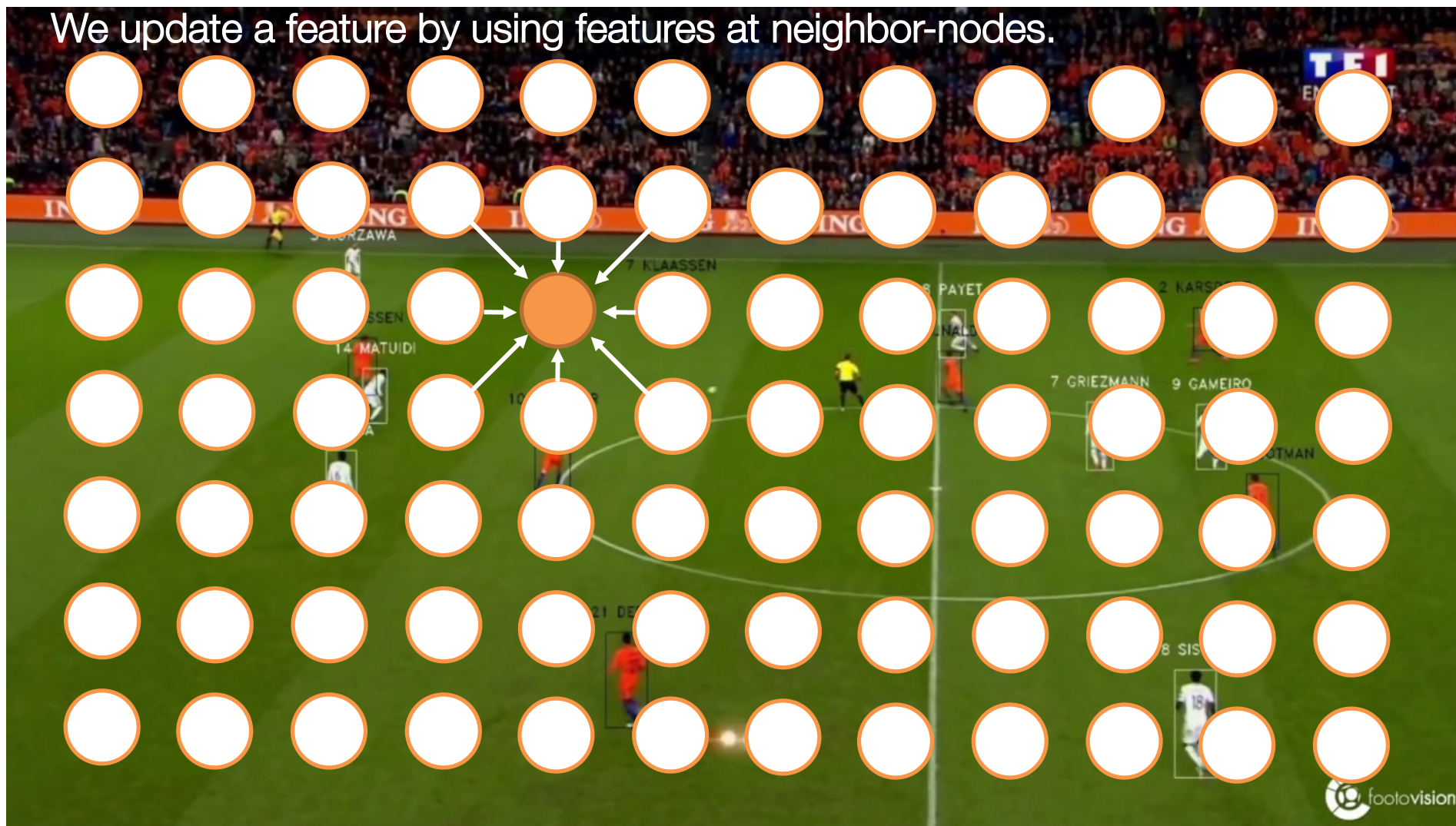
Object Detection by ConvNet



Pixel is a node w/ a feature



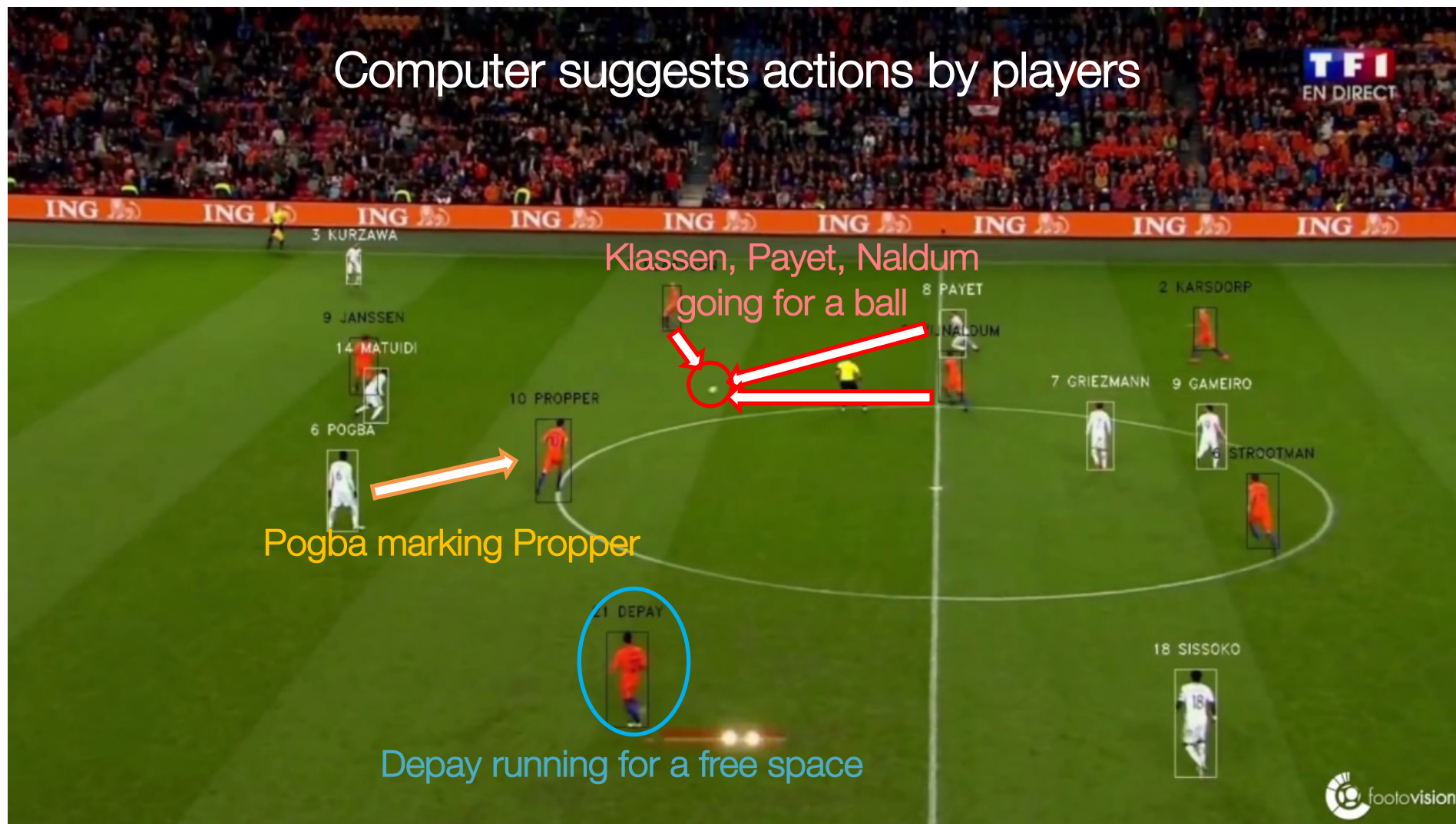
We update a feature by using features at neighbor-nodes.

