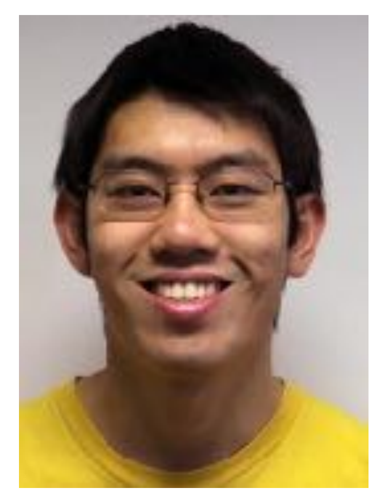
Covariant Neural Networks for Learning Graphs (and other things)

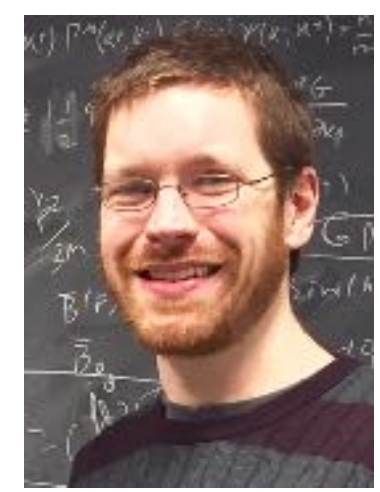
Risi Kondor
The University of Chicago

Shubhendu Trivedi

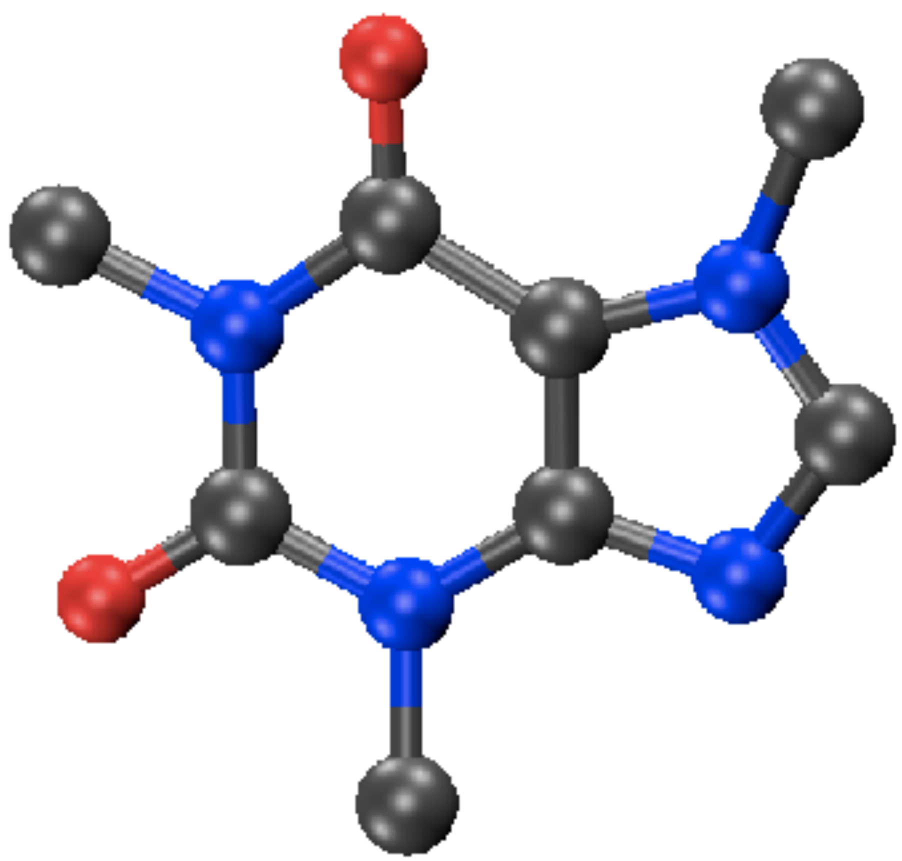
Hy Trong Son



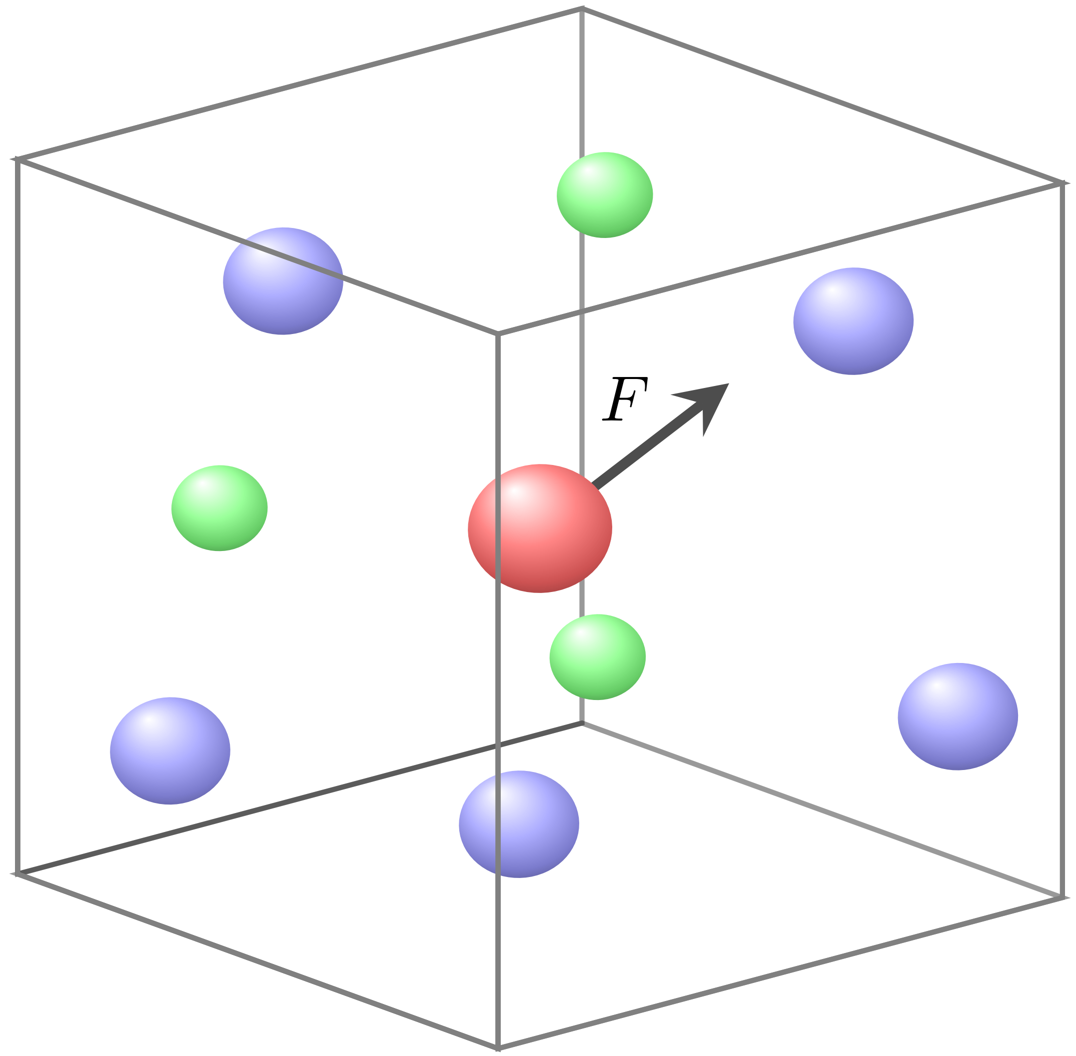
Horace Pan



Brandon Anderson



$$\mapsto \quad \phi(G)$$

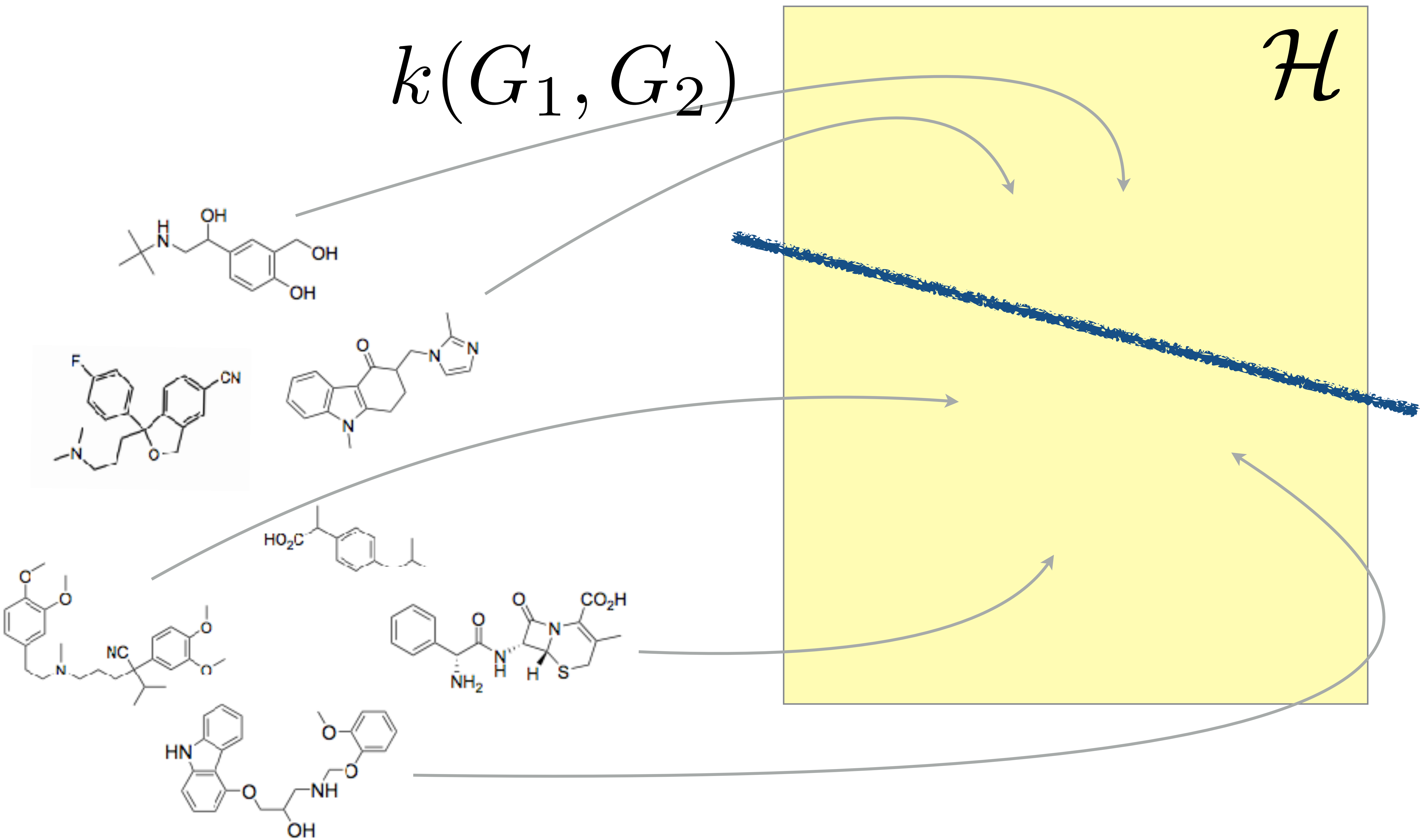


$$F(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_m)$$

[Ferre, Maillet & Stoltz, 2015]

