Simulation of complex physical system with Graph Networks

Pietropolli Gloria

Università degli studi di Trieste

03 marzo 2021

What are the topics of this seminar?

- **Graph Network** (GN)

What are the topics of this seminar?

- **Graph Network** (GN)
  - Definition

What are the topics of this seminar?

- **Graph Network** (GN)
  - Definition
  - Computational steps within a GN

What are the topics of this seminar?

- **Graph Network** (GN)
  - Definition
  - Computational steps within a GN
  - Strength of Graph Network:
    - Flexible representation
    - Configurable within-block structure
    - Composable multi-block architectures

What are the topics of this seminar?

- **Graph Network** (GN)
  - Definition
  - Computational steps within a GN
  - Strength of Graph Network:
    - Flexible representation
    - Configurable within-block structure
    - Composable multi-block architectures
- **Simulate complex physics with Graph Network**

What are the topics of this seminar?

- **Graph Network** (GN)
  - Definition
  - Computational steps within a GN
  - Strength of Graph Network:
    - Flexible representation
    - Configurable within-block structure
    - Composable multi-block architectures
- **Simulate complex physics with Graph Network**
  - why use GN to simulate complex physics?

What are the topics of this seminar?

- **Graph Network** (GN)
  - Definition
  - Computational steps within a GN
  - Strength of Graph Network:
    - Flexible representation
    - Configurable within-block structure
    - Composable multi-block architectures
- **Simulate complex physics with Graph Network**
  - why use GN to simulate complex physics?
  - GNS (graph network-based) simulators
    - definition
    - implementation details

Relational inductive biases

Relational inductive biases are defined as inductive biases which impose constraints on relationship and interactions among entities in a learning process.

Relational inductive biases

Relational inductive biases are defined as inductive biases which impose constraints on relationship and interactions among entities in a learning process.

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

Relational inductive biases

Relational inductive biases are defined as inductive biases which impose constraints on relationship and interactions among entities in a learning process.

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

Using *relational inductive biases* within deep learning architectures can facilitate learning about entities, relations and rules for composing them.

Relational inductive biases

Relational inductive biases are defined as inductive biases which impose constraints on relationship and interactions among entities in a learning process.

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

Using *relational inductive biases* within deep learning architectures can facilitate learning about entities, relations and rules for composing them.

Graph Network have a **strong relational inductive biases**

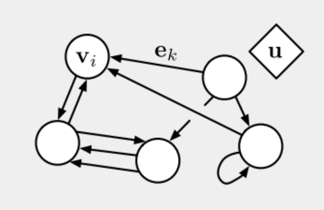
Graph Network

GN is a Neural Network that operates on graphs and structure their computation accordingly.

Graph Network

GN is a Neural Network that operates on graphs and structure their computation accordingly.

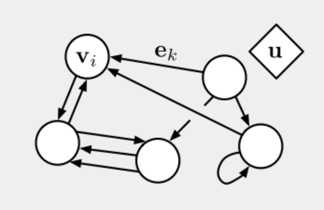
- *main unit:* GN block
- *entities:* nodes
- *relations:* edges
- *system proprieties:* global attributes



Graph Network

GN is a Neural Network that operates on graphs and structure their computation accordingly.

- *main unit:* GN block
- *entities:* nodes
- *relations:* edges
- *system proprieties:* global attributes



A **GN block** is a "graph-to-graph module" which:

1. takes a graph as *input*
2. performs *computation* over the structure
3. returns a graph as *output*

Internal Structure of a GN block

Internal Structure of a GN block

- $G = (u, V, E)$
  - u : *global attribute*
  - $V = \{v_i\}_{i=1:N^v}$: set of *nodes*
  - $E = \{(e_k, r_k, s_k)\}_{k=1:N^e}$: set of *edges*

Internal Structure of a GN block

- $G = (u, V, E)$
  - u : *global attribute*
  - $V = \{v_i\}_{i=1:N^v}$: set of *nodes*
  - $E = \{(e_k, r_k, s_k)\}_{k=1:N^e}$: set of *edges*

- <u>UPDATE FUNCTIONS</u>
  - $e'_k = \phi^e(e_k, v_{r_k}, v_{s_k}, u)$: compute per-edge updates
  - $v'_i = \phi^v(\bar{e}_i, v_i, u)$: compute per-nodes update
  - $u' = \phi^u(\bar{e}, \bar{v}, u)$: compute the global update

