Graph Neural Networks

Amog Kamsetty
January 30, 2019

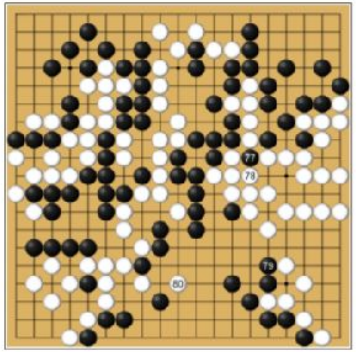
Motivations

Traditional Deep Learning

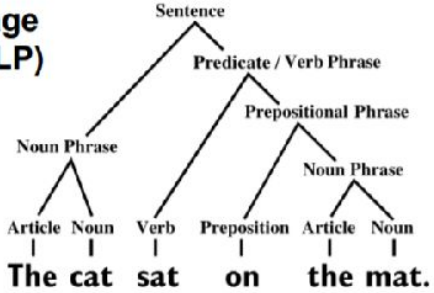
IMAGENET

Speech data

Grid games



Natural language processing (NLP)



...

Deep neural nets that exploit:

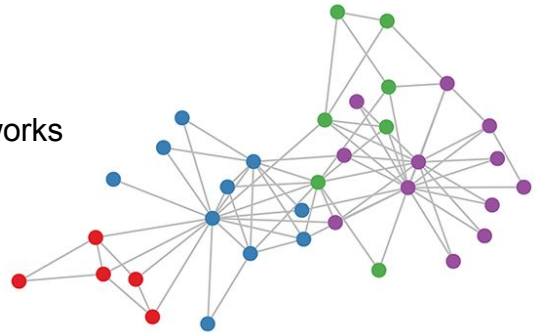
- translation equivariance (weight sharing)
- hierarchical compositionality

Slide from Thomas Kipf

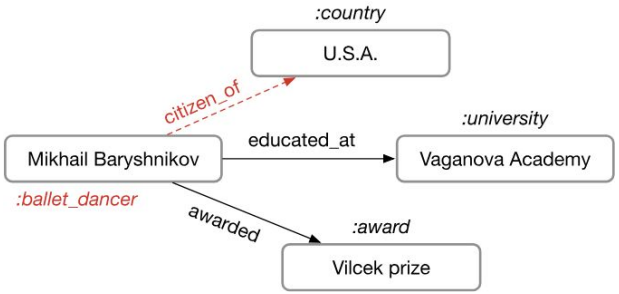
Graph Structured Data

A lot of real-world data does not live on “grids”

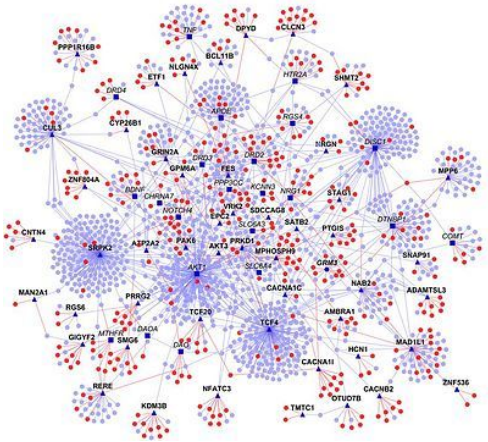
Social Networks
Citation Networks
Communication Networks
Multi-Agent Systems



Knowledge Graphs



Protein Interaction Networks



Slide from Thomas Kipf

Inspiration from CNNs

- Advantages of CNNs
  - Local Connections
  - Shared Weights
  - Use of multiple Layers
- But, hard to define convolutional and pooling layers for non-Euclidean data

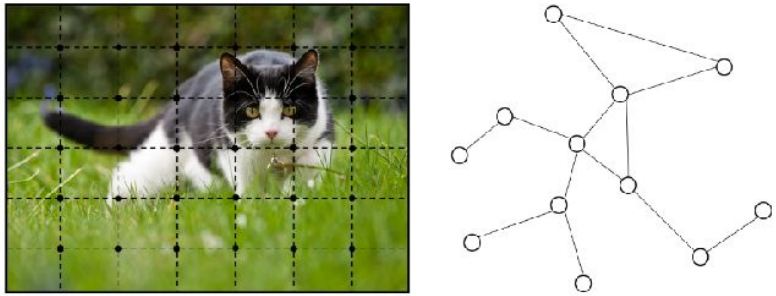


Fig. 1. Left: image in Euclidean space. Right: graph in non-Euclidean space

Deep Learning on Graphs

- Need architecture for machine learning on graph structured data
- Also need to take advantage of the structural relationships in graphs, similar to CNNs
- Collect information via message passing and then aggregation

