ML SESSION 02, PART 1
DEEP GRAPH NETWORK

Javen Qinfeng Shi

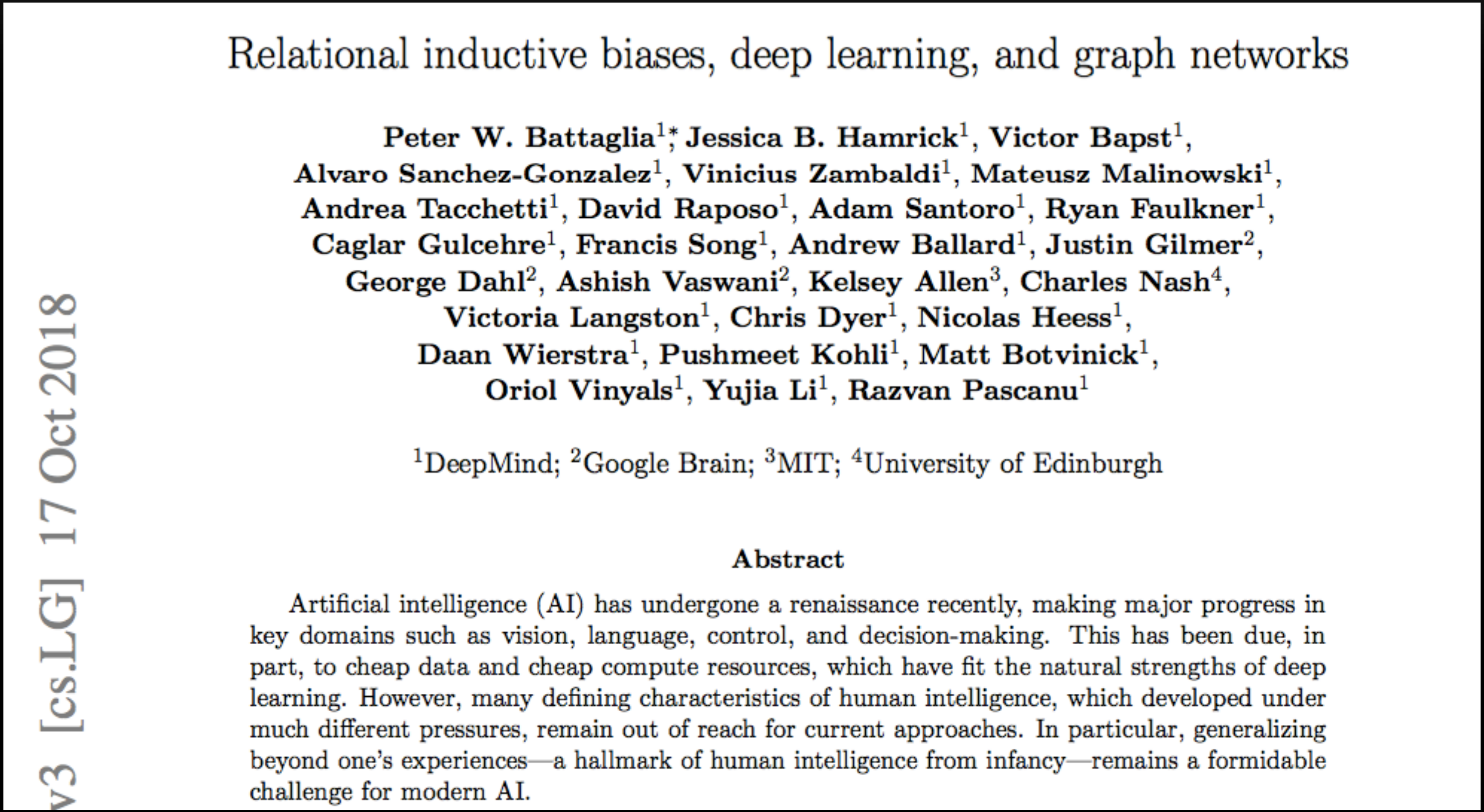
Associate Professor, The University of Adelaide (UoA)

Director and Founder, Probabilistic Graphical Model Group, UoA

Director of Advanced Reasoning and Learning, Australian Institute of Machine Learning (AIML), UoA

- Two types of deep neural networks
  - Deep belief networks are Markov random fields
  - CNN, RNN, … are Computational diagrams
- Two ways to combine graphical models and deep learning
  - Use a neural net to model the feature of a graphical model
  - Put a graphical model (or its approximation) into a neural net

DEEP MIND'S GRAPH NETWORKS (OCT. 2018)



v3 [cs.LG] 17 Oct 2018

Relational inductive biases, deep learning, and graph networks

Peter W. Battaglia[1]*, **Jessica B. Hamrick**[1], **Victor Bapst**[1], **Alvaro Sanchez-Gonzalez**[1], **Vinicius Zambaldi**[1], **Mateusz Malinowski**[1], **Andrea Tacchetti**[1], **David Raposo**[1], **Adam Santoro**[1], **Ryan Faulkner**[1], **Caglar Gulcehre**[1], **Francis Song**[1], **Andrew Ballard**[1], **Justin Gilmer**[2], **George Dahl**[2], **Ashish Vaswani**[2], **Kelsey Allen**[3], **Charles Nash**[4], **Victoria Langston**[1], **Chris Dyer**[1], **Nicolas Heess**[1], **Daan Wierstra**[1], **Pushmeet Kohli**[1], **Matt Botvinick**[1], **Oriol Vinyals**[1], **Yujia Li**[1], **Razvan Pascanu**[1]

[1]DeepMind; [2]Google Brain; [3]MIT; [4]University of Edinburgh

Abstract

Artificial intelligence (AI) has undergone a renaissance recently, making major progress in key domains such as vision, language, control, and decision-making. This has been due, in part, to cheap data and cheap compute resources, which have fit the natural strengths of deep learning. However, many defining characteristics of human intelligence, which developed under much different pressures, remain out of reach for current approaches. In particular, generalizing beyond one's experiences—a hallmark of human intelligence from infancy—remains a formidable challenge for modern AI.