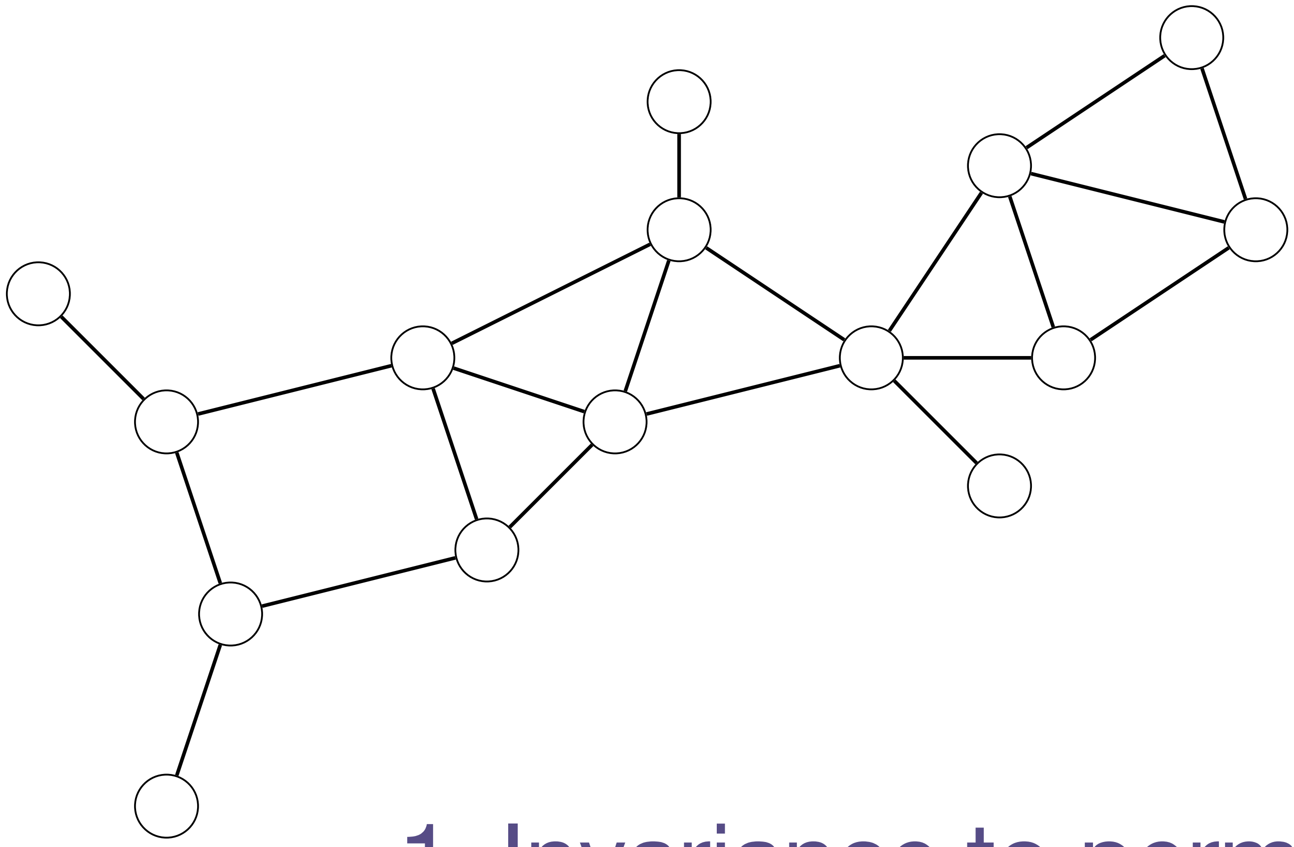
Learning graphs

Kernel approach



1. Random walks and spectral ideas
   [Gartner, 2002] [Vishwanathan et al., 2010]
2. Shortest Paths
   [Borgwardt & Kriegel, 2005]
3. Counting subgrapgs
   [Shervashidze et al., 2009] [Feragen et al., 2013]
3. Algebraic approach
   [K. & Borgwardt, 2008]
4. Label Propagation
   [Shervashidze et al., 2009]
5. Hierarchical
   [K.& Pan, 2016]

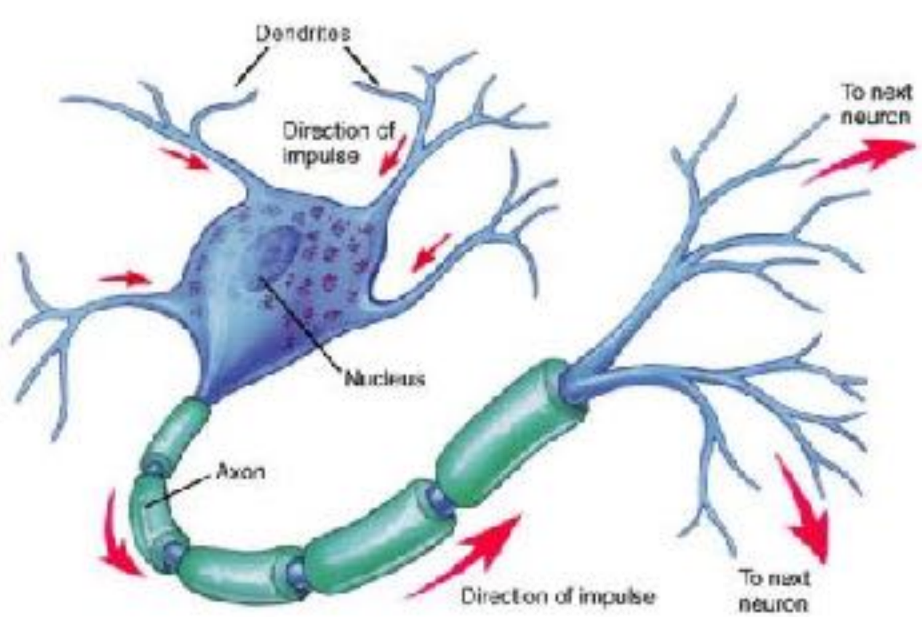
The kernel approach is an inherently fixed representation.

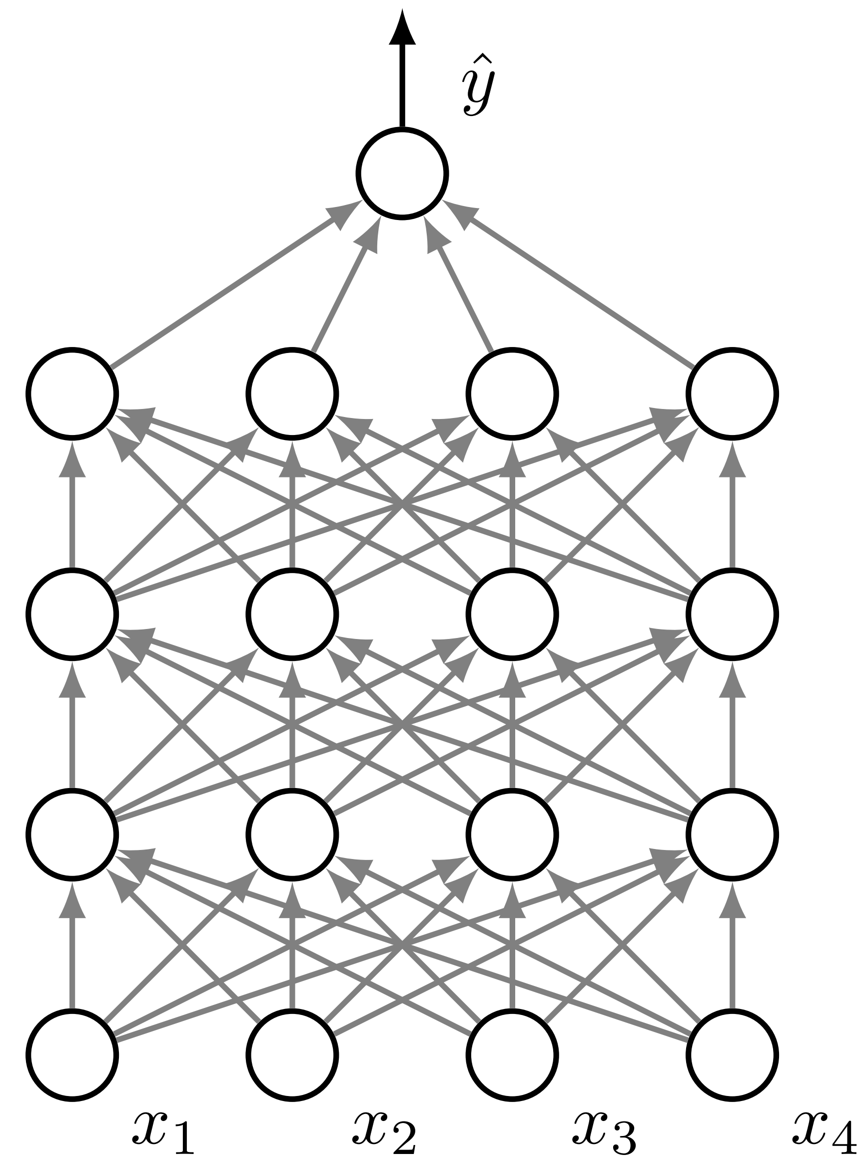


1. Invariance to permutations of vertices
2. Ability to capture structure at multiple scales

Feed-forward neural networks

$$f_{\text{out}} = \xi\Big(\sum_i w_i\, f_{\text{in}}^{(i)} + b\Big)$$



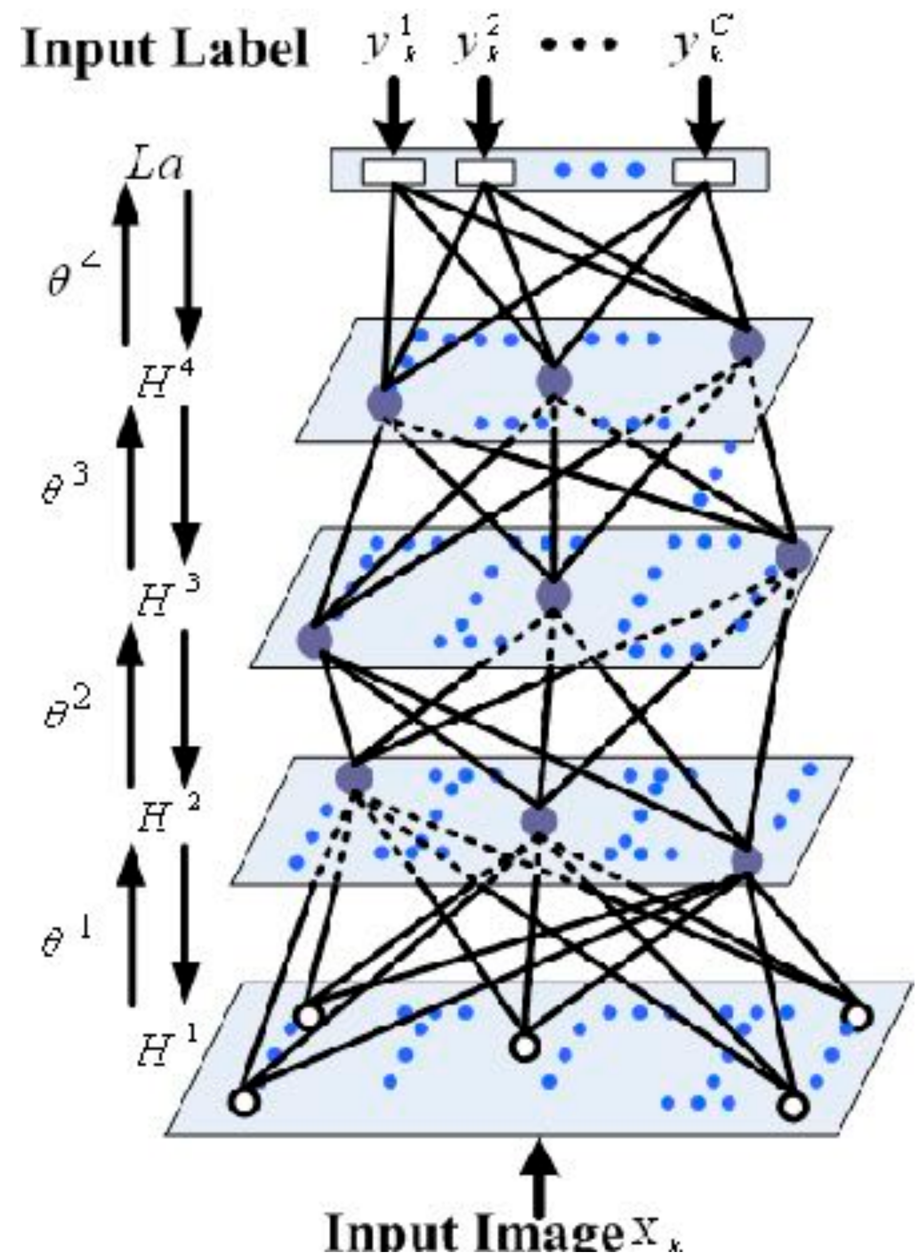


[McCullogh & Pitts, 1943] [Rumelhart, Hinton & Williams, 1986]
[Hinton, 2006]