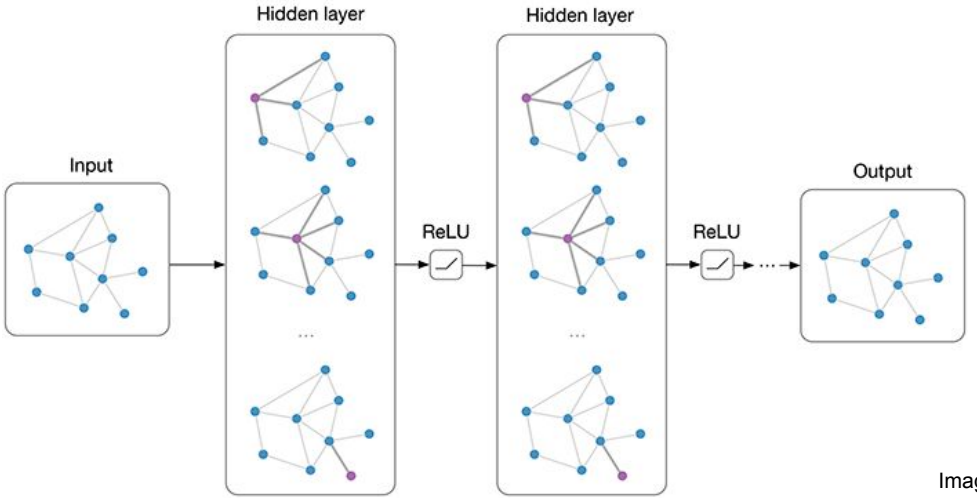


Image from Thomas Kipf

“Graph neural networks (GNNs) are connectionist models that capture the dependence of graphs via message passing between the nodes of graphs.”

Types of Graph Networks

The original Graph Neural Network (GNN)

- Each node is defined by its own features and those of its neighbors
- Learn some state embedding for each node
- Scarselli et al., 2009

$$h_v = f(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]})$$

$$o_v = g(h_v, x_v)$$

$$H^{t+1} = F(H^t, X)$$

The original Graph Neural Network (GNN)

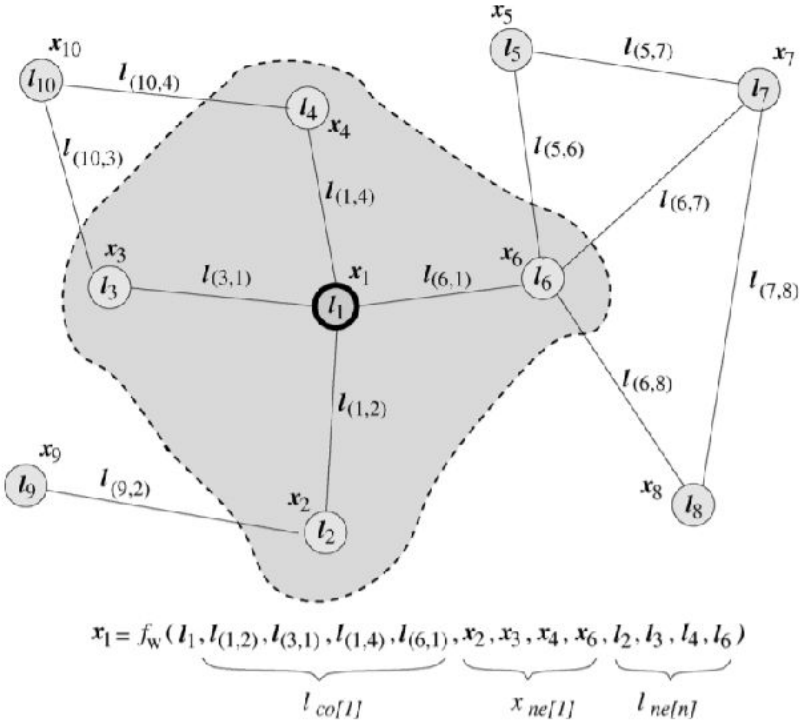
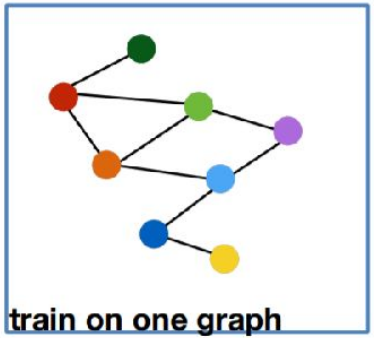