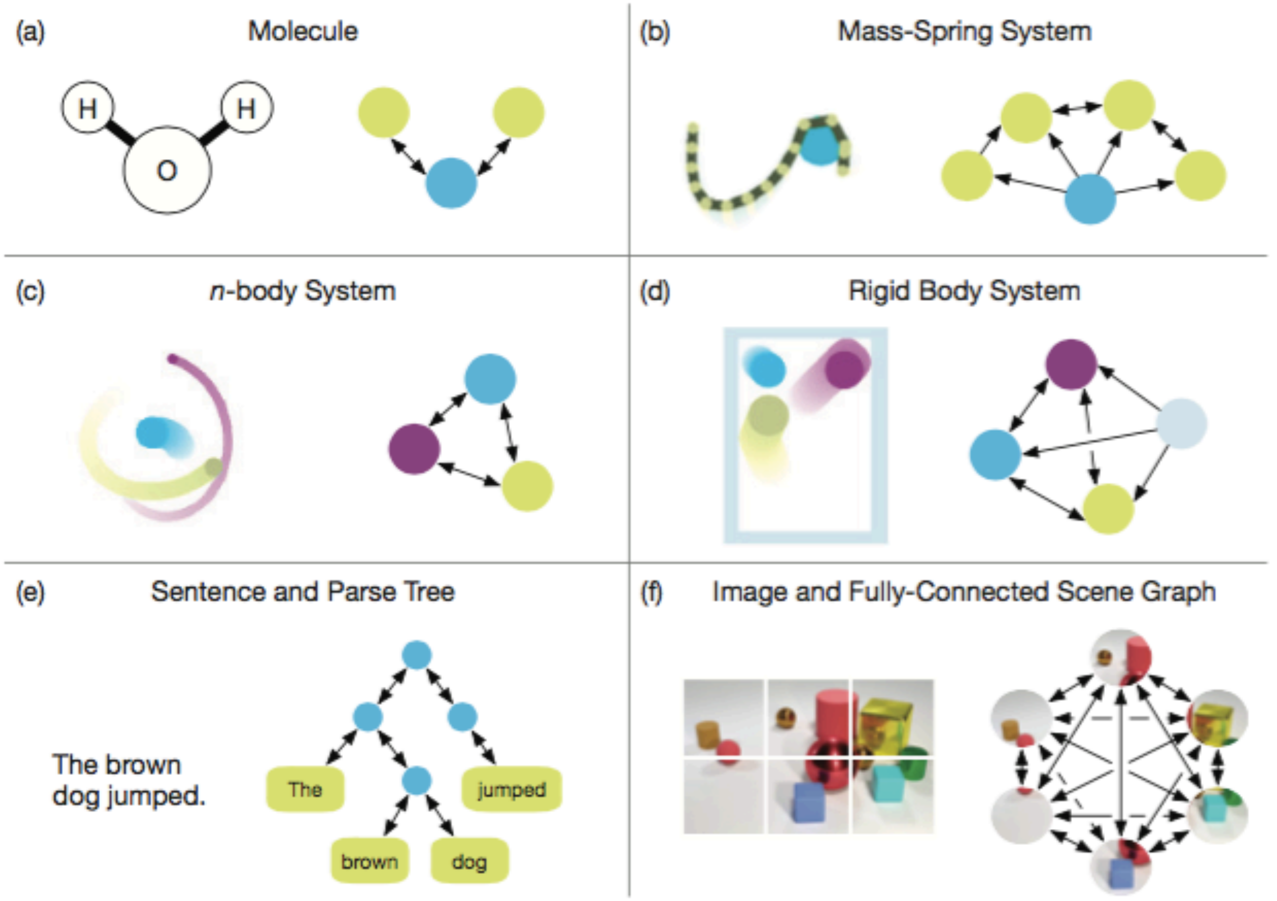
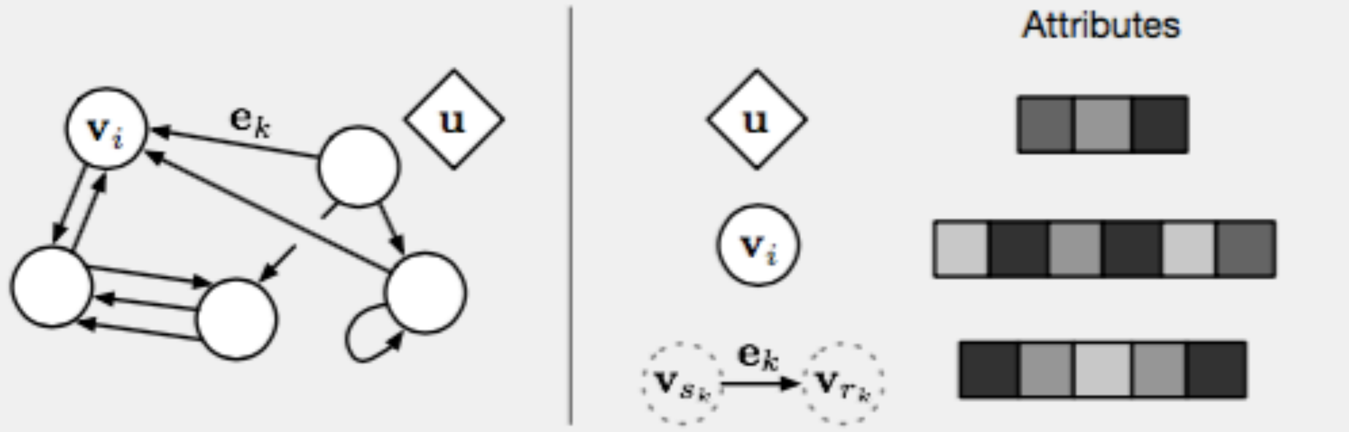


Figure 2: Different graph representations. (a) A molecule, in which each atom is represented as a node and edges correspond to bonds (e.g. Duvenaud et al., 2015). (b) A mass-spring system, in which the rope is defined by a sequence of masses which are represented as nodes in the graph (e.g. Battaglia et al., 2016; Chang et al., 2017). (c) A n-body system, in which the bodies are nodes and the underlying graph is fully connected (e.g. Battaglia et al., 2016; Chang et al., 2017). (d) A rigid body system, in which the balls and walls are nodes, and the underlying graph defines interactions between the balls and between the balls and the walls (e.g. Battaglia et al., 2016; Chang et al., 2017). (e) A sentence, in which the words correspond to leaves in a tree, and the other nodes and edges could be provided by a parser (e.g. Socher et al., 2013). Alternately, a fully connected graph could be used (e.g. Vaswani et al., 2017). (f) An image, which can be decomposed into image patches corresponding to nodes in a fully connected graph (e.g. Santoro et al., 2017; Wang et al., 2018c).