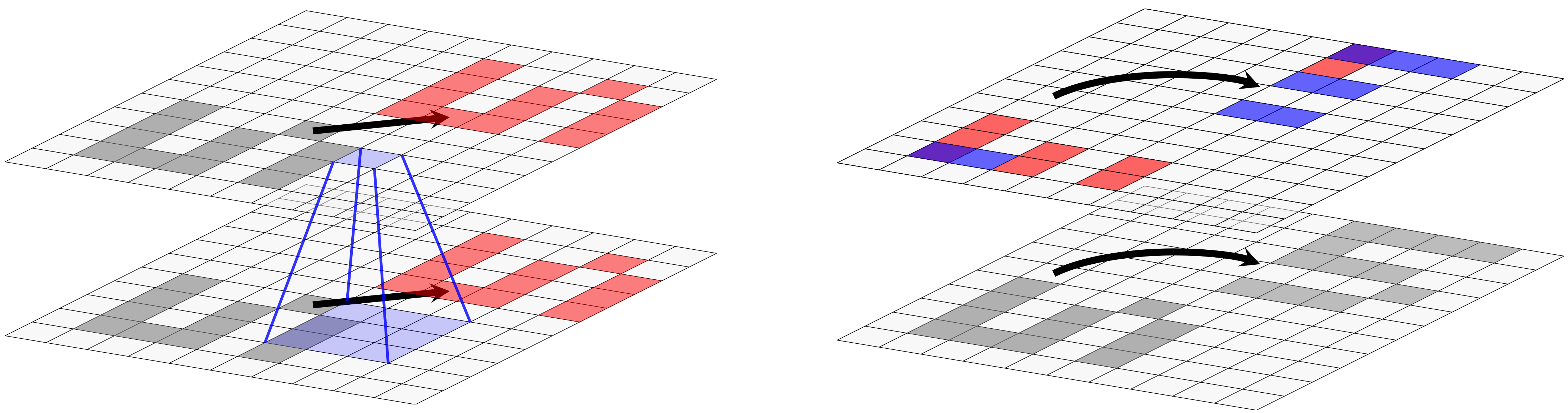
Convolutional Neural Networks

$$(f * g)(x) = \int f(x-y)\, g(y)\, dy$$

$$f_\ell(\mathbf{x}) = \xi\Big(\sum_{\mathbf{y}} f_\ell(\mathbf{x}-\mathbf{y})\, \chi_\ell(\mathbf{y})\Big)$$



Invariance and covariance (steerability)



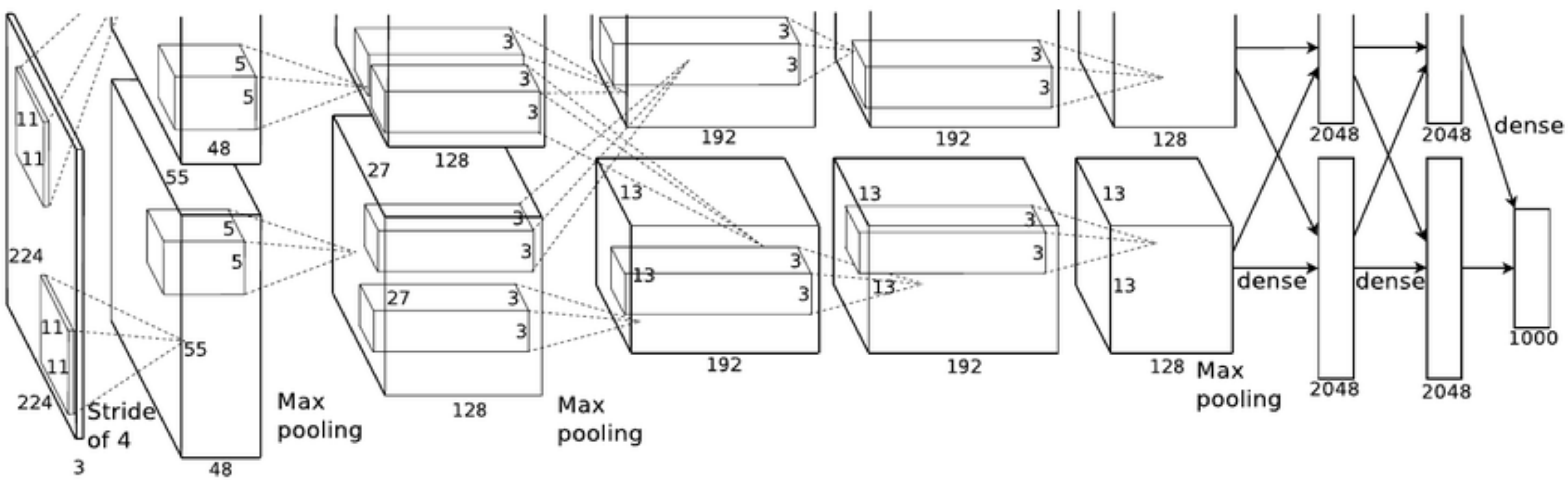
$$f'_\ell(T(\mathbf{x})) = f_\ell(\mathbf{x})$$

$$f'_\ell(T(\mathbf{x})) = R_T(f_\ell(\mathbf{x}))$$

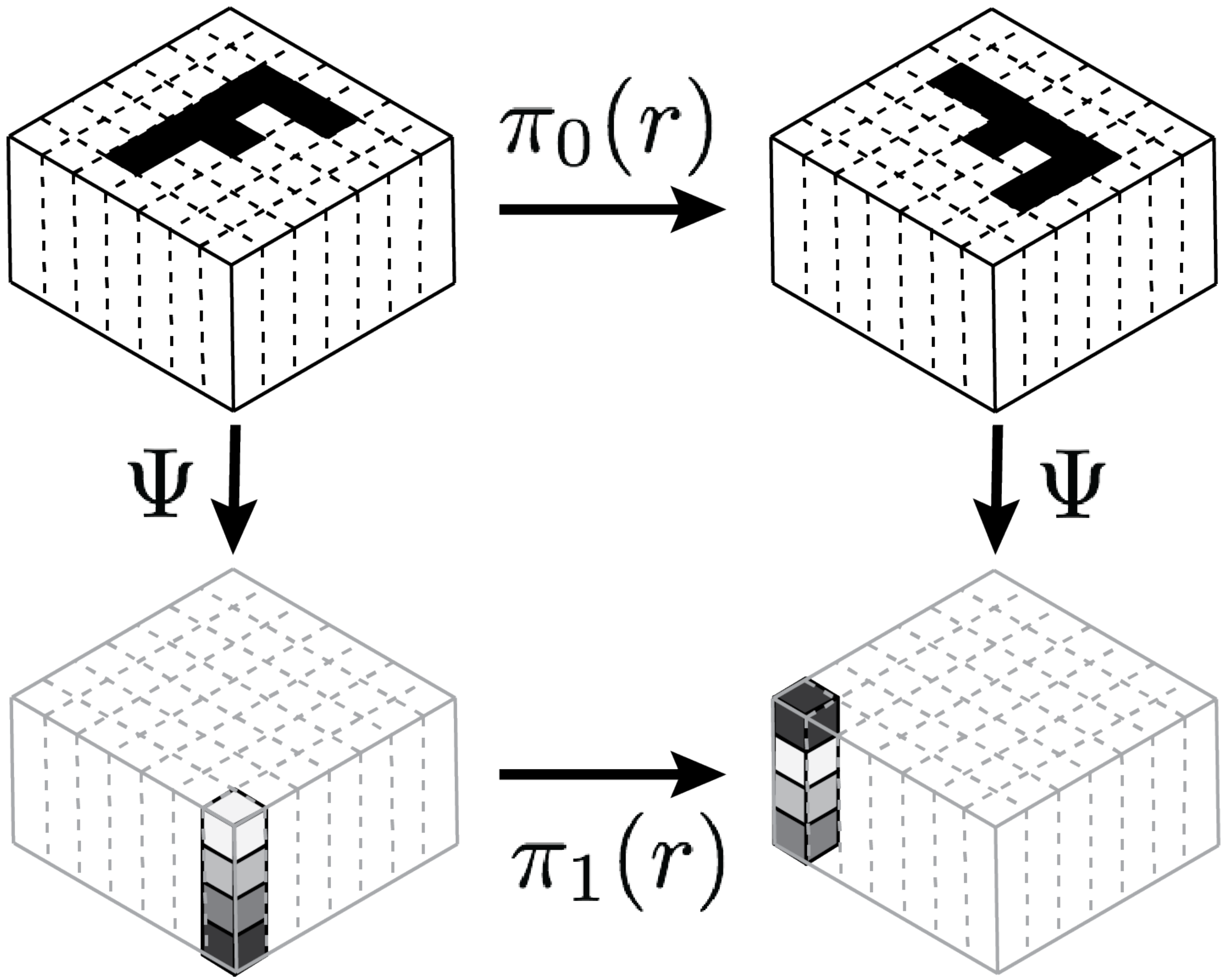
Equivariance to translations

$$\begin{tikzcd}
S_1 \arrow[r, "T_g"] \arrow[d, "\phi"'] & S_1 \arrow[d, "\phi"'] \\
S_2 \arrow[r, "T'_g"] & S_2
\end{tikzcd}$$

Alexnet

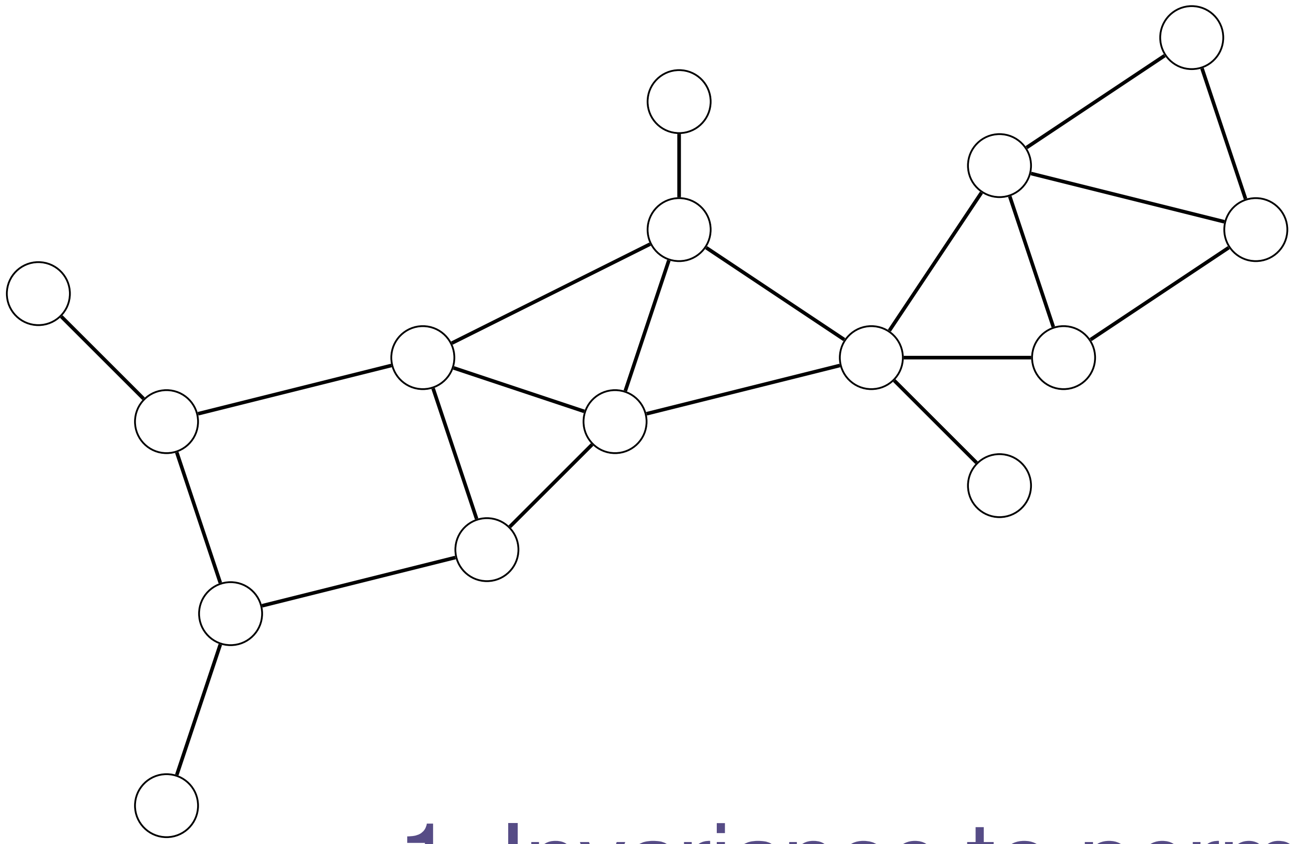


[Krizhevsky, Sutskever & Hinton, 2012]