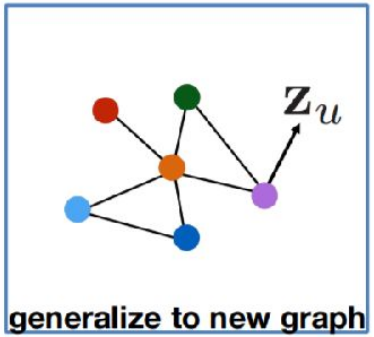
Fig. 2. Graph and the neighborhood of a node. The state $\boldsymbol{x}_1$ of the node 1 depends on the information contained in its neighborhood.

$$\mathbf{h}_i^k = \sum_{v_j \in \mathcal{N}(v_i)} \sigma(\mathbf{W}\mathbf{h}_j^{k-1} + \mathbf{b}),$$

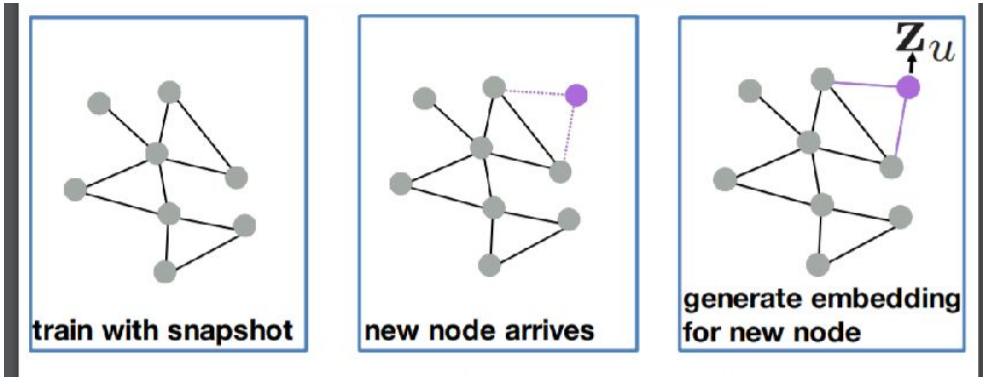
Update equation if parametrized by MLP

Inductive Capability





Inductive node embedding → generalize to entirely unseen graphs



Slide from Stanford Snap Lab

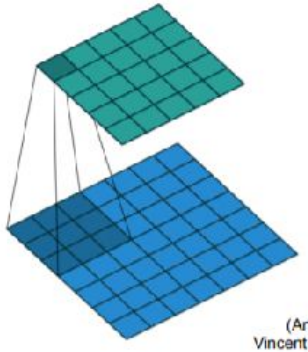
Limitations of GNN

- Very computationally intensive to recursively compute fixed point solution
- Same parameters are used in every timestep, while other neural networks use different parameters in each layer
- Informative edge features are difficult to model
    - Can’t alter message propagation depending on edge type

These pitfalls led to many variations of GNNs, particularly in how the propagation occurs

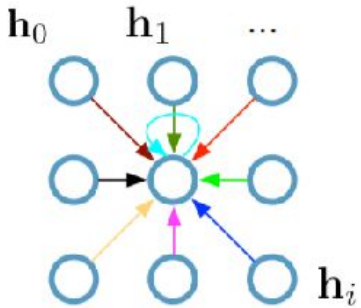
Graph Convolutional Networks

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)

Kipf & Welling, 2017



$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Slide from Thomas Kipf

Graph Convolutional Networks

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

use the same transformation matrix for self and neighbor embeddings

instead of simple average, normalization varies across neighbors

Slide from Stanford Snap Lab

Graph Convolutional Networks

- Semi-supervised learning on Zachary's Karate Club Network
- Only one node from each class is labeled
- 300 iterations

Video from Thomas Kipf

Graph Convolutional Networks

Table 2: Summary of results in terms of classification accuracy (in percent).