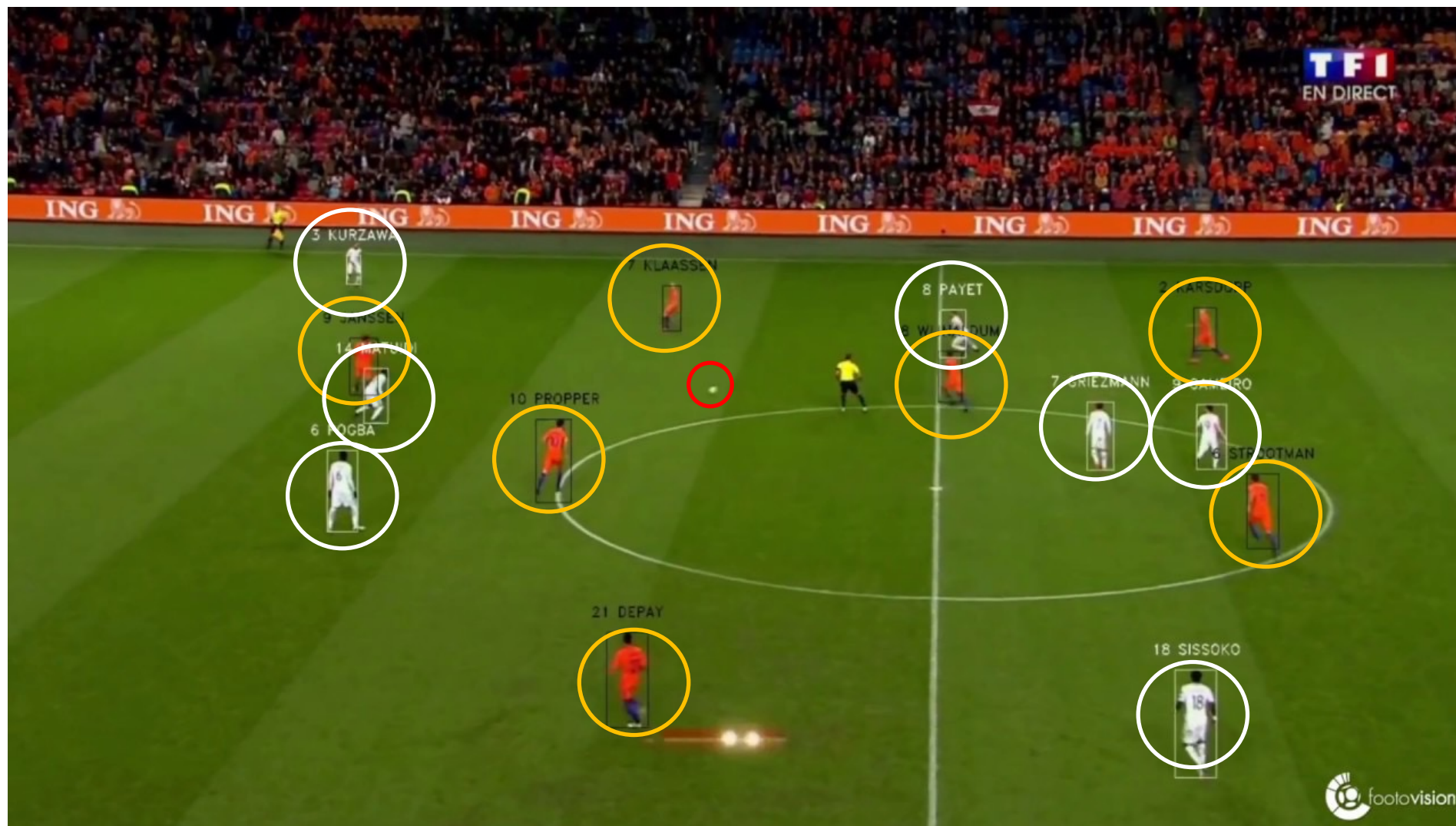


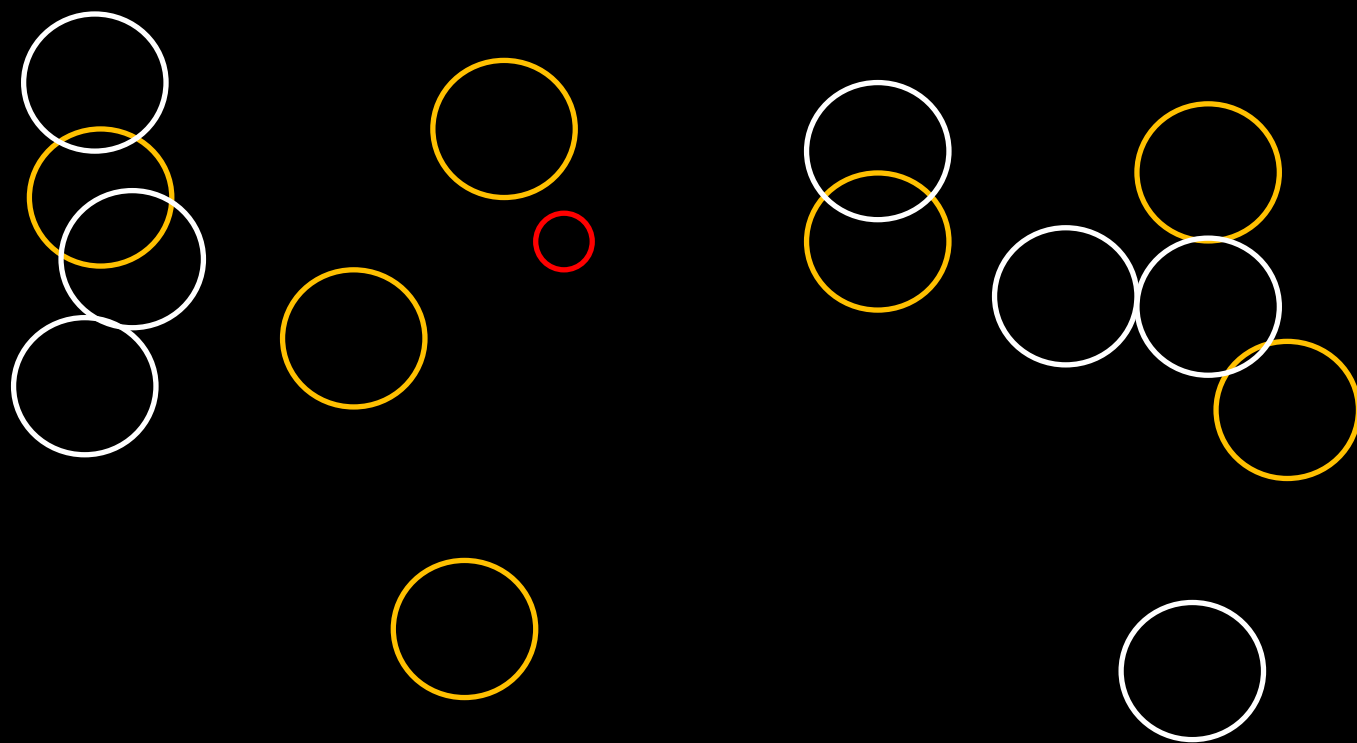
Object Detection by ConvNet

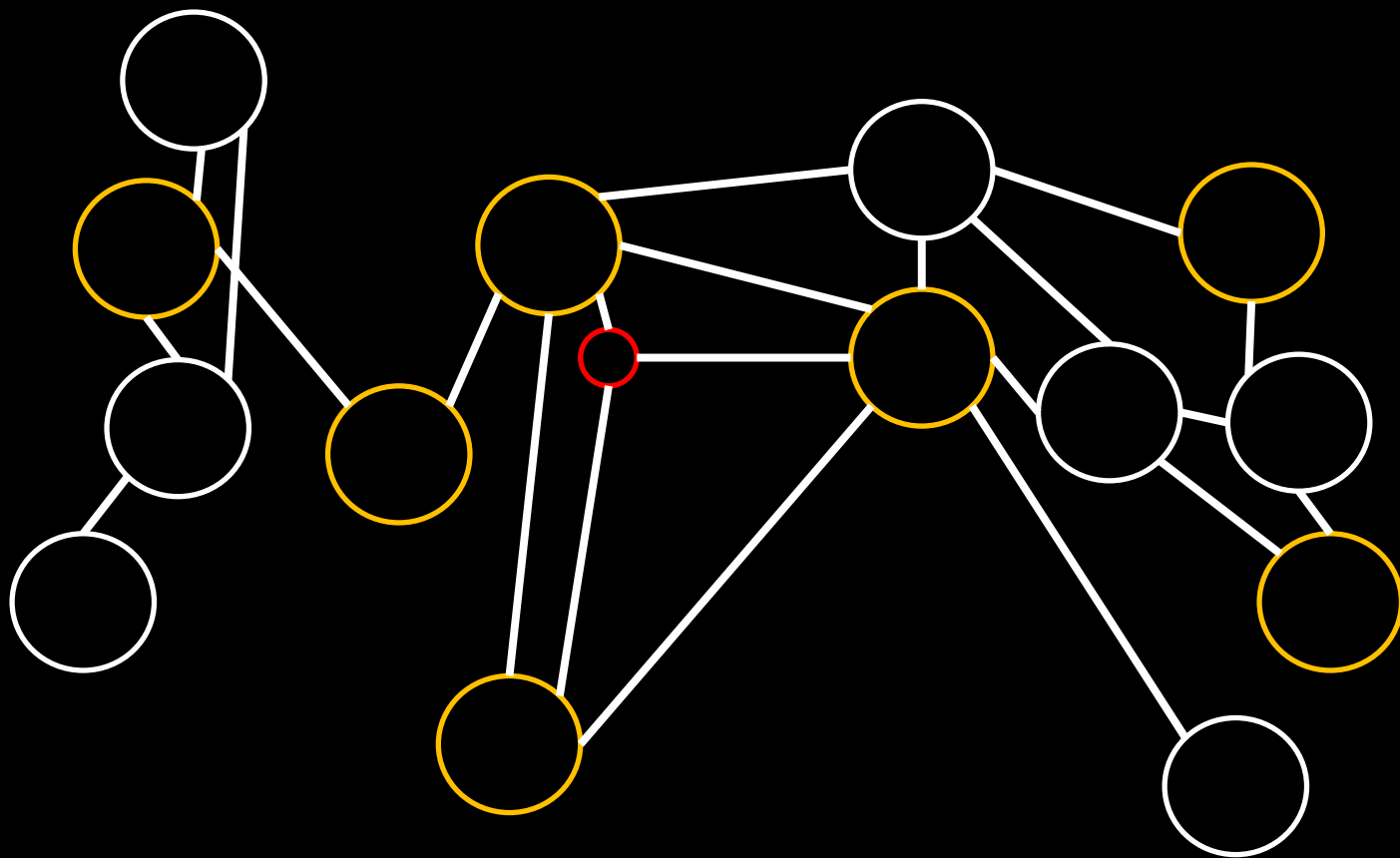


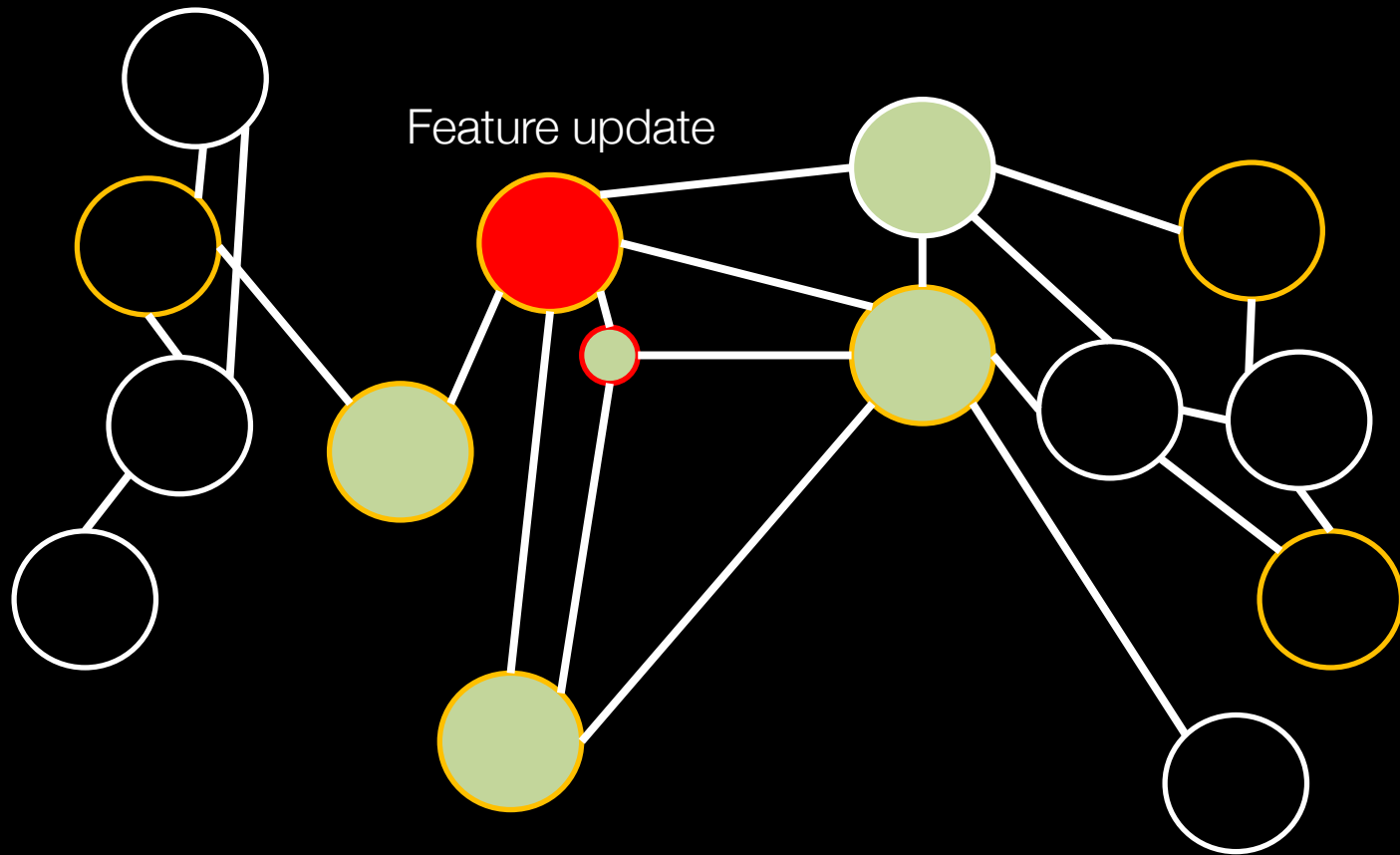
Computer suggests actions by players



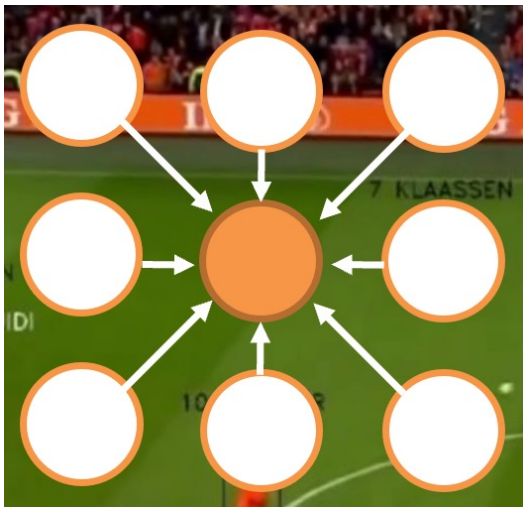






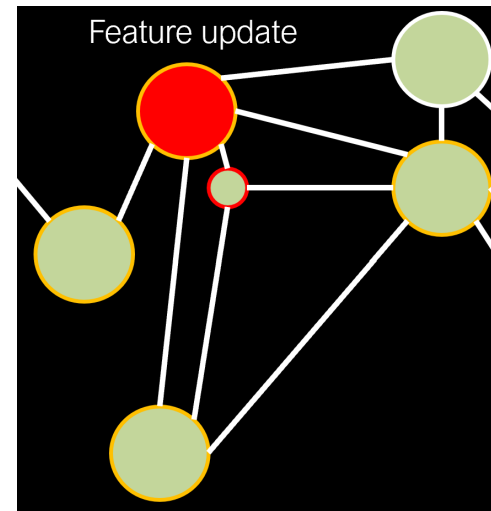


ConvNet (CNN)



