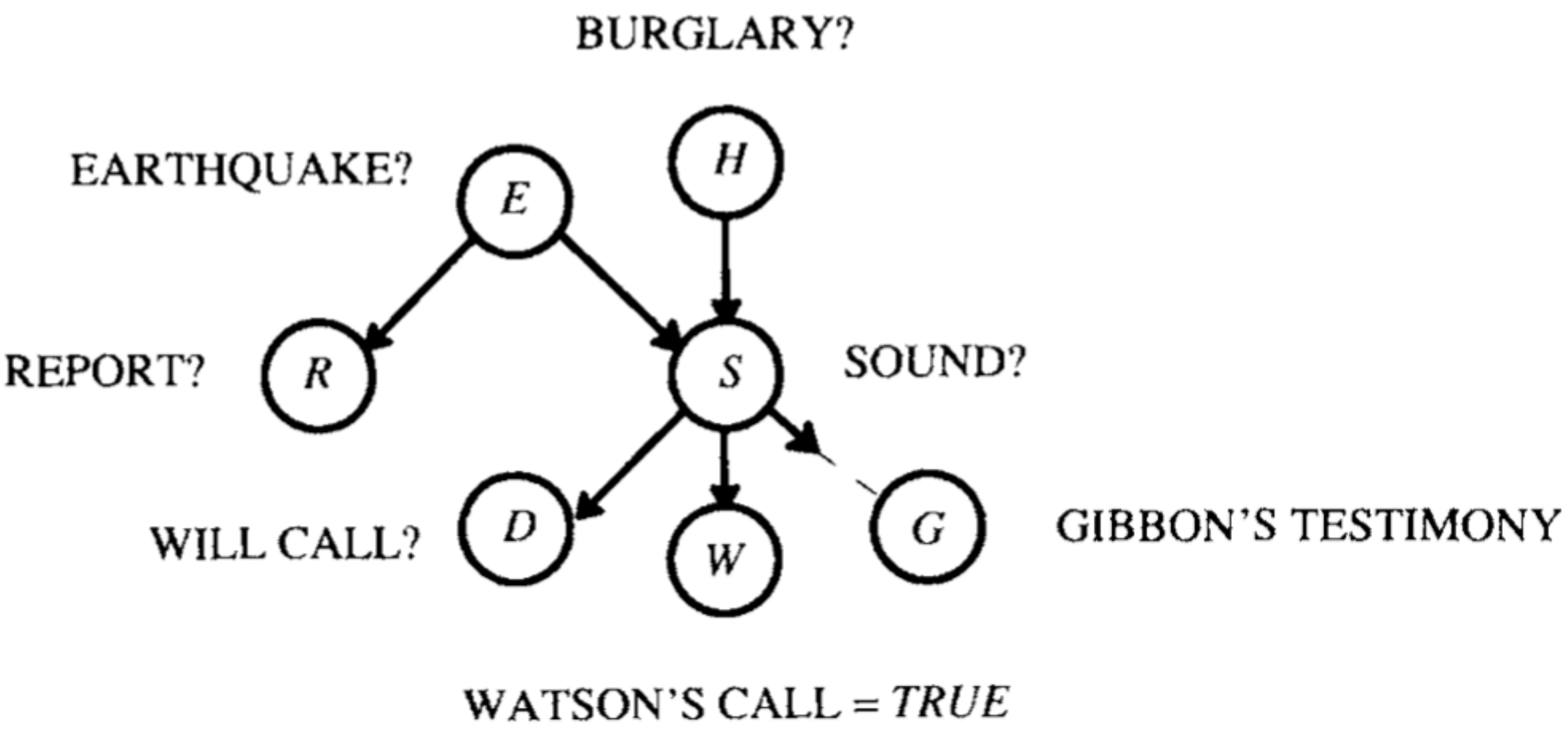
Graph Neural Networks

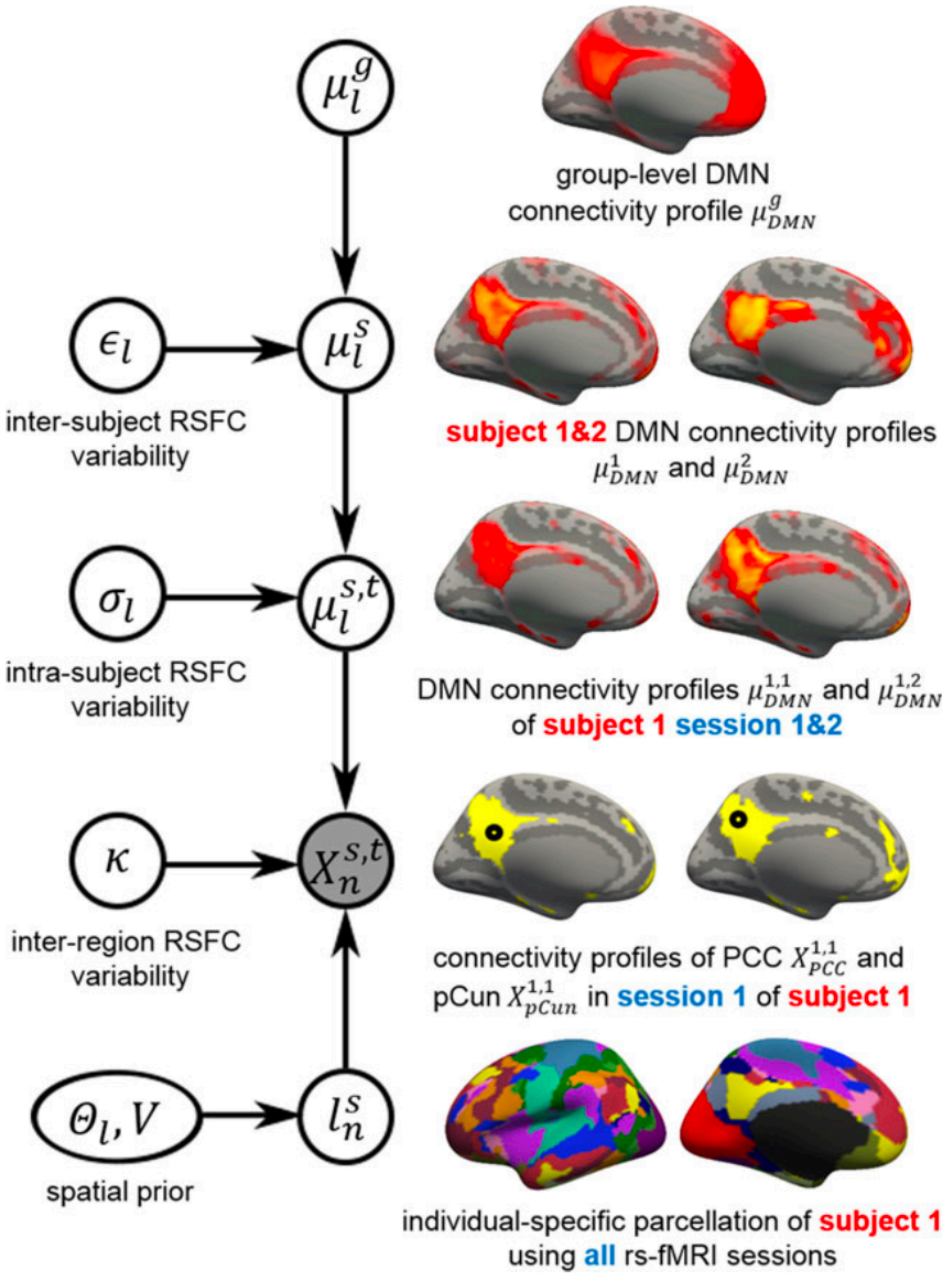
Demian Wassermann, Inria

Graphical Models: Discrete Inference and Learning

Introduction to DAG and their relationship with Probability Functions (Pearl)



[Pearl 1987]



[Kong et al 2019]

And the Usual Graph Slide

Image credit: Medium

Social Networks

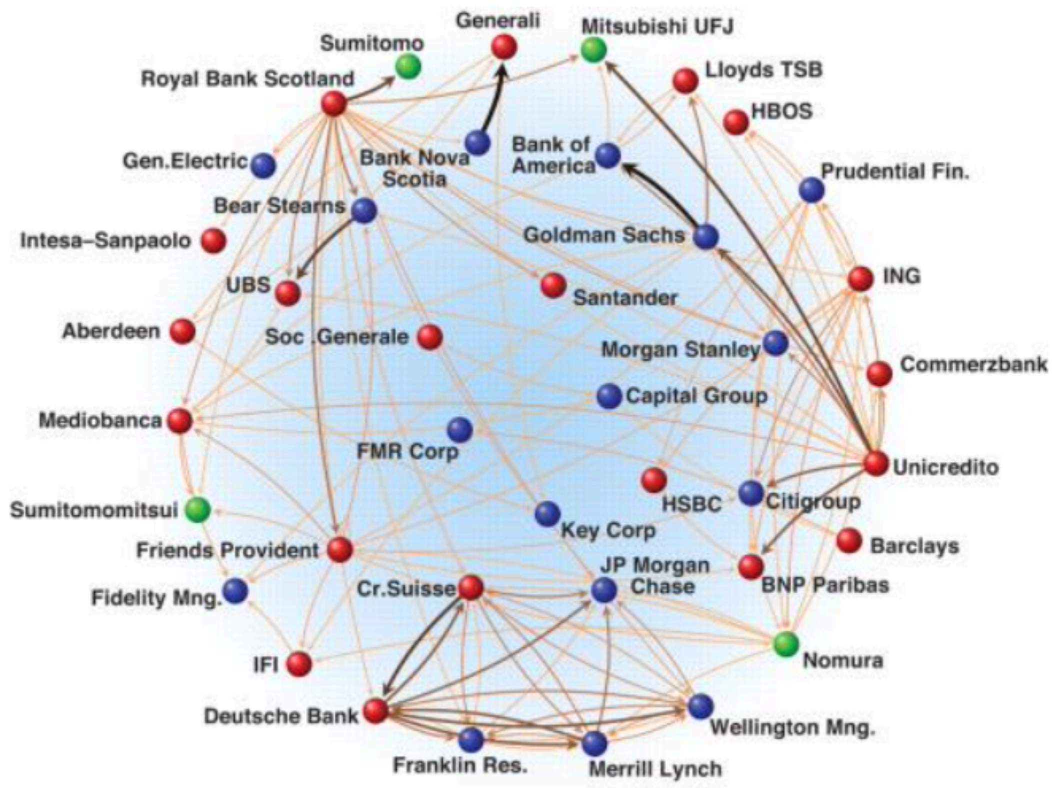
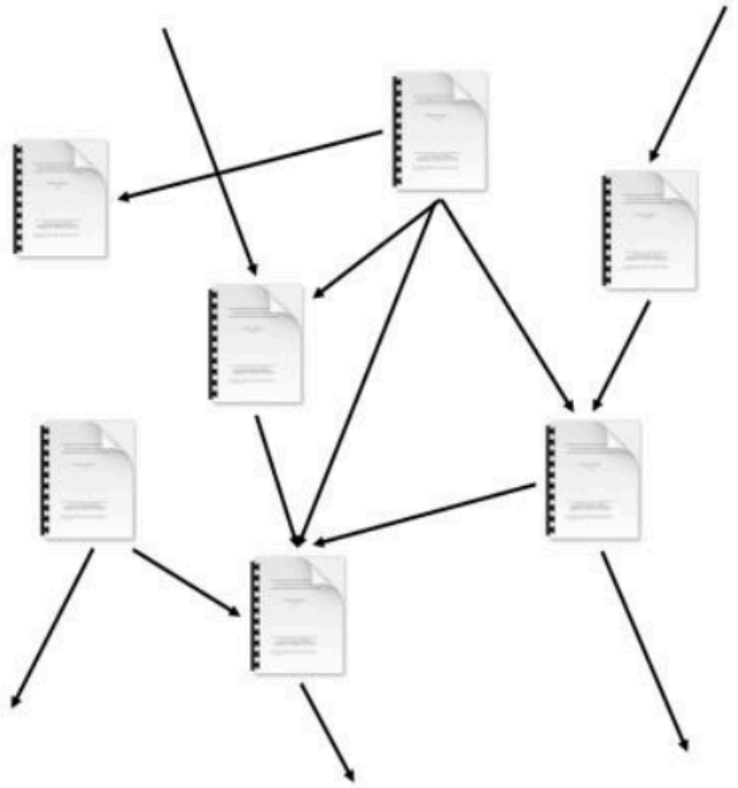


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks



Citation Networks

Image credit: Missoula Current News

Internet

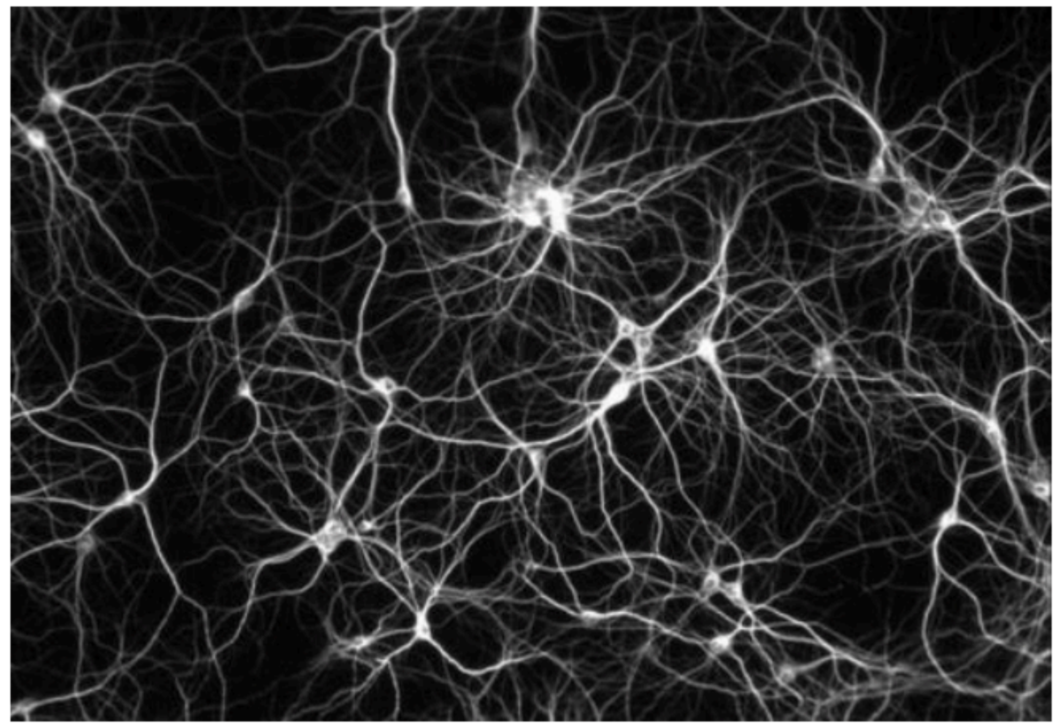
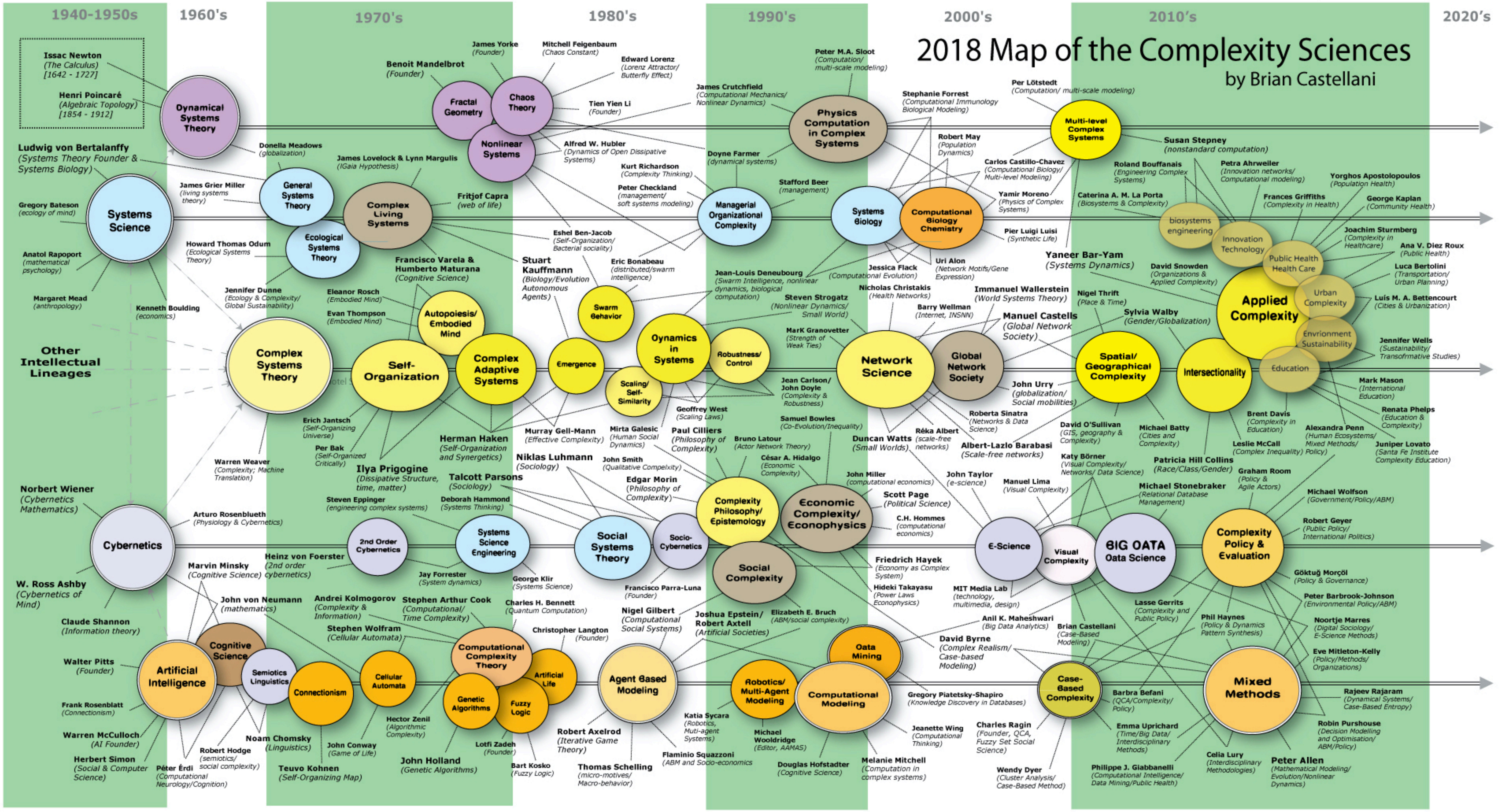