Box 3: Our definition of "graph"



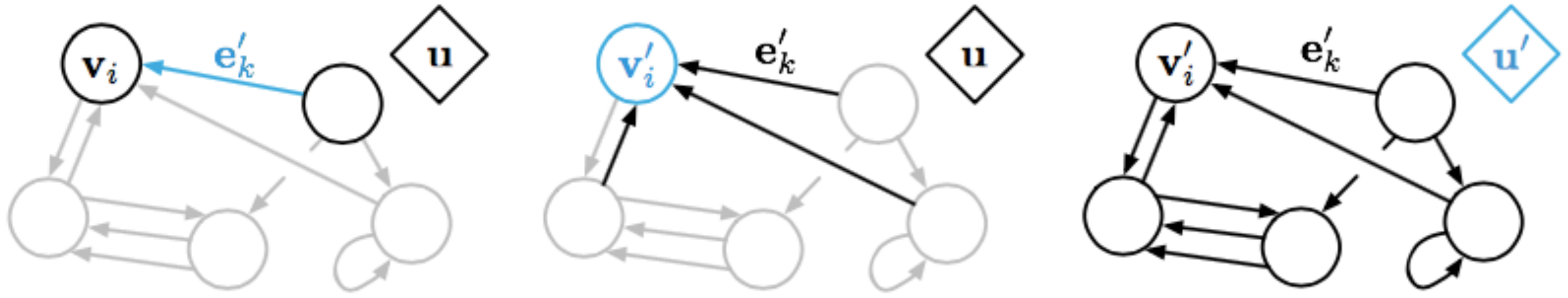
Here we use "graph" to mean a directed, attributed multi-graph with a global attribute. In our terminology, a node is denoted as $\mathbf{v}_i$, an edge as $\mathbf{e}_k$, and the global attributes as $\mathbf{u}$. We also use s_k and r_k to indicate the indices of the sender and receiver nodes (see below), respectively, for edge k. To be more precise, we define these terms as:

Directed : one-way edges, from a "sender" node to a "receiver" node.
Attribute : properties that can be encoded as a vector, set, or even another graph.
Attributed : edges and vertices have attributes associated with them.
Global attribute : a graph-level attribute.
Multi-graph : there can be more than one edge between vertices, including self-edges.

Figure 2 shows a variety of different types of graphs corresponding to real data that we may be interested in modeling, including physical systems, molecules, images, and text.

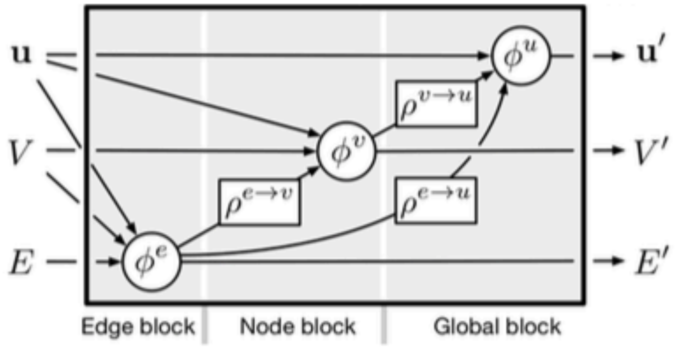
Algorithm 1 Steps of computation in a full GN block.

```
function GraphNetwork($E$, $V$, $\mathbf{u}$)
    for $k \in \{1 \dots N^e\}$ do
        $\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$        ▷ 1. Compute updated edge attributes
    end for
    for $i \in \{1 \dots N^n\}$ do
        let $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$
        $\bar{\mathbf{e}}'_i \leftarrow \rho^{e \to v}(E'_i)$        ▷ 2. Aggregate edge attributes per node
        $\mathbf{v}'_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$        ▷ 3. Compute updated node attributes
    end for
    let $V' = \{\mathbf{v}'\}_{i=1:N^v}$
    let $E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$
    $\bar{\mathbf{e}}' \leftarrow \rho^{e \to u}(E')$        ▷ 4. Aggregate edge attributes globally
    $\bar{\mathbf{v}}' \leftarrow \rho^{v \to u}(V')$        ▷ 5. Aggregate node attributes globally
    $\mathbf{u}' \leftarrow \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$        ▷ 6. Compute updated global attribute
    return $(E', V', \mathbf{u}')$
end function
```

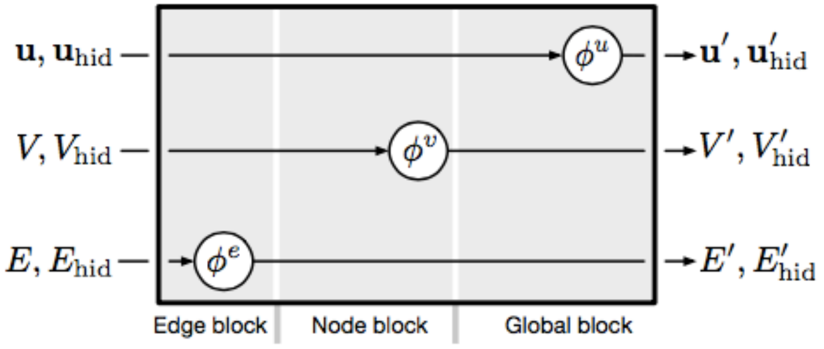


(a) Edge update (b) Node update (c) Global update

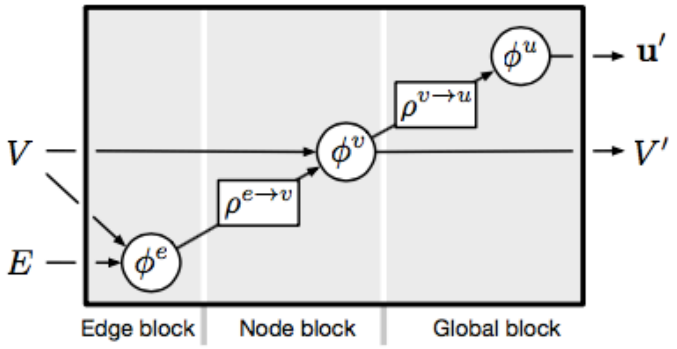
Figure 3: Updates in a GN block. Blue indicates the element that is being updated, and black indicates other elements which are involved in the update (note that the pre-update value of the