Fixed input dimension (3x3)

GraphNet (GNN)



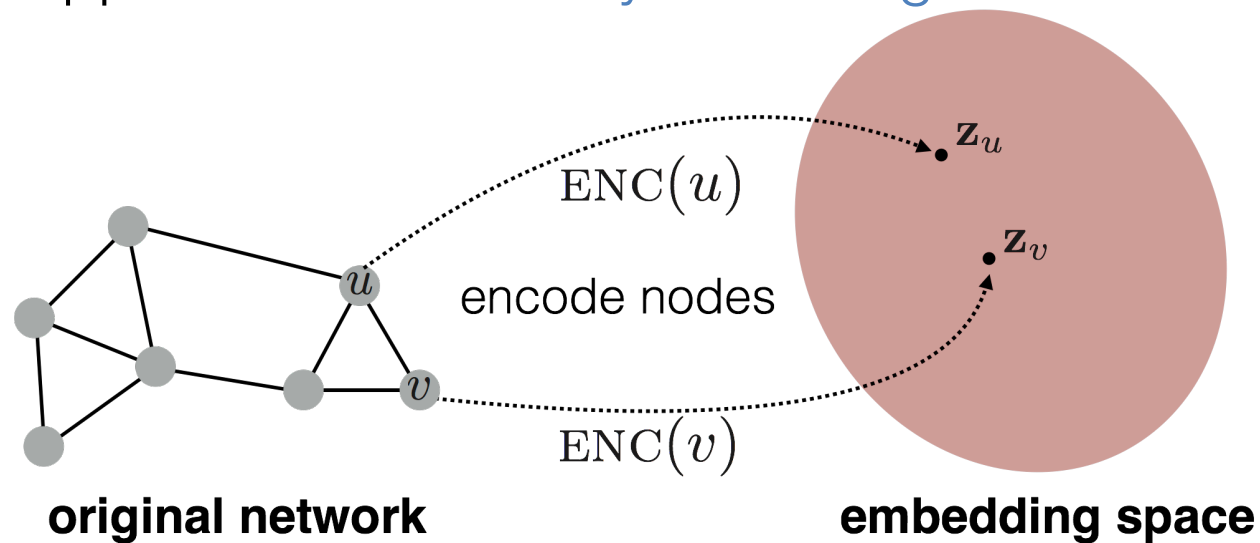
Arbitrary input dimension

Applications of GNN in CV

- Scene understanding (scene graphs)
- Human pose estimation
- Medical image analysis (cell relationship)
- Multi object tracking (space-time connections)
- Point cloud analysis (Lidar, SfM)
- ...

Embedding Nodes

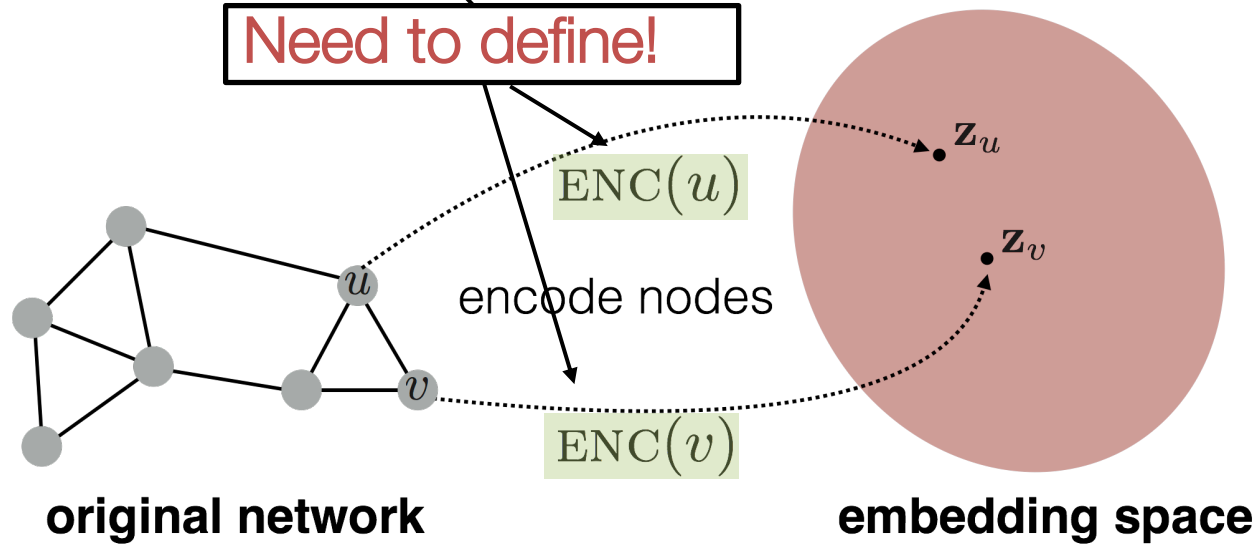
- Goal is to encode nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the original network.



Embedding Nodes

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

Need to define!



Outline for this Section

- We will now discuss “deep” methods based on **graph neural networks**.
 1. The Basics
 2. Graph Convolutional Networks (GCNs)
 3. GraphSAGE
 4. Gated Graph Neural Networks
 - ~~5. Subgraph Embeddings~~

The Basics: Graph Neural Networks

Based on material from:

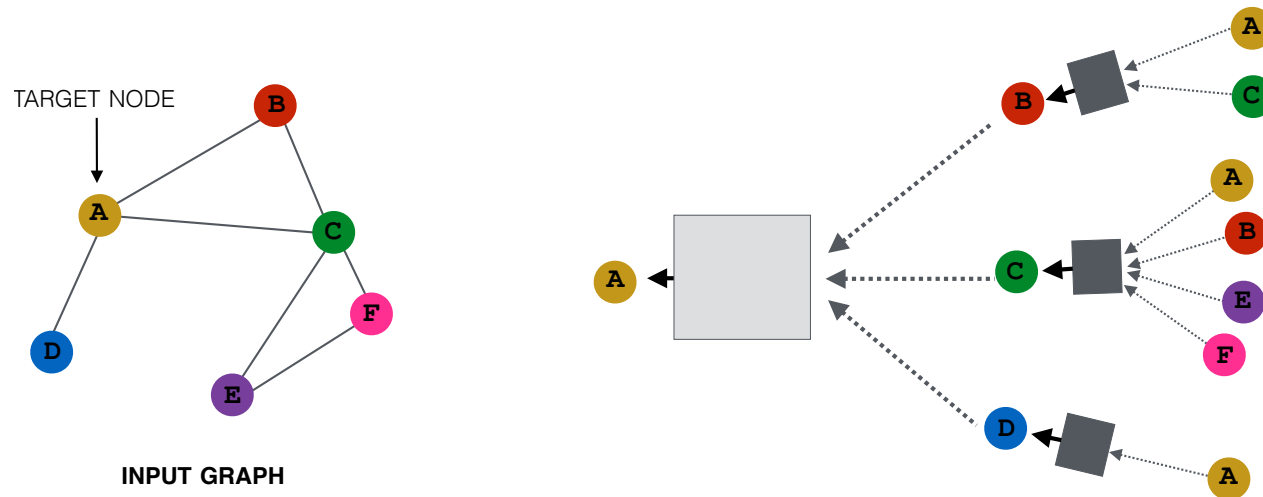
- Hamilton et al. 2017. [Representation Learning on Graphs: Methods and Applications](#). *IEEE Data Engineering Bulletin on Graph Systems*.
- Scarselli et al. 2005. [The Graph Neural Network Model](#). *IEEE Transactions on Neural Networks*.

Setup

- Assume we have a graph G :
 - V is the vertex set.
 - A is the adjacency matrix (assume binary).
 - $X \in \mathbb{R}^{m \times |V|}$ is a matrix of node features.
 - Categorical attributes, text, image data
 - E.g., profile information in a social network.
 - Node degrees, clustering coefficients, etc.
 - Indicator vectors (i.e., one-hot encoding of each node)

Neighborhood Aggregation

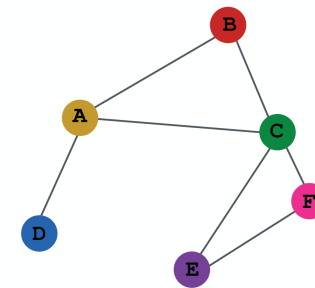
- **Intuition:** Nodes aggregate information from their neighbors using neural networks



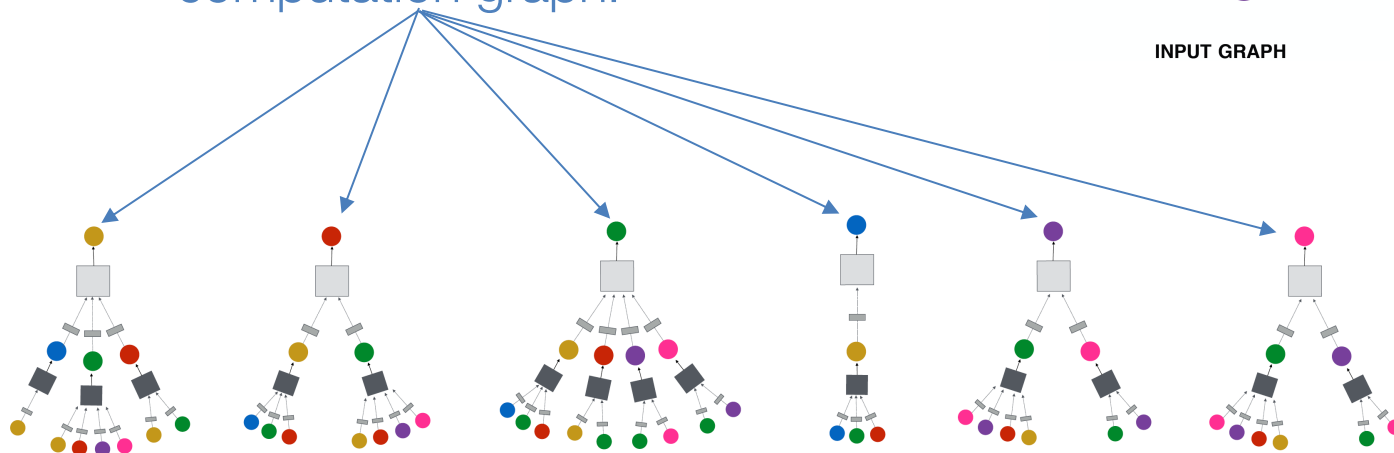
Neighborhood Aggregation

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

Every node defines a unique computation graph!