Internal Structure of a GN block

- $G = (u, V, E)$
  - u : *global attribute*
  - $V = \{v_i\}_{i=1:N^v}$: set of *nodes*
  - $E = \{(e_k, r_k, s_k)\}_{k=1:N^e}$: set of *edges*

- <u>UPDATE FUNCTIONS</u>
  - $e'_k = \phi^e(e_k, v_{r_k}, v_{s_k}, u)$: compute per-edge updates
  - $v'_i = \phi^v(\bar{e}_i, v_i, u)$: compute per-nodes update
  - $u' = \phi^u(\bar{e}, \bar{v}, u)$: compute the global update

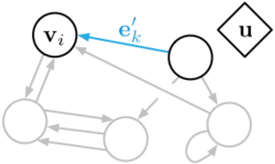
- <u>AGGREGATION FUNCTION</u>

take a set as an input and reduce it to a single element which represent the aggregated information

- $\bar{e}'_i = \rho^{e \to v}(E'_i)$
- $\bar{e}' = \rho^{e \to u}(E')$
- $\bar{v}' = \rho^{v \to u}(V')$

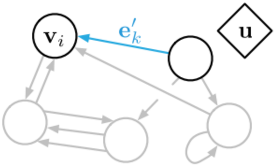
- $E'_i = \{(e'_k, r_k, s_k)\}_{r_k = i, k=1:N^e}$
- $E' = \bigcup_i E'_i = \{(e'_k, r_k, s_k)\}_{k=1:N^e}$
- $V' = \{v'_i\}_{i=1:N^v}$

Computational Steps within a Graph Network



(a) Edge update

Computational Steps within a Graph Network

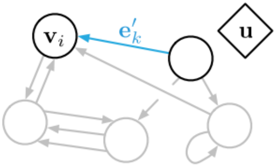


(a) Edge update

- ϕ^e is applied to compute an update edge attribute

$$\boxed{e'_k \leftarrow \phi^e(e_k, v_{r_k}, v_{s_k}, u)}$$

Computational Steps within a Graph Network



(a) Edge update

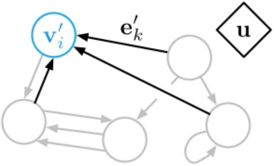
- ϕ^e is applied to compute an update edge attribute

$$\boxed{e'_k \leftarrow \phi^e(e_k, v_{r_k}, v_{s_k}, u)}$$

- $\rho^{e \to v}$ is applied and aggregates the edge updates for edge that project to vertex i

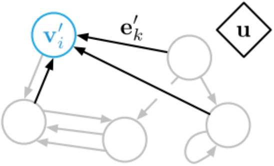
$$\boxed{\bar{e}'_i \leftarrow \rho^{e \to v}(E'_i)}$$

Computational Steps within a Graph Network



(b) Node update

Computational Steps within a Graph Network

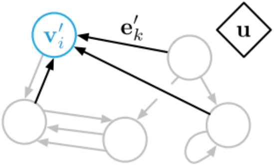


(b) Node update

- ϕ^v is applied to compute an update node attribute

$$\boxed{v'_i \leftarrow \phi^v(\bar{e}_i, v_i, u)}$$

Computational Steps within a Graph Network



(b) Node update

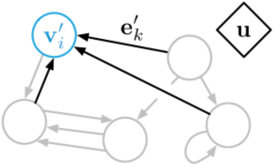
- ϕ^v is applied to compute an update node attribute

$$\boxed{v_i' \leftarrow \phi^v(\bar{e}_i, v_i, u)}$$

- $\rho^{e\rightarrow u}$ is applied and aggregates all edge updates

$$\boxed{\bar{e'} \leftarrow \rho^{e\rightarrow u}(E')}$$

Computational Steps within a Graph Network



(b) Node update

- ϕ^v is applied to compute an update node attribute

$$\boxed{v'_i \leftarrow \phi^v(\bar{e}_i, v_i, u)}$$

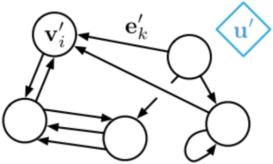
- $\rho^{e\rightarrow u}$ is applied and aggregates all edge updates

$$\boxed{\bar{e'} \leftarrow \rho^{e\rightarrow u}(E')}$$

- $\rho^{e\rightarrow v}$ is applied and aggregates all node updates

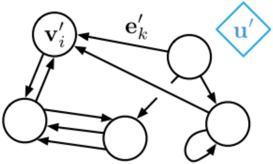
$$\boxed{\bar{v'} \leftarrow \rho^{v\rightarrow u}(V')}$$