Image credit: The Conversation

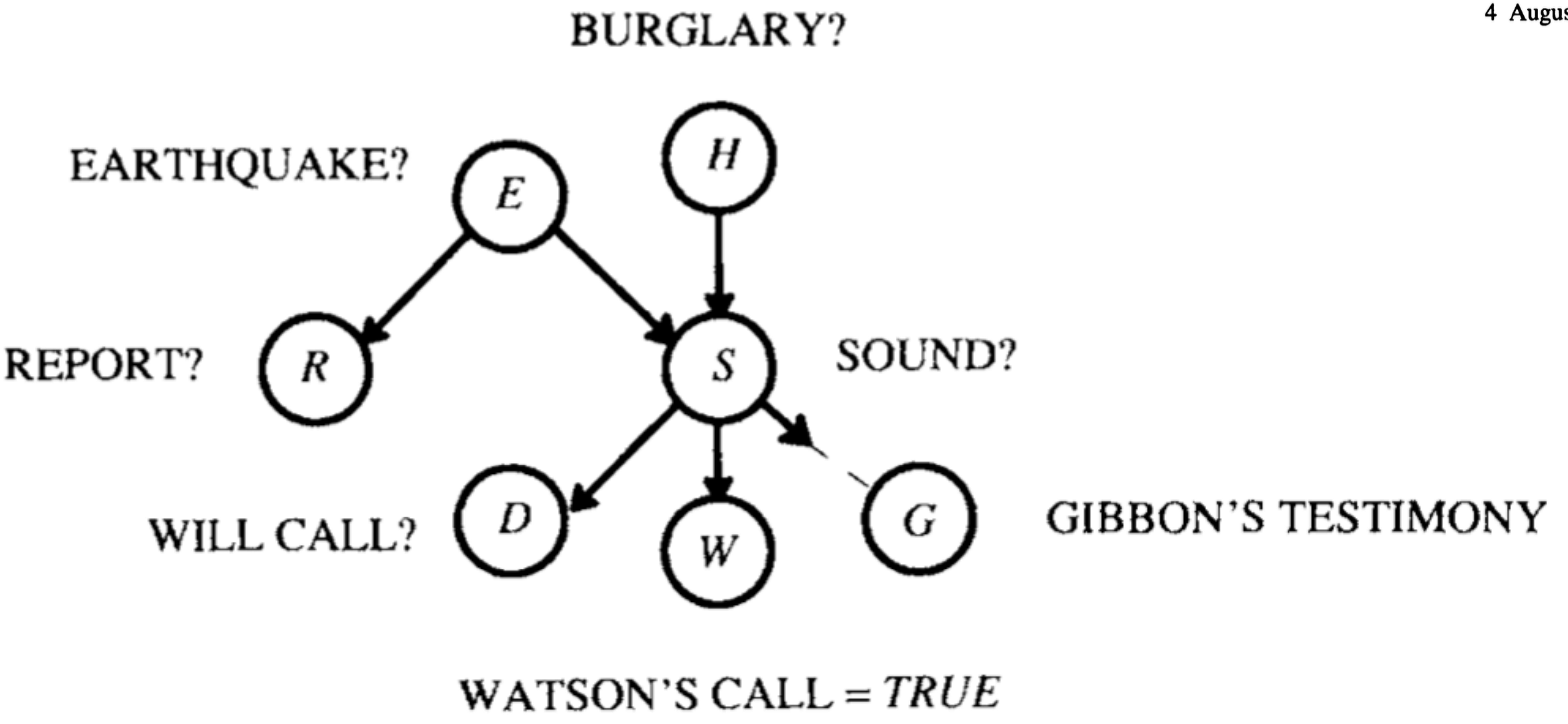
Networks of Neurons

Complex Systems to Understand the World



Wikipedia

Main Epistemological Angles on Graphs and Knowledge



[Pearl 1987]

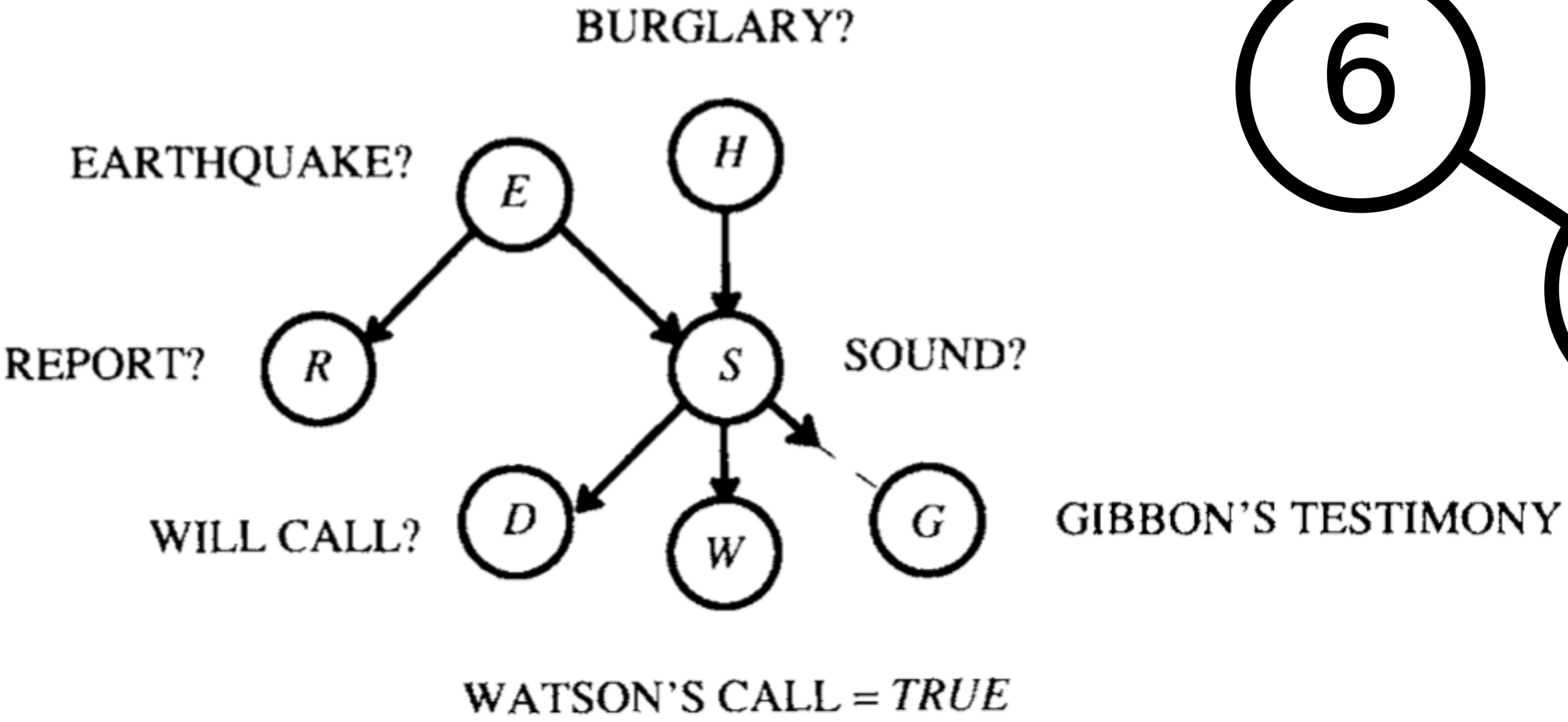
4 August 1972, Volume 177, Number 4047 **SCIENCE**

More Is Different

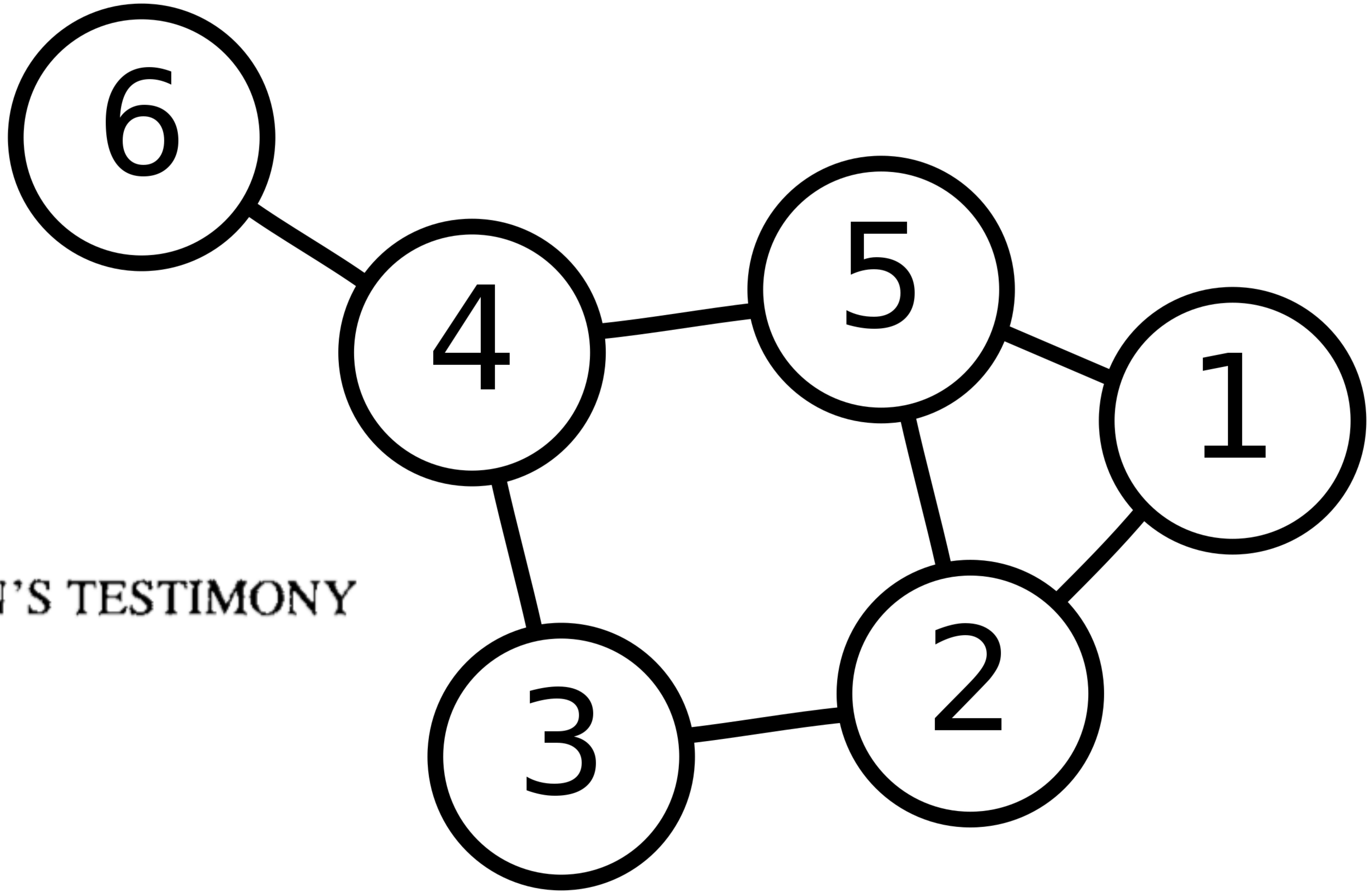
Broken symmetry and the nature of the hierarchical structure of science.

P. W. Anderson

Main Epistemological Angles on Graphs and Knowledge

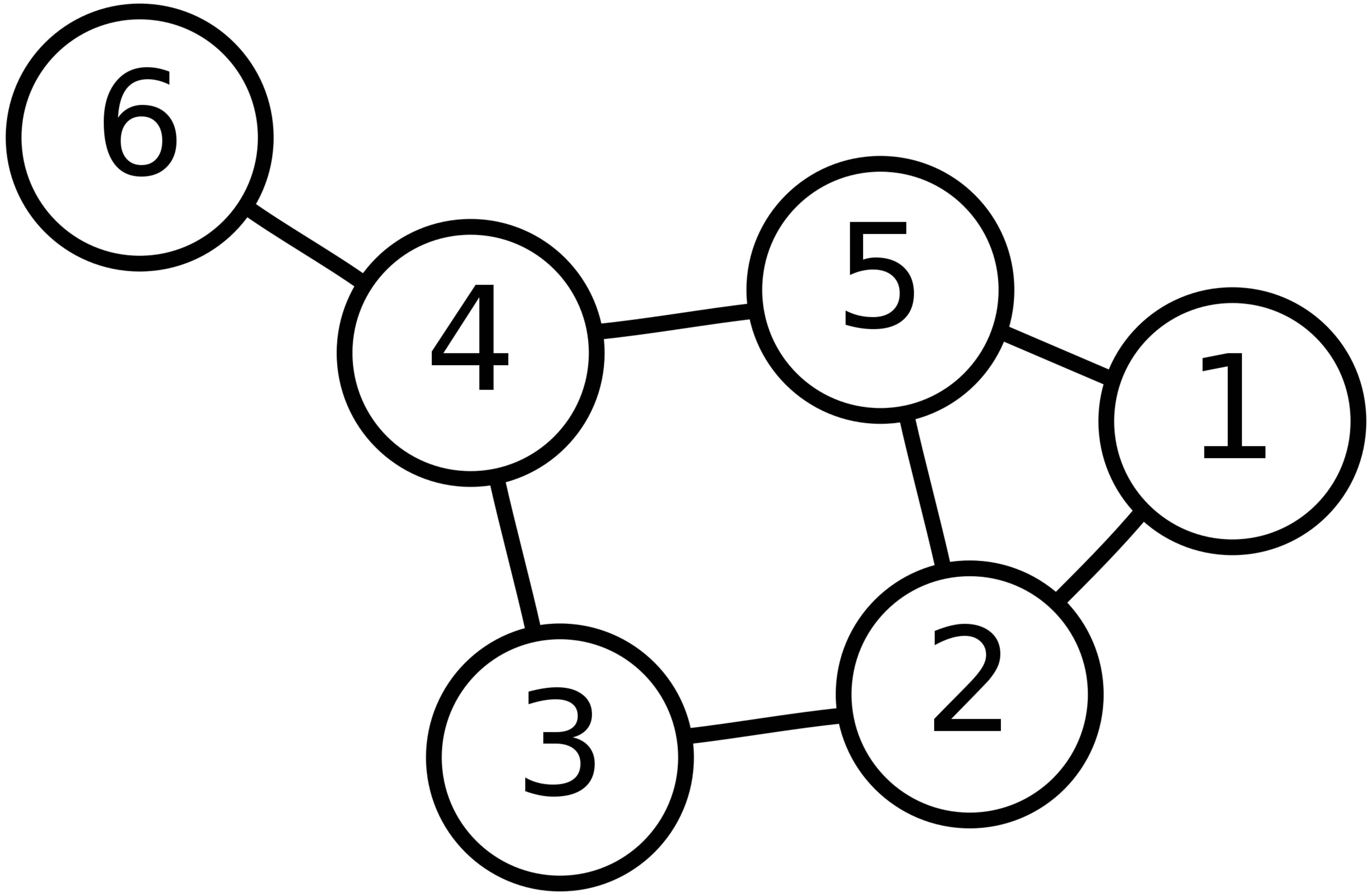


[Pearl 1987]



Alternative General Undirected Graphs

- Graph $G(V, E, f, g), E \subseteq V \times V, f : V \mapsto F, g : E \mapsto G$