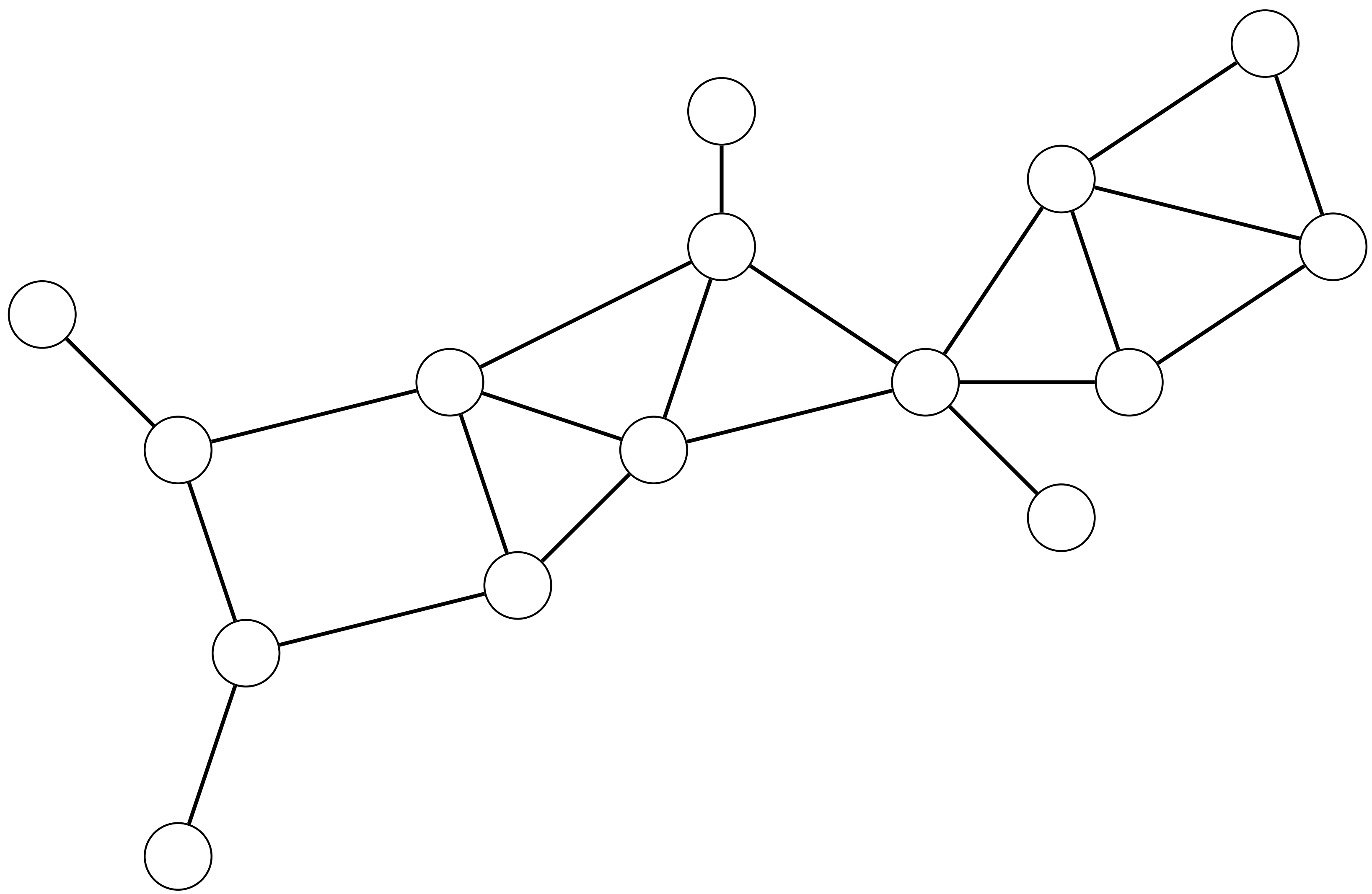
[Cohen & Welling, 2016]

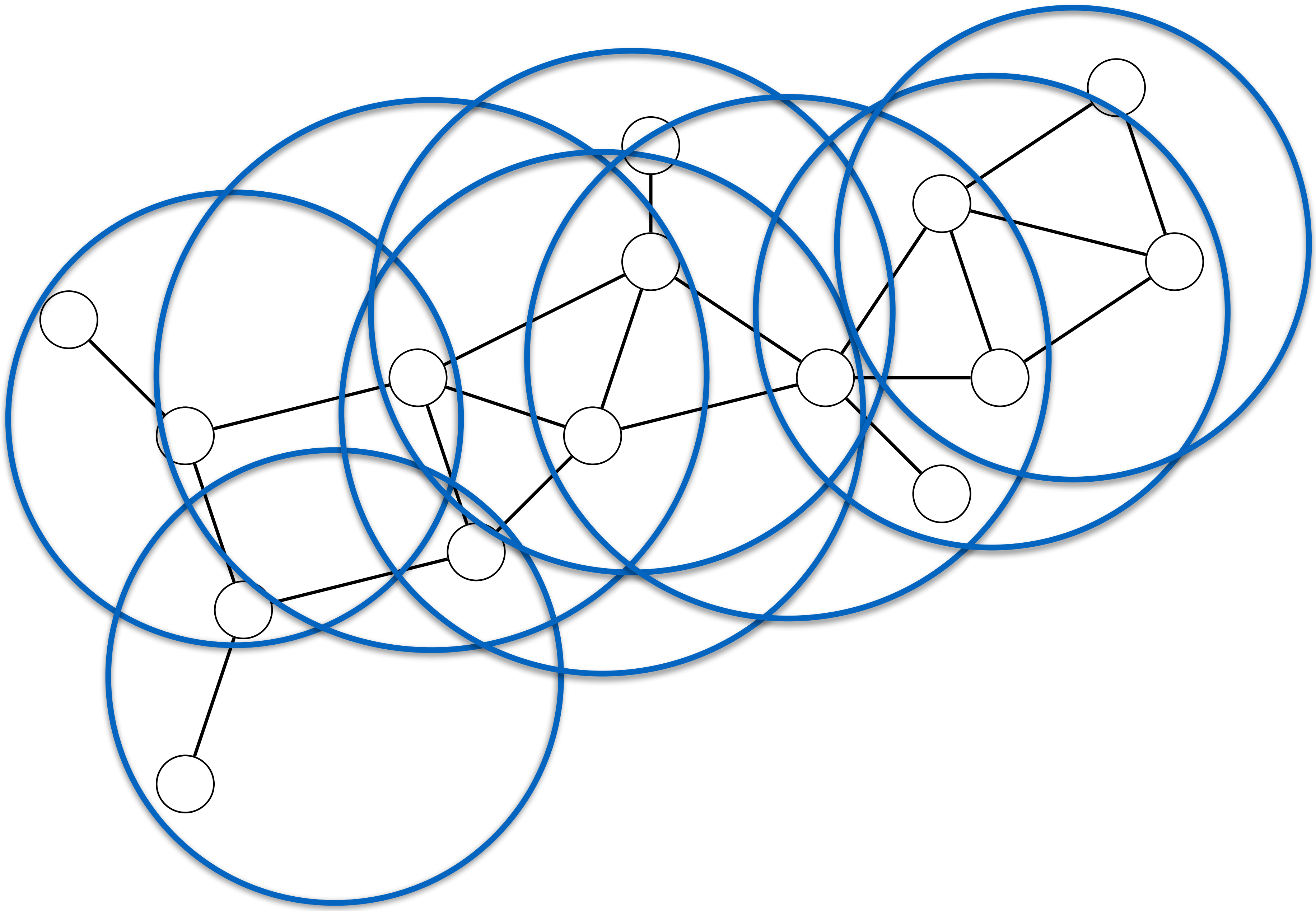
What is the analog of convolutions on graphs?

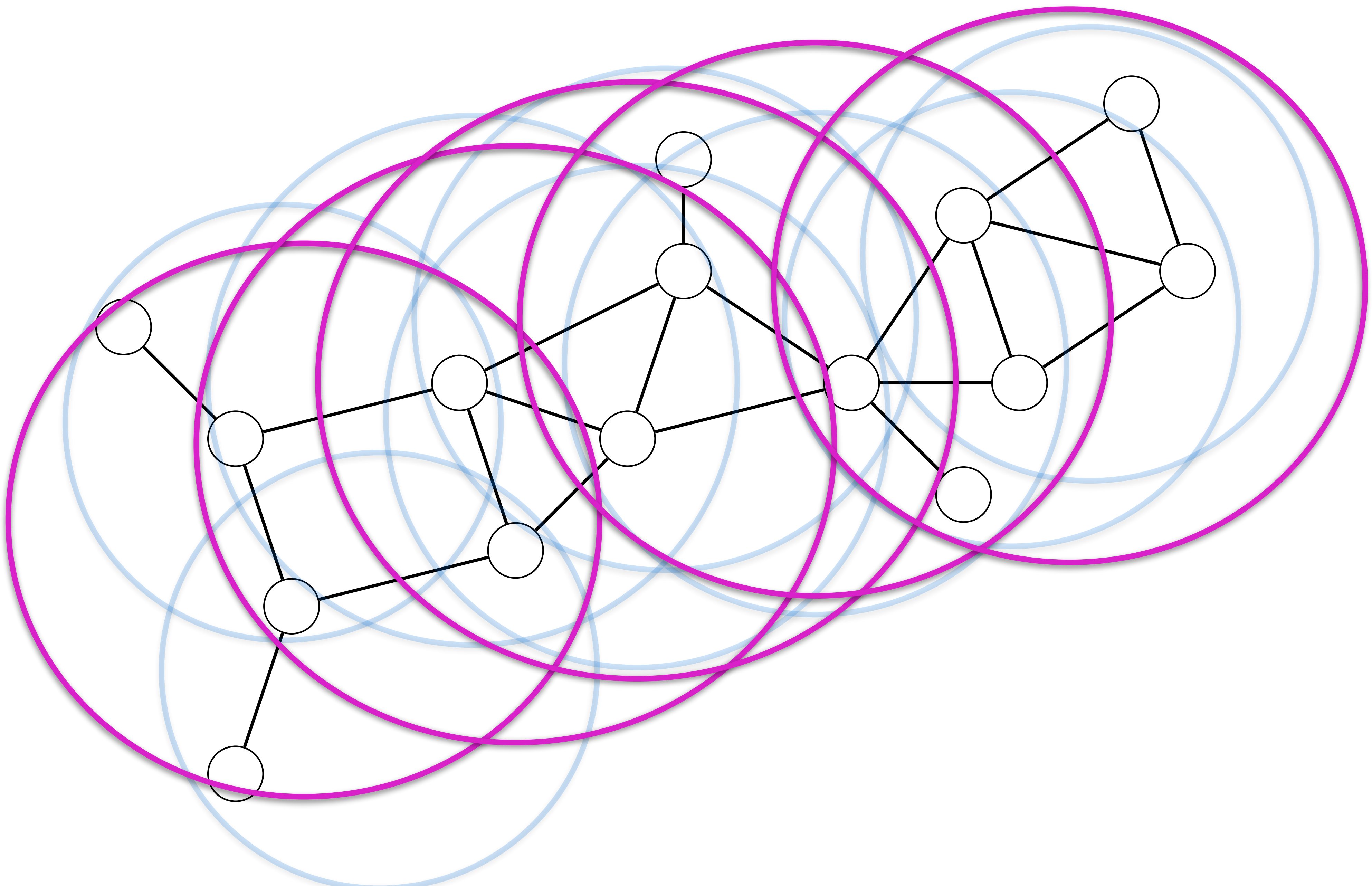


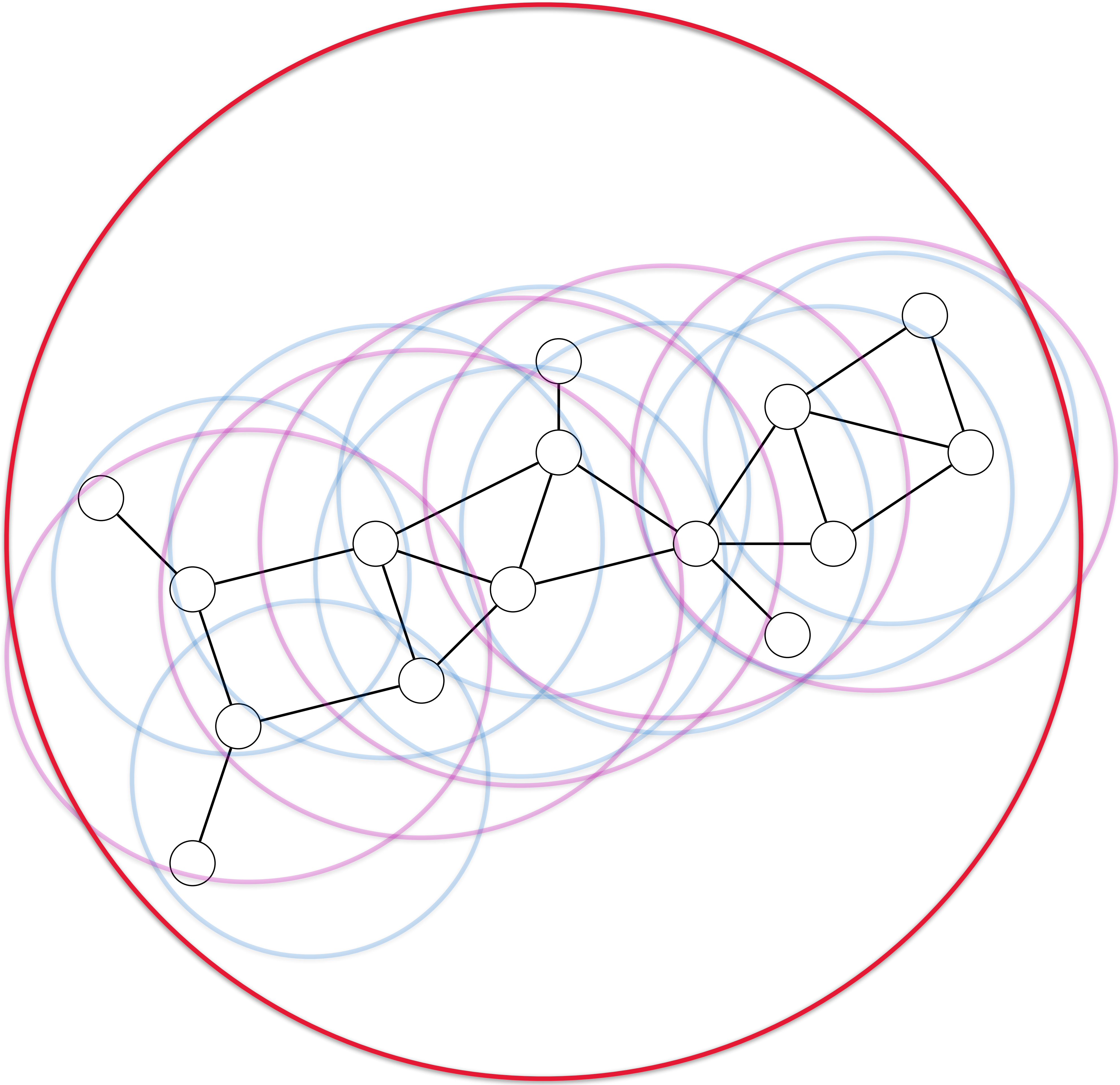
1. Invariance to permutations of vertices
2. Ability to capture structure at multiple scales