Computational Steps within a Graph Network



(c) Global update

Computational Steps within a Graph Network



(c) Global update

- ϕ^u is applied once per graph, to compute an update global attribute

$$\boxed{u' \leftarrow \phi^u(\bar{e}, \bar{v}, u)}$$

Relational inductive biases in graph network

GN framework imposes several strong **relational inductive biases**:

Relational inductive biases in graph network

GN framework imposes several strong **relational inductive biases**:

1. Graphs can express arbitrary relationship among entities

   ⇒ GN's input determines how representations *interact* and *are isolated*

   The assumption that two entities have a relationship is expressed by an edge between them

Relational inductive biases in graph network

GN framework imposes several strong **relational inductive biases**:

1. Graphs can express arbitrary relationship among entities

   $\Rightarrow$ GN's input determines how representations *interact* and *are isolated*

   The assumption that two entities have a relationship is expressed by an edge between them

2. Graphs represents entities and their relations as set (invariant to permutations)

   $\Rightarrow$ GNs is invariant to the *order* of these elements

Relational inductive biases in graph network

GN framework imposes several strong **relational inductive biases**:

1. Graphs can express arbitrary relationship among entities

   $\Rightarrow$ GN's input determines how representations *interact* and *are isolated*

   The assumption that two entities have a relationship is expressed by an edge between them

2. Graphs represents entities and their relations as set (invariant to permutations)

   $\Rightarrow$ GNs is invariant to the *order* of these elements

3. GN's per-edge and per-node functions are reused across all edges and nodes

   $\Rightarrow$ GNs automatically support a form of *combinatorial generalization*

   $\Rightarrow$ A single GN can operate on graphs of **different size** (number of edges/nodes) and **different shapes** (edge connectivity)

Strength of GN: FLEXIBLE REPRESENTATION

GN supports *flexible representations* in two way:

Strength of GN: FLEXIBLE REPRESENTATION

GN supports *flexible representations* in two way:

1. **ATTRIBUTES** the global/node/edge attributes of a GN block can use arbitrary representational formats:
   - real-valued vectors
   - tensor
   - sequences
   - set
   - graph
   - ...

   depending on the problem we are facing

Strength of GN: FLEXIBLE REPRESENTATION

GN supports *flexible representations* in two way:

1. **ATTRIBUTES** the global/node/edge attributes of a GN block can use arbitrary representational formats:
   - real-valued vectors
   - tensor
   - sequences
   - set
   - graph
   - ...

   depending on the problem we are facing

2. **GRAPH STRUCTURE** when defining how input data are represented as a graph there are two possibilities:
   - input explicitly specifies the relational structure
     - social network
     - optimization problems
     - chemical graphs
     - road networks
     - ...
   - the relational structure must be inferred or assumed

Strength of GN: CONFIGURABLE WITHIN-BLOCK STRUCTURE

The **functions** within a GN block can be configured in different ways

Strength of GN: CONFIGURABLE WITHIN-BLOCK STRUCTURE

The **functions** within a GN block can be configured in different ways

- The ϕ implementation used neural networks:
  - $\phi^e(e_k, v_{r_k}, v_{s_k}, u) = NN_e([e_k, v_{r_k}, v_{s_k}, u])$
  - $\phi^v(\bar{e}_i, v_i, u) = NN_v(\bar{e}_i, v_i, u)$
  - $\phi^u(\bar{e}, \bar{v}, u) = NN_u(\bar{e}, \bar{v}, u)$