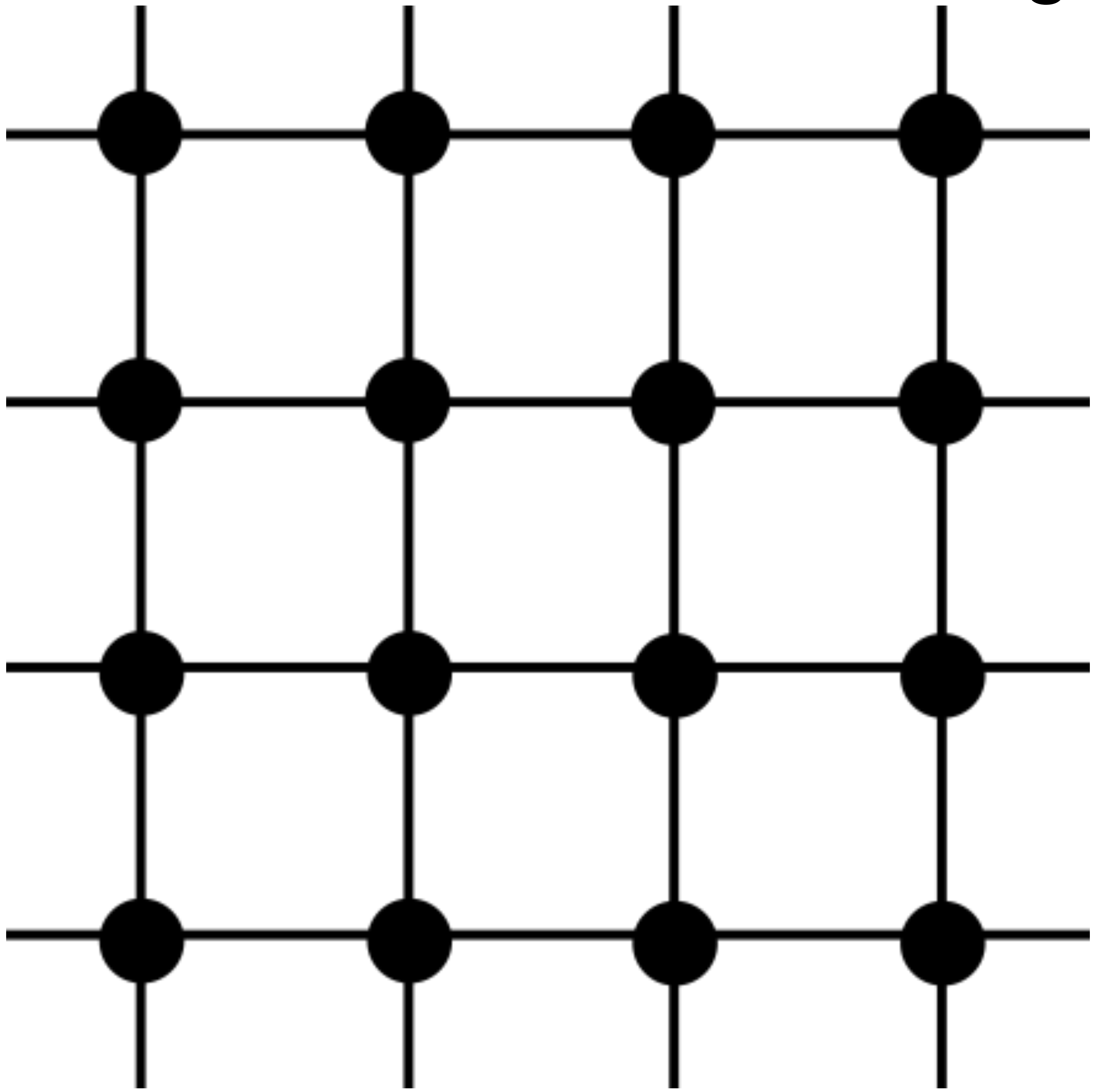


- V are the vertices
- E are the edges
- f is a mapping of features for vertices
- g is a mapping of features for edges

Alternative General Undirected Graphs

- Graph $G(V, E, f, g), E \subseteq V \times V, f : V \mapsto F, g : E \mapsto G$

The Lattice Case, or an Image

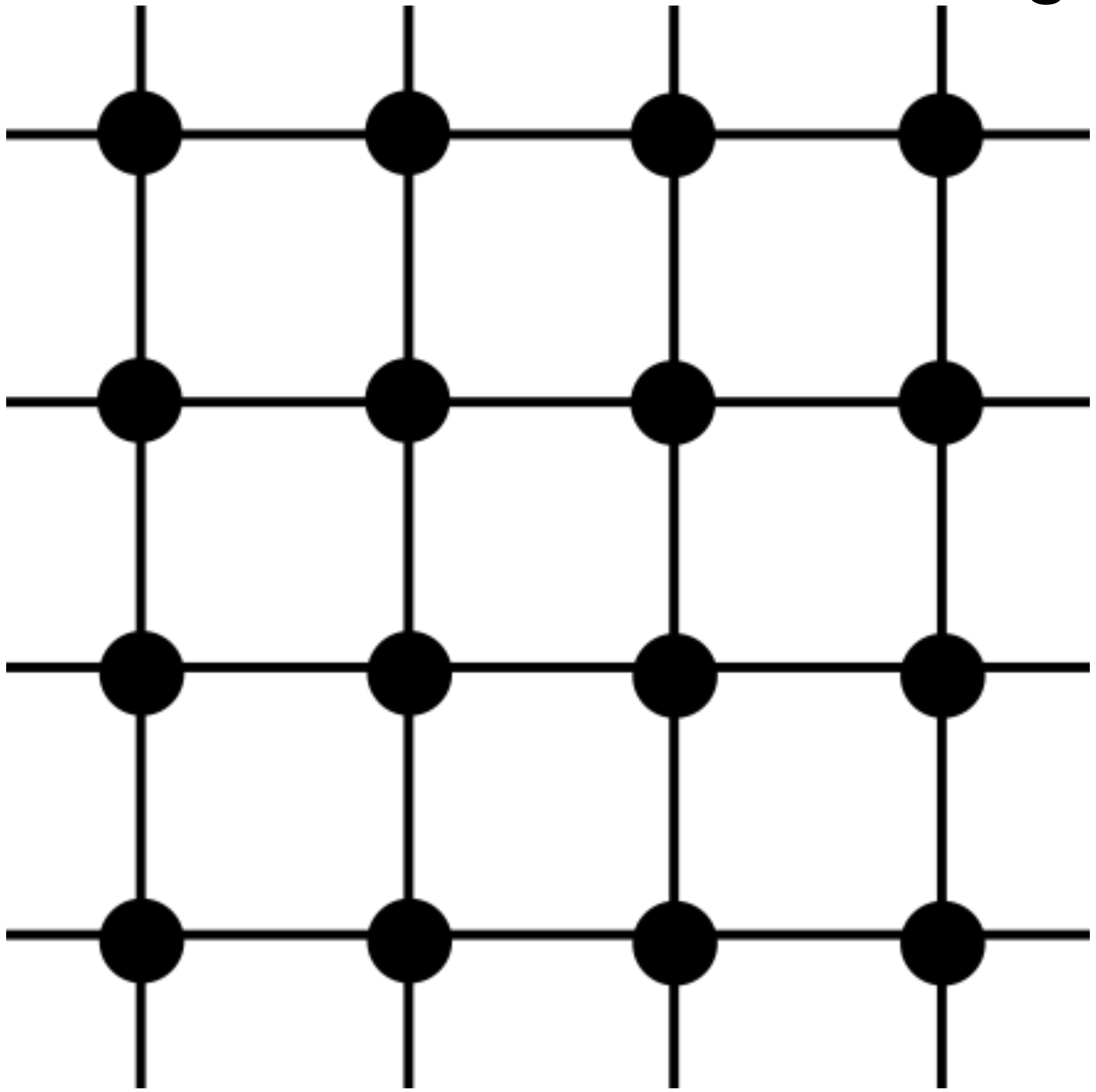


- V are the vertices
- E are the edges
- f is a mapping of features for vertices
- g is a mapping of features for edges

Alternative General Undirected Graphs

- Graph $G(V, E, f, g), E \subseteq V \times V, f : V \mapsto F, g : E \mapsto G$

The Lattice Case, or an Image



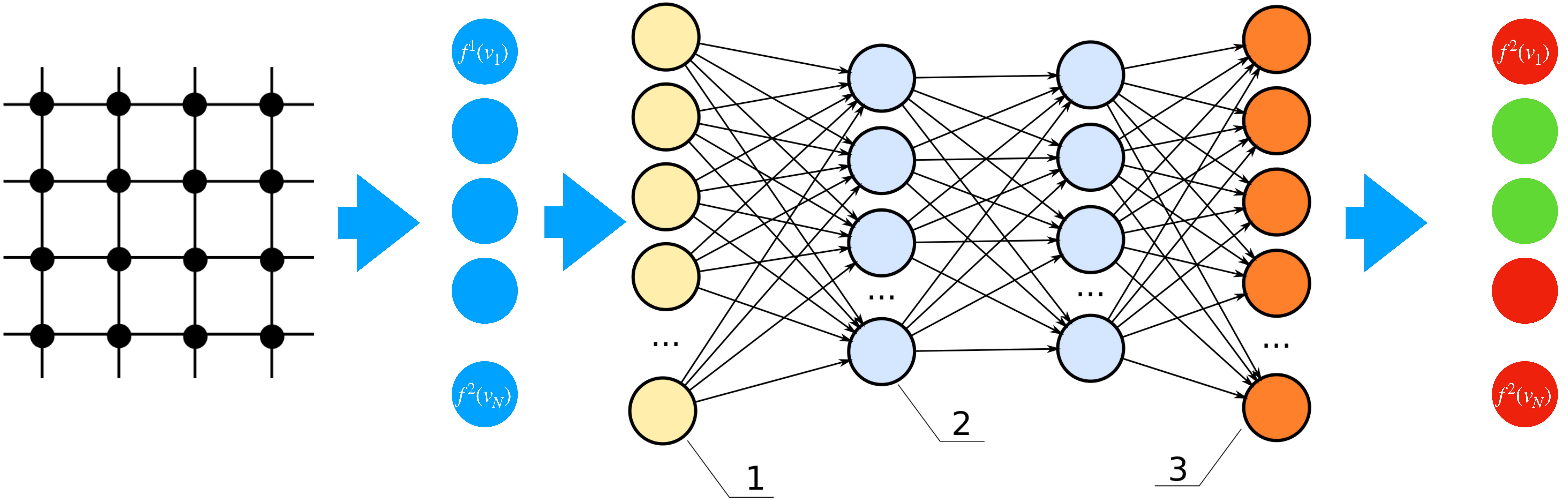
- V are the vertices
- E are the edges
- f is a mapping of features for vertices

- Learn $\arg\min_{\theta} \mathscr{L}([f(v_i)]_i, [\phi_\theta(v_i)]_i)$ for a fixed ordering i

Alternative General Undirected Graphs

- Learn $\arg\min_\theta \mathscr{L}([f^2(v_i)]_i, [\phi_\theta(f^1(v_i))]_i)$ for a fixed ordering i

The Lattice Case, or an Image

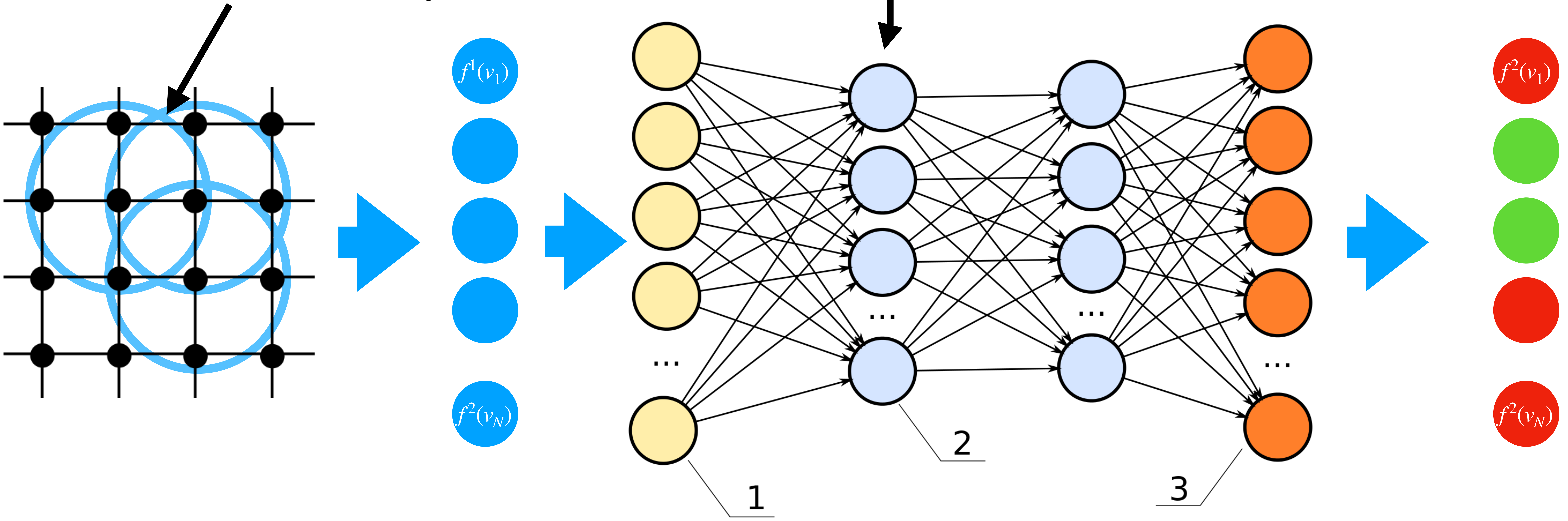


Alternative General Undirected Graphs

- Learn $\arg\min_{\theta} \mathscr{L}([f^2(v_i)]_i, [\phi_\theta(f^1(v_i))]_i)$ for a fixed ordering i

The Lattice Case, or an Image

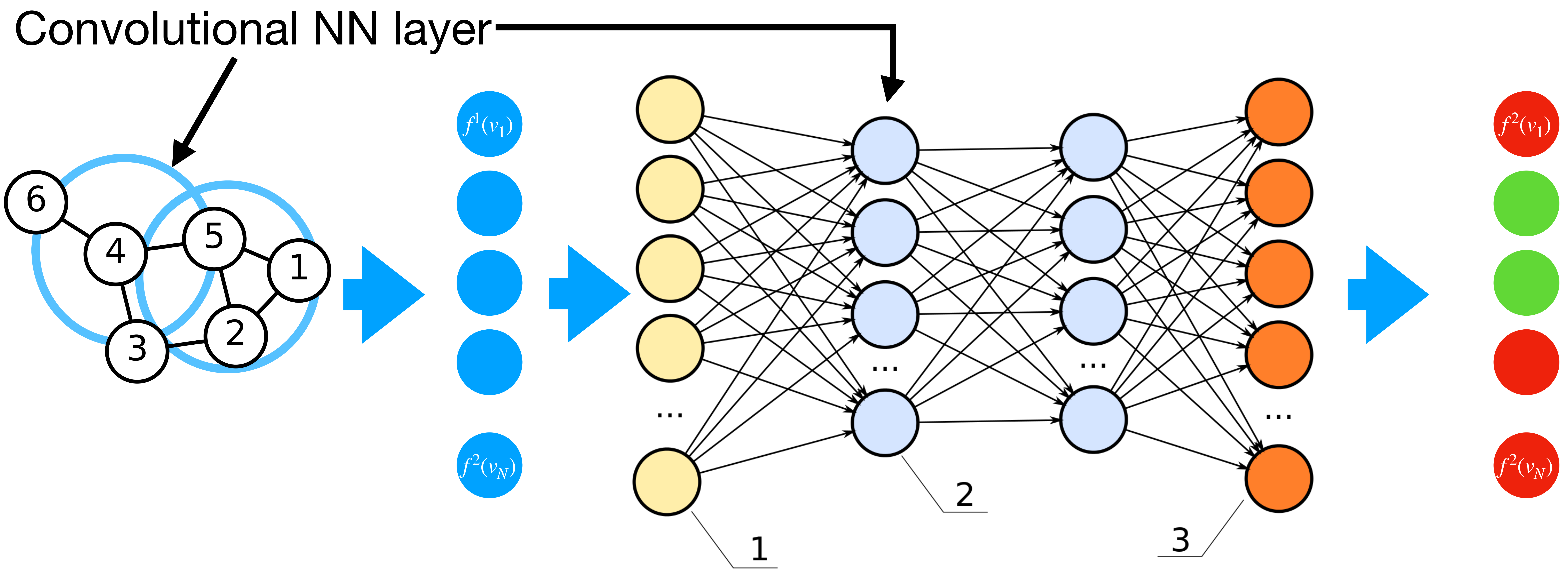
Convolutional NN layer



Alternative General Undirected Graphs

- Learn $\arg\min_{\theta} \mathcal{L}([f^2(v_i)]_i, [\phi_\theta(f^1(v_i))]_i)$ for a fixed structure

The Graph Case, and now?