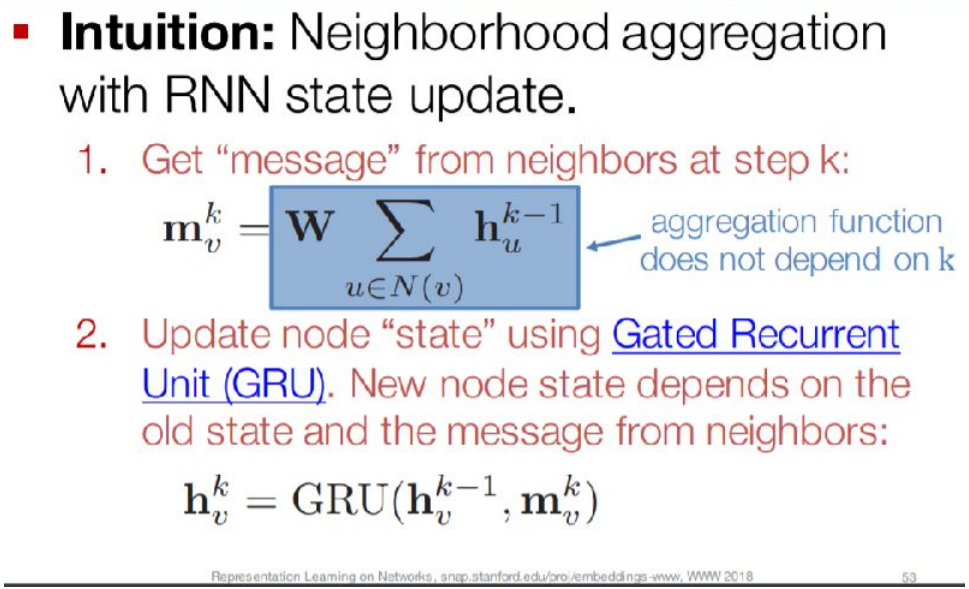
Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0**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

Table 3: Comparison of propagation models.

Description		Propagation model	Citeseer	Cora	Pubmed
Chebyshev filter (Eq. 5)	$K = 3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
	$K = 2$		69.6	81.2	73.8
1st-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
1st-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

Gated Graph Neural Networks

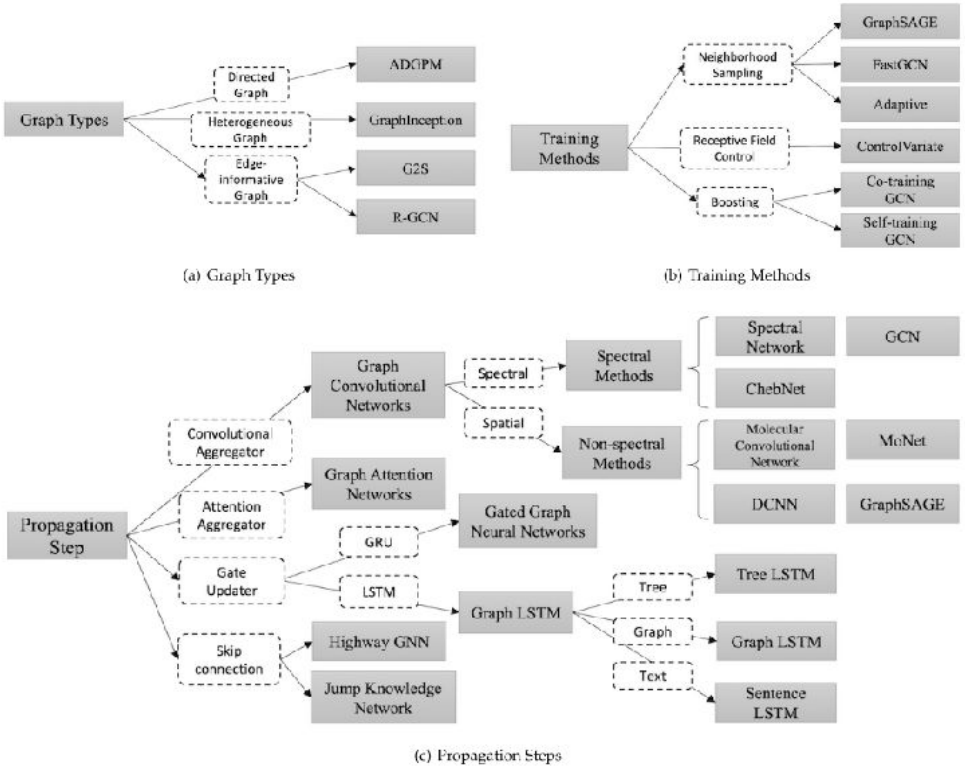
- Uses LSTM for aggregation
- Allows for deep graph networks without worrying about overfitting or vanishing/exploding gradient



