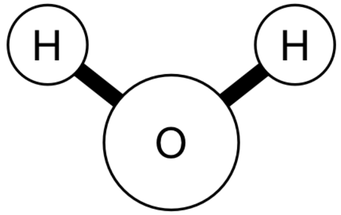
Graph networks for learning about complex systems

Peter Battaglia

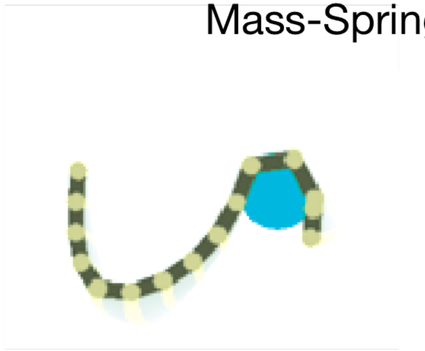
Workshop on Graph Neural Networks
Siena, Italy
July 22, 2019

Many complex systems are structured

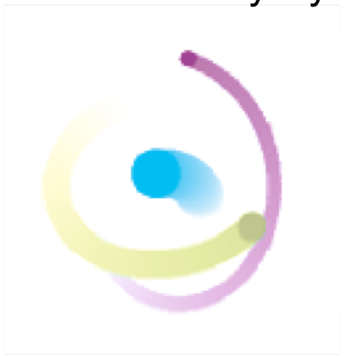
Molecule



Mass-Spring System



n-body System

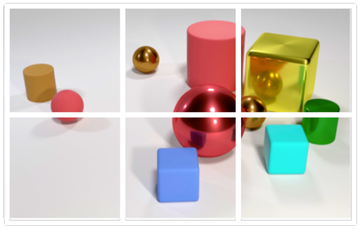


Rigid Body System

Sentence

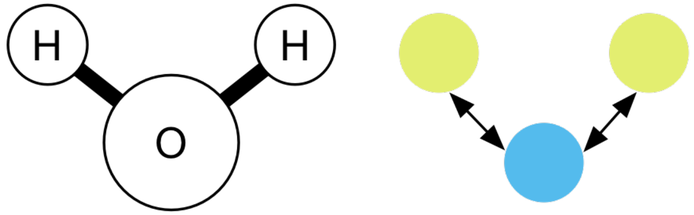
The brown
dog jumped.

Image

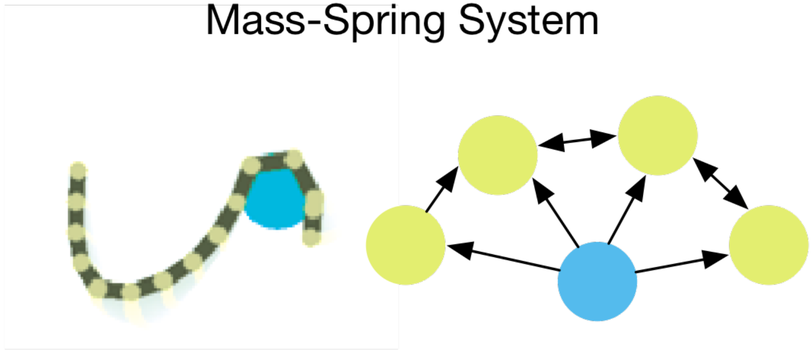


Many complex systems are structured

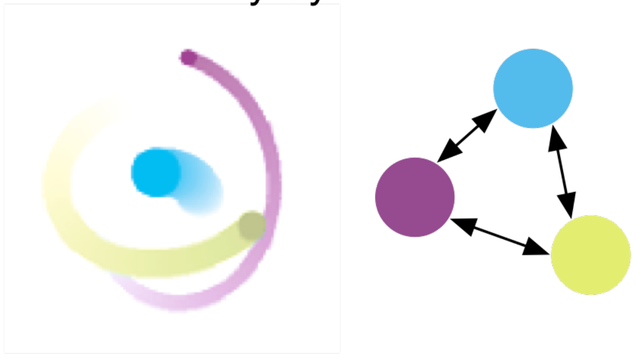
Molecule



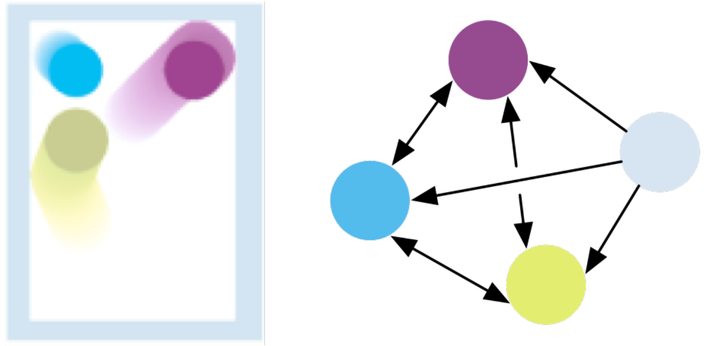
Mass-Spring System



n-body System



Rigid Body System



Sentence and Parse Tree

The brown dog jumped.