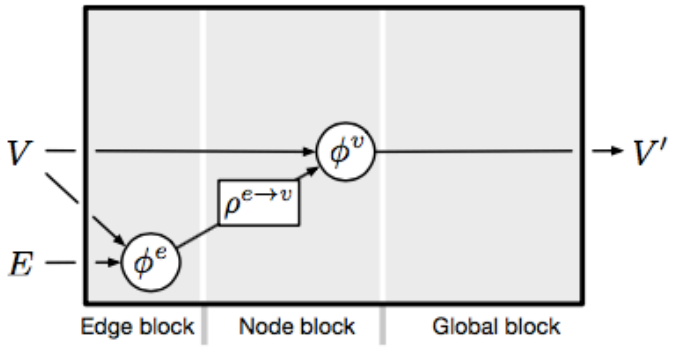
(a) Full GN block



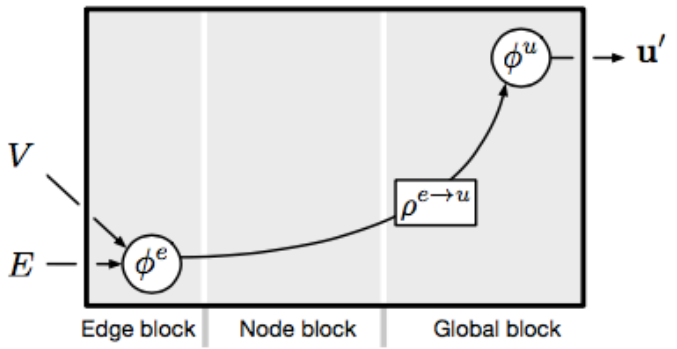
(b) Independent recurrent block



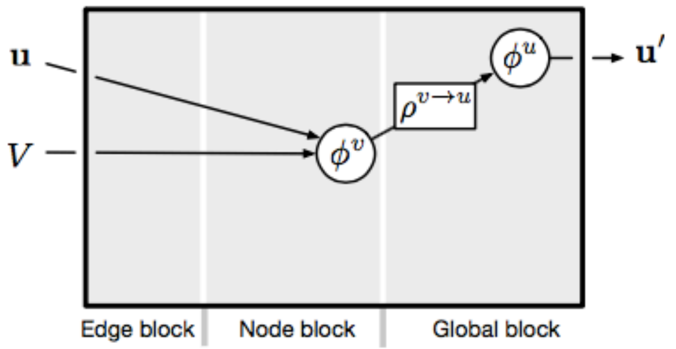
(c) Message-passing neural network



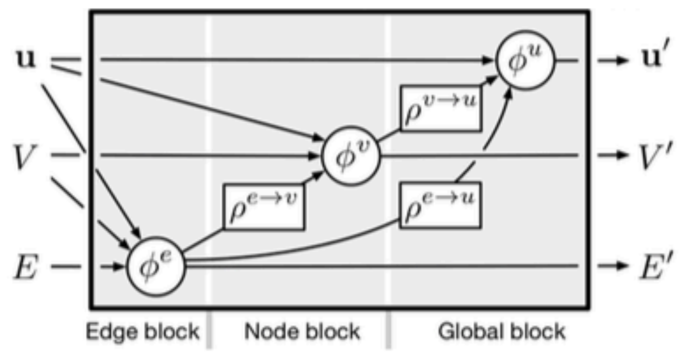
(d) Non-local neural network



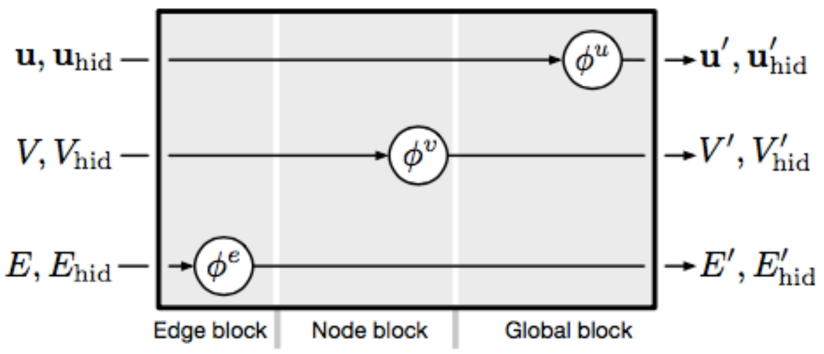
(e) Relation network



(f) Deep set



(a) Full GN block



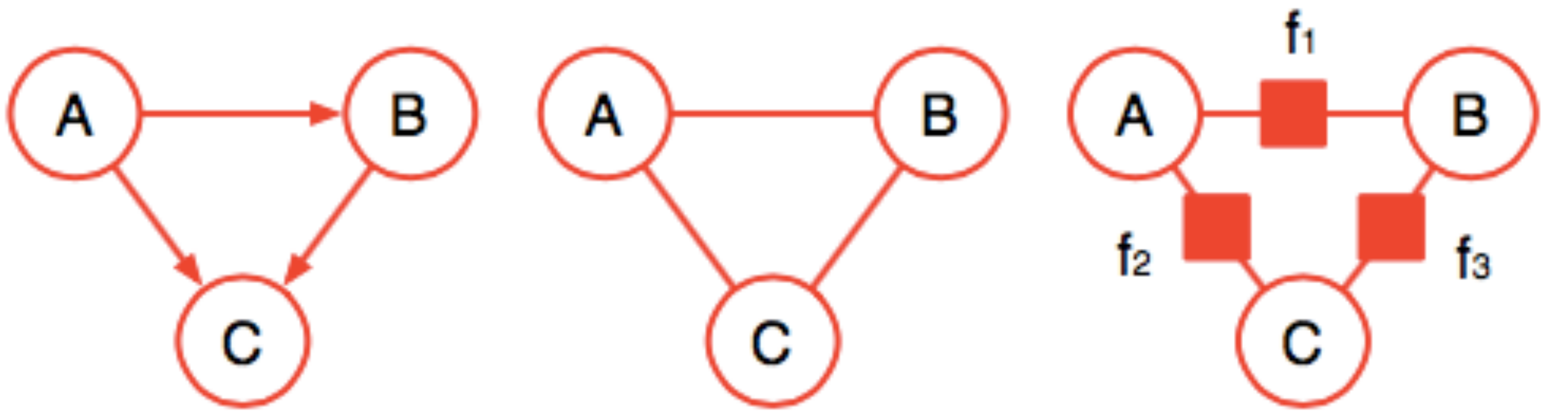
(b) Independent recurrent block

Algorithm 1 Steps of computation in a full GN block.

```
function GraphNetwork($E$, $V$, $\mathbf{u}$)
    for $k \in \{1 \ldots N^e\}$ do
        $\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$          ▷ 1. Compute updated edge attributes
    end for
    for $i \in \{1 \ldots N^n\}$ do
        let $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$
        $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\to v}(E'_i)$          ▷ 2. Aggregate edge attributes per node
        $\mathbf{v}'_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$          ▷ 3. Compute updated node attributes
    end for
    let $V' = \{\mathbf{v}'\}_{i=1:N^v}$
    let $E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$
    $\bar{\mathbf{e}}' \leftarrow \rho^{e\to u}(E')$          ▷ 4. Aggregate edge attributes globally
    $\bar{\mathbf{v}}' \leftarrow \rho^{v\to u}(V')$          ▷ 5. Aggregate node attributes globally
    $\mathbf{u}' \leftarrow \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$          ▷ 6. Compute updated global attribute
    return $(E', V', \mathbf{u}')$
end function
```