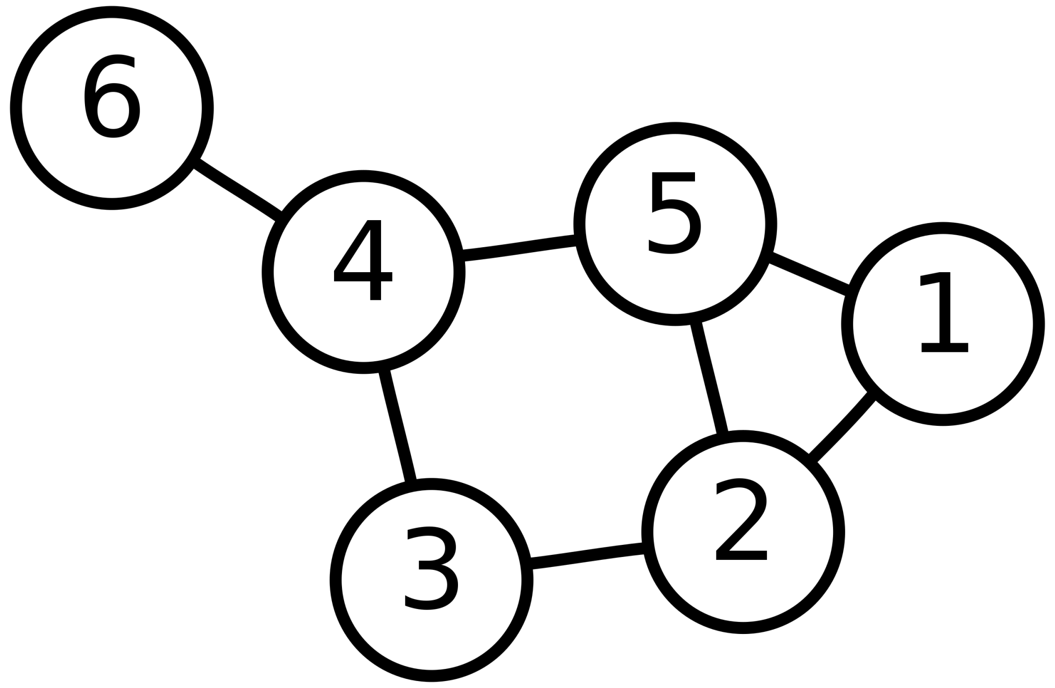


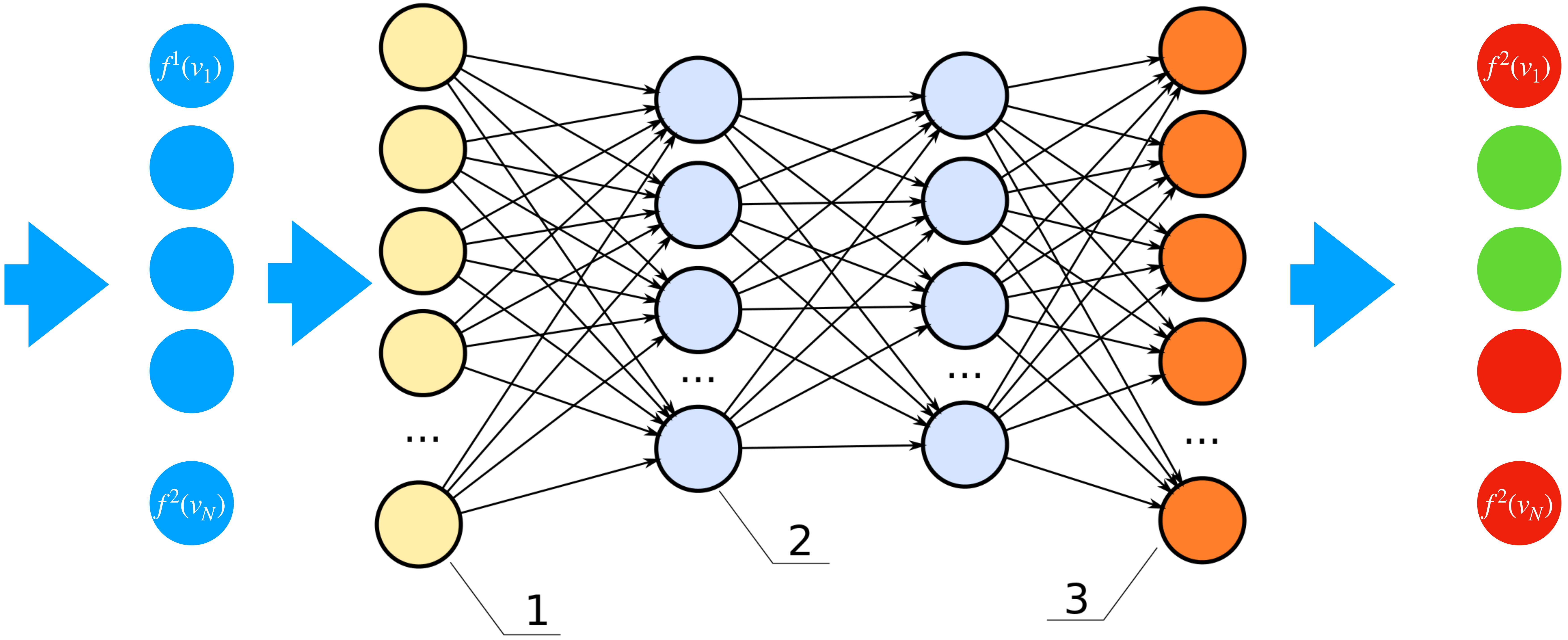
Alternative General Undirected Graphs

- Learn $\arg\min_{\theta} \mathscr{L}([f^2(v_i)]_i, [\phi_\theta(f^1(v_i))]_i)$ for a fixed structure

The Graph Case, use the affinity matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$





Alternative General Undirected Graphs

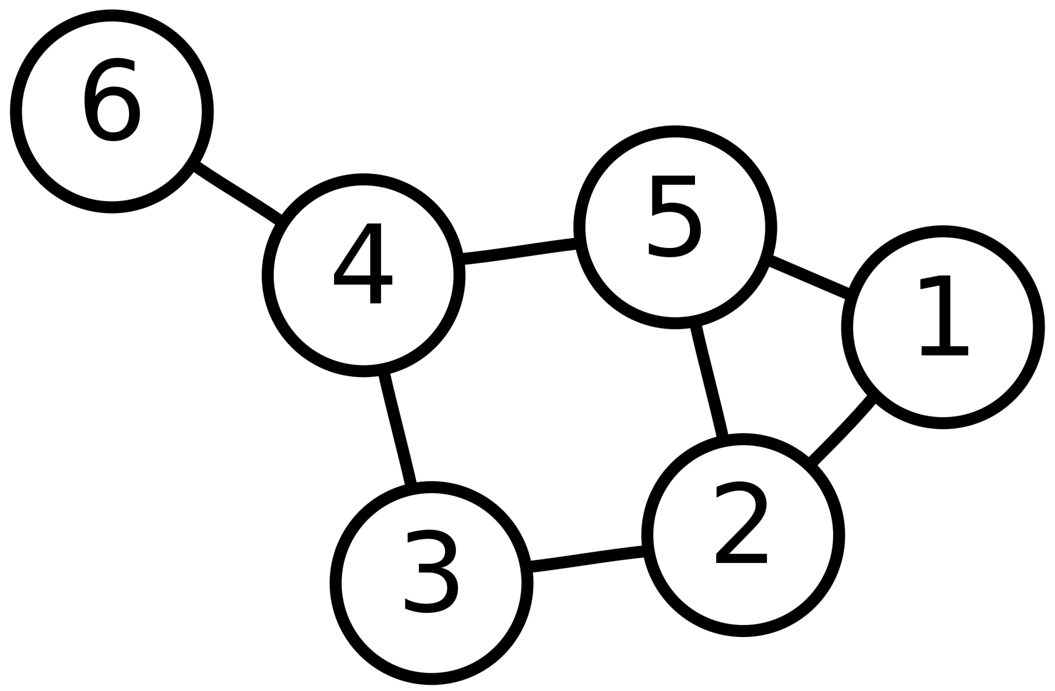
- Learn $\arg\min_{\theta} \mathscr{L}([f^2(v_i)]_i, [\phi_\theta(f^1(v_i))]_i)$ for a fixed structure

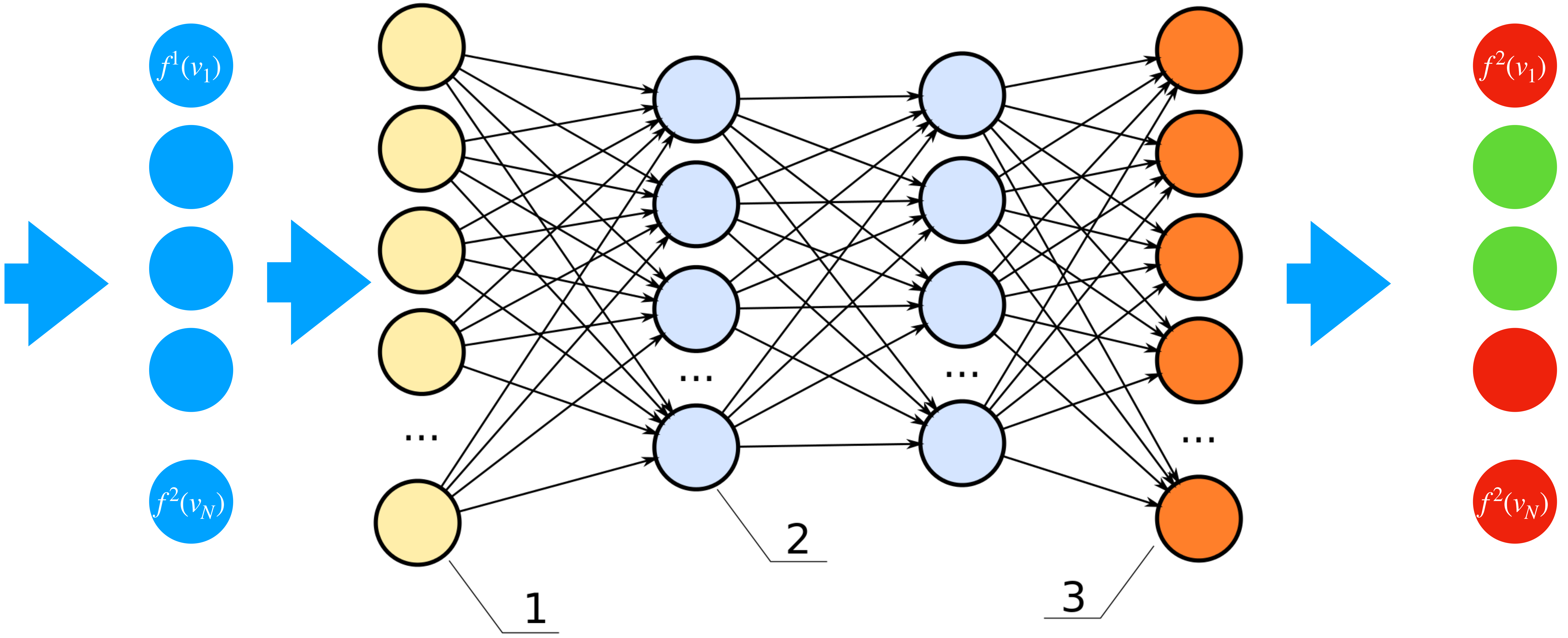
Issues

- O(|V|) Parameters
- Not applicable different sizes
- Sensitive to node ordering

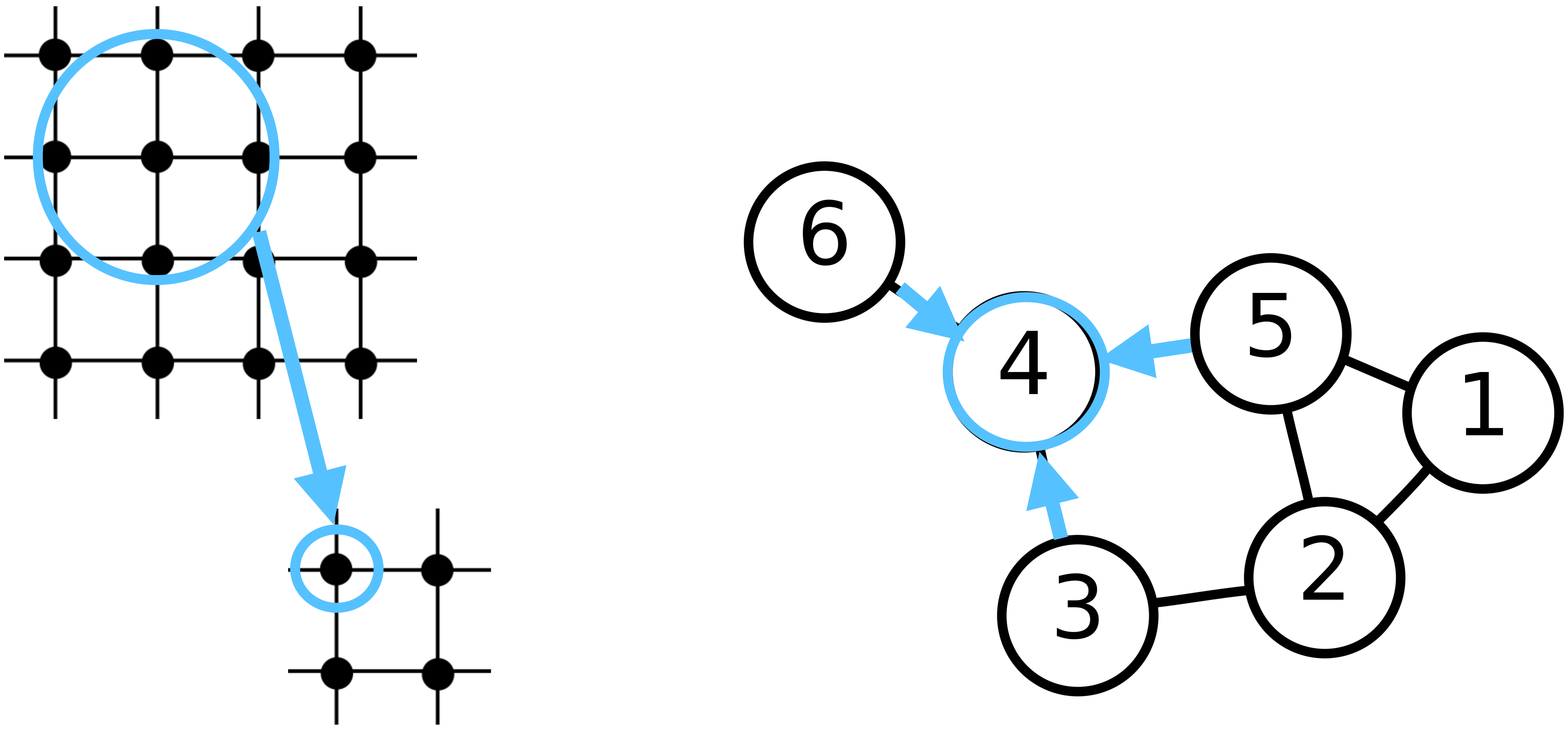
The Graph Case, use the affinity matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