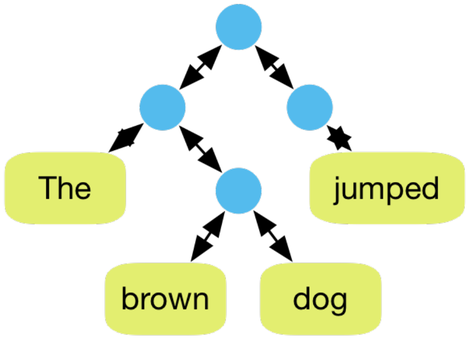
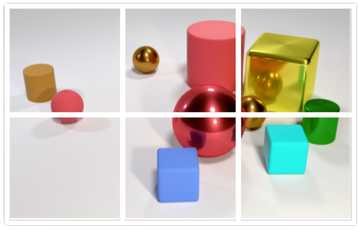
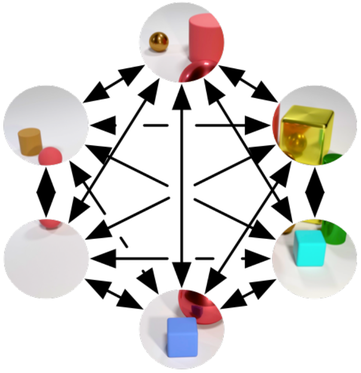


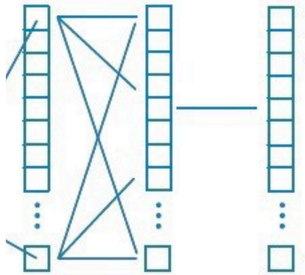
Image and Fully-Connected Scene Graph



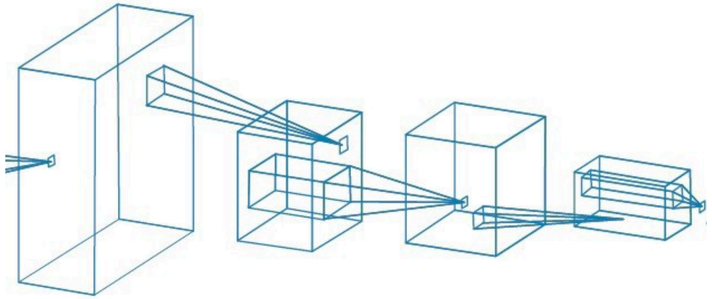


The standard deep learning toolkit…

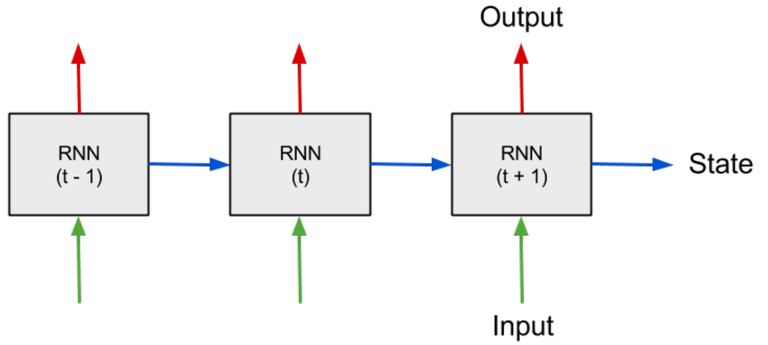
Multi-layer perceptron (MLP)



Convolutional neural network (CNN)



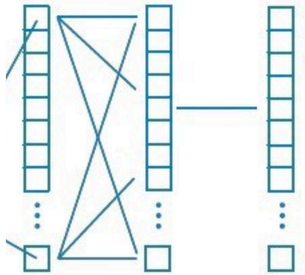
Recurrent neural network (RNN)



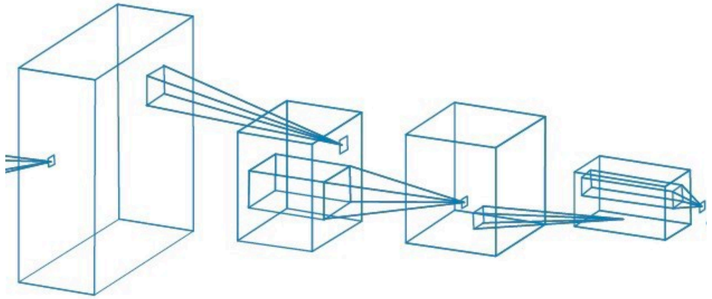
…is not well-suited to reasoning over structured representations.

