Graph networks for developing AI

Based on: Relational inductive biases, deep learning, and graph networks

Student: Andrey Smirdin

21.11.18

Introduction

“... the world is compositional, or at least, we understand it in compositional terms. When learning, we either fit new knowledge into our existing structured representations, or adjust the structure itself to better accommodate (and make use of) the new and the old. “

Modern deep learning

Done incredibly well avoiding explicit structure.

But seems to be finding its limitations in some areas.

We need a structured representation with structured computations!

What is structure?

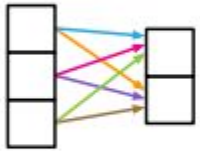
- An entity with attributes.
- A relation is a property between entities.
- A rule is a function that maps entities and relations to other entities and relations.

Relational inductive biases

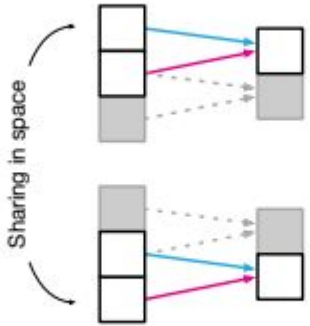
Inductive bias is how an algorithm prioritises one solution over another, independent of the observed data.

Relational inductive bias imposes constraints on relationships and interactions among entities.

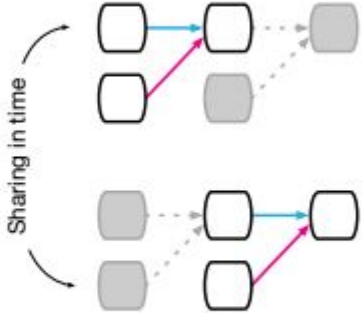
Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations



(a) Fully connected



(b) Convolutional



(c) Recurrent

Computations over sets and graphs

What if we want to find:

- Barycenter
- Velocities of planets
- Velocities of planets and their moons

Graph networks

Consist of:

- Nodes
- Edges
- Attributes

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

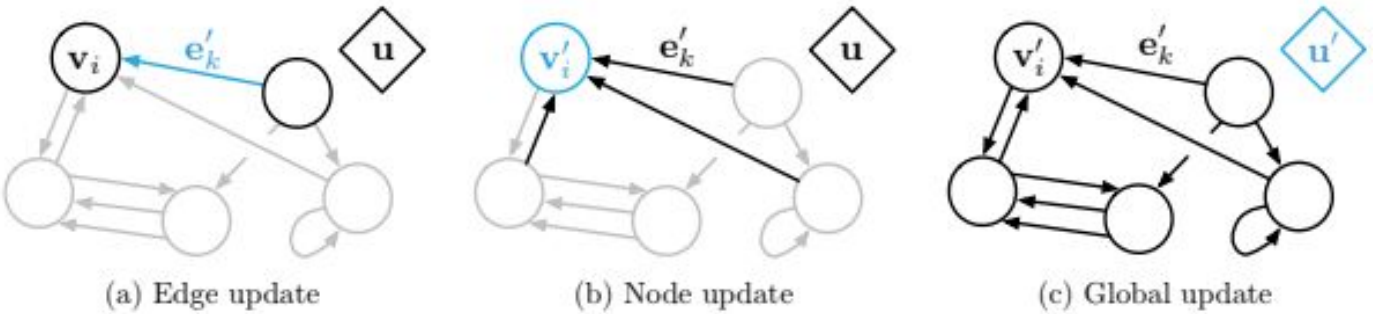
  $\bar{\mathbf{e}}' \leftarrow \rho^{e\rightarrow u}\left(E'\right)$ ▷ 4. Aggregate edge attributes globally

  $\bar{\mathbf{v}}' \leftarrow \rho^{v\rightarrow u}\left(V'\right)$ ▷ 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



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

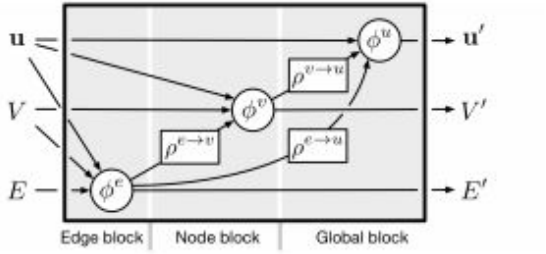
Let’s consider an example graph where nodes represent rubber balls and edges are springs connecting them.