Li et al., 2016

Slide from Stanford Snap Lab

Many more variants...perhaps too many



Generalizations

- Message Passing Neural Networks (MPNN)
  - Unifies GNN and GCN
  - Message Passing Phase that constructs message based on local neighborhood
  - Update function which interprets message and updates node’s hidden state
  - Readout phase computes feature vector for the whole graph
  - All of these functions can have different settings
- Non-local Neural Networks (NLNN)
  - Unification of self-attention style methods
  - Node update based on neighbor nodes, not edges

Graph Nets

Relational Inductive Biases

- Battaglia et al., 2018
- Combinatorial Generalization
  - New inferences, predictions, and behaviors from known building blocks
  - Relies on humans' mechanism for representing structure and reasoning about relations
- But, modern deep Learning relies on end-to-end design and does away with any explicit structure
- Instead, advocates for use end-to-end and hand-engineering jointly
  - Combination of "nature" and "nurture"
- Biases/constraints on structure and relations is necessary
- Structured representations have entities and relations (aka graphs) at their core
- Thus, argues for graph networks as foundational building block for AI

Relational Inductive Biases

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2.

Relational Inductive Biases

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2.

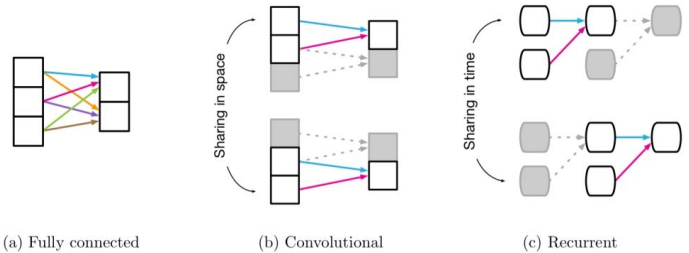


Figure 1: Reuse and sharing in common deep learning building blocks. (a) Fully connected layer, in which all weights are independent, and there is no sharing. (b) Convolutional layer, in which a local kernel function is reused multiple times across the input. Shared weights are indicated by arrows with the same color. (c) Recurrent layer, in which the same function is reused across different processing steps.

Graph Networks

- While RNNs and CNNs use relational inductive biases, they do not generalize to arbitrary relations
- Need a framework that can learn from graphs
- Proposes graph networks, which generalizes MPNN, NLNN, and other variants
- Graph is defined as a 3-tuple
- Where u is global attribute
- V is set of vertices with attributes
- E is set of edges with attributes

$$G = (u, V, E)$$

Graph Networks

- Attributes are updated in a sequence of updates and aggregations, as we've seen before
- Update functions can be parametrized with neural networks
- Same 6 functions are used for all nodes and edges

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

Graph Networks

Algorithm 1 Steps of computation in a full GN block.

function GRAPHNETWORK(E, V, $\mathbf{u}$)

  for $k \in \{1 \dots N^e\}$ **do**

    $\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$ ▷ 1. Compute updated edge attributes

  end for

  for $i \in \{1 \dots N^n\}$ **do**

    let $E'_i = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{r_k=i,\, k=1:N^e}$

    $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\rightarrow v}\left(E'_i\right)$ ▷ 2. Aggregate edge attributes per node

    $\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$ ▷ 3. Compute updated node attributes

  end for

  let $V' = \left\{\mathbf{v}'\right\}_{i=1:N^v}$

  let $E' = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{k=1:N^e}$

  $\bar{\mathbf{e}}' \leftarrow \rho^{e\rightarrow u}\left(E'\right)$ ▷ 4. Aggregate edge attributes globally

  $\bar{\mathbf{v}}' \leftarrow \rho^{v\rightarrow u}\left(V'\right)$ ▷ 5. Aggregate node attributes globally

  $\mathbf{u}' \leftarrow \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$ ▷ 6. Compute updated global attribute