The standard deep learning toolkit…

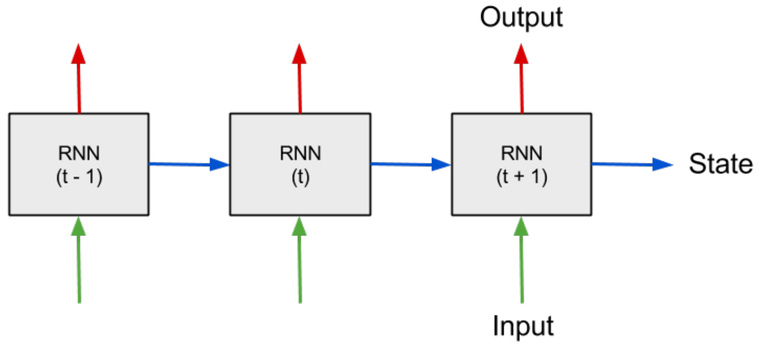
Multi-layer perceptron (MLP)



Convolutional neural network (CNN)



Recurrent neural network (RNN)

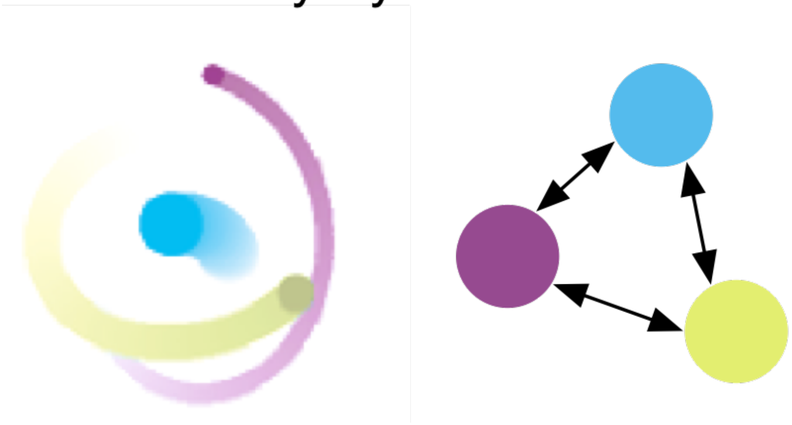


…is not well-suited to reasoning over structured representations.

But graph neural networks, which can learn a form of **message-passing on graphs**, are.

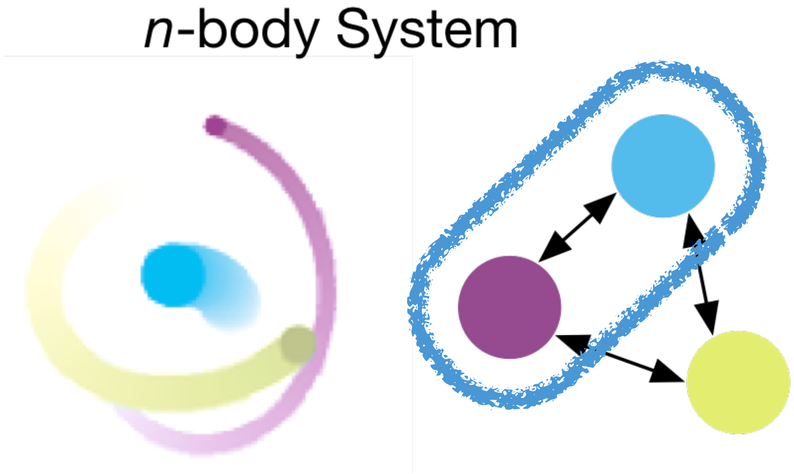
Interaction Network: Learning simulation as message-passing