Strength of GN: CONFIGURABLE WITHIN-BLOCK STRUCTURE

The **functions** within a GN block can be configured in different ways

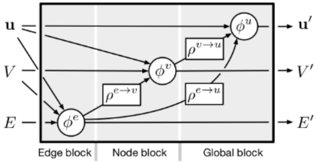
- The ϕ implementation used neural networks:
  - $\phi^e(e_k, v_{r_k}, v_{s_k}, u) = NN_e([e_k, v_{r_k}, v_{s_k}, u])$
  - $\phi^v(\bar{e}_i, v_i, u) = NN_v(\bar{e}_i, v_i, u)$
  - $\phi^u(\bar{e}, \bar{v}, u) = NN_u(\bar{e}, \bar{v}, u)$
- the ρ implementations used element-wise summation

Strength of GN: CONFIGURABLE WITHIN-BLOCK STRUCTURE

The **structure** within a GN block can be configured in different ways

Strength of GN: CONFIGURABLE WITHIN-BLOCK STRUCTURE

The **structure** within a GN block can be configured in different ways



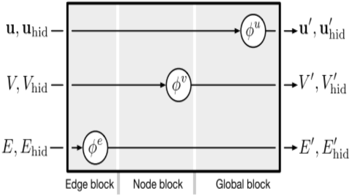
(a) Full GN block

Strength of GN: CONFIGURABLE WITHIN-BLOCK STRUCTURE

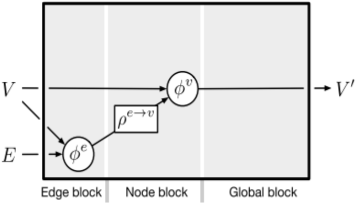
...But there are a variety of other architecture that we can express in a GN block!

Strength of GN: CONFIGURABLE WITHIN-BLOCK STRUCTURE

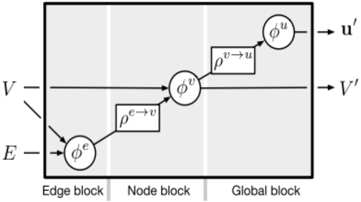
...But there are a variety of other architecture that we can express in a GN block!



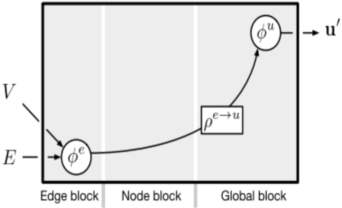
(b) Independent recurrent block



(d) Non-local neural network



(c) Message-passing neural network



(e) Relation network

Strength of GN: COMPOSABLE MULTI-BLOCK ARCHITECTURES

$\boxed{\textit{IDEA}}$ constructing complex architectures by composing GN blocks

Strength of GN: COMPOSABLE MULTI-BLOCK ARCHITECTURES

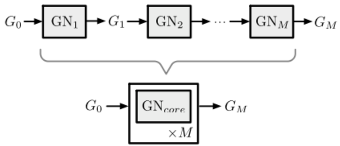
$\boxed{IDEA}$ constructing complex architectures by composing GN blocks

- The graph-to-graph I/O interface ensures that the **output** of one GN block can be passed as **input** to another
- Arbitrary numbers of GN blocks can be composed

Strength of GN: COMPOSABLE MULTI-BLOCK ARCHITECTURES

IDEA constructing complex architectures by composing GN blocks

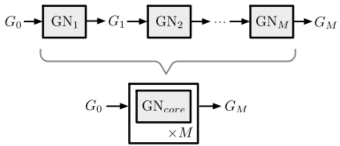
- The graph-to-graph I/O interface ensures that the **output** of one GN block can be passed as **input** to another
- Arbitrary numbers of GN blocks can be composed



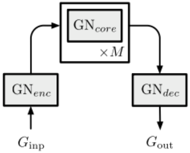
Strength of GN: COMPOSABLE MULTI-BLOCK ARCHITECTURES

$\boxed{\textit{IDEA}}$ constructing complex architectures by composing GN blocks

- The graph-to-graph I/O interface ensures that the **output** of one GN block can be passed as **input** to another
- Arbitrary numbers of GN blocks can be composed



A common architecture design is the *encode-process-decode* configuration



Why to simulate complex physics with Graph Networks?

1. WHY SIMULATE COMPLEX PHYSICS?

Why to simulate complex physics with Graph Networks?

1. WHY SIMULATE COMPLEX PHYSICS?

Traditional simulators can be **very expensive** to create and use

Why to simulate complex physics with Graph Networks?