Why GN are interesting?

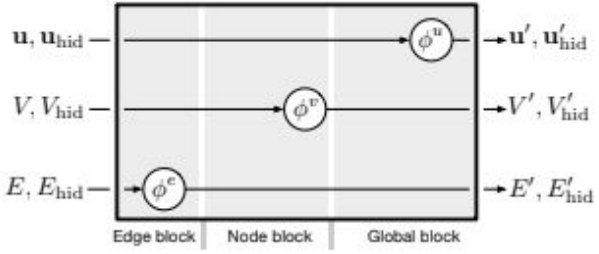
- The nodes and the edges between provide strong relational inductive biases (e.g. the absence of an edge between two nodes means that they should not directly influence each other).
- Entities and relations are represented as sets, and thus are invariant to ordering and permutations.
- Since the per-edge and per-node functions are reused across all edges and nodes respectively, GNs automatically support a form of combinatorial generalisation.

Design principles

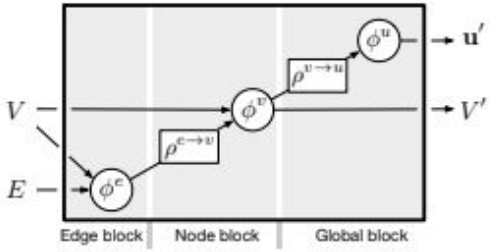
- Flexible representations.
- Configurable within-block structure
- Composable multi-block architectures