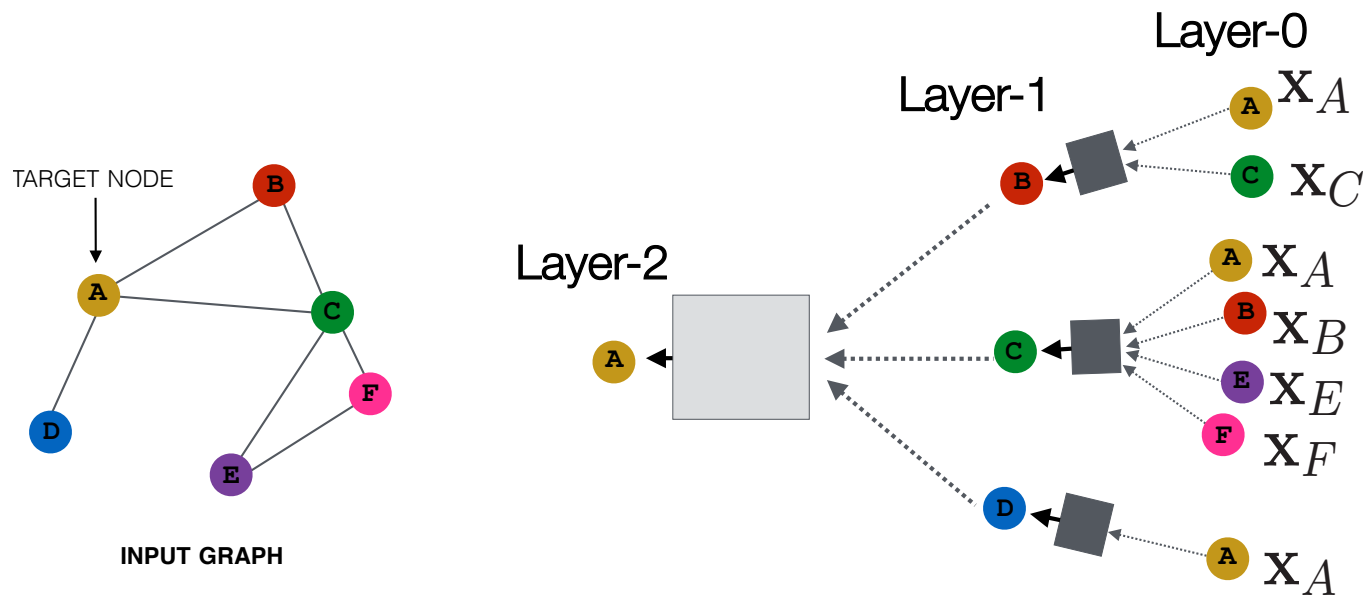


INPUT GRAPH



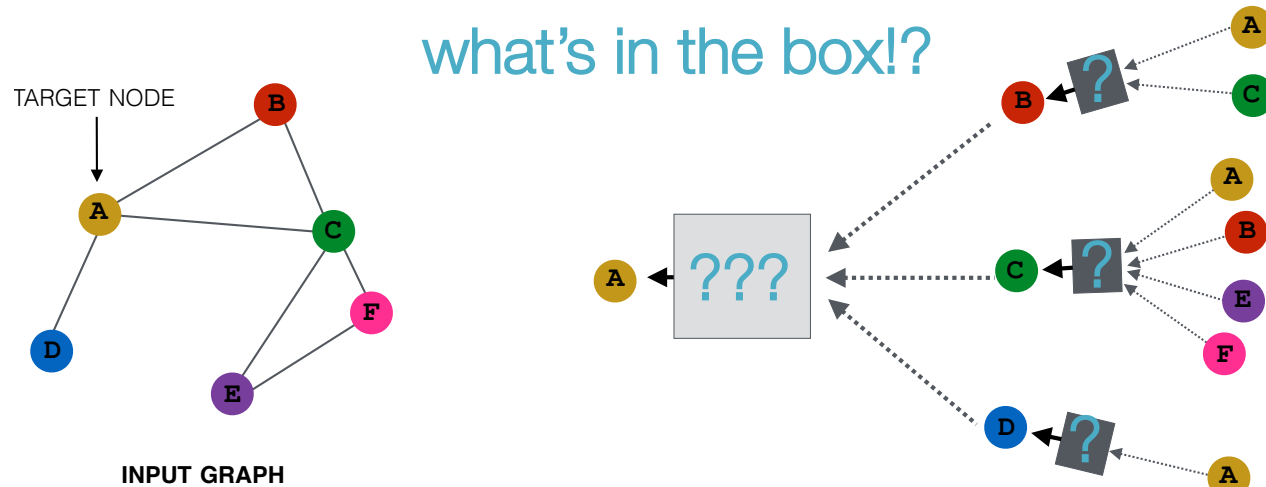
Neighborhood Aggregation

- Nodes have embeddings at each layer.
- Model can be arbitrary depth.
- “layer-0” embedding of node u is its input feature, i.e. x_u .



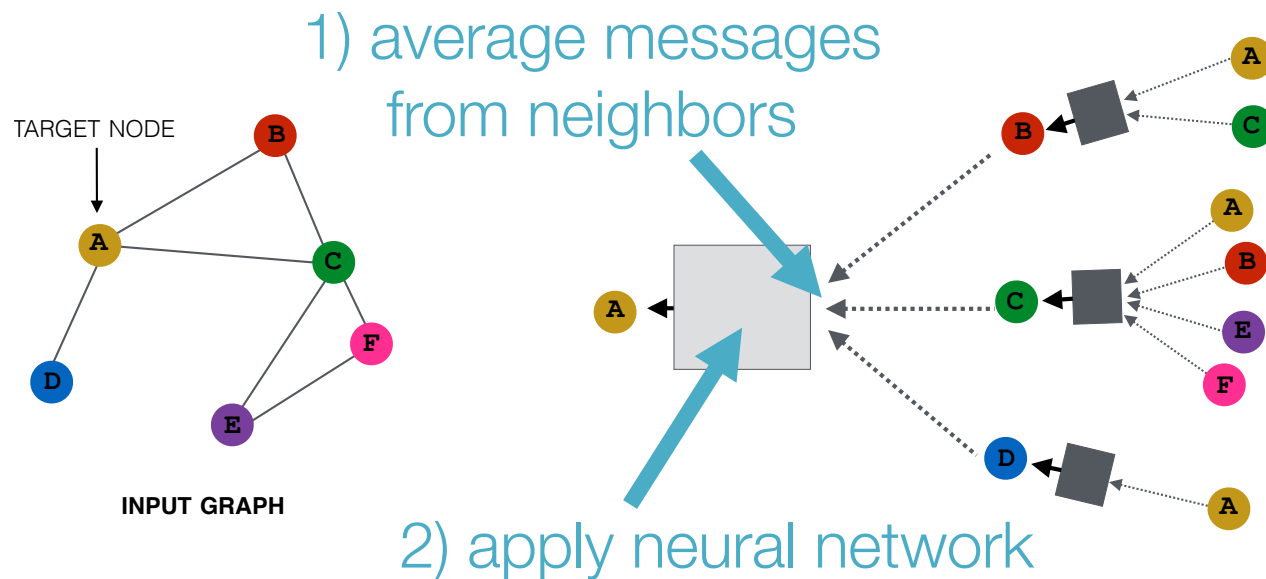
Neighborhood Aggregation

- Key distinctions are in how different approaches aggregate information across the layers.



Neighborhood Aggregation

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



The Math

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

Initial “layer 0” embeddings are equal to node features

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

previous layer embedding of v

$$\mathbf{h}_v^k = \sigma \left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1} \right), \quad \forall k > 0$$

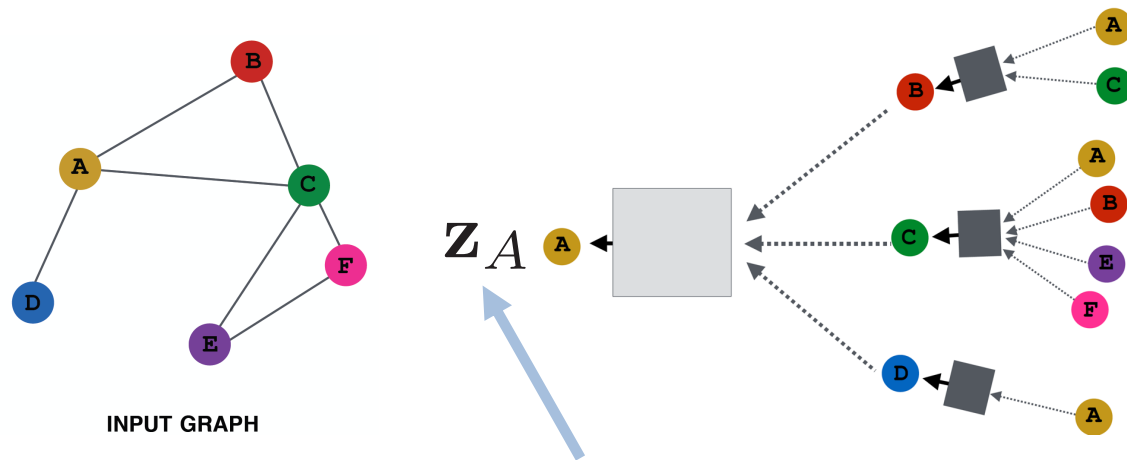
kth layer embedding of v

non-linearity (e.g., ReLU or tanh)

average of neighbor's previous layer embeddings

Training the Model

- How do we train the model to generate “high-quality” embeddings?



Need to define a loss function on the embeddings, $\mathcal{L}(z_u)$!

Training the Model

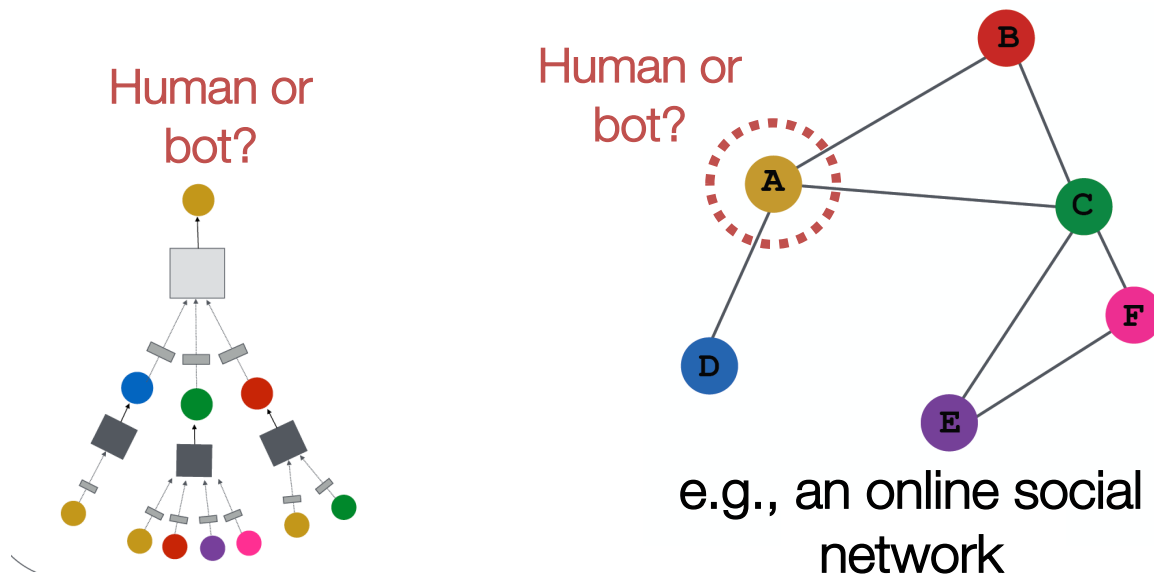
trainable matrices
(i.e., what we learn)

$$\mathbf{h}_v^0 = \mathbf{x}_v$$
$$\mathbf{h}_v^k = \sigma \left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1} \right), \quad \forall k \in \{1, \dots, K\}$$