1. WHY SIMULATE COMPLEX PHYSICS?

   Traditional simulators can be **very expensive** to create and use

2. WHY USE GRAPH NETWORK-based SIMULATORS (GNS)?

   - This framework imposes **strong inductive biases**

Why to simulate complex physics with Graph Networks?

1. WHY SIMULATE COMPLEX PHYSICS?

   Traditional simulators can be **very expensive** to create and use

2. WHY USE GRAPH NETWORK-based SIMULATORS (GNS)?

   - This framework imposes **strong inductive biases**
   - **Rich physical state** can be represented by graph of interacting particles

Why to simulate complex physics with Graph Networks?

1. WHY SIMULATE COMPLEX PHYSICS?

   Traditional simulators can be **very expensive** to create and use

2. WHY USE GRAPH NETWORK-based SIMULATORS (GNS)?

   - This framework imposes **strong inductive biases**
   - **Rich physical state** can be represented by graph of interacting particles
   - **Complex dynamics** can be approximated by learned message-passing among nodes

Why to simulate complex physics with Graph Networks?

1. WHY SIMULATE COMPLEX PHYSICS?

   Traditional simulators can be **very expensive** to create and use

2. WHY USE GRAPH NETWORK-based SIMULATORS (GNS)?

   - This framework imposes **strong inductive biases**
   - **Rich physical state** can be represented by graph of interacting particles
   - **Complex dynamics** can be approximated by learned message-passing among nodes
   - GNS could learn to simulate a **wide range** of physical system (where fluids, rigid solids and deformable materials interact with one another)

Why to simulate complex physics with Graph Networks?

1. WHY SIMULATE COMPLEX PHYSICS?

   Traditional simulators can be **very expensive** to create and use

2. WHY USE GRAPH NETWORK-based SIMULATORS (GNS)?

   - This framework imposes **strong inductive biases**
   - **Rich physical state** can be represented by graph of interacting particles
   - **Complex dynamics** can be approximated by learned message-passing among nodes
   - GNS could learn to simulate a **wide range** of physical system (where fluids, rigid solids and deformable materials interact with one another)
   - GNS **generalized** well:
     - too much larger system
     - longer time scales

     than those on which was trained

GNS model framework : LEARNABLE SIMULATOR

What is a simulator?

GNS model framework : LEARNABLE SIMULATOR

What is a simulator?

- $X^t \in \mathcal{X}$: *state of the world at time t*

GNS model framework : LEARNABLE SIMULATOR

What is a simulator?

- $X^t \in \mathcal{X}$: *state of the world at time t*
- $X^{t_0:K} = (X^{t_0}, \cdots, X^{t_k})$: *trajectory*

GNS model framework : LEARNABLE SIMULATOR

What is a simulator?

- $X^t \in \mathcal{X}$: *state of the world at time t*
- $X^{t_0:K} = (X^{t_0}, \cdots, X^{t_k})$: *trajectory*
- $s : \mathcal{X} \to \mathcal{X}$: *simulator*

GNS model framework : LEARNABLE SIMULATOR

What is a simulator?

- $X^t \in \mathcal{X}$: *state of the world at time t*
- $X^{t_0:K} = (X^{t_0}, \cdots, X^{t_k})$: *trajectory*
- $s : \mathcal{X} \rightarrow \mathcal{X}$: *simulator*
- $\tilde{X}^{t_0:K} = (X^{t_0}, \cdots, \tilde{X}^{t_k})$: *simulated trajectory*, computed by $\tilde{X}^{t_{k+1}} = s(\tilde{X}^{t_k})$

GNS model framework : LEARNABLE SIMULATOR

What is a simulator?

- $X^t \in \mathcal{X}$: *state of the world at time t*
- $X^{t_0:K} = (X^{t_0}, \cdots, X^{t_k})$: *trajectory*
- $s : \mathcal{X} \to \mathcal{X}$: *simulator*
- $\tilde{X}^{t_0:K} = (X^{t_0}, \cdots, \tilde{X}^{t_k})$: *simulated trajectory*, computed by $\tilde{X}^{t_{k+1}} = s(\tilde{X}^{t_k})$

A learnable simulator s_θ computes dynamic information with a *parametrized function approximator*:

$$\boxed{d_\theta : \mathcal{X} \to \mathcal{Y}}$$

- $Y \in \mathcal{Y}$ represents dynamic information
- the parameters θ can be optimized for some training objective