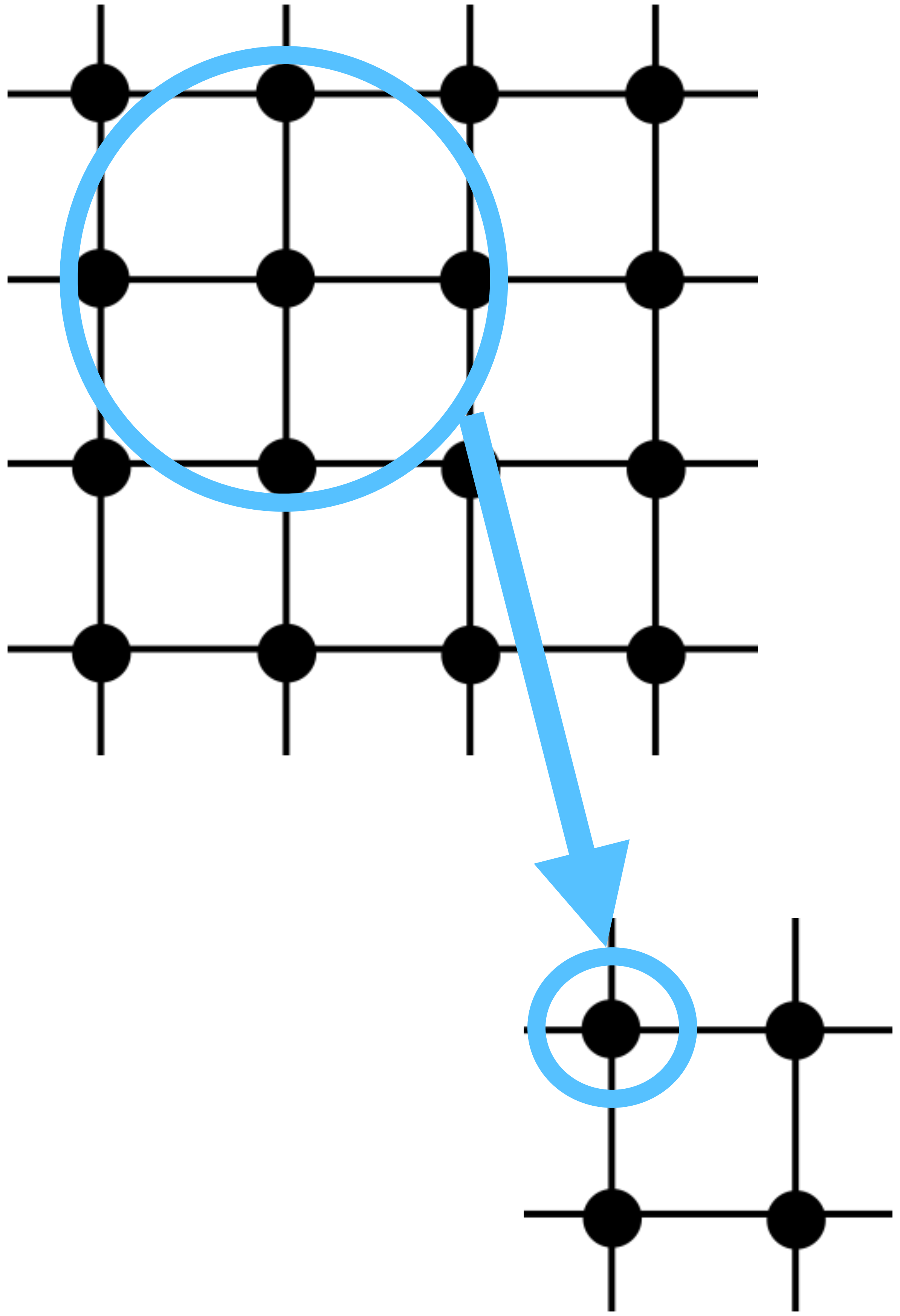


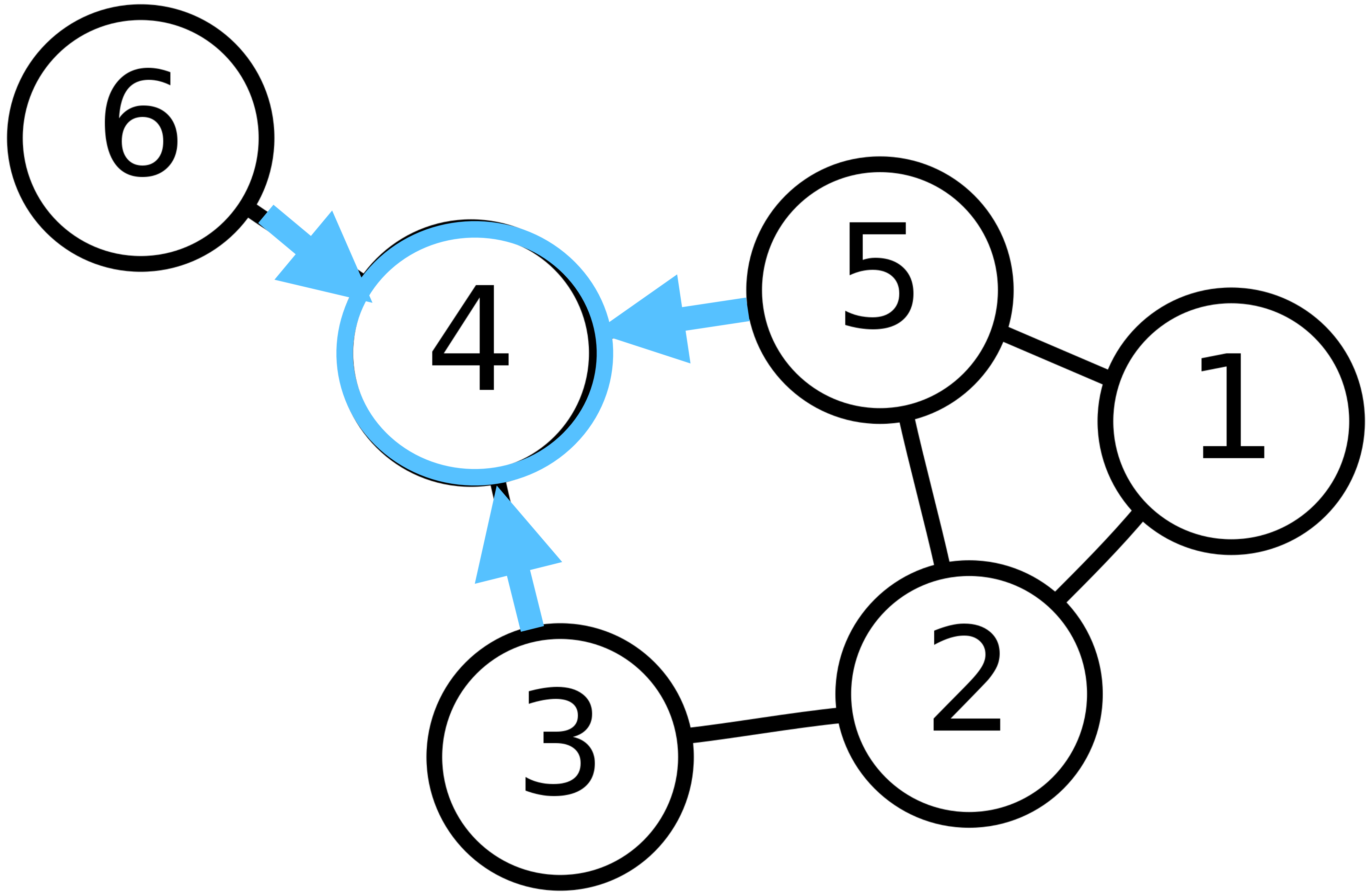


Convolutions on Graphs: Message Passing

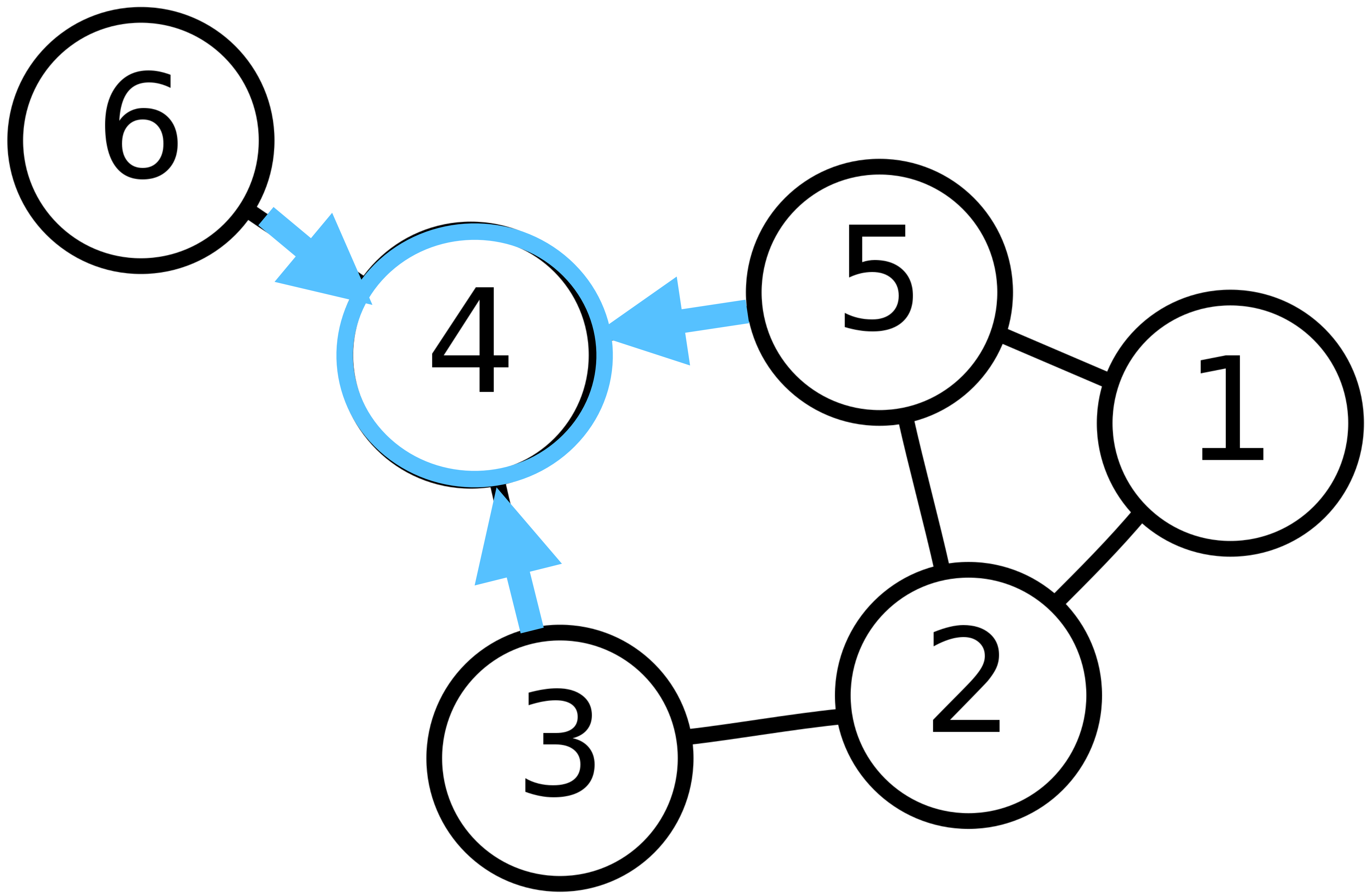


Convolutions on Graphs: Message Passing





Convolutions on Graphs: Message Passing



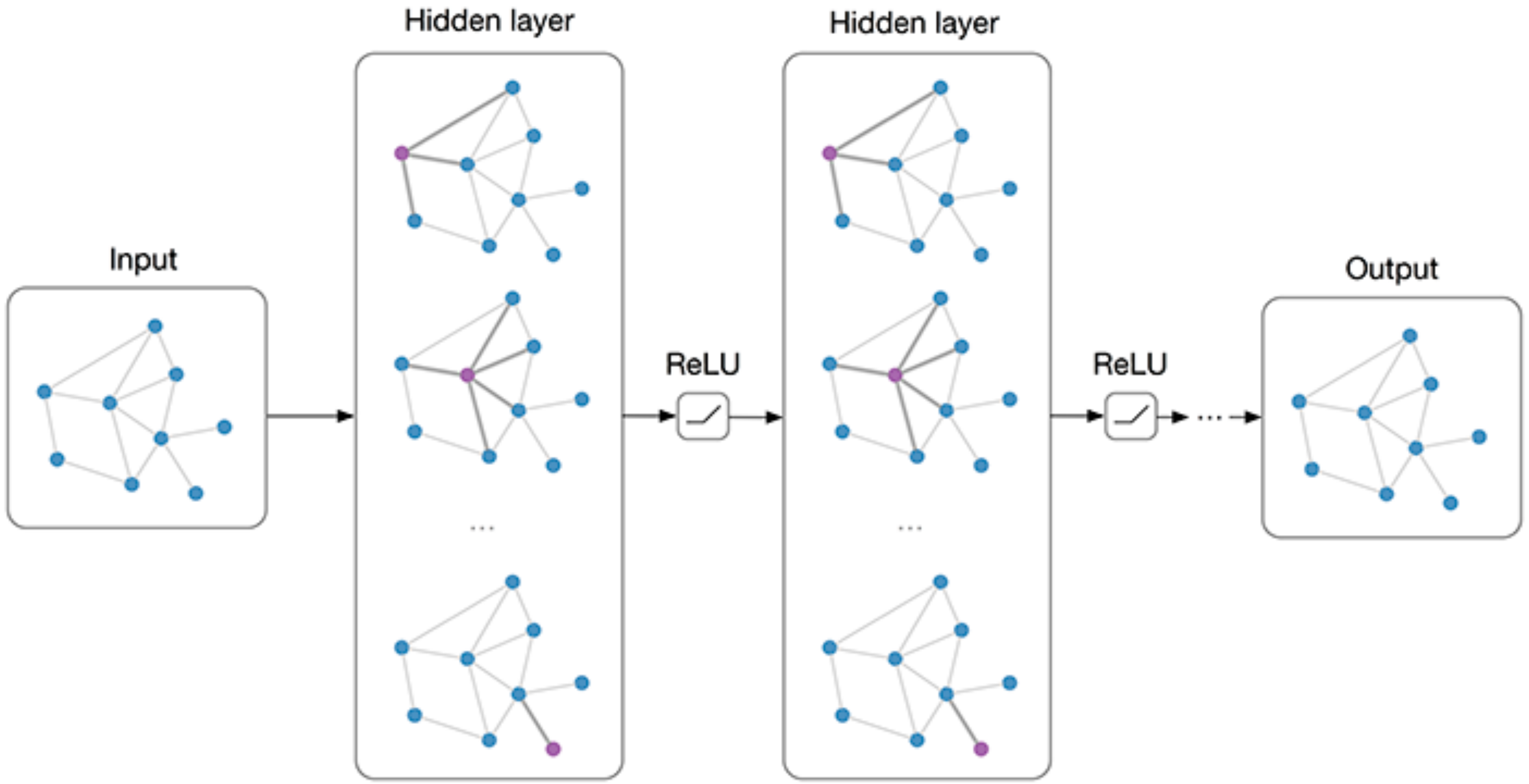
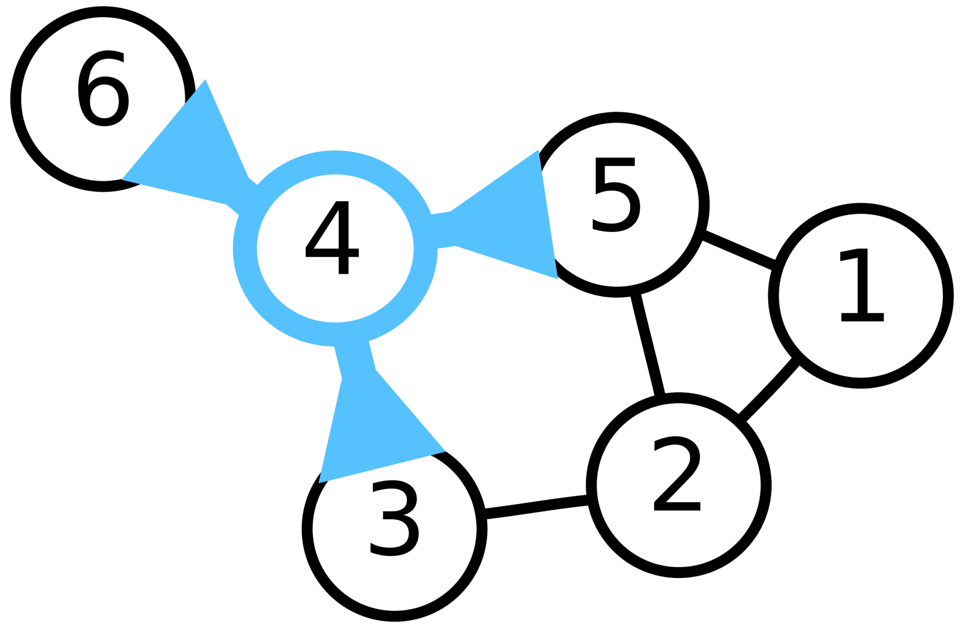


Image from T. Kipf's blog. Kipf et al. ICLR 2017

Convolutions on Graphs: Message Passing

$$h_u^{(k+1)} = \text{update}^{(k)}\left(h_u^{(k)}, \text{agg}^{(k)}_{v \in \mathcal{N}(u)} h_v^{(k)}\right)$$