- Why inference in Graph Net is this way?
- How is Message Passing (Inference) done in a graphical model?

GRAPHICAL MODELS



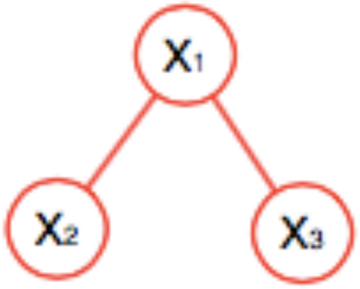
(a) Directed graph (b) Undirected graph (c) Factor graph

- Nodes represent random variables
- Edges reflect dependencies between variables
- Factors explicitly show which variables are used in each factor *i.e.* $f_1(A, B) f_2(A, C) f_3(B, C)$

VARIABLE ELIMINATION (MARGINAL INFERENCE)

Assume $$P(x_1, x_2, x_3) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_1)\psi(x_2)\psi(x_3)$$

$$\begin{aligned}P(x_1) &= \frac{1}{Z}\psi(x_1)\sum_{x_2}\Big(\psi(x_1, x_2)\psi(x_2)\Big)\sum_{x_3}\Big(\psi(x_1, x_3)\psi(x_3)\Big)\\ &= \frac{1}{Z}\psi(x_1)m_{2\to 1}(x_1)m_{3\to 1}(x_1)\end{aligned}$$



$$\begin{aligned}P(x_2) &= \frac{1}{Z}\psi(x_2)\sum_{x_1}\Big(\psi(x_1, x_2)\psi(x_1)\sum_{x_3}\Big[\psi(x_1, x_3)\psi(x_3)\Big]\Big)\\ &= \frac{1}{Z}\psi(x_2)\sum_{x_1}\psi(x_1, x_2)\psi(x_1)m_{3\to 1}(x_1)\\ &= \frac{1}{Z}\psi(x_2)m_{1\to 2}(x_2)\end{aligned}$$

$m_{3\to 1}(x_1)$ can be reused instead of computing twice.

SUM-PRODUCT

Can we compute all messages first, and then use them to compute all marginal distributions?

Yes, it's called sum-product.

In general,

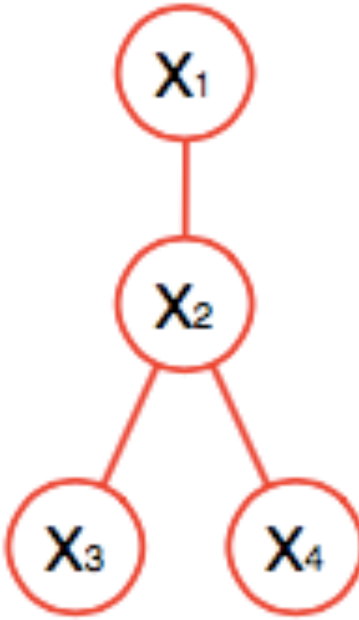
$$P(x_i) = \frac{1}{Z}\left(\psi(x_i) \prod_{j \in Ne(i)} m_{j \to i}(x_i)\right)$$

$$m_{j \to i}(x_i) = \sum_{x_j}\left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in Ne(j) \setminus \{i\}} m_{k \to j}(x_j)\right)$$

$Ne(i)$: neighbouring nodes of i (*i.e.* nodes that connect with i).

MAP INFERENCE

$$\max_{x_1,x_2,x_3,x_4} P(x_1,x_2,x_3,x_4)$$
$$= \max_{x_1,x_2,x_3,x_4} \psi(x_1,x_2)\psi(x_2,x_3)\psi(x_2,x_4)\psi(x_1)\psi(x_2)\psi(x_3)\psi(x_4)$$
$$= \max_{x_1,x_2} \left[\ldots \max_{x_3} \left(\psi(x_2,x_3)\psi(x_3) \right) \max_{x_4} \left(\psi(x_2,x_4)\psi(x_4) \right) \right]$$
$$= \max_{x_1} \left[\psi(x_1) \max_{x_2} \left(\psi(x_2)\psi(x_1,x_2) m_{3\to 2}(x_2) m_{4\to 2}(x_2) \right) \right]$$
$$= \max_{x_1} \left(\psi(x_1) m_{2\to 1}(x_1) \right)$$



$$x_1^* = \operatorname{argmax}_{x_1} \left(\psi(x_1) m_{2\to 1}(x_1) \right)$$
$$x_2^* = \operatorname{argmax}_{x_2} \left(\psi(x_2)\psi(x_1^*, x_2) m_{3\to 2}(x_2) m_{4\to 2}(x_2) \right)$$
$$x_3^* = \operatorname{argmax}_{x_3} \left(\psi(x_2^*, x_3)\psi(x_3) \right)$$
$$x_4^* = \operatorname{argmax}_{x_4} \left(\psi(x_2^*, x_4)\psi(x_4) \right)$$

MAX-PRODUCT

Variable elimination for MAP $\Rightarrow$ Max-product:

$$x_i^* = \operatorname*{argmax}_{x_i} \left(\psi(x_i) \prod_{j \in Ne(i)} m_{j \to i}(x_i) \right)$$

$$m_{j \to i}(x_i) = \max_{x_j} \left(\psi(x_j) \psi(x_i, x_j) \prod_{k \in Ne(j) \setminus \{i\}} m_{k \to j}(x_j) \right)$$