Simulation as Message-Passing on Graph

- **Particle-based representation** of the physical system : $X = (x_0, x_1, \cdots, x_N)$
- Physical dynamics are approximated by **interactions** among the particles

Simulation as Message-Passing on Graph

- **Particle-based representation** of the physical system : $X = (x_0, x_1, \cdots, x_N)$
- Physical dynamics are approximated by **interactions** among the particles
- Particle-based simulation can be viewed as message-passing on a graph :
  - nodes corresponds to particles
  - edges corresponds to pairwise relations among particles

$\Downarrow$

Tanks to this correspondence we can define d_θ based on GNs

Let's see GNS implementation details, i.e.

- **INPUT** AND **OUTPUT** representations
- **ENCODER** details
- **PROCESSOR** details
- **DECODER** details

INPUT AND OUTPUT REPRESENTATIONS

Particle **INPUT**

$$x_i^{t_k} = [p_i^{t_k}, \dot{p}_i^{t_k - C+1}, \dots, \dot{p}_i^{t_k}, f_i]$$

- $p_i^{t_k}$ position
- $\dot{p}_i^{t_k - C+1}, \dots, \dot{p}_i^{t_k}$ sequence of C $(= 5)$ previous velocities
- f_i features that capture static material properties

INPUT AND OUTPUT REPRESENTATIONS

Particle **INPUT**

$$x_i^{t_k} = [p_i^{t_k}, \dot{p}_i^{t_k - C+1}, \ldots, \dot{p}_i^{t_k}, f_i]$$

- $p_i^{t_k}$ position
- $\dot{p}_i^{t_k - C+1}, \ldots, \dot{p}_i^{t_k}$ sequence of C $(= 5)$ previous velocities
- f_i features that capture static material properties

Particle **OUTPUT**

The prediction targets for supervised learning are the per-particle average acceleration $\ddot{p}_i$

Simulation as message-passing on a graph

ENCODER

The **ENCODER** : $\mathcal{X} \to \mathcal{G}$ embeds the particle-based state representation X as a latent graph:

$$\boxed{G^0 = ENCODER(X)} \quad \text{where } G = (V, E, u),\ v_i \in V,\ e_{i,j} \in E$$

ENCODER

The <u>**ENCODER**</u> : $\mathcal{X} \rightarrow \mathcal{G}$ embeds the particle-based state representation X as a latent graph:

$$\boxed{G^0 = ENCODER(X)} \quad \text{where } G = (V, E, u),\ v_i \in V,\ e_{i,j} \in E$$

The graph structure is construct:

- assigning a **node** to each particle
- adding **edges** between particles within a *connectivity radius R*
  - reflect local interactions of particles
  - kept constant for all simulation

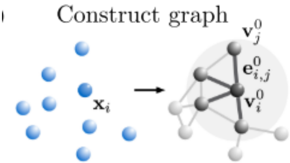
ENCODER

The **ENCODER** : $\mathcal{X} \rightarrow \mathcal{G}$ embeds the particle-based state representation X as a latent graph:

$$\boxed{G^0 = ENCODER(X)} \quad \text{where } G = (V, E, u),\ \ v_i \in V,\ \ e_{i,j} \in E$$

The graph structure is construct:

- assigning a **node** to each particle
- adding **edges** between particles within a *connectivity radius R*
  - reflect local interactions of particles
  - kept constant for all simulation



The **ENCODER** implements:

- the node embeddings, $v_i = \epsilon^v(x_i)$
- the edge embeddings, $e_{i,j} = \epsilon^e(r_{i,j})$

as *multilayer perceptons* (MLP)

PROCESSOR

The **PROCESSOR** : $\mathcal{G} \rightarrow \mathcal{G}$ computes interactions among nodes via M steps of learned message-passing.
Is generated a sequence of updated latent graphs:

$$G = (G^1, \cdots, G^M) \quad \text{where } G^{m+1} = GN^{m+1}(G^m)$$

It returns the final graph:

$$\boxed{G^M = PROCESSOR(G^0)}$$

PROCESSOR

The **PROCESSOR** : $\mathcal{G} \rightarrow \mathcal{G}$ computes interactions among nodes via M steps of learned message-passing.
Is generated a sequence of updated latent graphs:

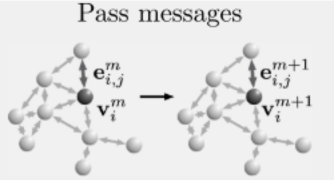
$$G = (G^1, \cdots, G^M) \quad \text{where } G^{m+1} = GN^{m+1}(G^m)$$

It returns the final graph:

$$\boxed{G^M = PROCESSOR(G^0)}$$

The processors use a stack of M GNs

- with **identical structure**
- **MLPs** as internal edge and node update functions



DECODER

The **DECODER** : $\mathcal{G} \rightarrow \mathcal{Y}$ extracts dynamics information from the nodes of the final latent graph

$$\boxed{Y = DECODER(G^M)}$$

DECODER

The **DECODER** : $\mathcal{G} \rightarrow \mathcal{Y}$ extracts dynamics information from the nodes of the final latent graph

$$\boxed{Y = DECODER(G^M)}$$

The **DECODER** implements:

- the function, $y_i = \delta^v(v_i^M)$, that extracts dynamics informations from the nodes of the final graph

as *multilayer perceptons* (MLP)

DECODER

The **DECODER** : $\mathcal{G} \to \mathcal{Y}$ extracts dynamics information from the nodes of the final latent graph

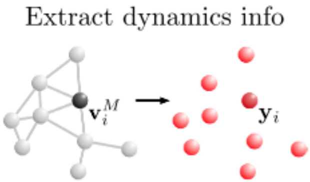
$$\boxed{Y = DECODER(G^M)}$$

The **DECODER** implements:

- the function, $y_i = \delta^v(v_i^M)$, that extracts dynamics informations from the nodes of the final graph

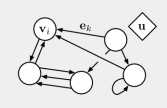
as *multilayer perceptons* (MLP)

After the DECODER, the **future position** and **velocity** are updated using EULER integrator



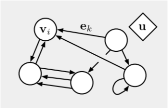
TO SUM UP

- Introduction of the concept of Graph Network

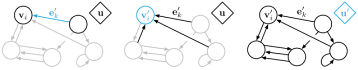


TO SUM UP

- Introduction of the concept of Graph Network



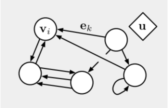
- Computational steps within a Graph Network



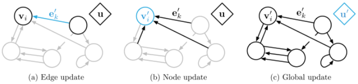
(a) Edge update (b) Node update (c) Global update

TO SUM UP

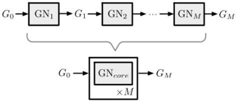
- Introduction of the concept of Graph Network



- Computational steps within a Graph Network



- Possibility to construct complex architecture by composing GN blocks



TO SUM UP

Rich physical state can be represented by graph of interacting particles
Particle-based simulations can be viewed as message-passing on a graph

TO SUM UP

Rich physical state can be represented by graph of interacting particles
Particle-based simulations can be viewed as message-passing on a graph

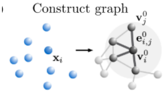
$\Rightarrow$ The learnable simulator d_θ can be defined by using the following *encode-process-decode* configuration

TO SUM UP

Rich physical state can be represented by graph of interacting particles
Particle-based simulations can be viewed as message-passing on a graph

$\Rightarrow$ The learnable simulator d_θ can be defined by using the following *encode-process-decode* configuration

- How to construct the graph starting from the particle-based representations

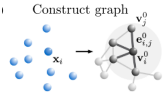


TO SUM UP

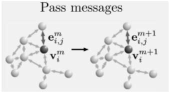
Rich physical state can be represented by graph of interacting particles
Particle-based simulations can be viewed as message-passing on a graph

$\Rightarrow$ The learnable simulator d_θ can be defined by using the following *encode-process-decode* configuration

- How to construct the graph starting from the particle-based representations



- How to compute interactions among nodes by using steps of learned message-passing



TO SUM UP

Rich physical state can be represented by graph of interacting particles
Particle-based simulations can be viewed as message-passing on a graph

$\Rightarrow$ The learnable simulator d_θ can be defined by using the following *encode-process-decode* configuration

- How to construct the graph starting from the particle-based representations
- How to compute interactions among nodes by using steps of learned message-passing
- How to extract dynamic information from the nodes of the final latent graph