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

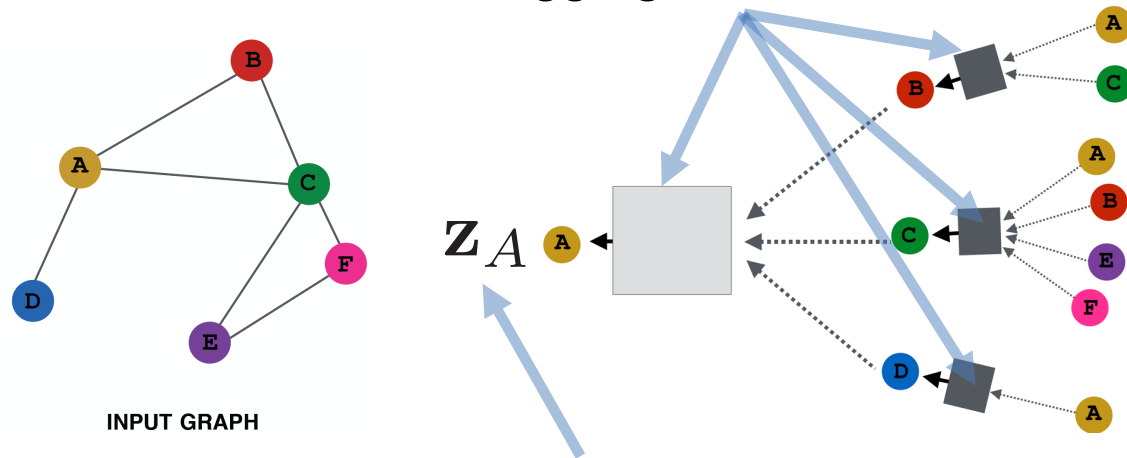
- After K-layers of neighborhood aggregation, we get output embeddings for each node.
- We can feed these embeddings into any loss function and run stochastic gradient descent to train the aggregation parameters.

Training the Model



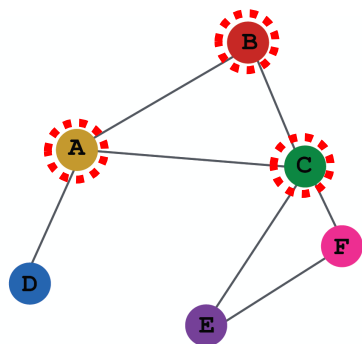
Overview of Model Design

1) Define a neighborhood aggregation function.



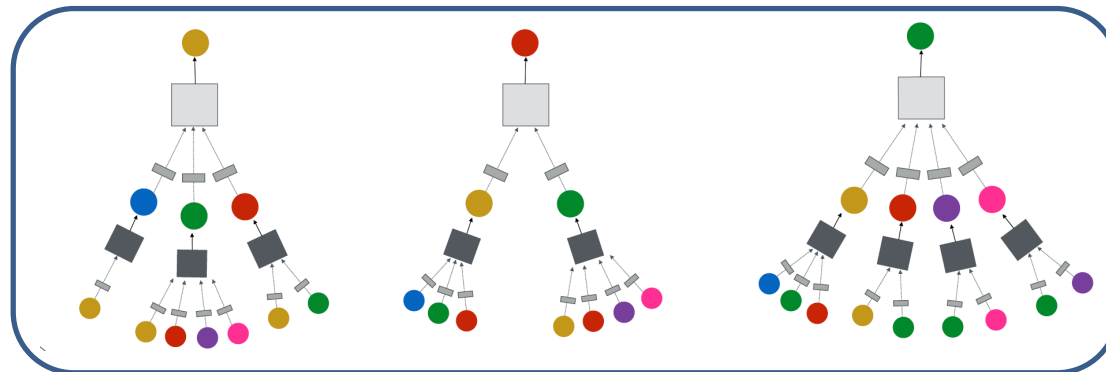
2) Define a loss function on the embeddings, $\mathcal{L}(z_u)$

Overview of Model Design



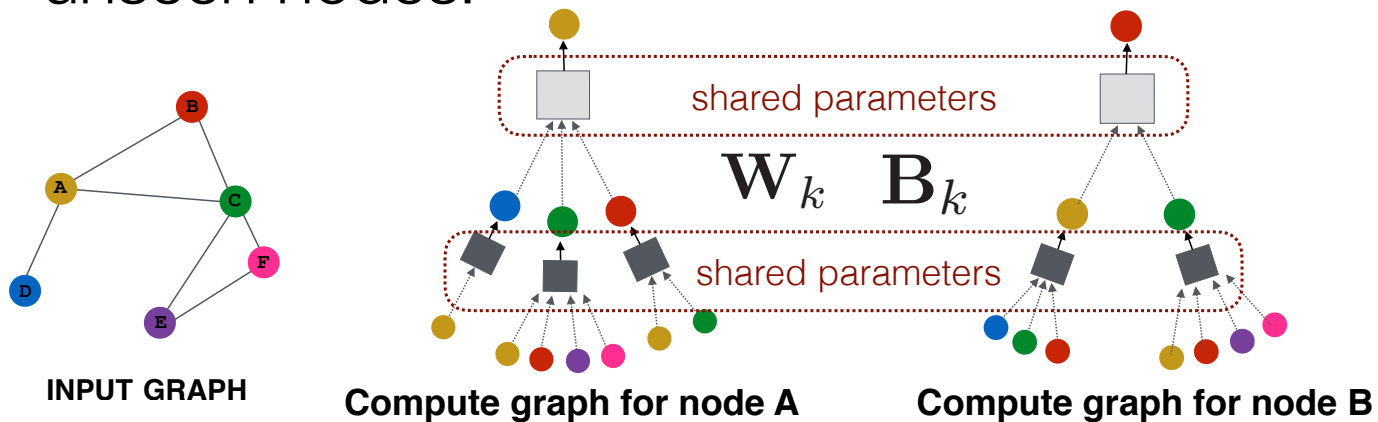
INPUT GRAPH

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

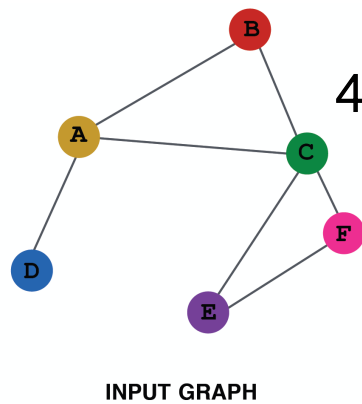


Inductive Capability

- 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!

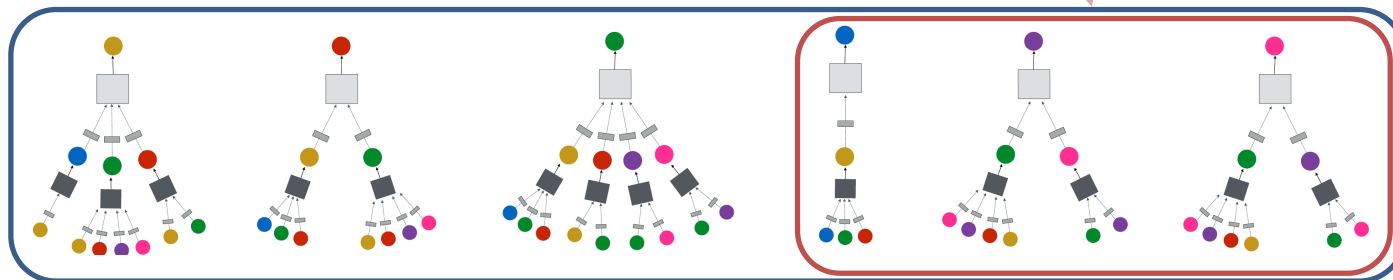


Overview of Model

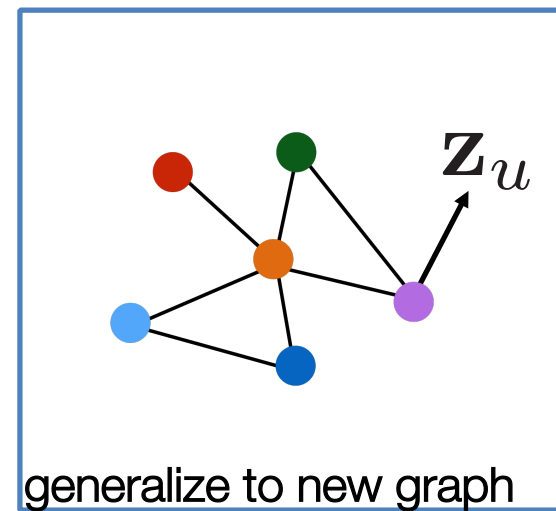
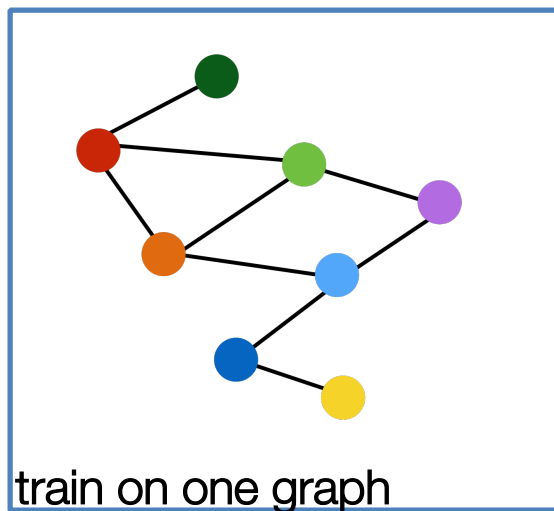


4) Generate embeddings for nodes as needed

Even for nodes we never trained on!!!!



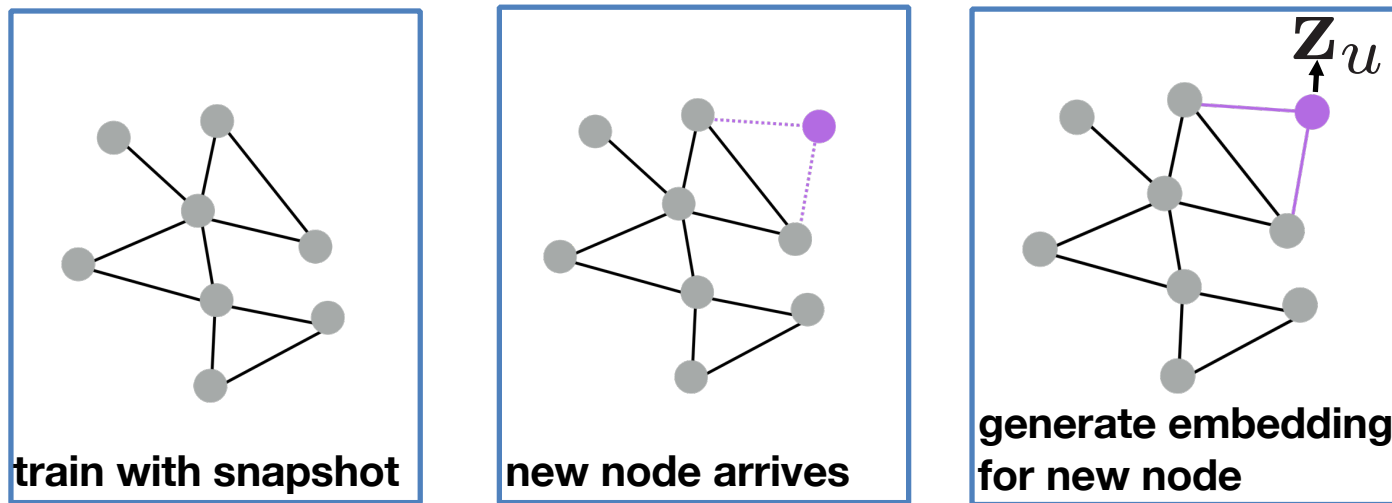
Inductive Capability



Inductive node embedding → generalize to entirely unseen graphs

e.g., train on protein interaction graph from model organism A and generate embeddings on newly collected data about organism B