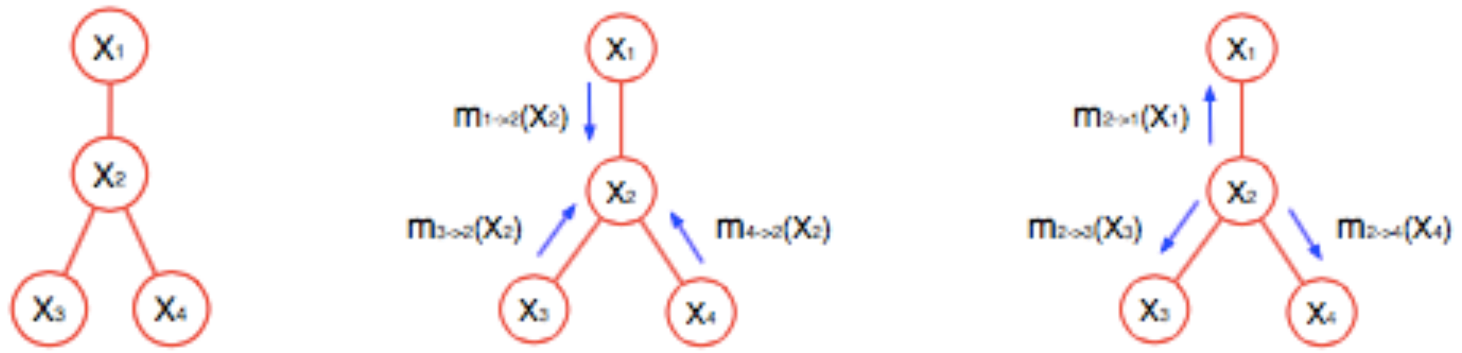
$Ne(i)$: neighbouring nodes of i (*i.e.* nodes that connect with i).

$Ne(j) \setminus \{i\} = \emptyset$ if j has only one edge connecting it. *e.g.* x_1, x_3, x_4. For such node j,

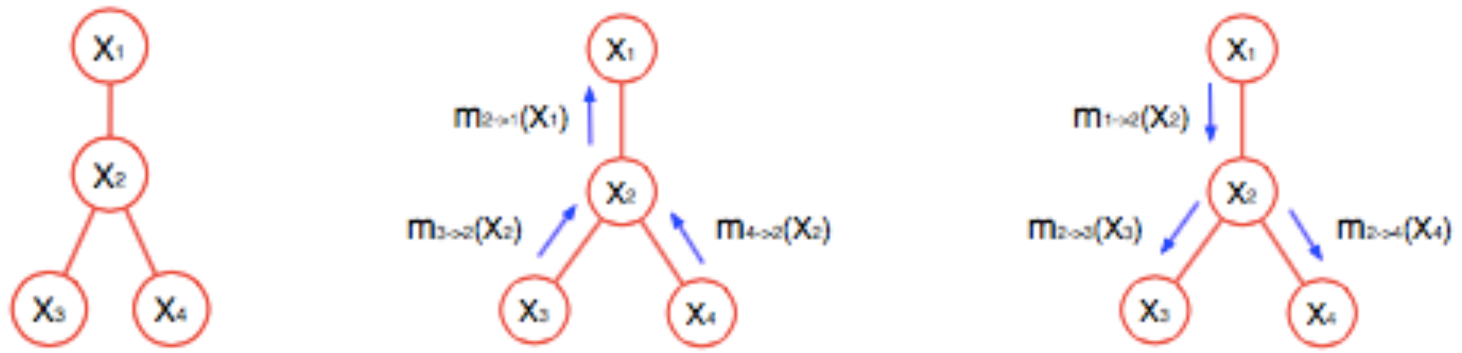
$$m_{j \to i}(x_i) = \max_{x_j} \left(\psi(x_j) \psi(x_i, x_j) \right)$$

MESSAGE PASSING

Order matters: message $m_{2 \to 3}(x_3)$ requires $m_{1 \to 2}(x_2)$ and $m_{4 \to 2}(x_2)$.



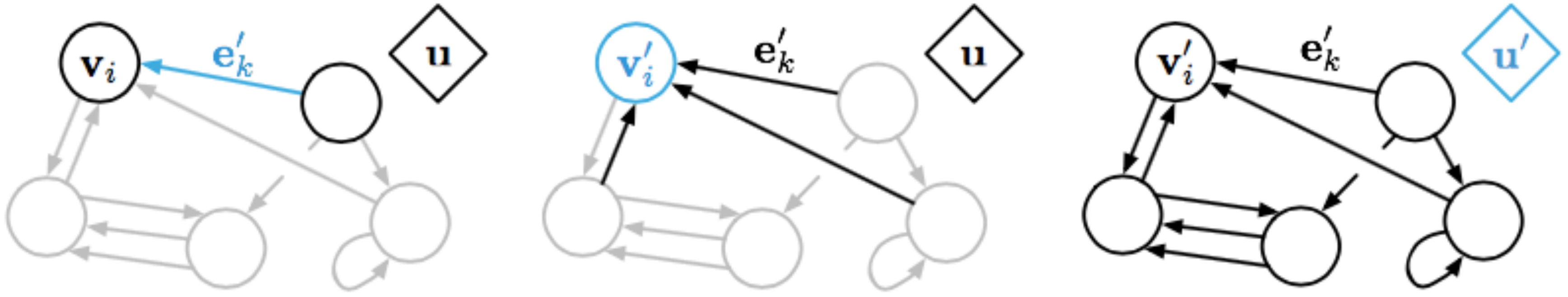
Alternatively, leaves to root, and root to leaves.



BACK TO DEEP GRAPH NET

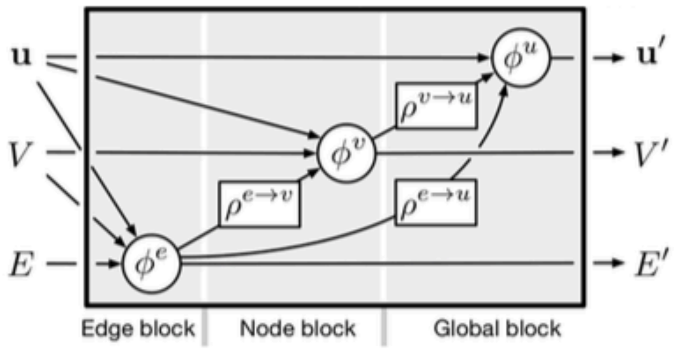
Algorithm 1 Steps of computation in a full GN block.

```
function GraphNetwork($E$, $V$, $\mathbf{u}$)
    for $k \in \{1 \ldots N^e\}$ do
        $\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$            ▷ 1. Compute updated edge attributes
    end for
    for $i \in \{1 \ldots N^n\}$ do
        let $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$
        $\bar{\mathbf{e}}'_i \leftarrow \rho^{e \to v}(E'_i)$            ▷ 2. Aggregate edge attributes per node
        $\mathbf{v}'_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$            ▷ 3. Compute updated node attributes
    end for
    let $V' = \{\mathbf{v}'\}_{i=1:N^v}$
    let $E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$
    $\bar{\mathbf{e}}' \leftarrow \rho^{e \to u}(E')$            ▷ 4. Aggregate edge attributes globally
    $\bar{\mathbf{v}}' \leftarrow \rho^{v \to u}(V')$            ▷ 5. Aggregate node attributes globally
    $\mathbf{u}' \leftarrow \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$            ▷ 6. Compute updated global attribute
    return $(E', V', \mathbf{u}')$
end function
```