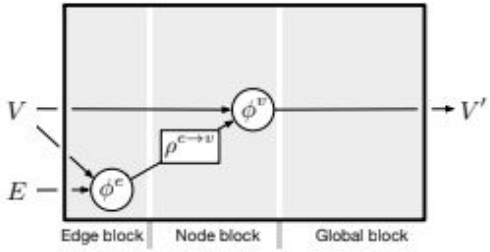
(a) Full GN block



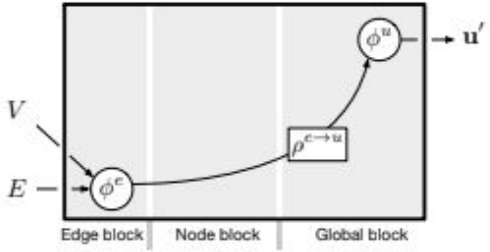
(b) Independent recurrent block



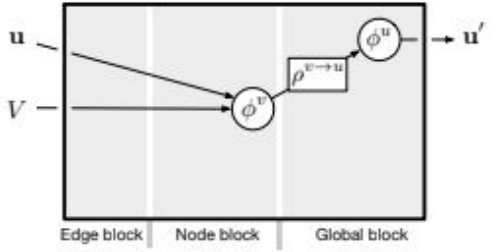
(c) Message-passing neural network



(d) Non-local neural network

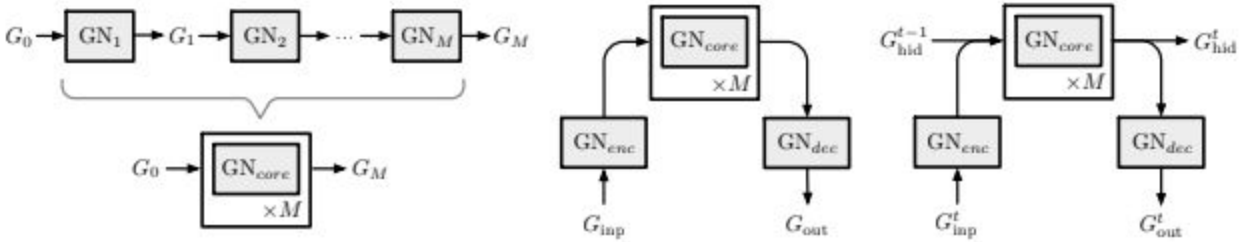


(e) Relation network



(f) Deep set

GN architectures

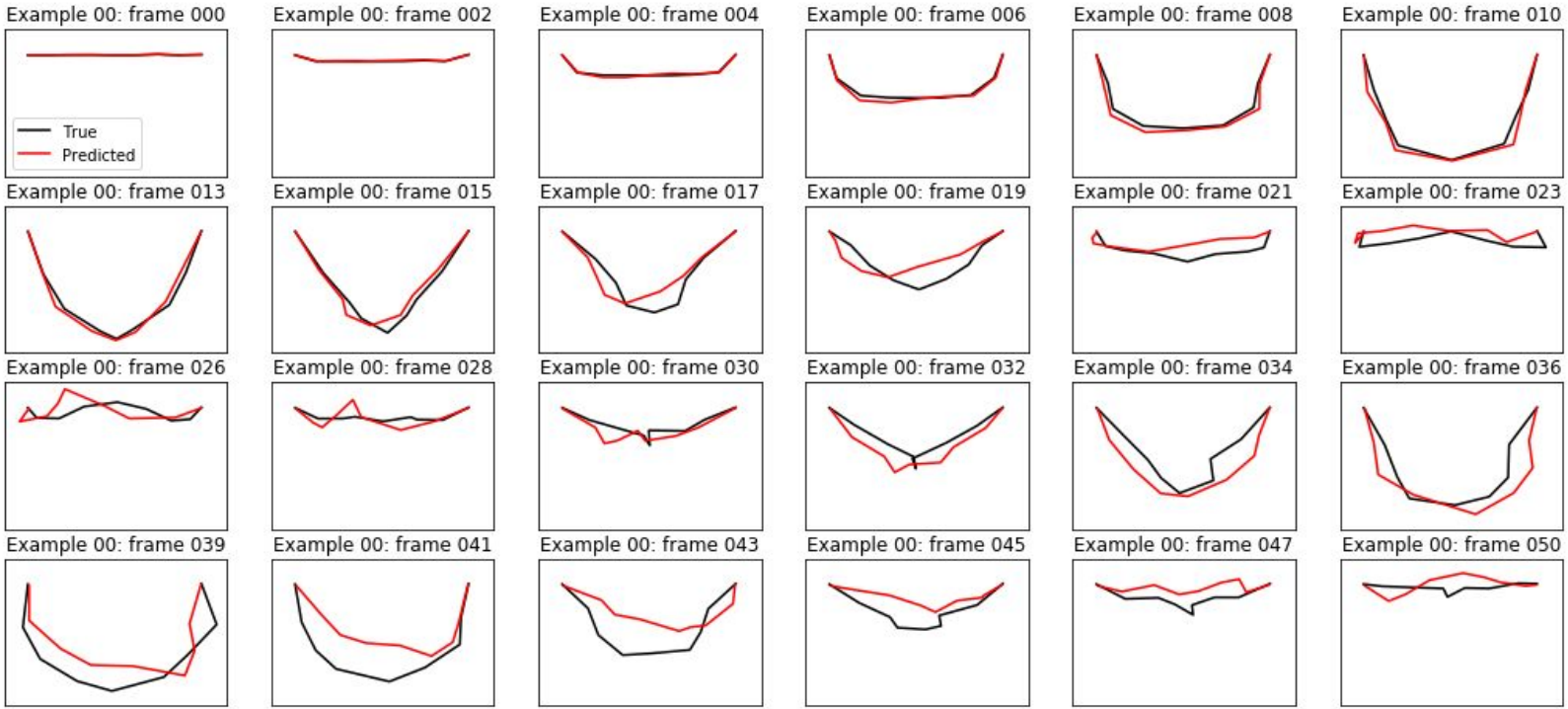


(a) Composition of GN blocks (b) Encode-process-decode (c) Recurrent GN architecture

Graph Nets open-source software library

Examples:

- Shortest path
- Sort
- Physical system


Example 00: frame 000
True
Predicted
Example 00: frame 002
Example 00: frame 004
Example 00: frame 006
Example 00: frame 008
Example 00: frame 010
Example 00: frame 013
Example 00: frame 015
Example 00: frame 017
Example 00: frame 019
Example 00: frame 021
Example 00: frame 023
Example 00: frame 026
Example 00: frame 028
Example 00: frame 030
Example 00: frame 032
Example 00: frame 034
Example 00: frame 036
Example 00: frame 039
Example 00: frame 041
Example 00: frame 043
Example 00: frame 045
Example 00: frame 047
Example 00: frame 050

Some problems

- Where do the for building GN comes from?
- How to implement recursion, control flow, and conditional iteration?
- How to adaptively modify graph structures during the course of computation?

Thank you!