Inductive Capability



Many application settings constantly encounter previously unseen nodes.
e.g., Reddit, YouTube, GoogleScholar,

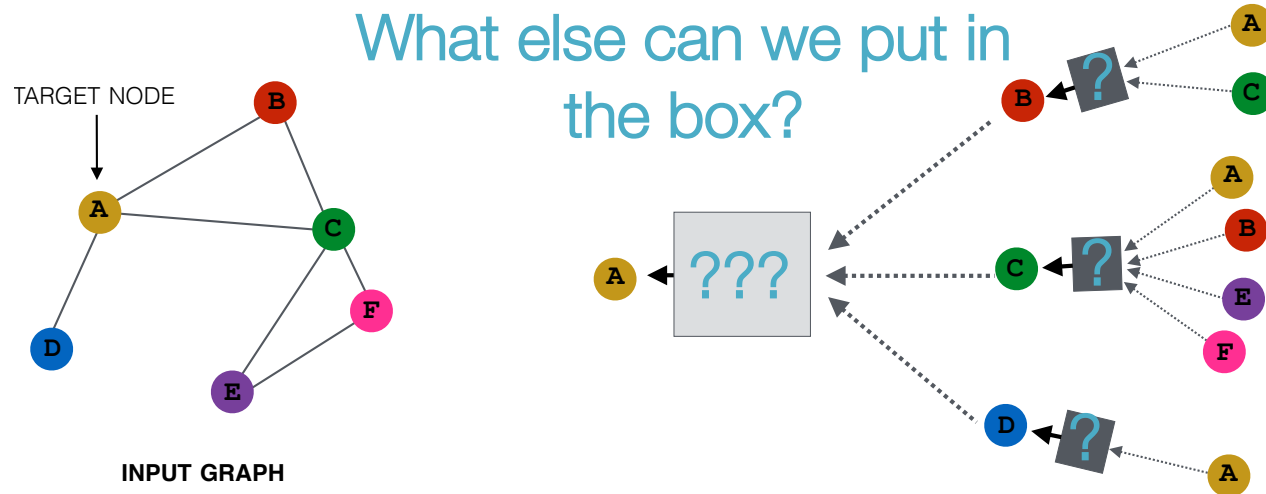
Need to generate new embeddings “on the fly”

Quick Recap

- **Recap:** Generate node embeddings by aggregating neighborhood information.
 - Allows for parameter sharing in the encoder.
 - Allows for inductive learning.
- We saw a **basic variant** of this idea...
now we will cover some state of the art variants from the literature.

Neighborhood Aggregation

- Key distinctions are in how different approaches aggregate messages



Outline for this Section

1. The Basics ✓
2. Graph Convolutional Networks
3. GraphSAGE
4. Gated Graph Neural Networks
- ~~5. Subgraph Embeddings~~



Graph Convolutional Networks

Based on material from:

- Kipf et al., 2017. [Semisupervised Classification with Graph Convolutional Networks](#). *ICLR*.

Graph Convolutional Networks

- [Kipf et al.'s Graph Convolutional Networks \(GCNs\)](#) are a slight variation on the neighborhood aggregation idea:

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

Graph Convolutional Networks

Basic Neighborhood Aggregation

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

VS.

GCN Neighborhood Aggregation

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

Graph Convolutional Networks

- Empirically, this gives the best results
 - More parameter sharing
 - Down-weights high degree neighbors

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

use the same transformation matrix for self and neighbor embeddings

instead of simple average, normalization varies across neighbors

Batch Implementation

- Can be efficiently implemented using sparse batch operations:


$$\mathbf{H}^{(k+1)} = \sigma \left(\mathbf{D}^{-\frac{1}{2}} \tilde{\mathbf{A}} \mathbf{D}^{-\frac{1}{2}} \mathbf{H}^{(k)} \mathbf{W}_k \right)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$

$$\mathbf{D}_{ii} = \sum_j \mathbf{A}_{i,j}$$

- $O(|E|)$ time complexity overall.

Outline for this Section

1. The Basics ✓
2. Graph Convolutional Networks ✓
3. GraphSAGE 
4. Gated Graph Neural Networks
- ~~5. Subgraph Embeddings~~

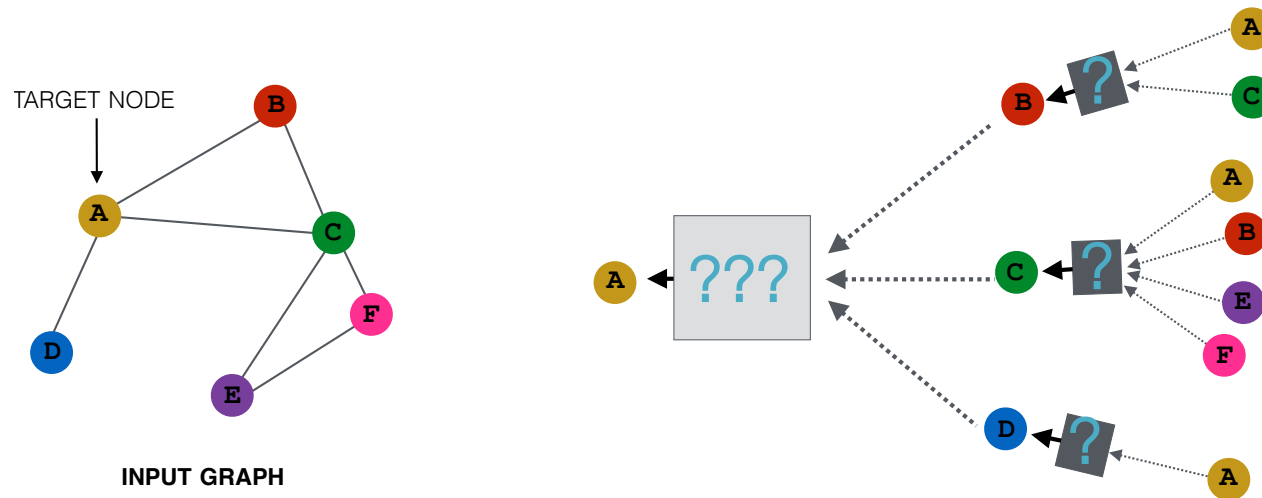
GraphSAGE

Based on material from:

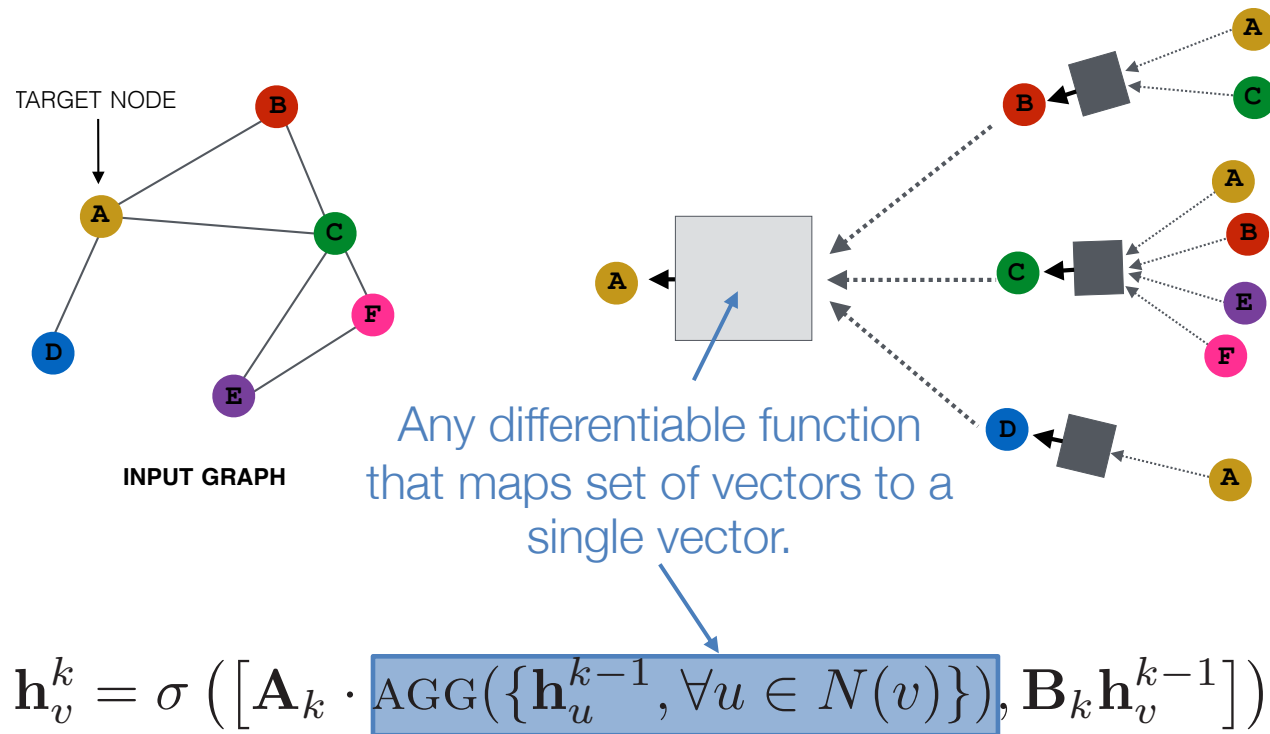
- Hamilton et al., 2017. [Inductive Representation Learning on Large Graphs](#). *NIPS*.

GraphSAGE Idea

- So far we have aggregated the neighbor messages by taking their (weighted) average, can we do better?



GraphSAGE Idea



Find the
mistake!!

GraphSAGE Differences

- Simple neighborhood aggregation:

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

- GraphSAGE:

concatenate self embedding and
neighbor embedding

$$\mathbf{h}_v^k = \sigma \left(\left[\mathbf{W}_k \cdot \text{AGG} \left(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\} \right), \mathbf{B}_k \mathbf{h}_v^{k-1} \right] \right)$$

generalized aggregation

GraphSAGE

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 $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$; non-linearity σ ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations \mathbf{z}_v for all $v \in \mathcal{V}$

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$  ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma \left( \mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k) \right)$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

GraphSAGE Variants

- **Mean:**
$$\text{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- **Max (w/ matrix mult):**

$$\text{AGG} = \text{mean}(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\})$$

$$\text{AGG} = \text{max}(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\})$$

- **LSTM:**

- Apply LSTM to random permutation of neighbors.


$$\text{AGG} = \text{LSTM}([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))])$$

GraphSAGE Variants

Table 1: Prediction results for the three datasets (micro-averaged F1 scores). Results for unsupervised and fully supervised GraphSAGE are shown. Analogous trends hold for macro-averaged scores.