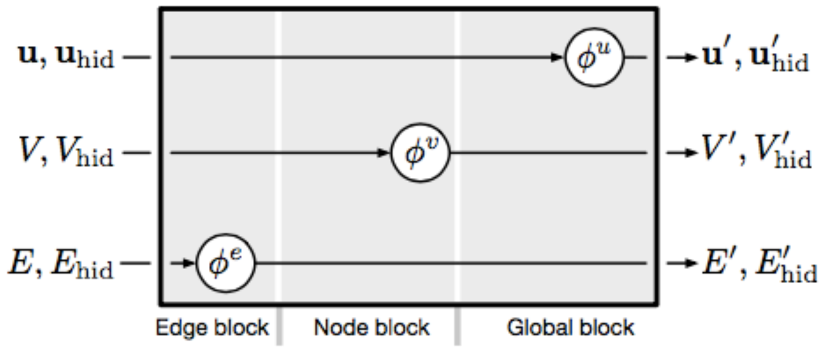


(a) Edge update (b) Node update (c) Global update

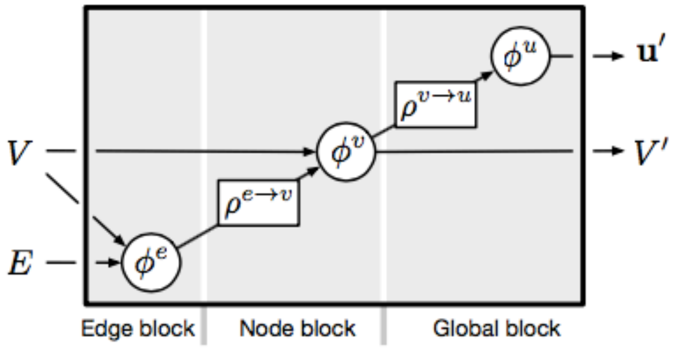
Figure 3: Updates in a GN block. Blue indicates the element that is being updated, and black indicates other elements which are involved in the update (note that the pre-update value of the