Compositional approach

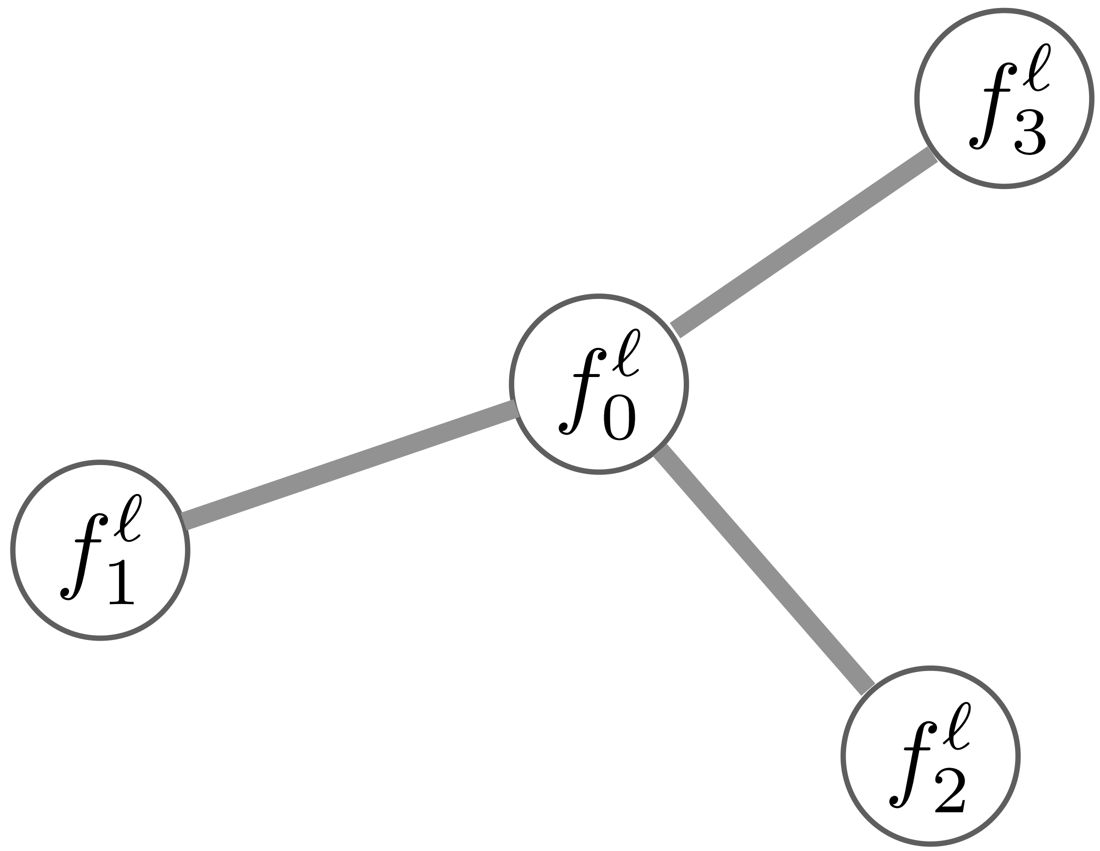




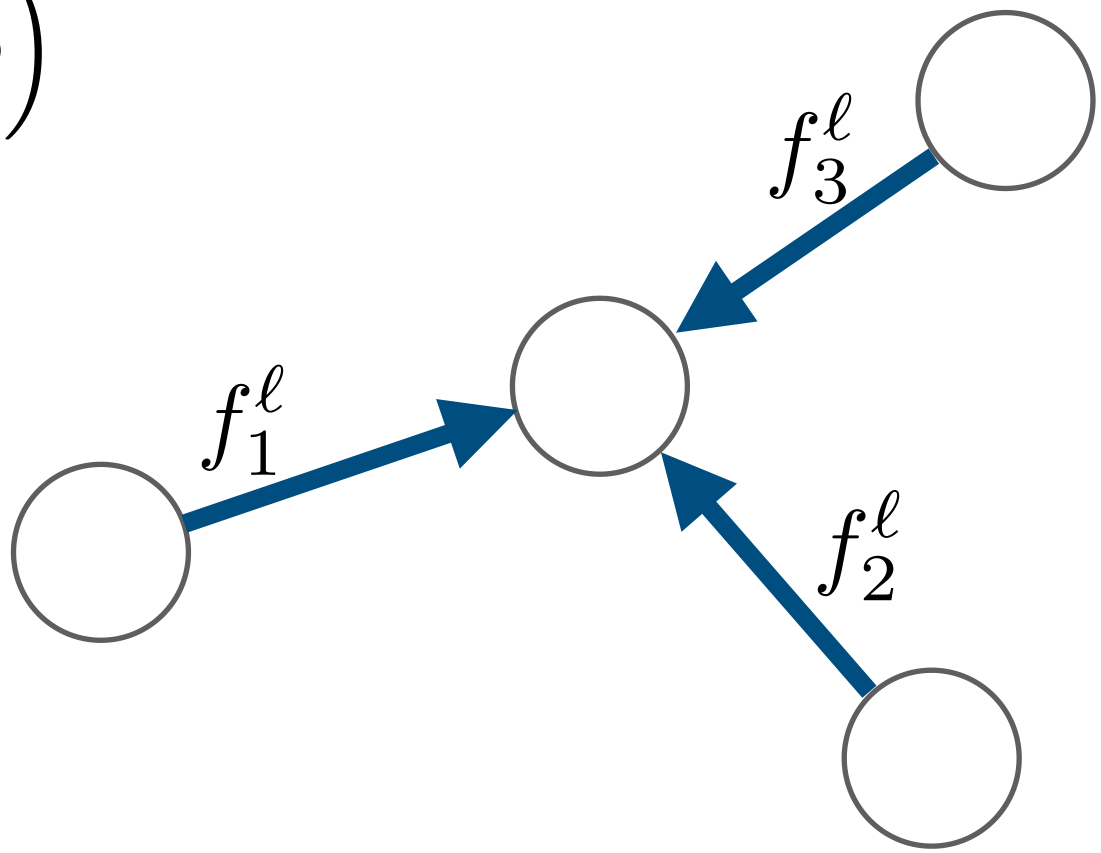




Label propagation schemes

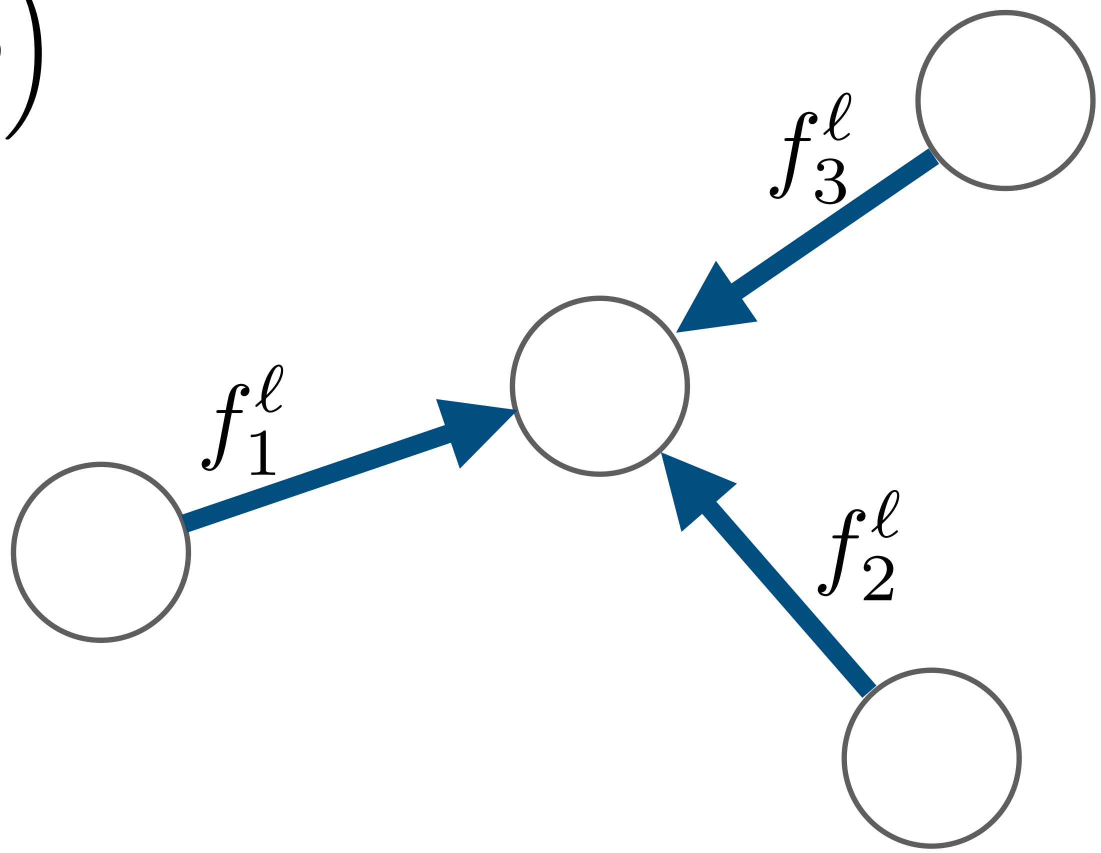


$$f_i^{\ell+1} = \xi\Big(W \sum_{j \in \mathcal{N}(i)} f_j^\ell + b\Big)$$

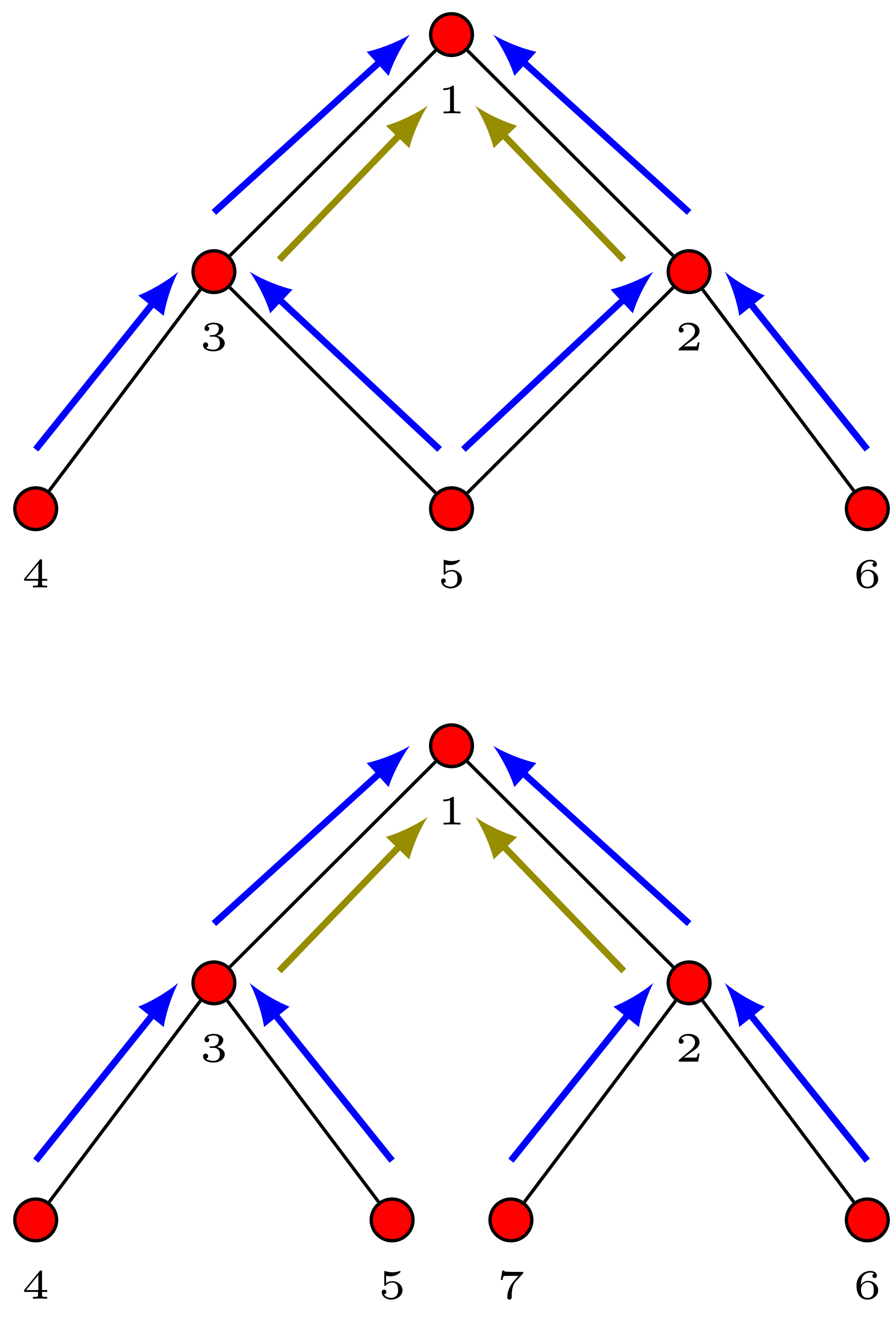


[Gilmer et al, ’17] [Kriege, ’16] [Niepert, ’16] [Duvenaud et al., ’15]
[Dai, Dai & Song, ’16]

$$f_i^{\ell+1} = \xi\Big(W \sum_{j \in \mathcal{N}(i)} f_j^{\ell} + b\Big)$$



1. Satisfies permutation invariance
2. Aggregates information at multiple different scales
3. Does not fully account for topology

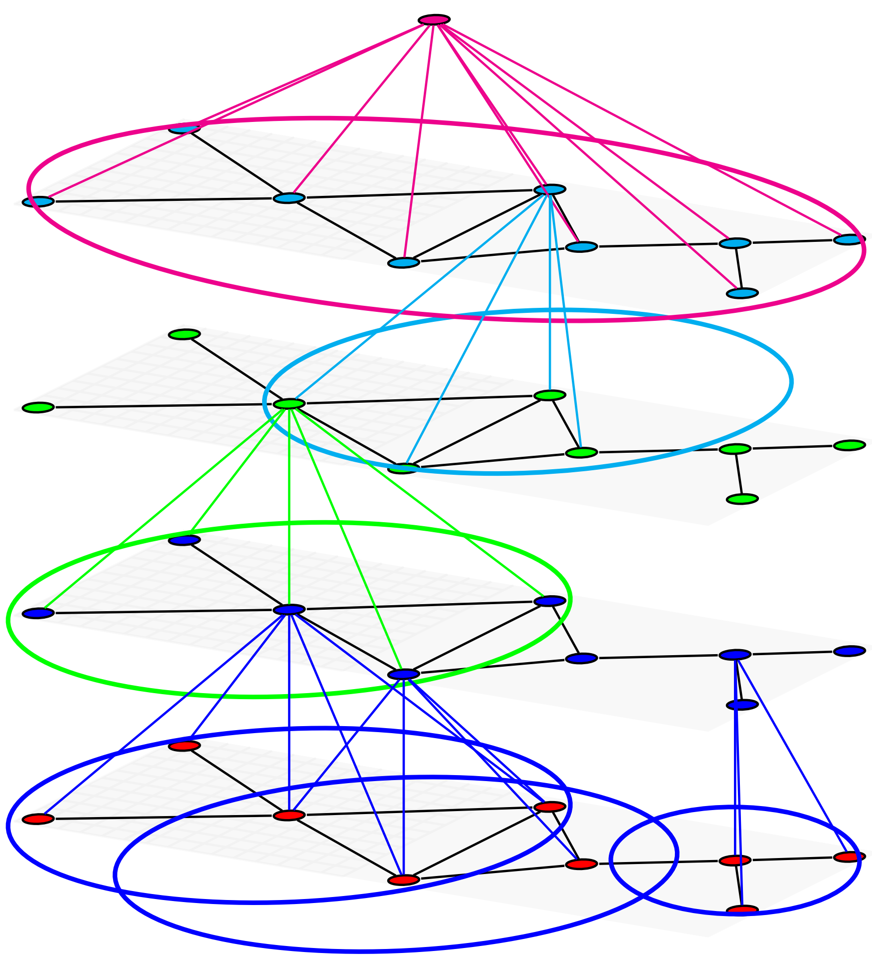