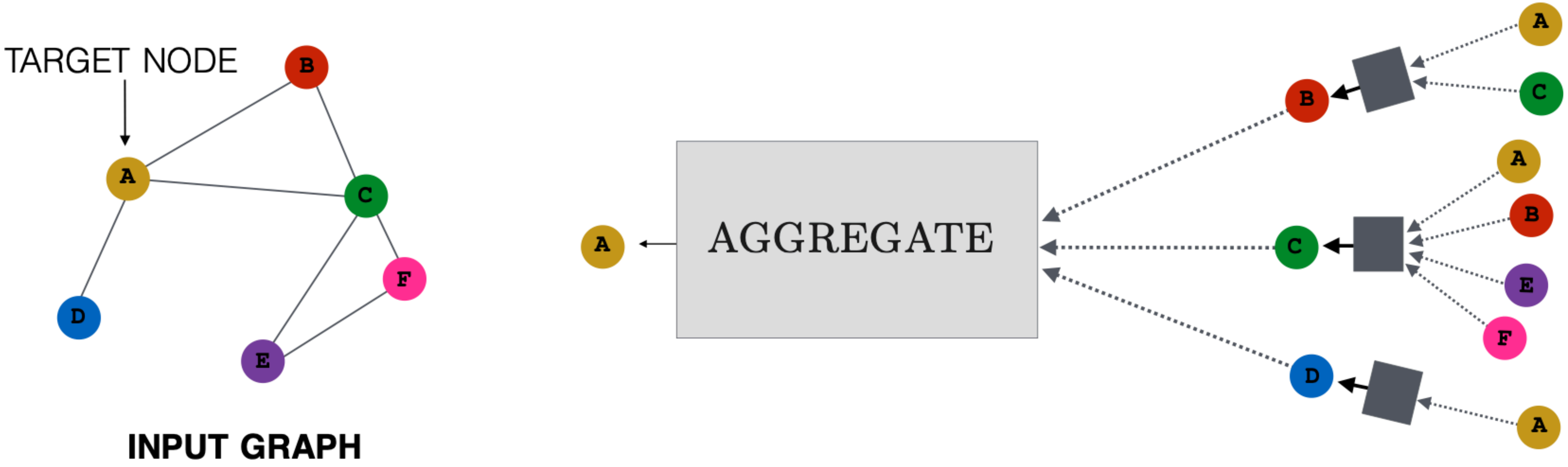


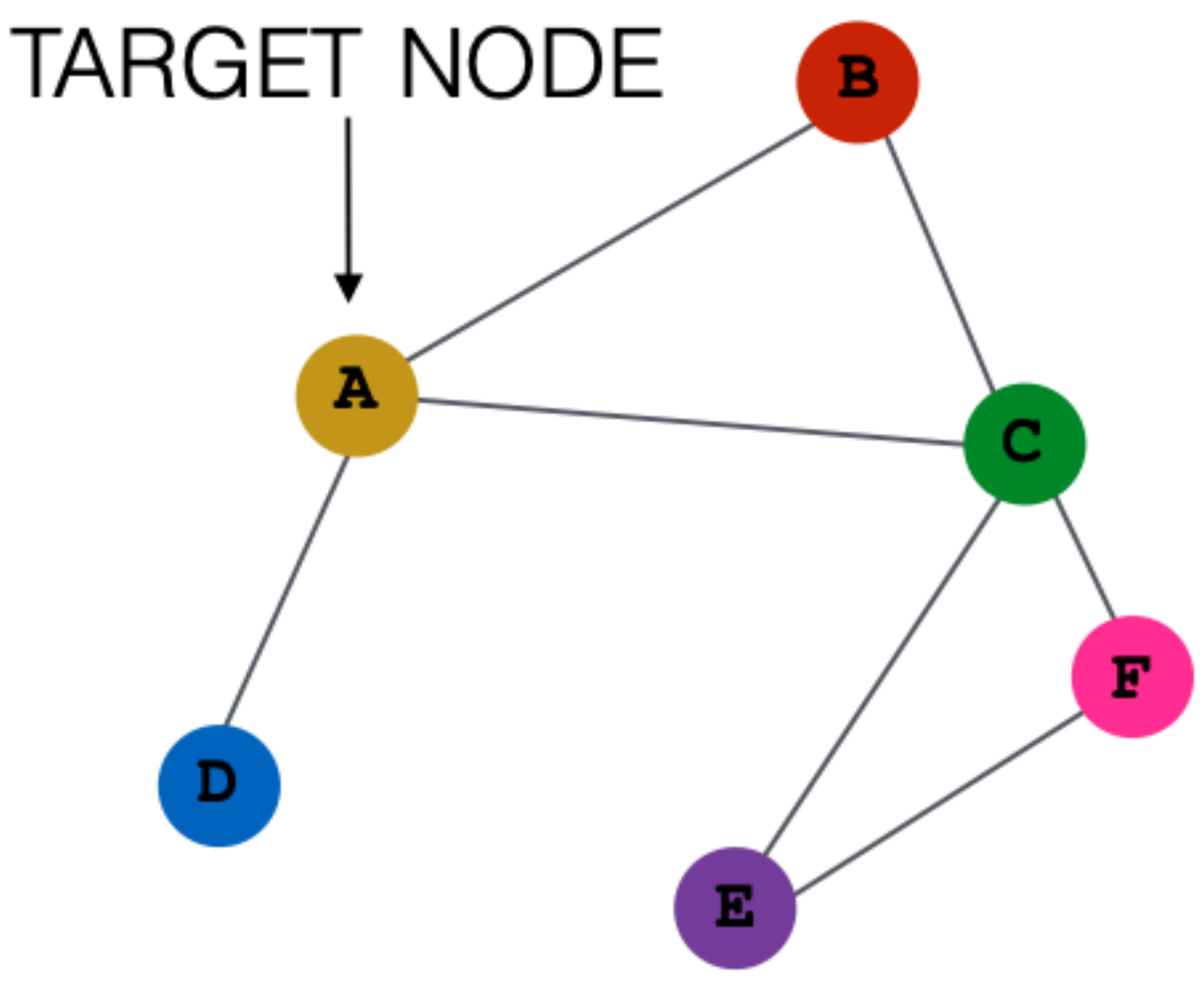
Image from Hamilton, “Graph Representation Learning Book”

Convolutions on Graphs: Message Passing

$$h_u^{(k+1)} = \text{update}^{(k)}(h_u^{(k)}, \text{agg}^{(k)}_{v \in \mathcal{N}(u)} h_v^{(k)})$$

Main Points

- Each gray block is a trainable network
- Each node has a different architecture
- Some will have the same
- We can learn over different architectures



INPUT GRAPH

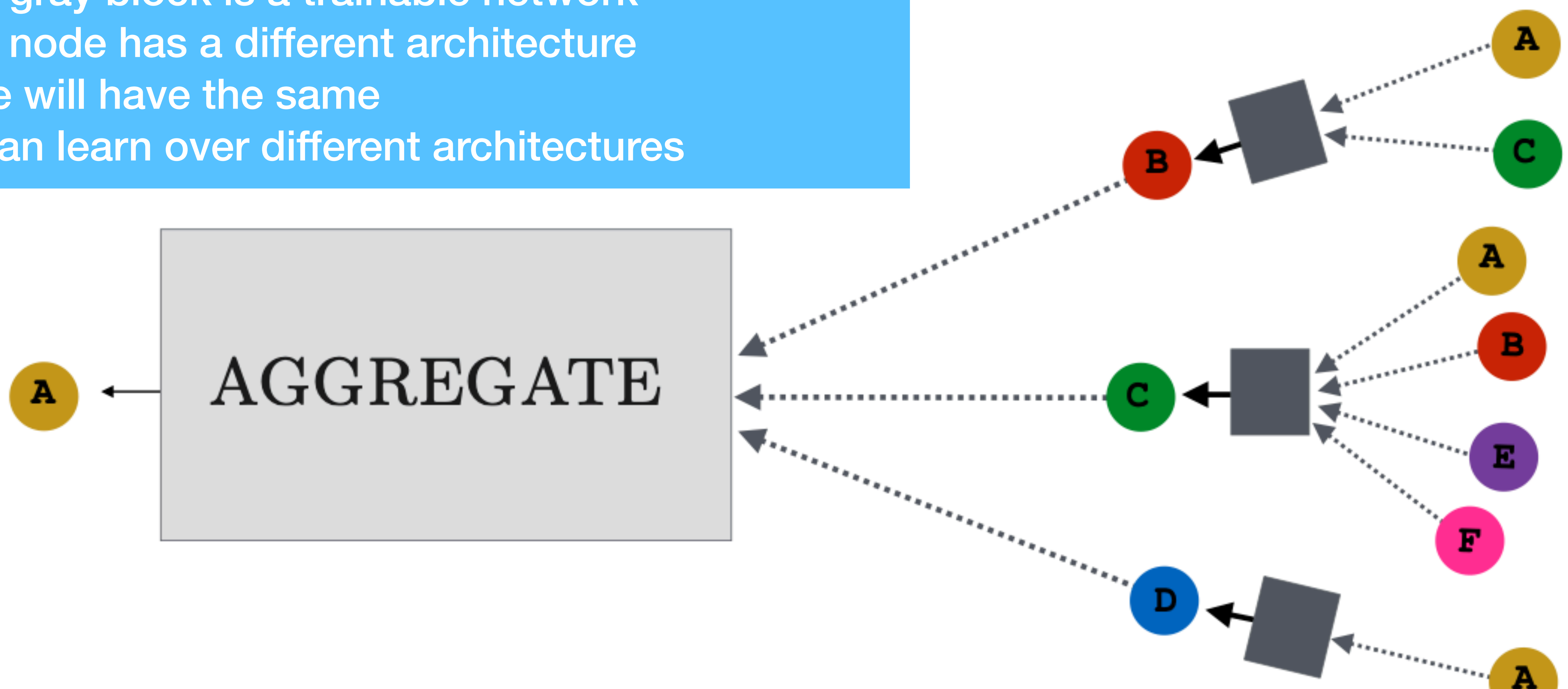


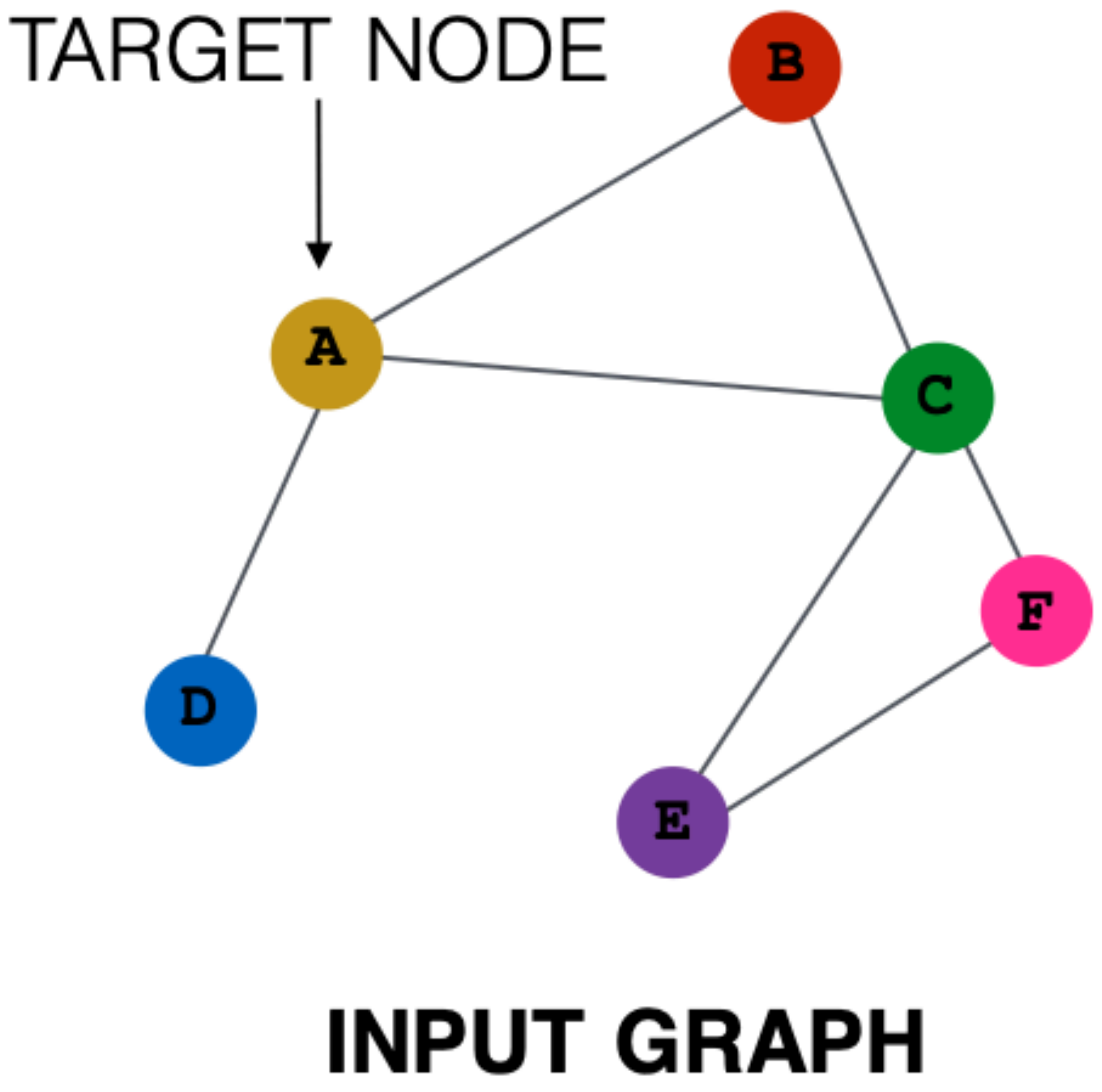
Image from Hamilton, "Graph Representation Learning Book"

Convolutions on Graphs: Message Passing

$$h_u^{(k+1)} = \text{update}^{(k)}\left(h_u^{(k)}, \text{agg}^{(k)}_{v \in \mathcal{N}(u)} h_v^{(k)}\right)$$

Main Points

- Each layer incorporates information from nodes k-hops away
- Neighbours need to be aggregated need to be permutation invariant
- Different aggregations define different networks



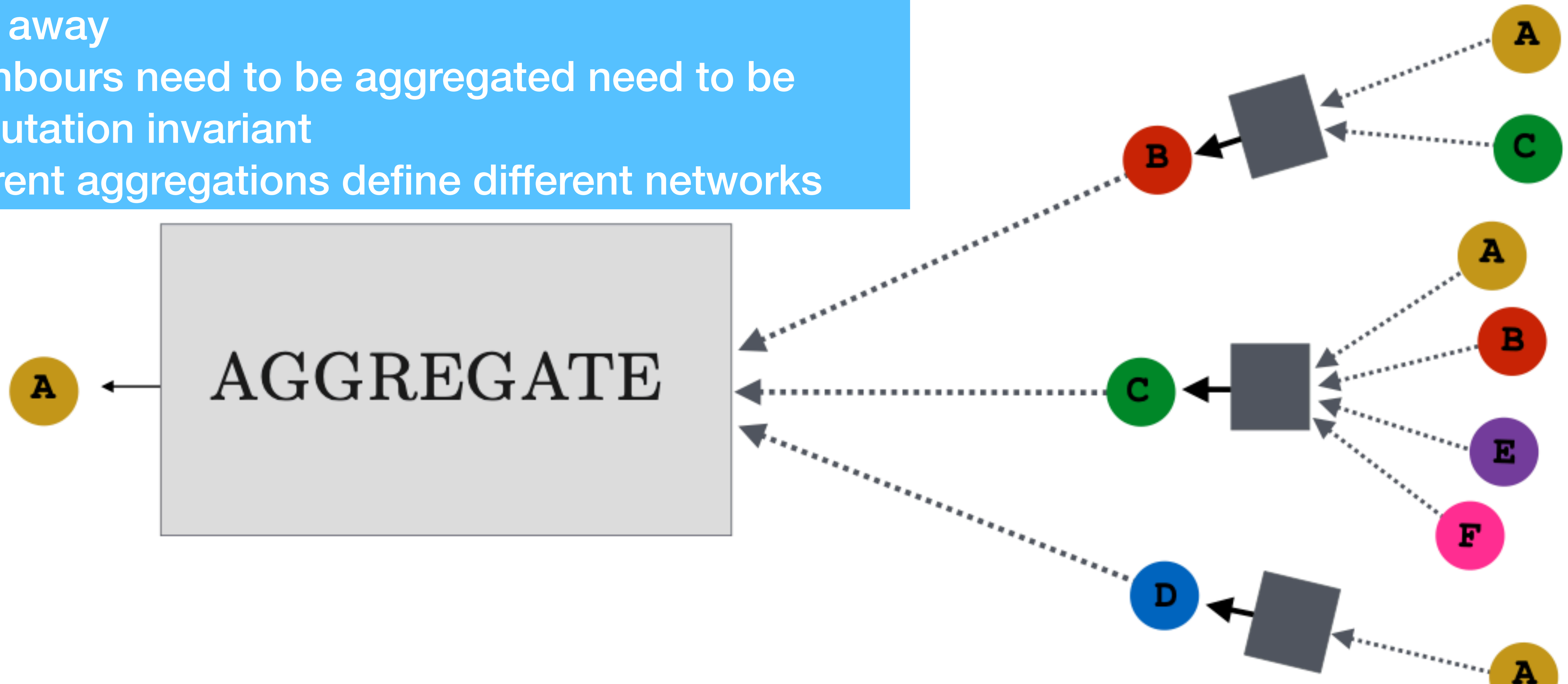


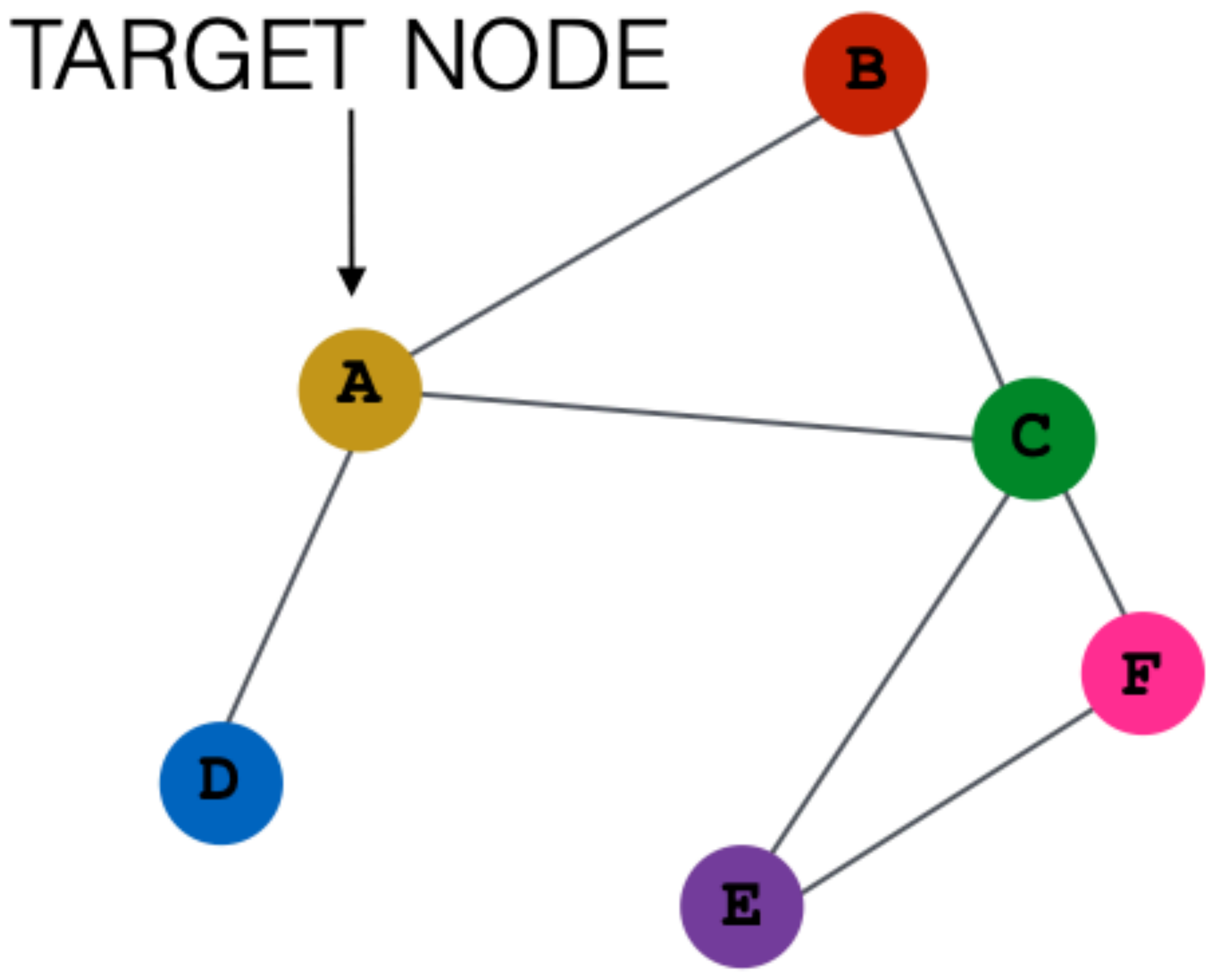
Image from Hamilton, "Graph Representation Learning Book"