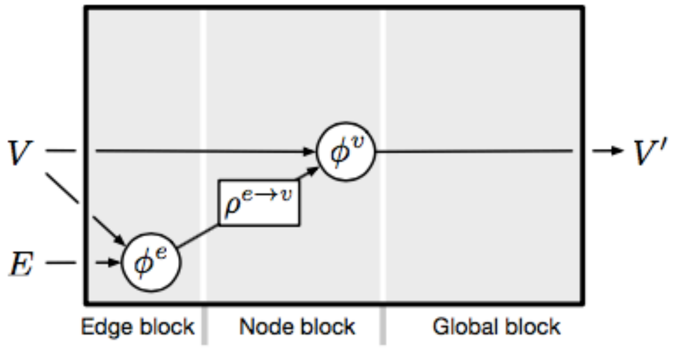
(a) Full GN block



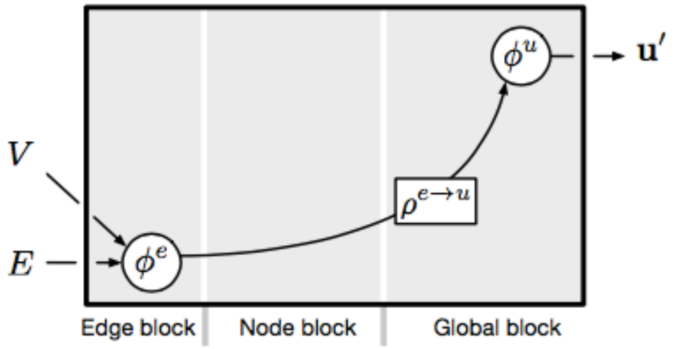
(b) Independent recurrent block



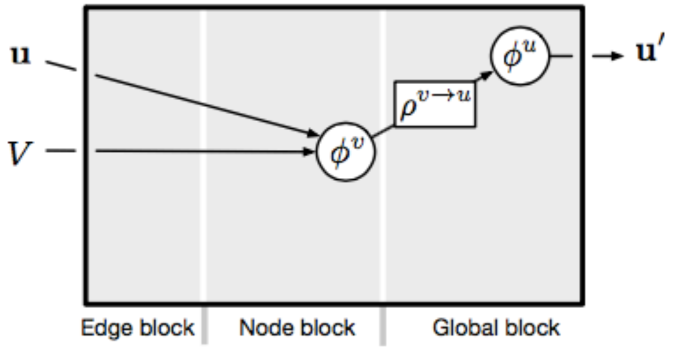
(c) Message-passing neural network



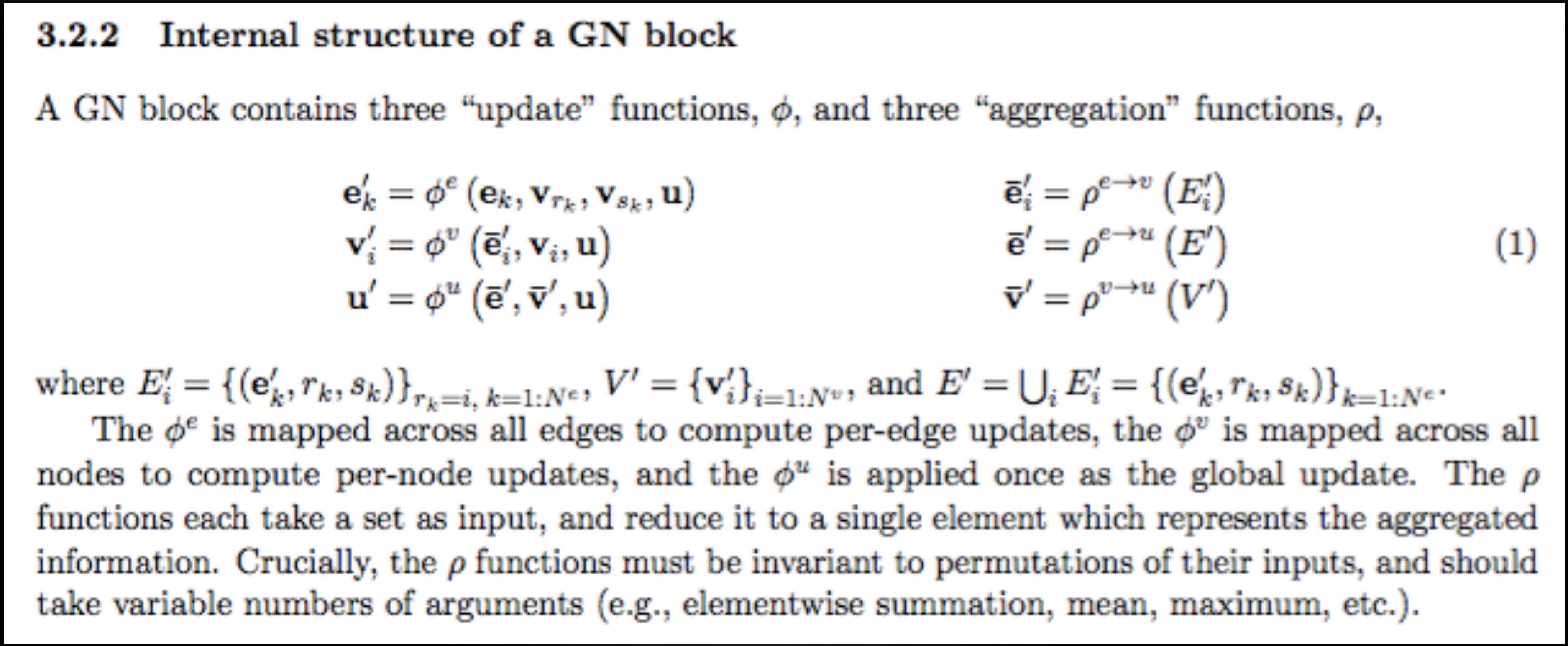
(d) Non-local neural network



(e) Relation network



(f) Deep set



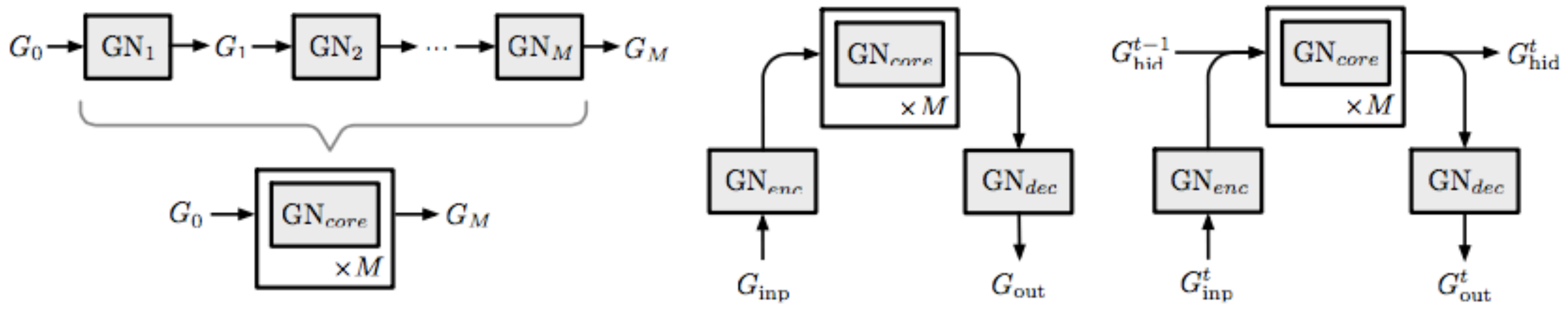
3.2.2 Internal structure of a GN block

A GN block contains three "update" functions, ϕ, and three "aggregation" functions, ρ,

$$
\begin{aligned}
\mathbf{e}'_k &= \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) & \bar{\mathbf{e}}'_i &= \rho^{e\rightarrow v}\left(E'_i\right) \\
\mathbf{v}'_i &= \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) & \bar{\mathbf{e}}' &= \rho^{e\rightarrow u}\left(E'\right) \\
\mathbf{u}' &= \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) & \bar{\mathbf{v}}' &= \rho^{v\rightarrow u}\left(V'\right)
\end{aligned}
\tag{1}
$$

where $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$, $V' = \{\mathbf{v}'_i\}_{i=1:N^v}$, and $E' = \bigcup_i E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$.

The ϕ^e is mapped across all edges to compute per-edge updates, the ϕ^v is mapped across all nodes to compute per-node updates, and the ϕ^u is applied once as the global update. The ρ functions each take a set as input, and reduce it to a single element which represents the aggregated information. Crucially, the ρ functions must be invariant to permutations of their inputs, and should take variable numbers of arguments (e.g., elementwise summation, mean, maximum, etc.).


G_0 → GN_1 → G_1 → GN_2 → ⋯ → GN_M → G_M
G_0 → GN_{core} ×M → G_M
G_{inp} → GN_{enc} → GN_{core} ×M → GN_{dec} → G_{out}
G^{t-1}_{hid}, G^t_{inp} → GN_{enc} → GN_{core} ×M → G^t_{hid}, GN_{dec} → G^t_{out}

(a) Composition of GN blocks (b) Encode-process-decode (c) Recurrent GN architecture

Figure 6: (a) An example composing multiple GN blocks in sequence to form a GN "core". Here, the GN blocks can use shared weights, or they could be independent. (b) The *encode-process-decode* architecture, which is a common choice for composing GN blocks (see Section 4.3). Here, a GN encodes an input graph, which is then processed by a GN core. The output of the core is decoded by a third GN block into an output graph, whose nodes, edges, and/or global attributes would be used for task-specific purposes. (c) The encode-process-decode architecture applied in a sequential setting in which the core is also unrolled over time (potentially using a GRU or LSTM architecture), in addition to being repeated within each time step. Here, merged lines indicate concatenation, and split lines indicate copying.