Name	Citation		Reddit		PPI	
	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	0.908	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	0.907	0.954	0.482	0.612
GraphSAGE-pool	0.798	0.839	0.892	0.948	0.502	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

Outline for this Section

1. The Basics ✓
2. Graph Convolutional Networks ✓
3. GraphSAGE ✓
4. Gated Graph Neural Networks 
- ~~5. Subgraph Embeddings~~

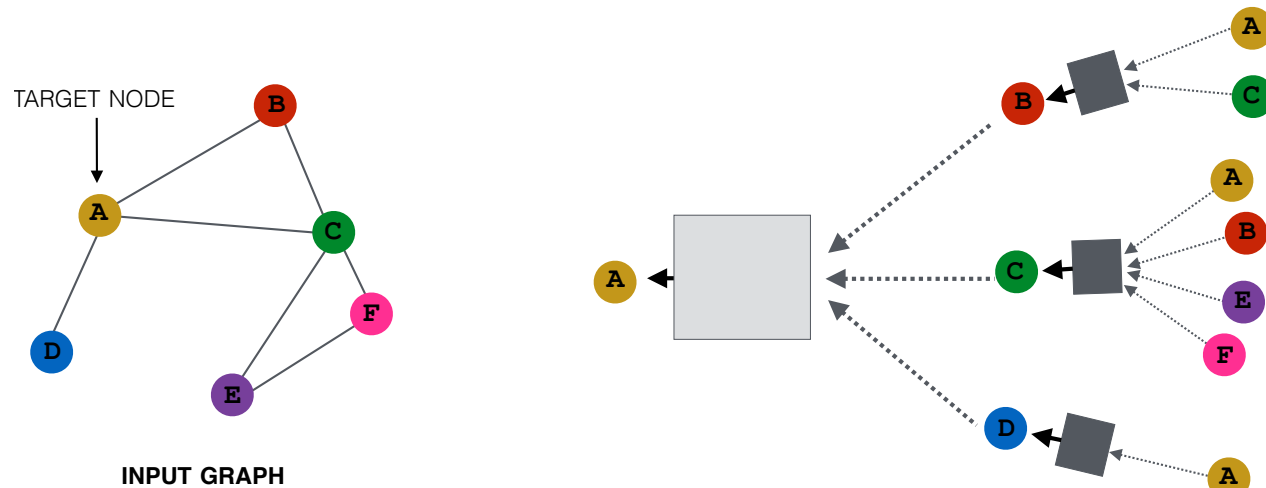
Gated Graph Neural Networks

Based on material from:

- Li et al., 2016. [Gated Graph Sequence Neural Networks](#). *ICLR*.

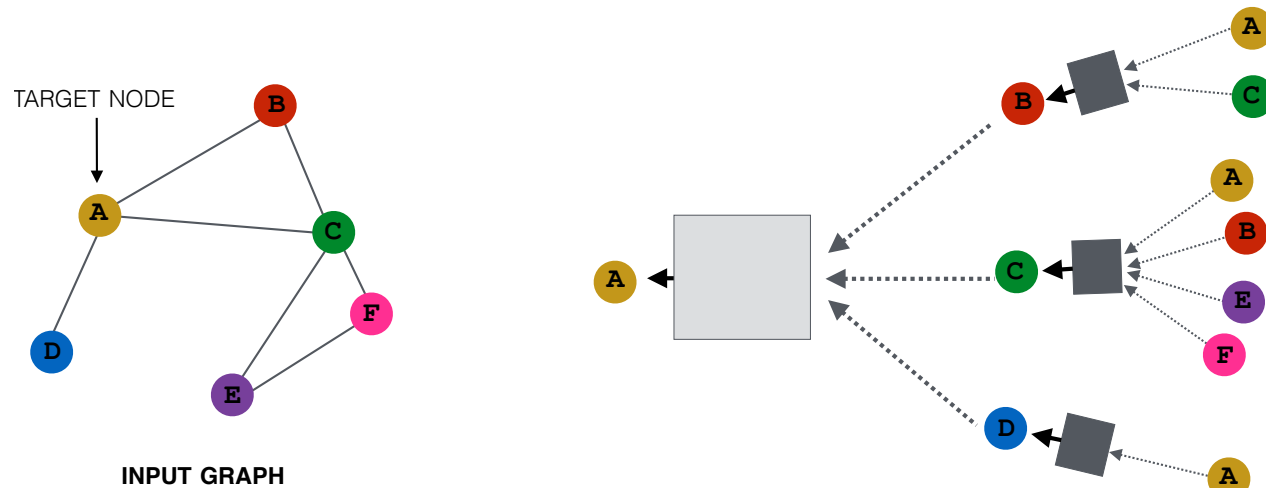
Neighborhood Aggregation

- **Basic idea:** Nodes aggregate “messages” from their neighbors using neural networks



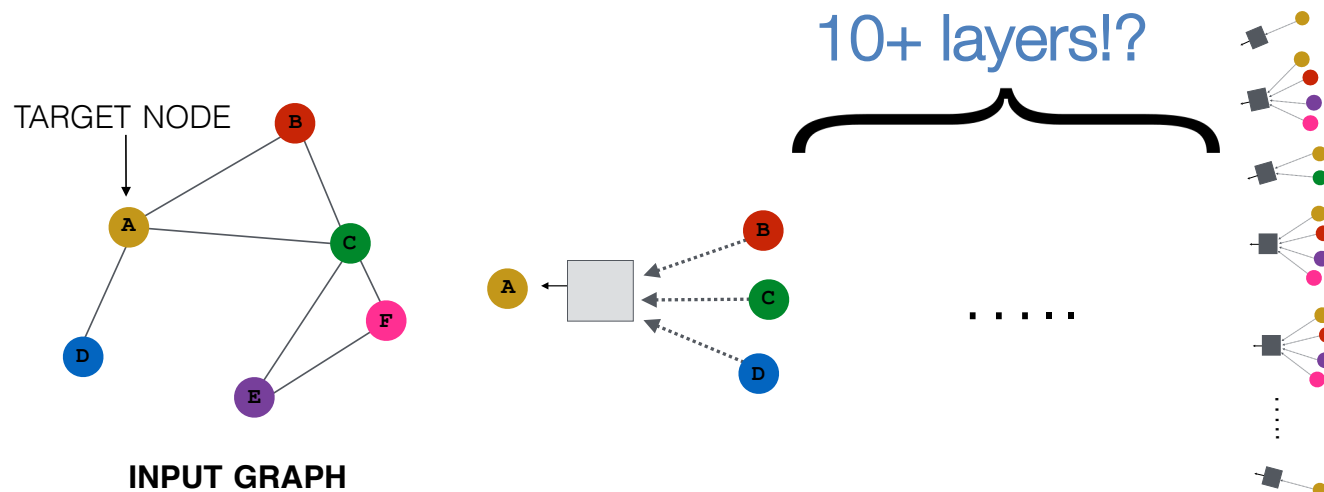
Neighborhood Aggregation

- GCNs and GraphSAGE generally only 2-3 layers deep.



Neighborhood Aggregation

- But what if we want to go deeper?



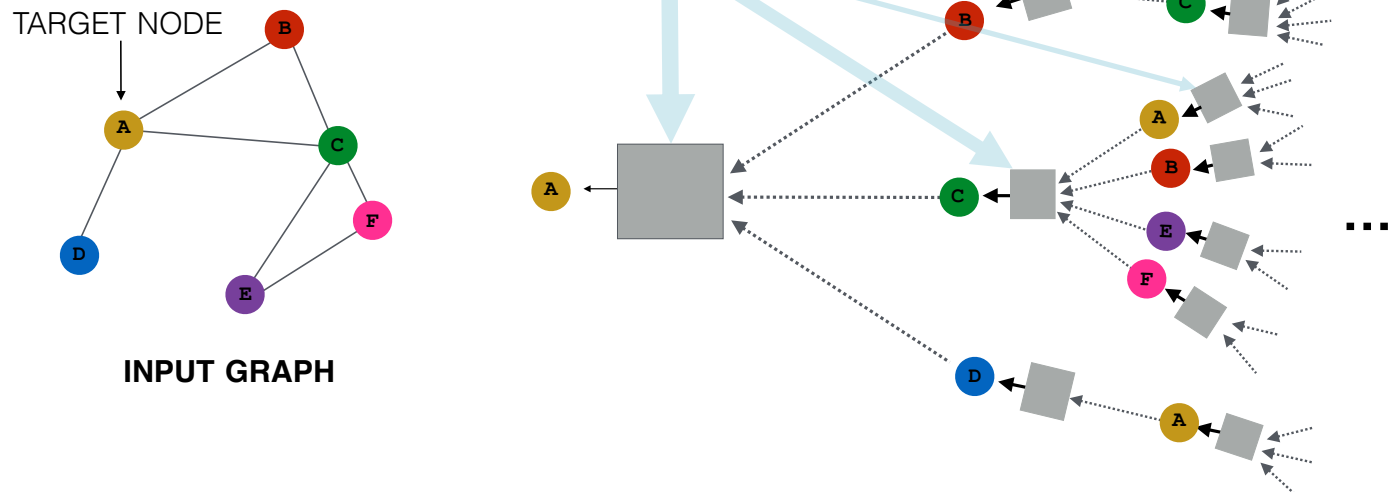
Gated Graph Neural Networks

- How can we build models with many layers of neighborhood aggregation?
- **Challenges:**
 - Overfitting from too many parameters.
 - Vanishing/exploding gradients during backpropagation.
- **Idea:** Use techniques from modern recurrent neural networks!

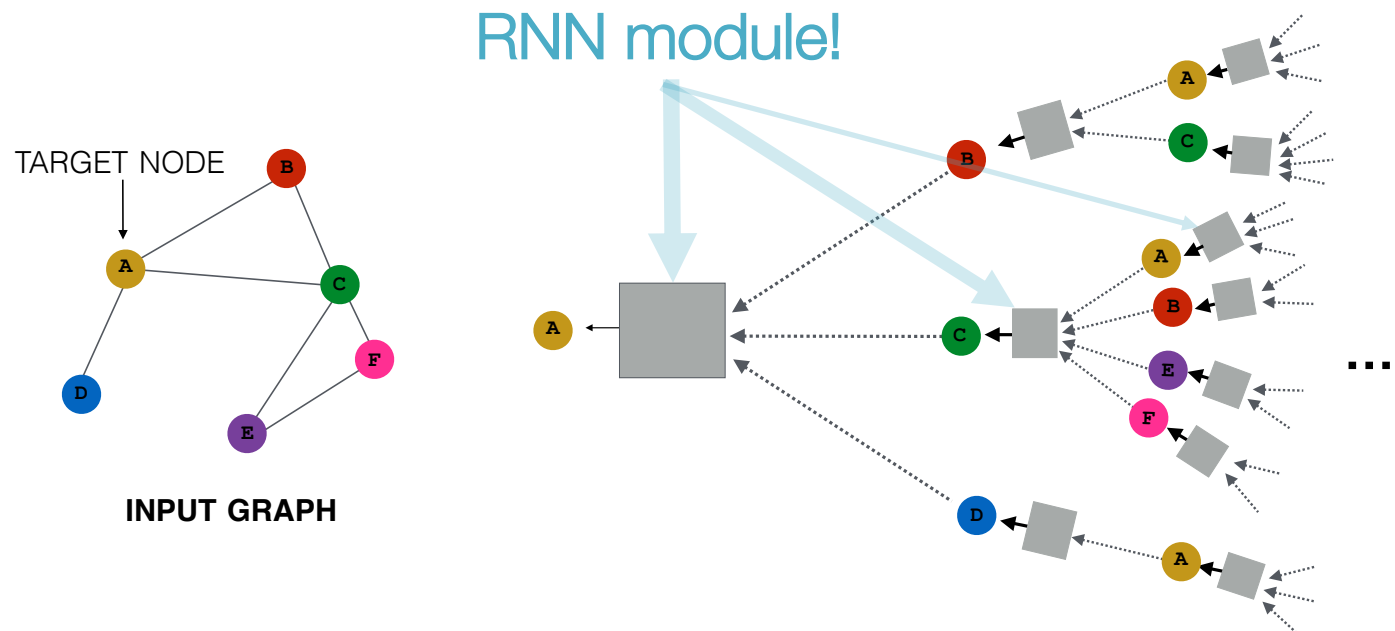
Gated Graph Neural Networks

- Idea 1: Parameter sharing across layers.