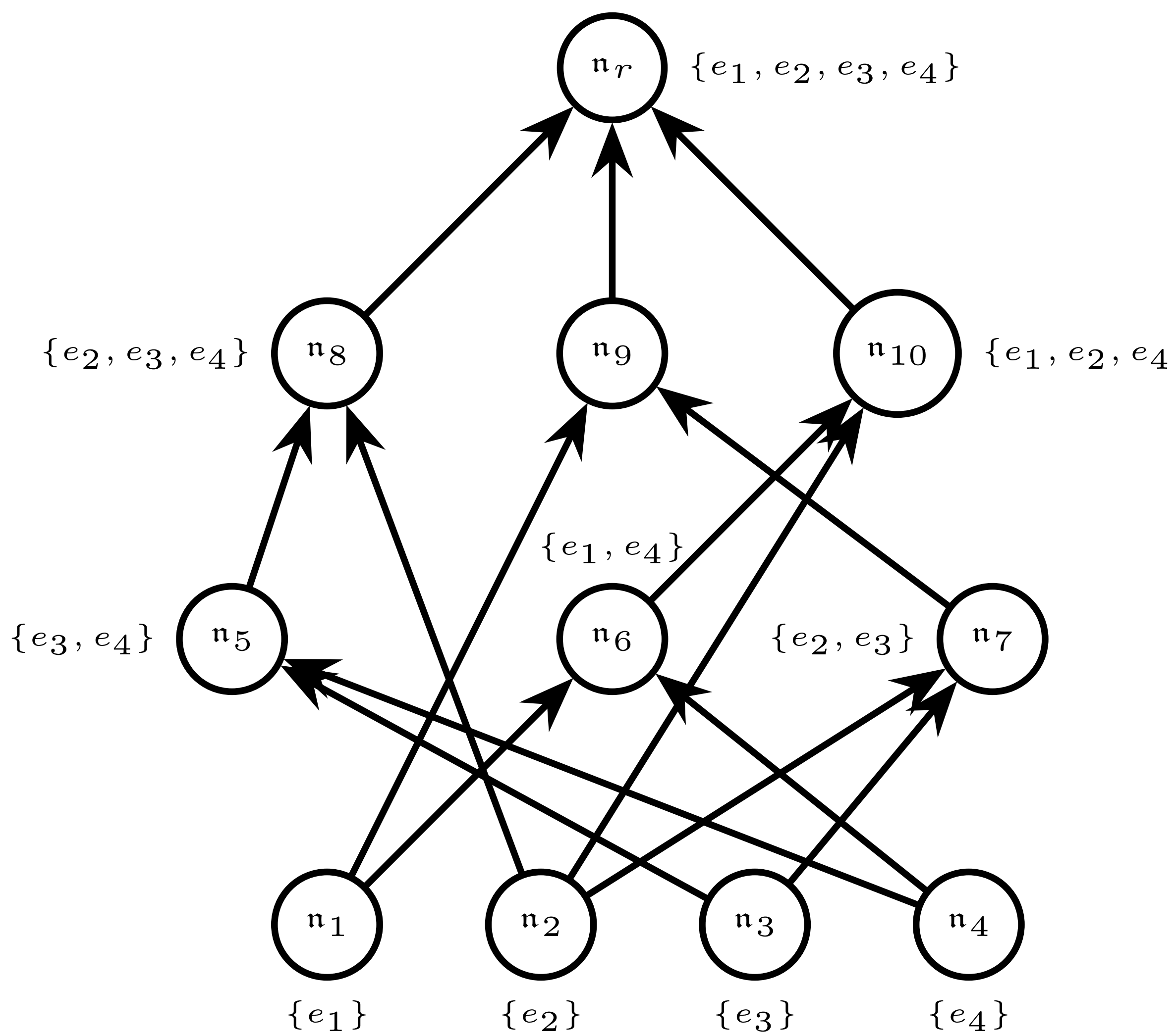
1
3
2
4
5
6
1
3
2
4
5
7
6

Compositional neural networks

[K., Pan, Hy-Truong, Trivedi & Anderson]

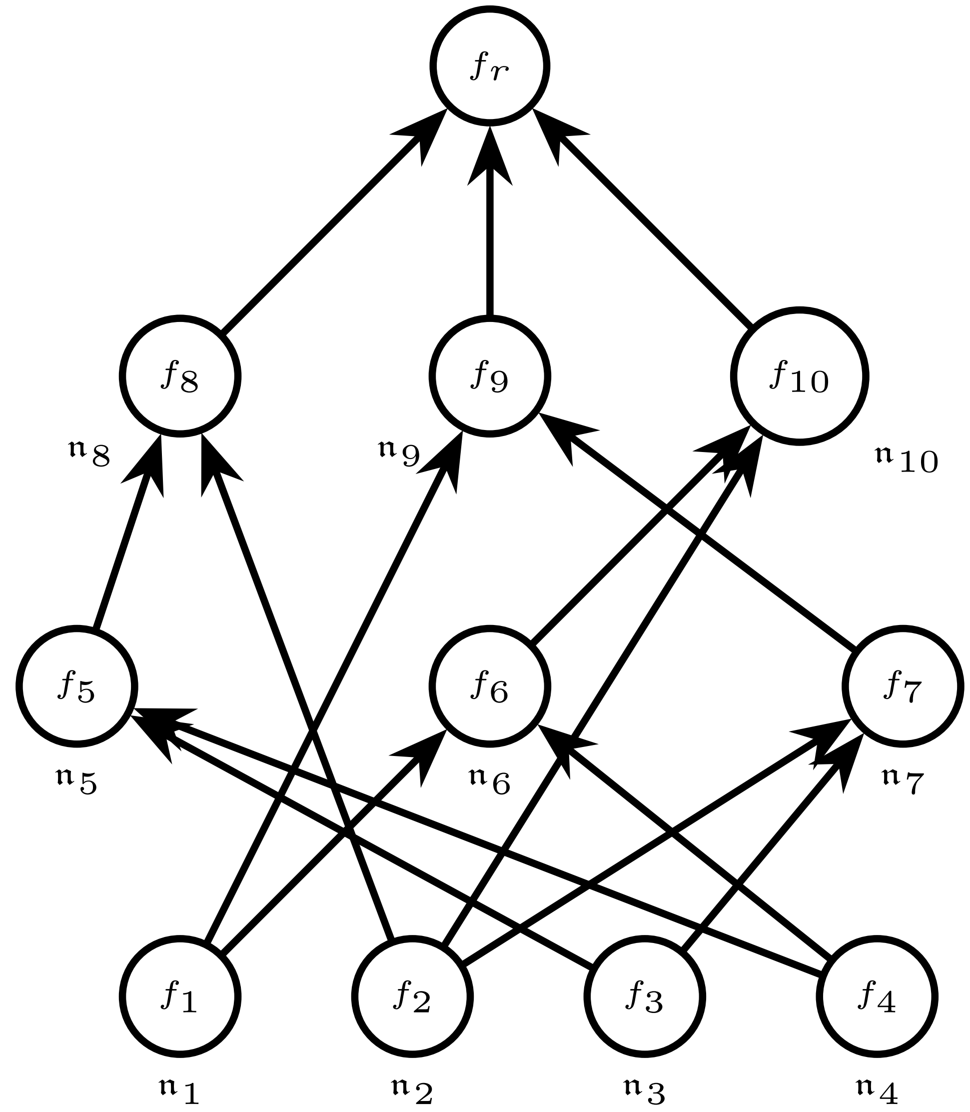
Composition scheme





Compositional networks (comp-nets)

$$f_i = \Phi(f_{c_1}, f_{c_2}, \ldots, f_{c_k})$$



Covariant Compositional network (CCN)

Quasi-invariant:

$$\Phi(f_{c_{\sigma(1)}}, f_{c_{\sigma(2)}}, \ldots, f_{c_{\sigma(k)}}) = \Phi(f_{c_1}, f_{c_2}, \ldots, f_{c_k})$$

Covariant:

$$\Phi(f_{c_{\sigma(1)}}, f_{c_{\sigma(2)}}, \ldots, f_{c_{\sigma(k)}}) = R_\sigma(\Phi(f_{c_1}, f_{c_2}, \ldots, f_{c_k}))$$

Here R_σ is a **representation** of $\mathbb{S}_k$.