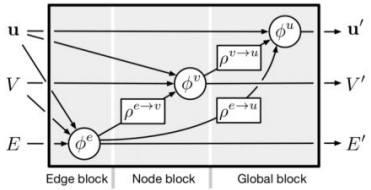
  return $\left(E', V', \mathbf{u}'\right)$

end function

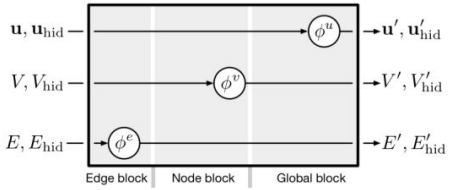
Graph Networks

- 3 main properties
  - Nodes and edges provide a strong relational bias
  - Entities and relations are represented as sets and are thus order invariant
  - Per-edge and per-node functions are shared across the entire network
- 3 design principles
  - Flexible representations
  - Configurable within block structure
  - Composable multi-block architectures

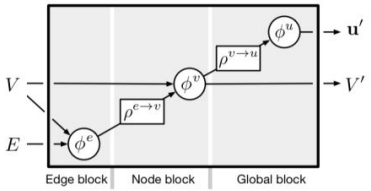
Flexible Representations



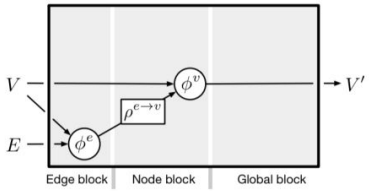
(a) Full GN block



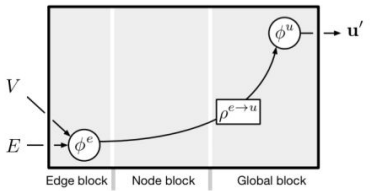
(b) Independent recurrent block



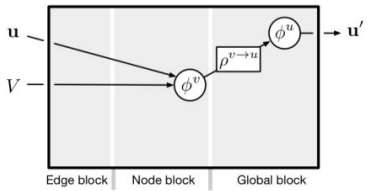
(c) Message-passing neural network



(d) Non-local neural network

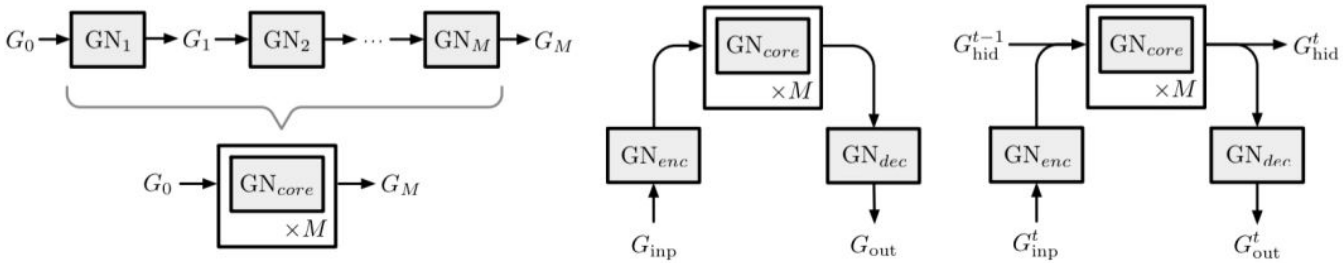


(e) Relation network



(f) Deep set

Composable Multi-block Architectures



(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.

Impact and Challenges

Impact

- Graph networks have applications in many areas
- Works well for any data that can be represented as graphs
  - Node/Graph classification
  - Link prediction
  - Clustering
- Use case in biology, chemistry, physical systems
- Can be used for non-structured data but difficult to generate graphs from raw data
- Whether the idea of relational inductive biases will be adopted remains to be seen

Challenges

1. Shallow Structure
2. Dynamic Graphs
3. Non-structural scenarios
4. Scalability

References

- “Graph Neural Networks: A Review of Methods and Applications” Zhou et al. 2019
- “Gated Graph Sequence Neural Networks” Li et al. 2017
- “The Graph Neural Network Model” Scarselli et al. 2009
- “Relational inductive biases, deep learning ,and graph networks” Battaglia et al. 2018
- The morning paper blog, Adrian Coyler
- Structured Deep Models: Deep Learning on Graphs and Beyond, talk by Thomas Kipf
- “Convolutional Networks on Graphs for Learning Molecular Fingerprints” Duvenaud et al. 2015
- “Semi-Supervised Classification with Graph Convolutional Networks” Kipf & Welling 2017
- “Graph Convolutional Networks”, Kipf 2016
- “Representation Learning on Graphs: Methods and Applications” Hamilton et al. 2017
- “Geometric Deep Learning: Going Beyond Euclidean Data” Bronstein et al. 2017
- “Gated Graph Sequence Neural Networks” Li et al., 2016