Convolutions on Graphs: Message Passing

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

Main Points

- Each layer incorporates information from nodes k-hops away
- Neighbours need to be aggregated need to be permutation invariant
- Different aggregations define different networks



INPUT GRAPH

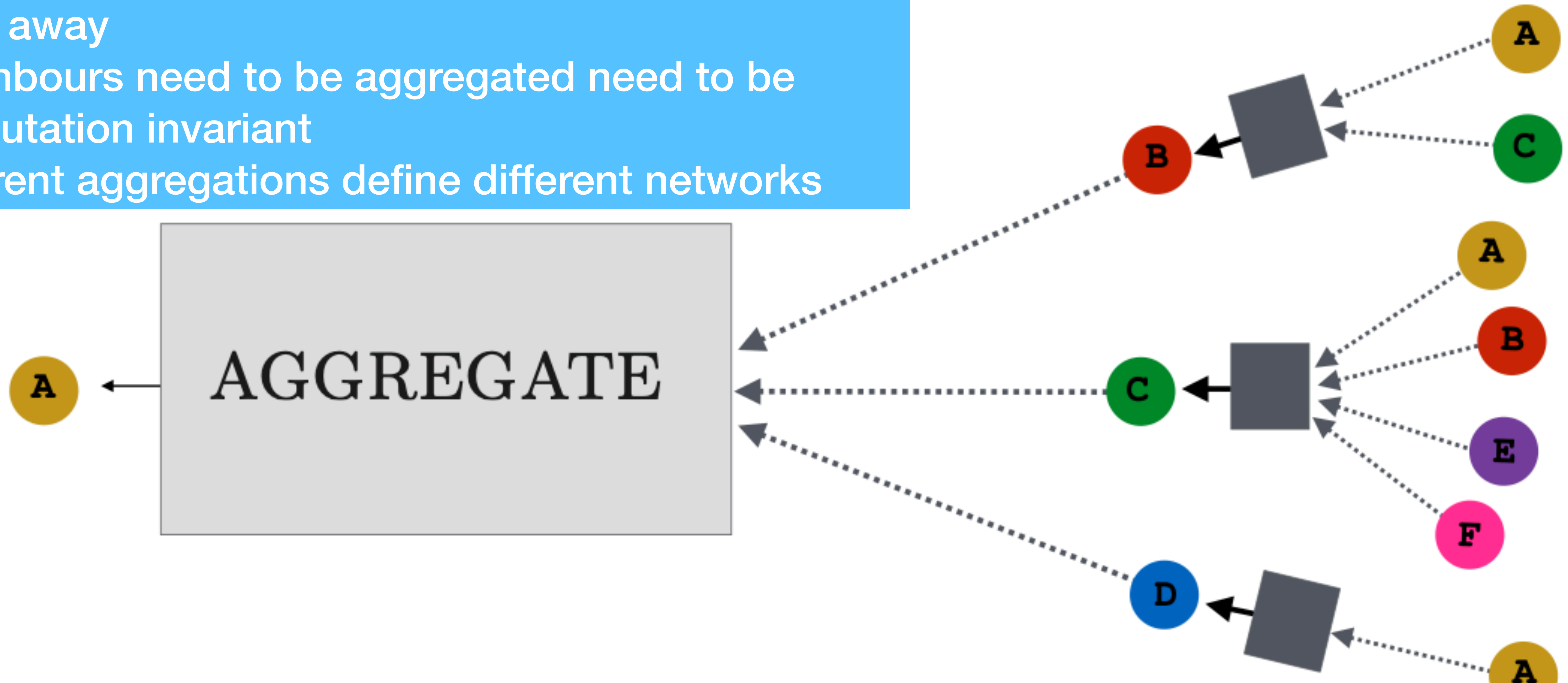


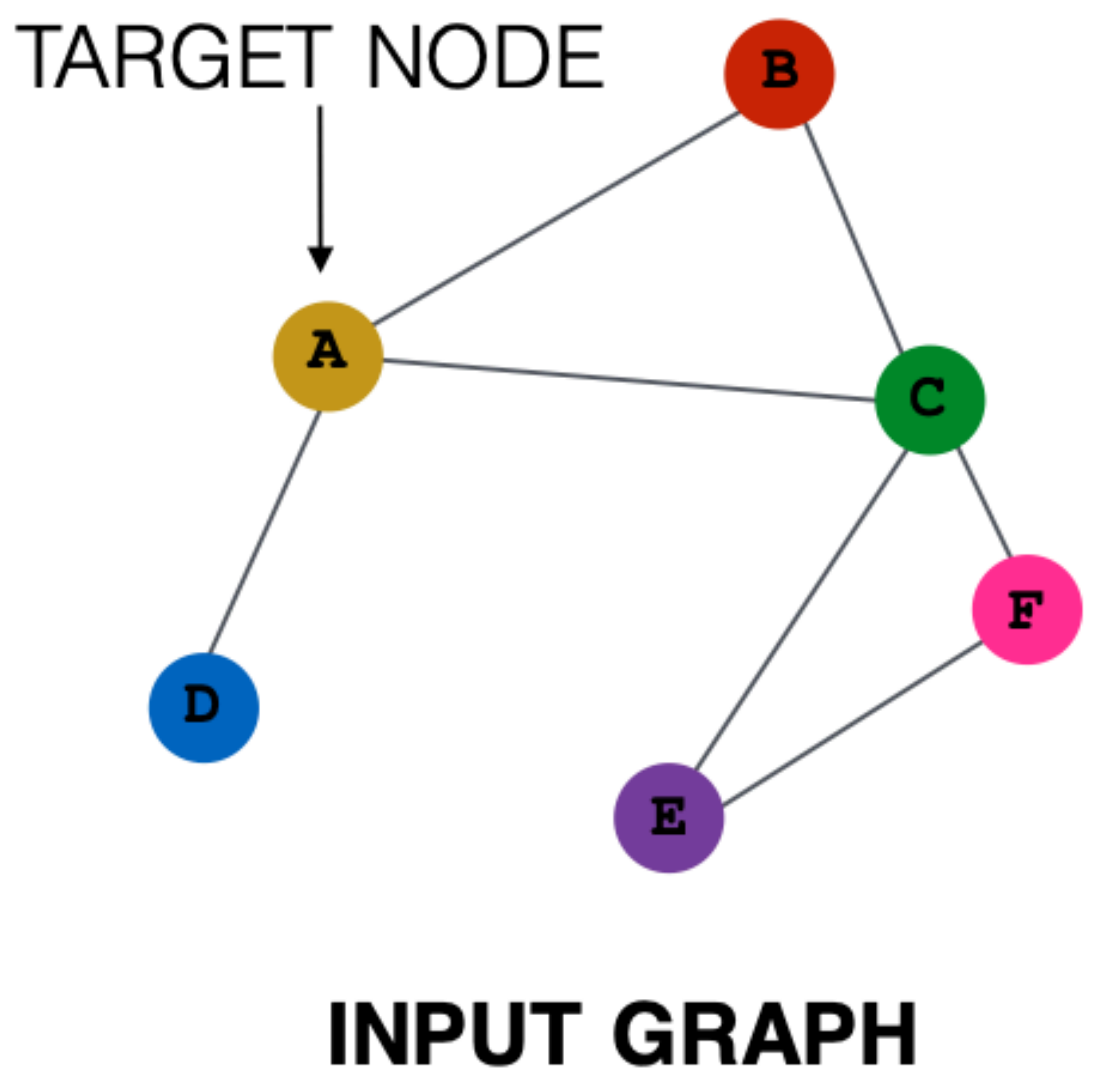
Image from Hamilton, "Graph Representation Learning Book"

Convolutions on Graphs: Deep Encoder

$$h_u^{(k+1)} = \sigma\left(W^{(k)} \sum_{v \in \mathcal{N}(u)} \frac{h_v^{(k)}}{|\mathcal{N}(u)|} + B^{(k)} h_u^{(k)} \right), h^0_{(u)} = x_v, h_u^{(L)} = z_v$$

Main Points

- Each layer incorporates information from nodes k-hops away
- Neighbours need to be aggregated need to be permutation invariant
- Different aggregations define different networks



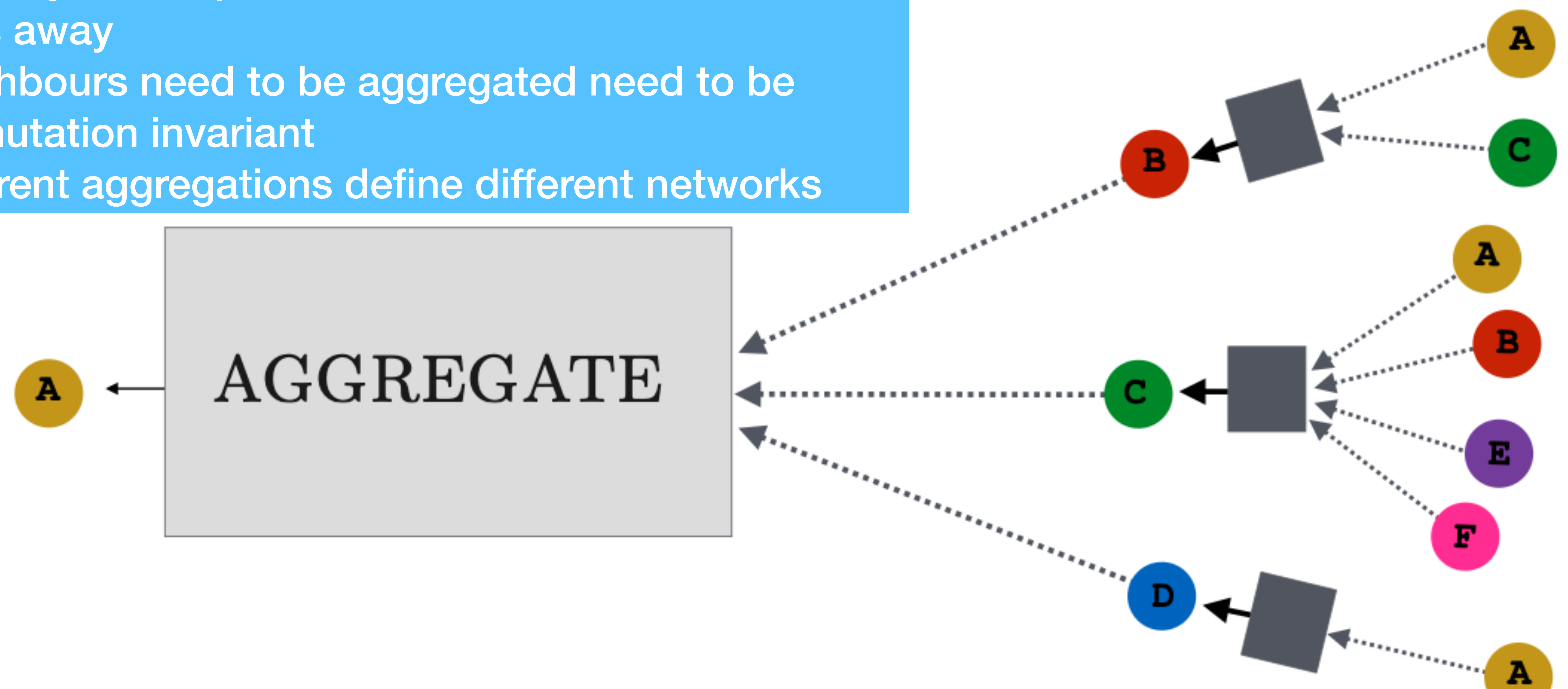


Image from Hamilton, “Graph Representation Learning Book”

Convolutions on Graphs: Deep Encoder

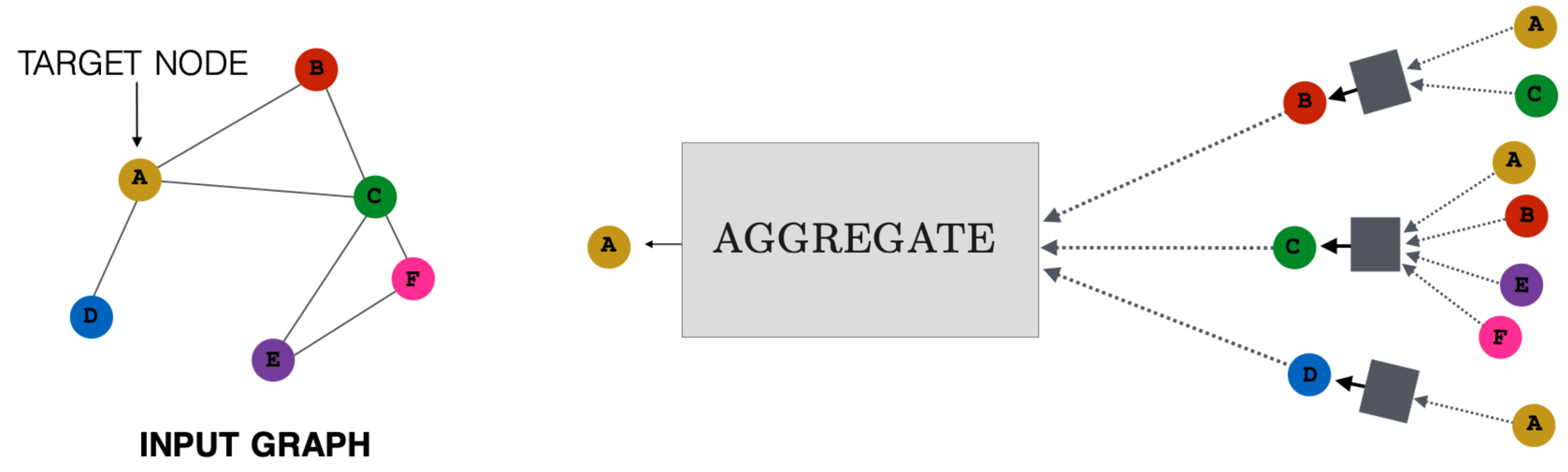
$$h_u^{(k+1)} = \sigma\left(W^{(k)} \sum_{v \in \mathcal{N}(u)} \frac{h_v^{(k)}}{|\mathcal{N}(u)|} + B^{(k)} h_u^{(k)}\right), h_{(u)}^0 = x_v, h_u^{(L)} = z_v$$

To train this model:

- Feed $W^{(k)}$ and $B^{(k)}$ to a loss and minimise with Stochastic gradient descent
- The matrices need to be shared across nodes
- In general, we can do this in matrix form $H^{(k)} = [h_i^{(k)}]_i$, define the diagonal degree matrix $D_{u,u} = Deg(u) = |\mathcal{N}(u)|$
- Then, $H^{(k+1)} = \sigma(D^{-1} A H^{(k)} W^{k^T} + H^{(k)} B^{k^T})$
- $D^{-1}A$ is sparse
- If the aggregation function is complex, the matrix formulation does not work

Convolutions on Graphs: Deep Encoder

$$H^{(k+1)} = \sigma(D^{-1}AH^{(k)}W^{k^T} + H^{(k)}B^{k^T}), h_u^{(L)} = z_v$$



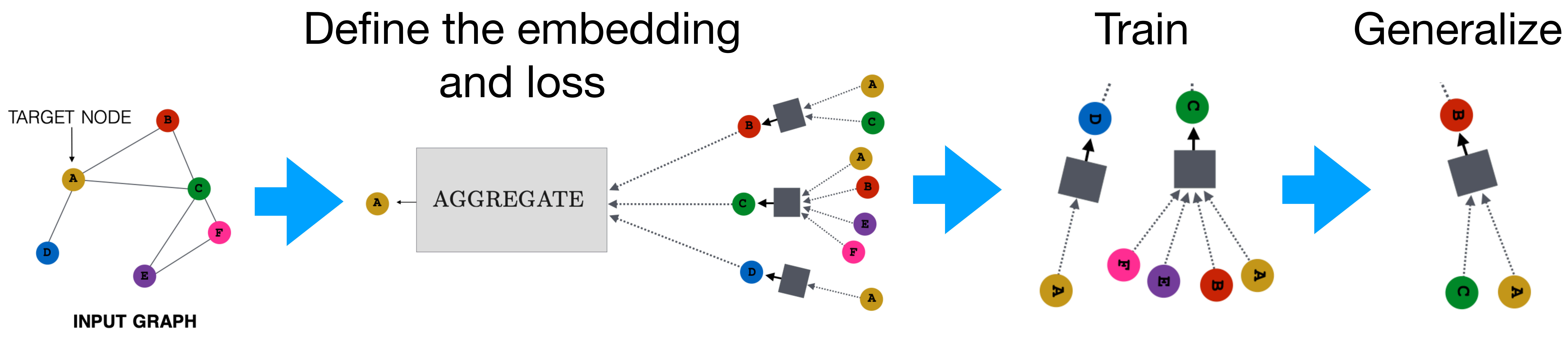
To train this model:

- Supervised learning $\arg\min_\theta \mathscr{L}(Y, f(z_v))$
  - Example for classification with cross entropy loss

$$\mathscr{L} = \sum_u y_u \log(\sigma(z_u^T\theta)) + (1 - y_u)\log(1 - \sigma(z_u^T\theta))$$

- For unsupervised learning $\mathscr{L} = sum_{u,v} CE(y_{u,v}, DEC(z_u, z_v))$ where $y_{u,v} = 1$ if nodes u and v are similar

Convolutions on Graphs: Deep Encoder Design



Design The model:

1. Define the embedding: define a neighbourhood aggregation function
2. Define a loss function on the embedding and batch-train
3. Train the model on batch-computed graphs. Which are selected from a node batch + subgraph
4. Generate embeddings for nodes. This is applicable to different graphs and nodes

Convolutions on Graphs: Results

Published as a conference paper at ICLR 2017

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Summary of results in terms of classification accuracy (in percent).