0th order:

$$F_i \stackrel{\sigma}{\longmapsto} F_i$$

1st order:

$$F_i \stackrel{\sigma}{\longmapsto} P_\sigma F_i$$

2nd order:

$$F_i \stackrel{\sigma}{\longmapsto} P_\sigma F_i P_\sigma^\top$$

k'th order:

$$F_{i_1,i_2,\ldots,i_k} \stackrel{\sigma}{\longmapsto} [P_\sigma]_{i_1}^{j_1} [P_\sigma]_{i_2}^{j_2} \ldots [P_\sigma]_{i_k}^{j_k} F_{j_1,j_2,\ldots,j_k}$$

$$C = A \otimes B \qquad C_{i_1, i_2, \ldots, i_{k+p}} = A_{i_1, i_2, \ldots, i_k} \, B_{i_{k+1}, i_{k+2}, \ldots, i_{k+p}}$$

$$C = A \odot_{(a_1, \ldots, a_p)} B \qquad C_{i_1, i_2, \ldots, i_k} = A_{i_1, i_2, \ldots, i_k} \, B_{i_{a_1}, i_{a_2}, \ldots, i_{a_p}}$$

$$C = A{\downarrow}_{a_1, \ldots, a_p} \qquad C_{i_1, i_2, \ldots, i_k} = \sum_{i_{a_1}} \sum_{i_{a_2}} \cdots \sum_{i_{a_p}} A_{i_1, i_2, \ldots, i_k},$$

$$C_{i_1, i_2, \ldots, i_k} = A_{i_1, i_2, \ldots, i_k} \delta^{i_a, i_b}$$

$$C_{i_1, i_2, \ldots, i_k} = \sum_j A_{i_1, \ldots, i_{a-1}, j, i_{a+i}, \ldots, i_{b-1}, j, i_{b+1}, \ldots, k}$$

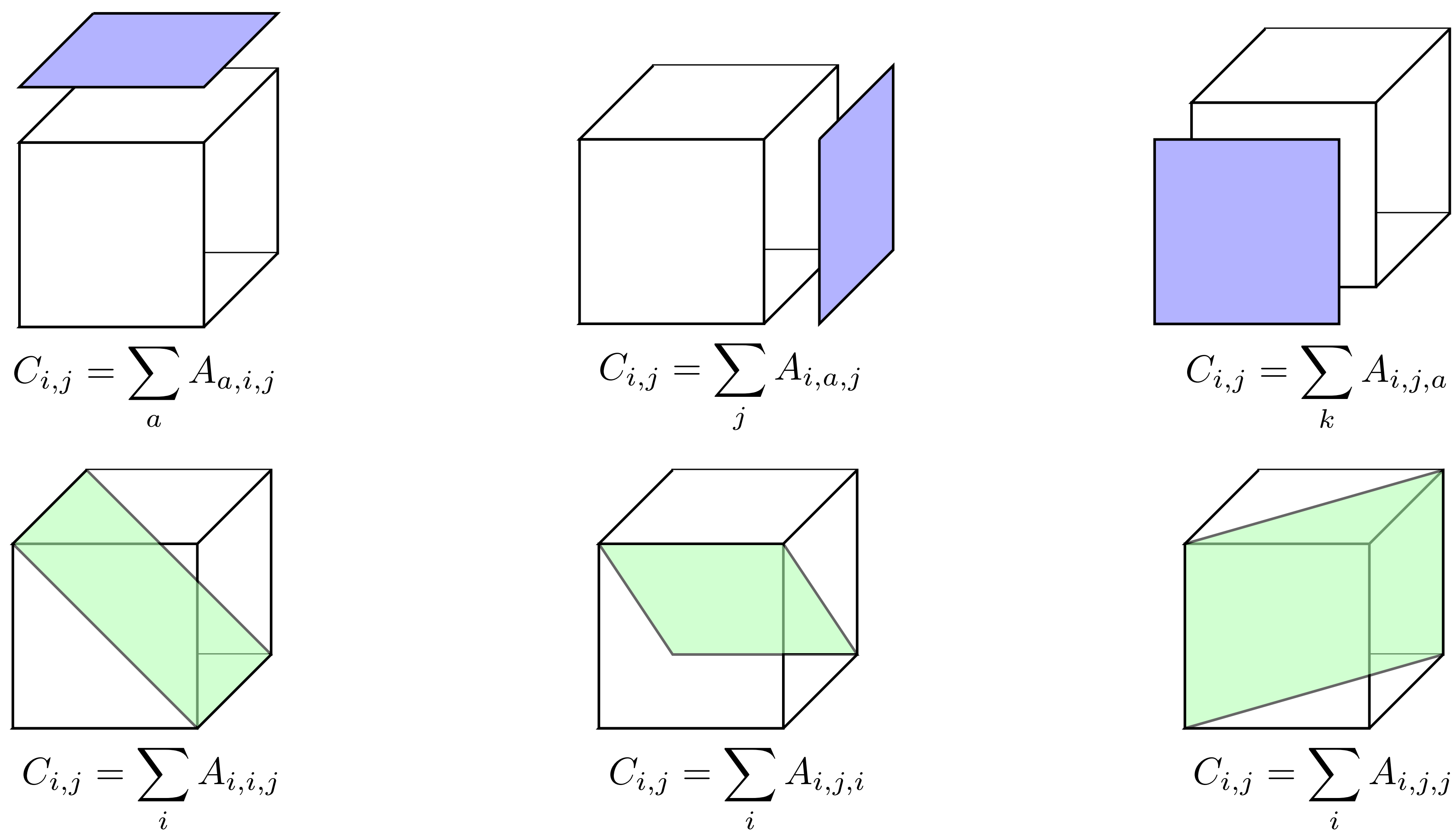


Figure 1: There are six different ways of covariantly reducing a third order tensor to a second order tensor: three different ways of projecting along each of its dimensions, and three different ways of taking the "trace" along a pair of dimensions.

Proposition. Assume that A and B are k'th and p'th order P–tensors, respectively. Then

1. $A \otimes B$ is a $k+p$'th order P–tensor.

2. $A \odot_{(a_1,\ldots,a_p)} B$ is a k'th order P–tensor.

3. $A{\downarrow}_{a_1,\ldots,a_p}$ is a $k-p$'th order P–tensor.

4. $A_{i_1,i_2,\ldots,i_k} \, \delta^{a^1_1,\ldots,a^1_{p_1}} \ldots \delta^{a^q_1,\ldots,a^q_{p_q}}$ is a $k-\sum_j p_j$'th order P–tensor.

In addition, if $A_1, \ldots, A_u$ are P–tensors and $\alpha_1, \ldots, \alpha_u$ are scalars, then $\sum_j \alpha_j A_j$ is a P–tensor.

Proposition Assume that node $\mathfrak{n}_a$ is a descendant of node $\mathfrak{n}_b$ in a comp-net $\mathcal{N}$, $\mathcal{P}_a = (e_{p_1}, \ldots, e_{p_m})$ and $\mathcal{P}_b = (e_{q_1}, \ldots, e_{q_{m'}})$ are the corresponding ordered receptive fields, and $\chi^{a \to b} \in \mathbb{R}^{m \times m'}$ is an indicator matrix defined

$$\chi^{a \to b}_{i,j} = \begin{cases} 1 & \text{if} \;\; q_i = p_j \\ 0 & \text{otherwise.} \end{cases}$$

Assume that F is a k'th order P–tensor with respect to permutations of $(e_{p_1}, \ldots, e_{p_m})$. Then, dropping the ${}^{a \to b}$ superscript for clarity,

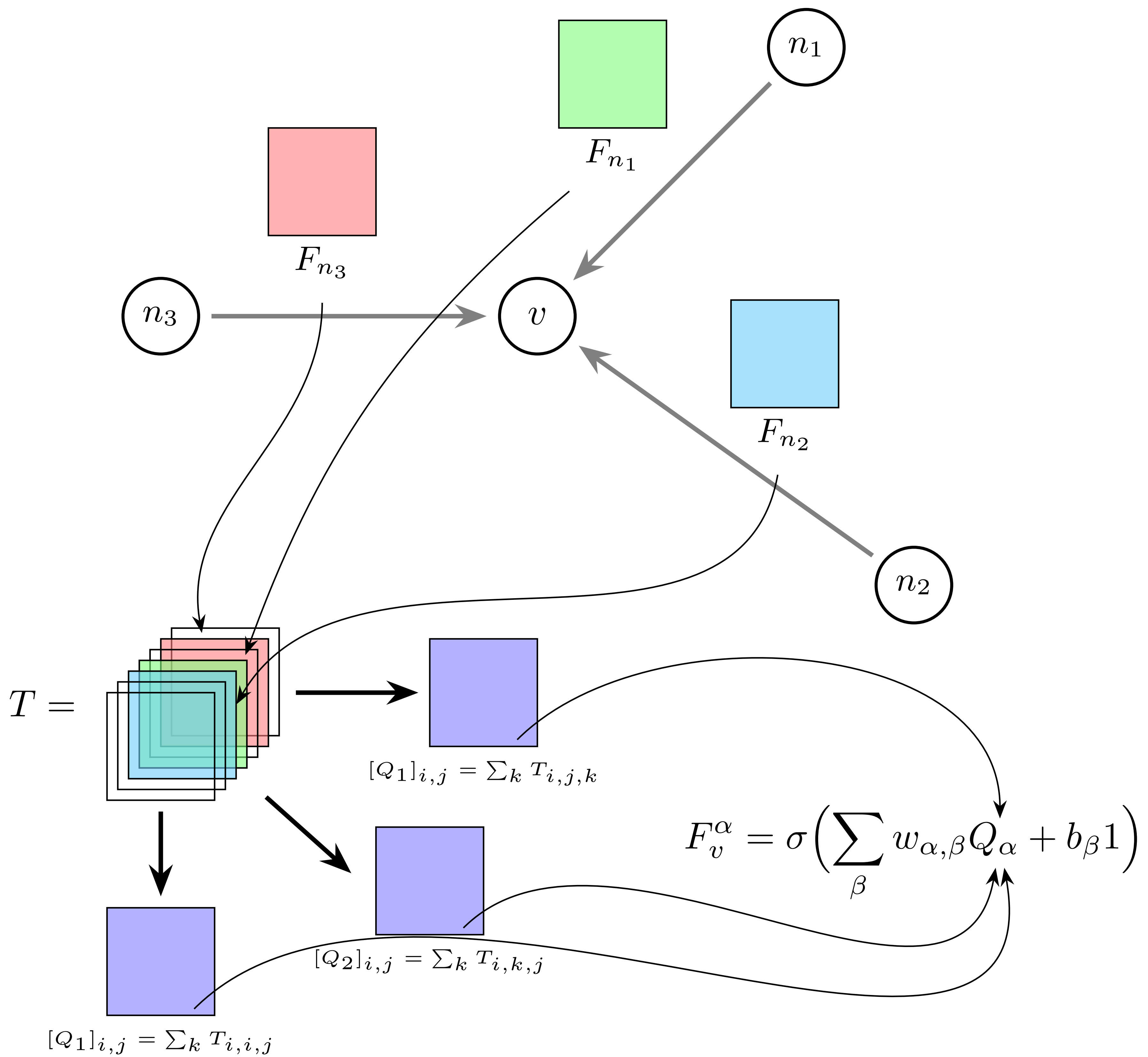
$$\widetilde{F}_{i_1, \ldots, i_k} = \chi_{i_1}^{\; j_1} \, \chi_{i_2}^{\; j_2} \, \ldots \, \chi_{i_k}^{\; j_k} \, F_{j_1, \ldots, j_k} \tag{1}$$

is a k'th order P–tensor with respect to permutations of $(e_{q_1}, \ldots, e_{q_{m'}})$.