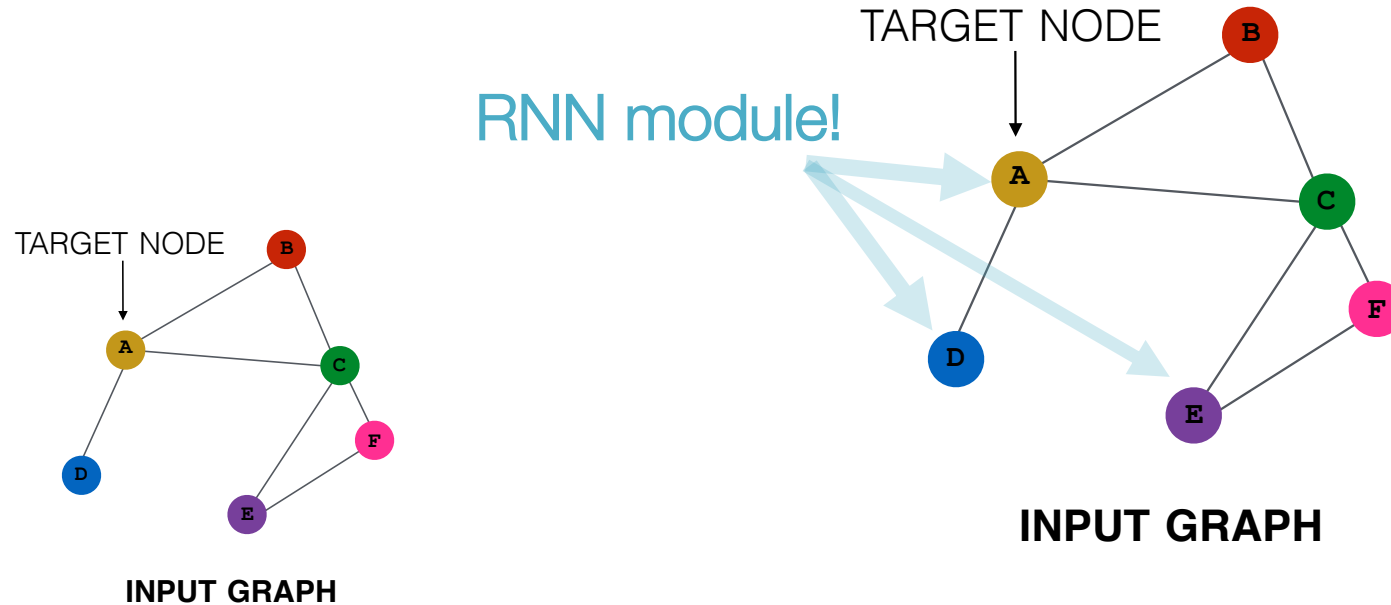
same neural network
across layers



Gated Graph Neural Networks



Gated Graph Neural Networks



The Math

- **Intuition:** Neighborhood aggregation with RNN state update.

1. Get “message” from neighbors at step k :

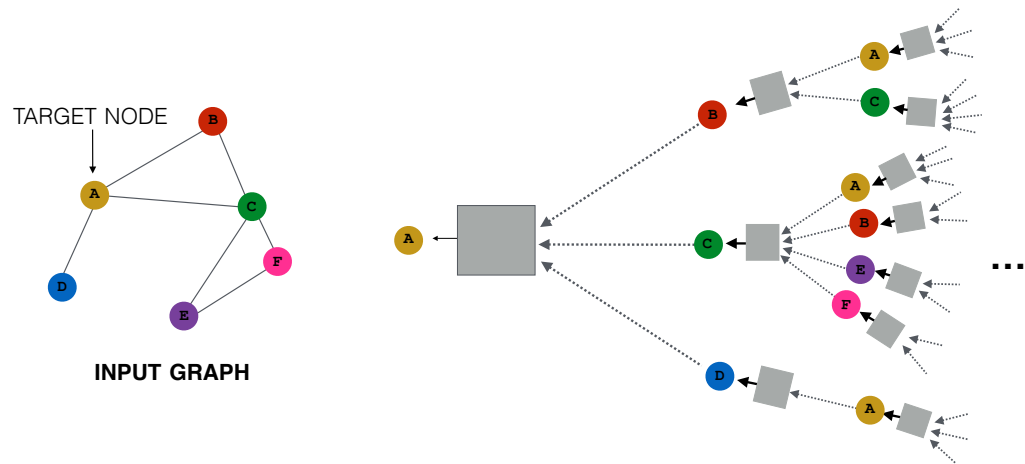
$$\mathbf{m}_v^k = \mathbf{W} \sum_{u \in N(v)} \mathbf{h}_u^{k-1}$$

← aggregation function does not depend on k

2. Update node “state” using Gated Recurrent Unit (GRU). New node state depends on the old state and the message from neighbors:

$$\mathbf{h}_v^k = \text{GRU}(\mathbf{h}_v^{k-1}, \mathbf{m}_v^k)$$

Gated Graph Neural Networks



- Can handle models with >20 layers.
- Most real-world networks have small diameters (e.g., less than 7).
- Allows for complex information about global graph structure to be propagated to all nodes.

Summary

- **Graph convolutional networks**
 - Average neighborhood information and stack neural networks.
- **GraphSAGE**
 - Generalized neighborhood aggregation.
- **Gated Graph Neural Networks**
 - Neighborhood aggregation + RNNs



Michael Bronstein

@mmbronstein

Head of Graph Learning Research

Only on Twitter

@Twitter

#OnlyOnTwitter

Insights

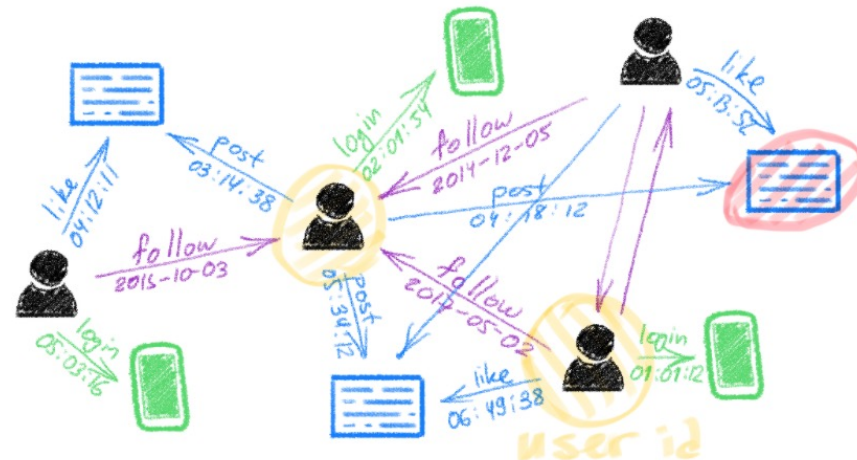
Graph ML at Twitter

By **Michael Bronstein**

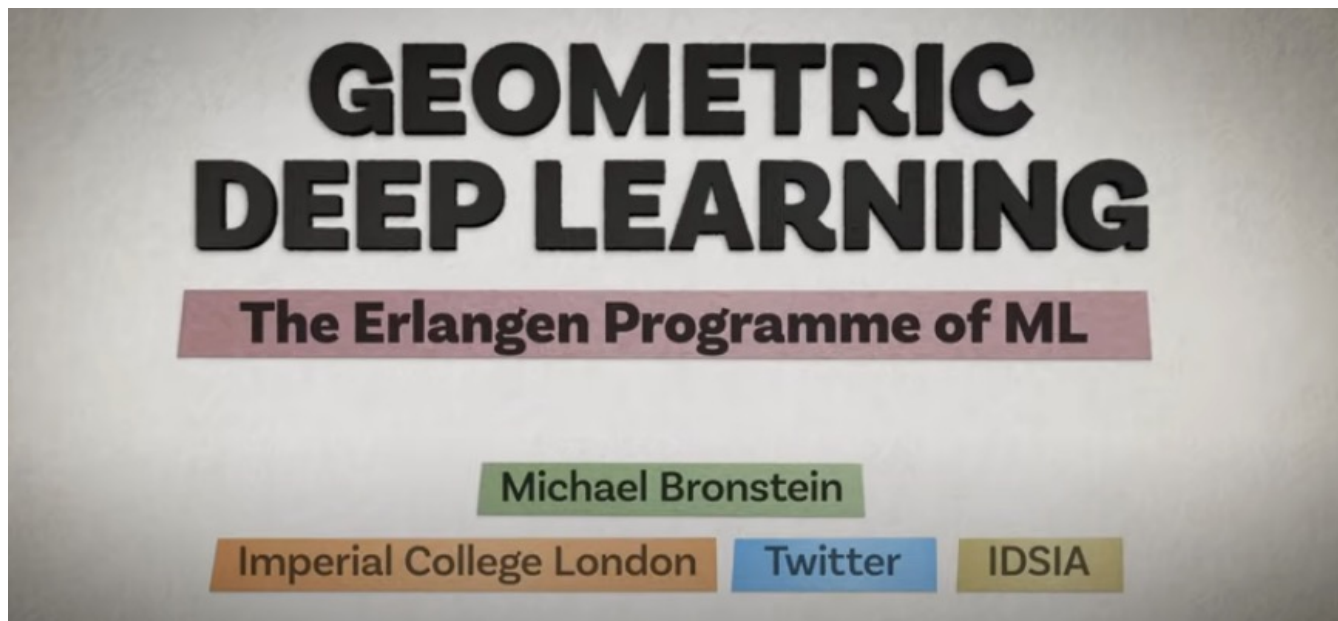
Wednesday, 2 September 2020 [Twitter](#) [Facebook](#) [LinkedIn](#) [Email](#)

Deep learning on graphs — also known as Geometric deep learning (GDL)¹, Graph representation learning (GRL), or relational inductive biases² — has recently become one of the hottest topics in machine learning. While early works on graph learning go back at least a decade³, if not two⁴, it is undoubtedly the past few years' progress that has taken these methods from a niche interest into the **spotlight of the ML community**, complete with dedicated **workshops** attracting large crowds.

Graphs are mathematical abstractions of complex systems of relations and interactions. A graph represents a network of objects (called nodes or vertices) with pairwise connections (edges). Graphs are ubiquitously used in fields as diverse as biology^{5 6 7}, quantum chemistry⁸, and high-energy physics⁹. Social networks like Twitter are examples of very large-scale complex graphs where the nodes model users and Tweets, while the edges model interactions such as replies, Retweets, or favs. Public conversations happening on our platform typically generate hundreds of millions of Tweets and Retweets every day. This makes Twitter perhaps one of the largest producers of graph-structured data in the world, second perhaps only to the **Large Hadron Collider**.



Amazing Talk – Keynote @ ICLR'21



<https://www.youtube.com/watch?v=w6Pw4MOzMuo>

Recent advances in graph neural nets (not covered in detail here)

- Attention-based neighborhood aggregation:
 - Graph Attention Networks ([Velickovic et al., 2018](#))
 - GeniePath ([Liu et al., 2018](#))
- Generalizations based on spectral convolutions:
 - Geometric Deep Learning ([Bronstein et al., 2017](#))
 - Mixture Model CNNs ([Monti et al., 2017](#))
- Speed improvements via subsampling:
 - FastGCNs ([Chen et al., 2018](#))
 - Stochastic GCNs ([Chen et al., 2017](#))