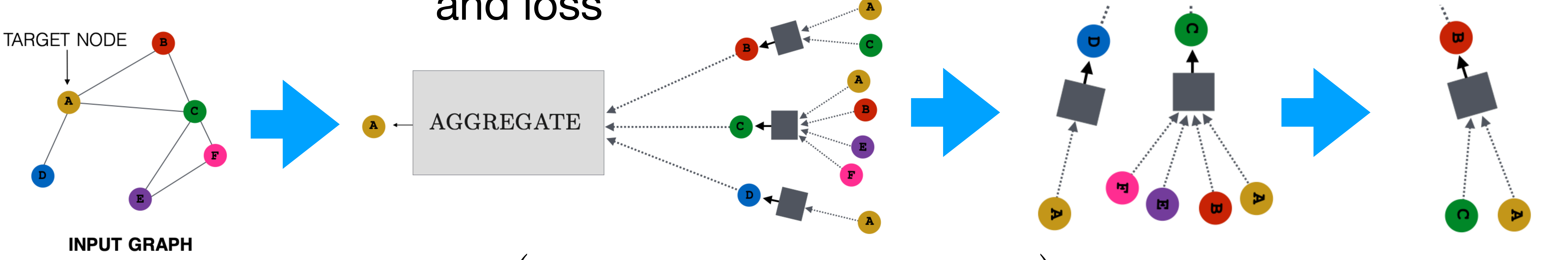
Method	**Citeseer**	**Cora**	**Pubmed**	**NELL**
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

Convolutions on Graphs: Deep Encoder General Case Across Different GNNs

Define the embedding and loss

Train

Generalize



$$h_u^{(k+1)} = \sigma\left(\mathrm{Agg}_{v\in\mathcal{N}(u)}(\mathrm{Msg}\,\phi(h_v^{(k)})), \psi(h_u^{(k)})\right), h_{(u)}^0 = x_v, h_u^{(L)} = z_v$$

Change over different GNNs

Convolutions on Graphs: Deep Encoder General Case Across Different GNNs

$$h_u^{(k+1)} = \sigma\left(\mathrm{Agg}_{v \in \mathcal{N}(u)}(\mathrm{Msg}\,\phi(h_v^{(k)})), \psi(h_u^{(k)})\right), h_{(u)}^0 = x_v, h_u^{(L)} = z_v$$

GCN

- σ is the sigmoid function
- Msg is the weighting of nodes $W^{(k+1)}h_v^{(k)}$
- Agg is the average
- ϕ is the identity function
- ψ is null

GraphSage

- σ is the sigmoid function
- Msg is the weighting of nodes $W^{(k+1)}h_v^{(k)}$
- Agg is two-stage
  - Aggregate from networks
  - Different aggregation with the node itself
- ϕ is the identity function
- ψ is null

Convolutions on Graphs: Deep Encoder General Case Across Different GNNs

$$h_u^{(k+1)} = \sigma\left(\mathrm{Agg}_{v\in\mathcal{N}(u)}(\mathrm{Msg}\,\phi(h_v^{(k)})), \psi(h_u^{(k)})\right), h_{(u)}^0 = x_v, h_u^{(L)} = z_v$$

GraphSage

- σ is the sigmoid function

  sometimes $\sigma(x) = \sigma\left(\frac{x}{\|x\|_2}\right)$
- Msg is the weighting of nodes $W^{(k+1)}h_v^{(k)}$
- Agg is two-stage
  - Aggregate from networks
  - Different aggregation with the node itself
- ϕ is the identity function
- ψ is null

GraphSage Aggregations

- Mean, like GCN
- Pool, by applying a non-identity ϕ
- LSTM aggregations

GraphSage: Results

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

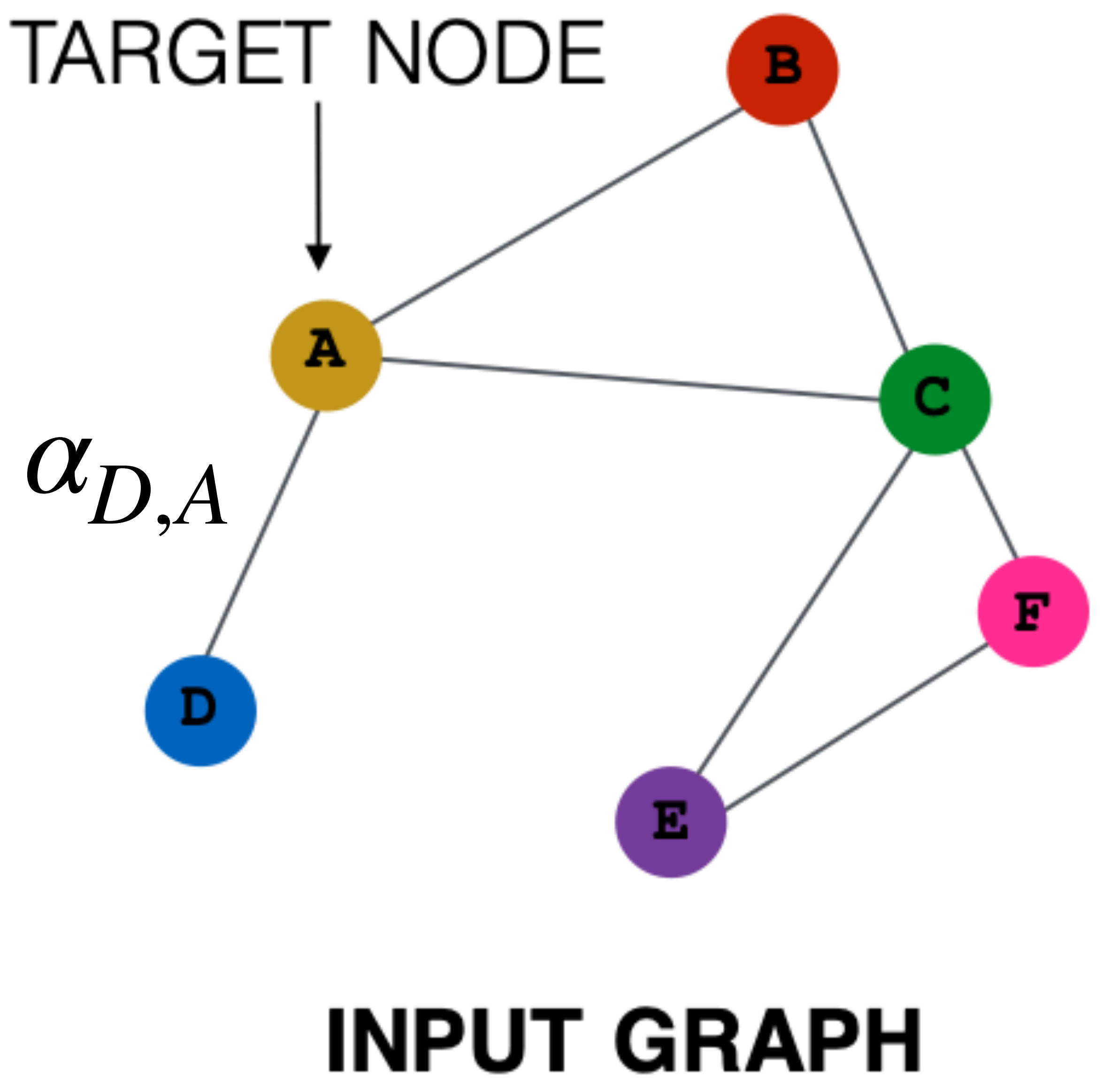
Department of Computer Science
Stanford University
Stanford, CA, 94305

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.

	Citation		Reddit		PPI	
Name	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%

Convolutions on Graphs: Deep Encoder General Case Across Different GNNs

$$h_u^{(k+1)} = \sigma\left(\mathrm{Agg}_{v\in\mathcal{N}(u)}(\mathrm{Msg}\,\phi(h_v^{(k)})), \psi(h_u^{(k)})\right), h_{(u)}^0 = x_v, h_u^{(L)} = z_v$$



Graph Attention Network (GAT)

- Most parameters are arbitrary
- Msg is the weighting of nodes $\alpha_{uv} W^{(k+1)} h_v^{(k)}$
- this learnable weighting $\alpha_{u,v}$ will learn which nodes are more important for any give node embedding.

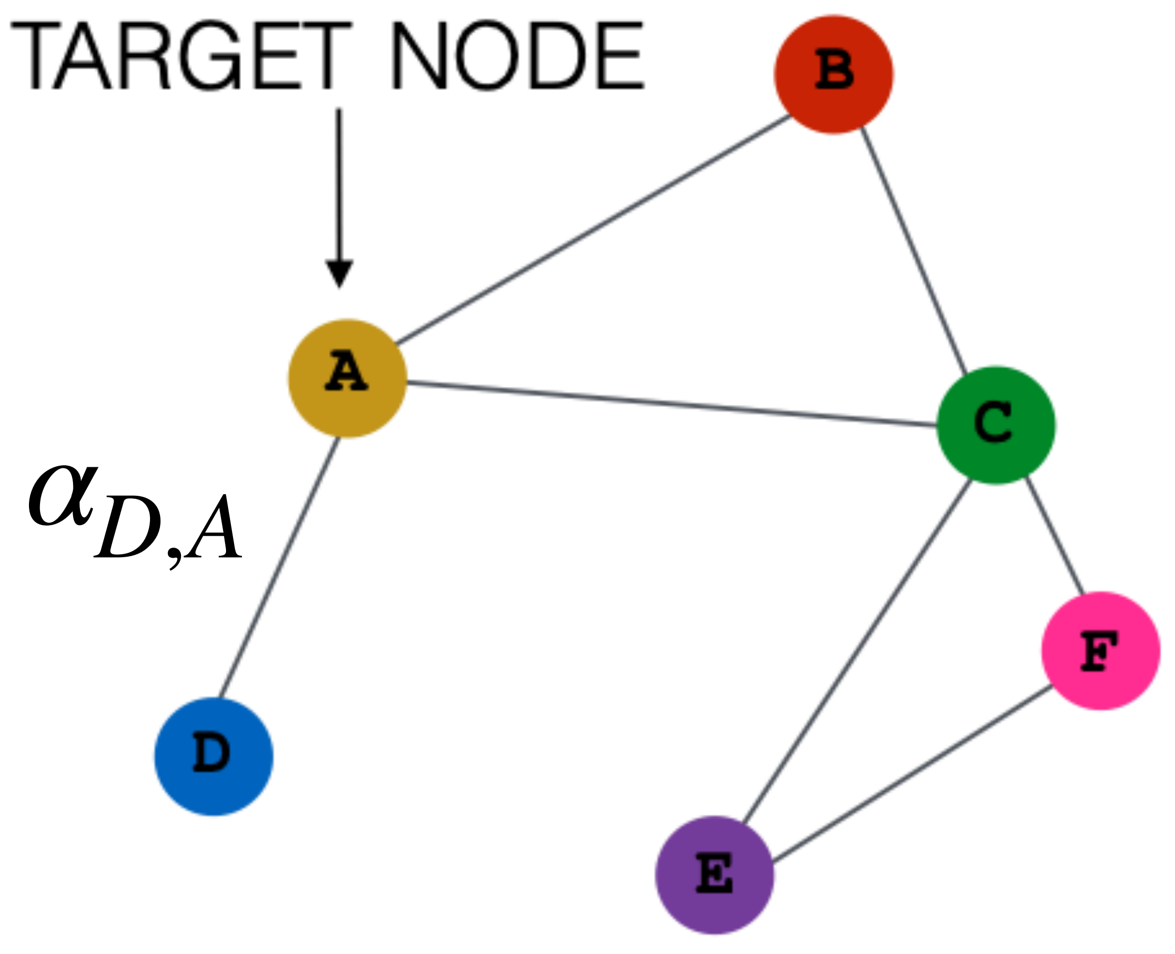
$$h_u^{(k+1)} = \sigma\left(\sum_{v\in\mathcal{N}(u)} \alpha_{u,v} W^{(k)} \frac{h_v^{(k)}}{|\mathcal{N}(u)|} + B^{(k)} h_u^{(k)}\right), h_{(u)}^0 = x_v, h_u^{(L)} = z_v, \alpha_{u,\cdot} = \sum_{v\in\mathcal{N}(v)} \alpha_{u,v} = 1$$

Convolutions on Graphs: Deep Encoder General Case Across Different GNNs

$$h_u^{(k+1)} = \sigma\left(\mathrm{Agg}_{v\in\mathcal{N}(u)}(\mathrm{Msg}\,\phi(h_v^{(k)})), \psi(h_u^{(k)})\right), h_{(u)}^0 = x_v, h_u^{(L)} = z_v$$

Graph Attention Network (GAT)

- Main Benefits:
  - Implicit importance of neighbours
  - Computationally efficient
  - Storage efficient O(V+E) entries and fixed
  - Localised
  - Inductive, it doesn't depend on the graph structure



INPUT GRAPH

Graphic Attention Network Results

Transductive

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

Published as a conference paper at ICLR 2018

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Inductive

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	$\mathbf{0.973} \pm 0.002$

Summarising

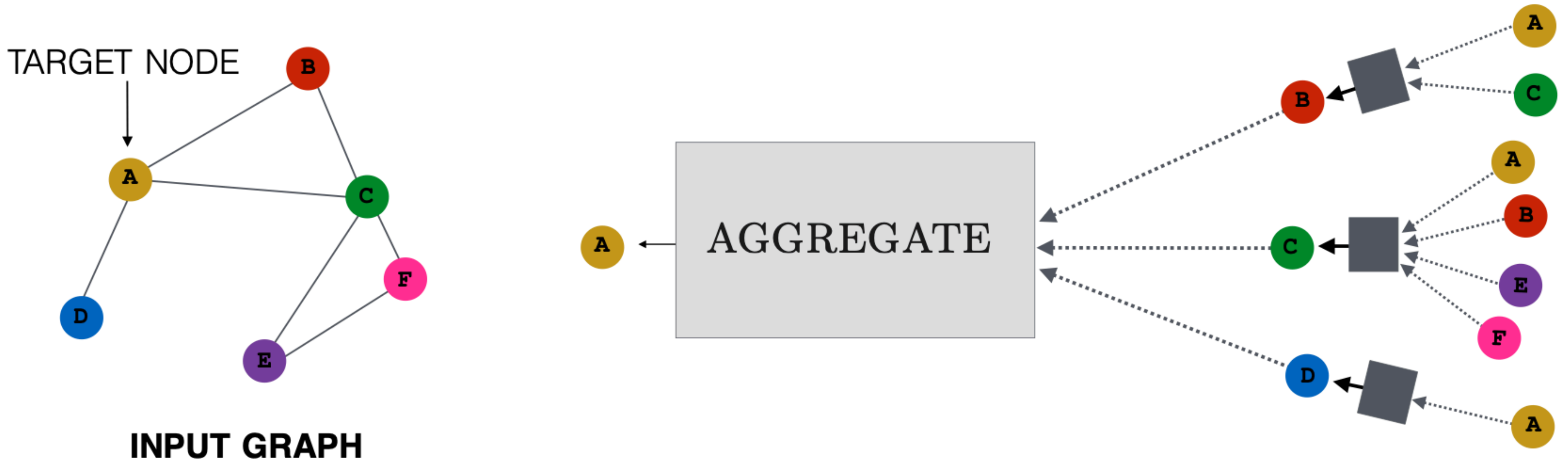


Image from Hamilton, “Graph Representation Learning Book”

- Graphs are good representations of data support and relationships
- Going from grid/lattices to graphs in non-trivial, we need permutation invariance, scale invariance, and sometimes topological invariance
- The main trick, is detect patches or motifs and generalise them to global structure
- There’s ample evidence that taking into account heterogeneous structure improves supervised/semi-supervised tasks
- Novel techniques in graph networks, like GCNN, GraphSage, GAT, and others clearly improve on the results that don’t take into account structure