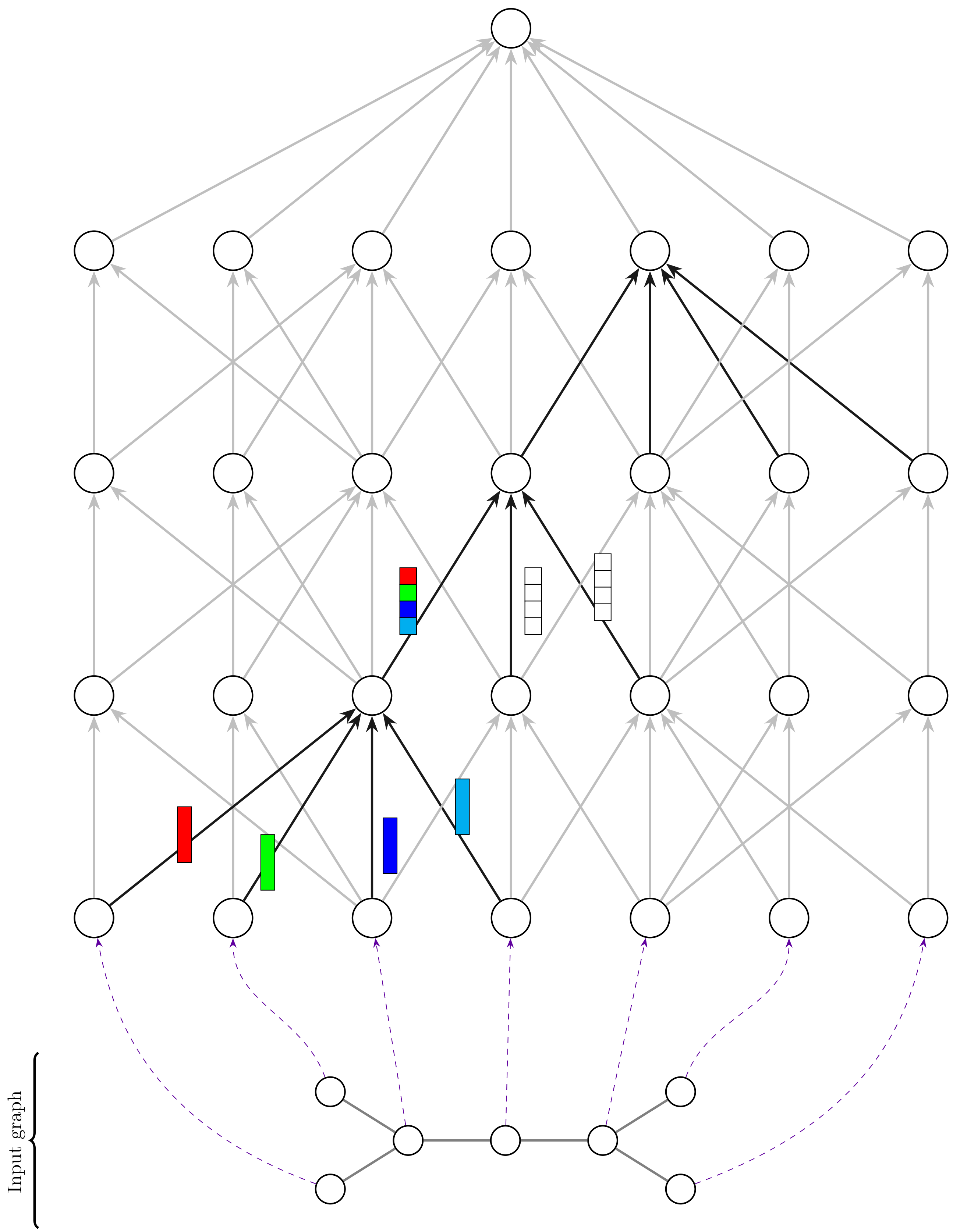
1. Collect all the k'th order activations $F_{c_1}, \ldots, F_{c_s}$ of the children.
2. Promote each activation to $\widetilde{F}_{c_1}, \ldots, \widetilde{F}_{c_s}$.
3. Stack $\widetilde{F}_{c_1}, \ldots, \widetilde{F}_{c_s}$ together into a $k+1$ order tensor T.
4. Optionally form the tensor product of T with $A{\downarrow_{\mathcal{P}_t}}$ to get a $k+3$ order tensor H (otherwise just set $H = T$).
5. Contract H along some number of combinations of dimensions to get s separate lower order tensors $Q_1, \ldots, Q_s$.
6. Mix $Q_1, \ldots, Q_s$ with a matrix $W \in \mathbb{R}^{s' \times s}$ and apply a nonlinearity Υ to get the final activation of the neuron, which consists of the s' output tensors

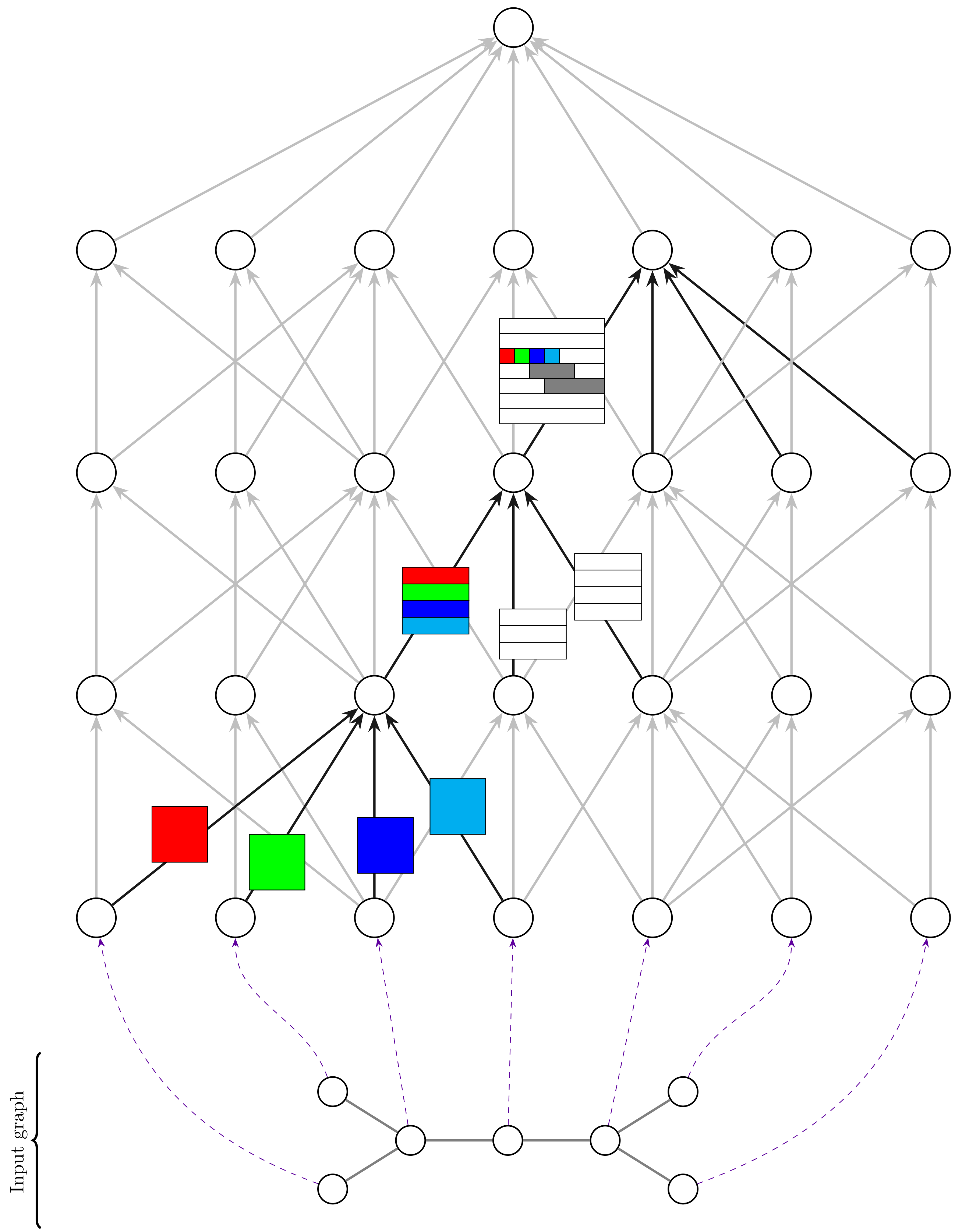
$$F^{(i)} = \Upsilon\Big[\sum_{j=1}^{s} W_{i,j}\, Q_j + b_i\Big] \qquad\qquad i = 1, 2, \ldots s',$$

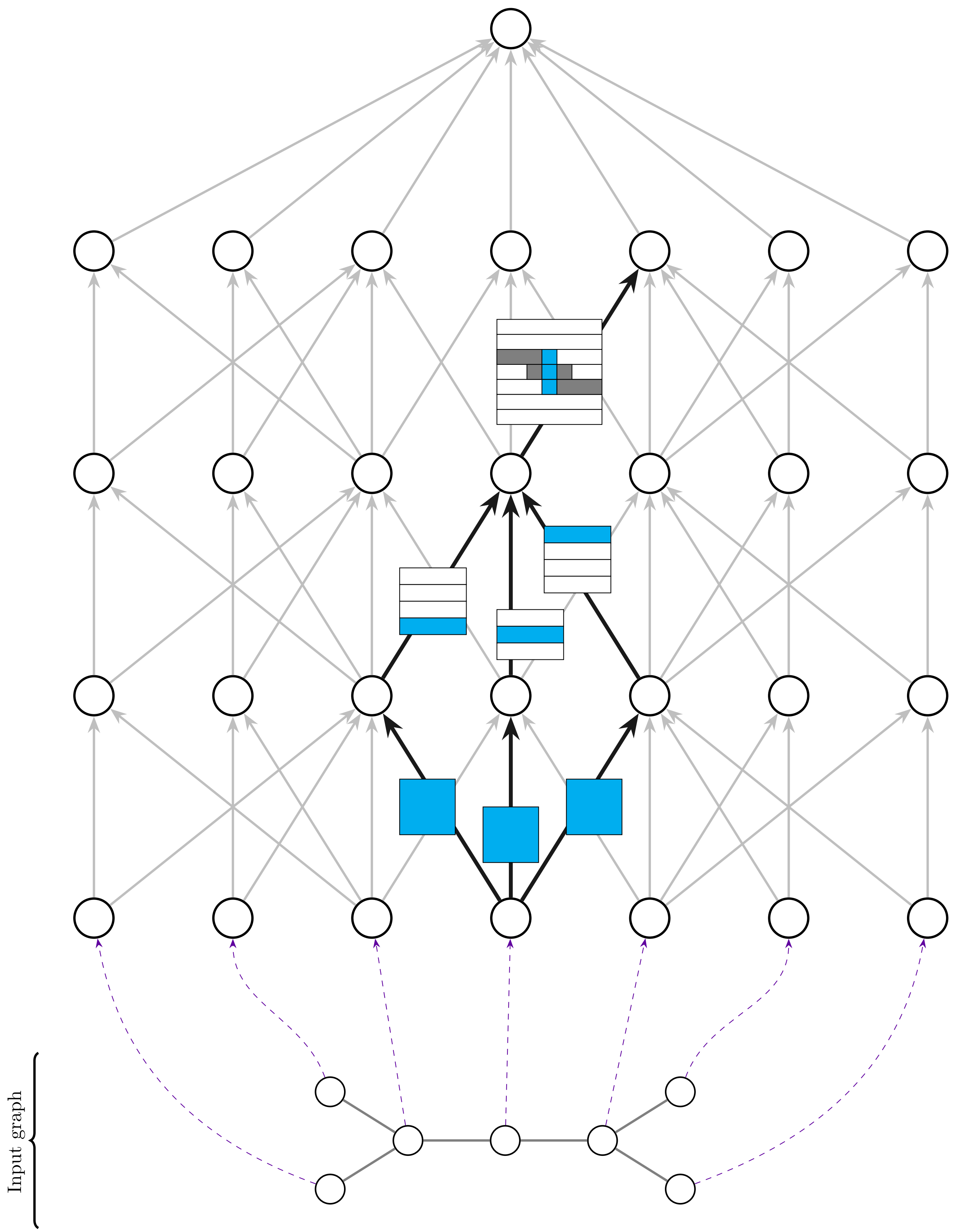
where the b_i scalars are bias terms.



n_1
F_{n_1}
F_{n_3}
n_3
v
F_{n_2}
n_2
$T =$
$[Q_1]_{i,j} = \sum_k T_{i,j,k}$
$F_v^\alpha = \sigma\Big(\sum_\beta w_{\alpha,\beta} Q_\alpha + b_\beta 1\Big)$
$[Q_2]_{i,j} = \sum_k T_{i,k,j}$
$[Q_1]_{i,j} = \sum_k T_{i,i,j}$


Input graph


Input graph


Input graph

HCEP results

Method	Train MAE	Train RMSE	Test MAE	Test RMSE
Lasso	0.863	1.190	0.867	1.437
Ridge Regression	0.849	1.164	0.854	1.376
Random Forest	0.999	1.331	1.004	1.799
Gradient Boosted Tree	0.676	0.939	0.704	1.005
Weisfeiler-Lehman Graph Kernel	0.805	1.111	0.805	1.096
Neural Graph Fingerprint	0.848	1.187	0.851	1.177
Learning Convolution Neural Network	0.704	0.972	0.718	0.973
CCN 2D	0.562	0.773	0.570	0.773

[Duvenaud et al., 2015] [Kriege, 2016] [Niepert, 2016]
[Hachmann et al., 2011]