n-body System



Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



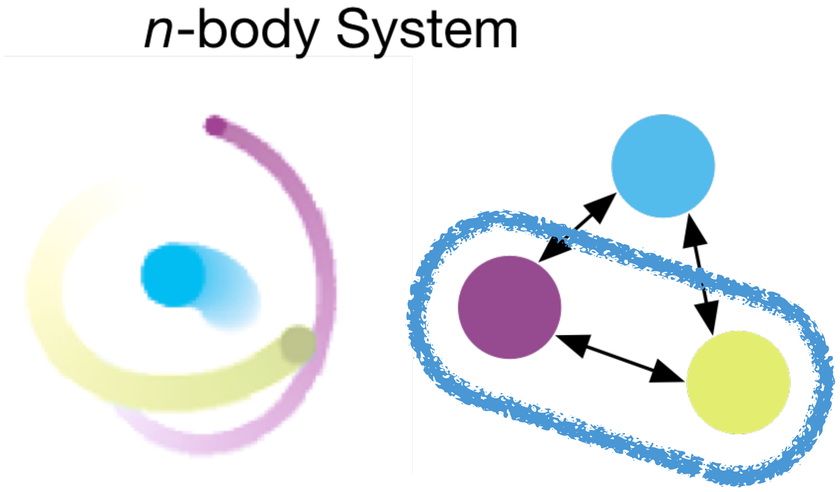
Edge function

$$\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k})$$

- Compute “message” from node and edge attributes associated with an edge

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



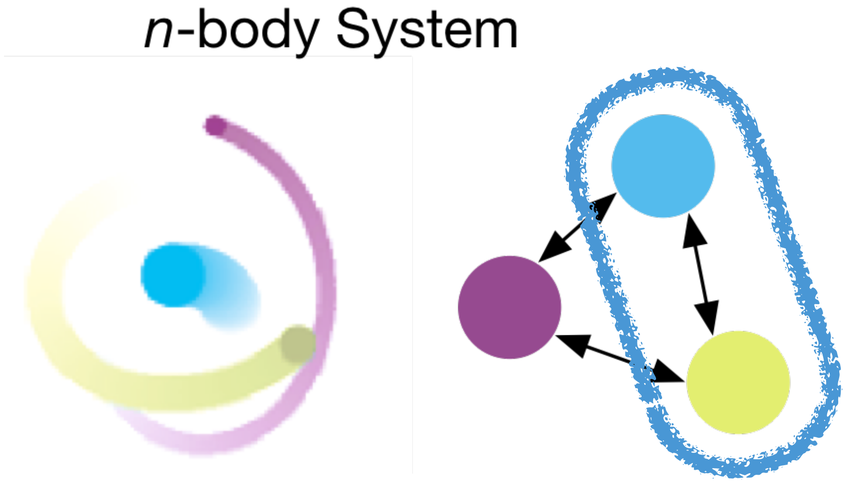
Edge function

$$\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k})$$

- Compute “message” from node and edge attributes associated with an edge

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



Edge function

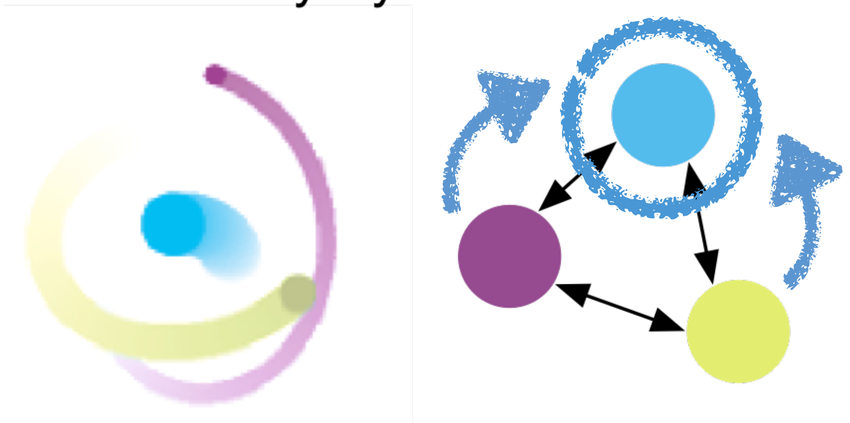
$$\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k})$$

- Compute “message” from node and edge attributes associated with an edge

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing

n-body System



Edge function

$$\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k})$$

- Compute “message” from node and edge attributes associated with an edge

Node function

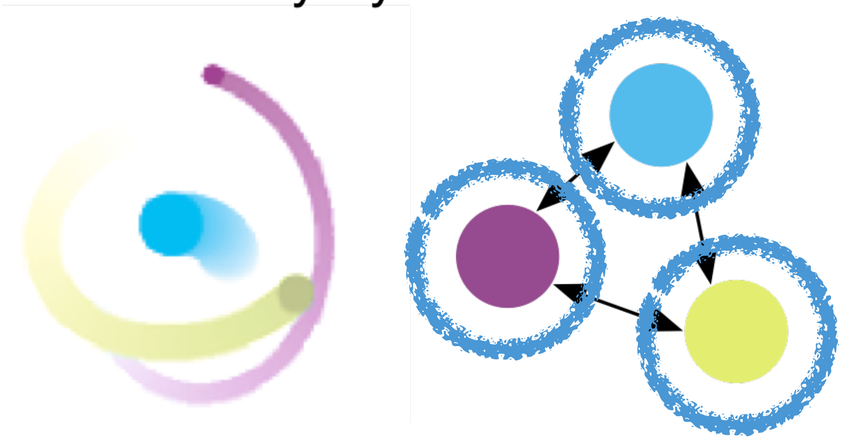
$$\mathbf{v}'_k \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$$

- Update node info from previous node state and aggregated “messages”

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing

n-body System



Edge function

$$\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k})$$

- Compute “message” from node and edge attributes associated with an edge

Node function

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

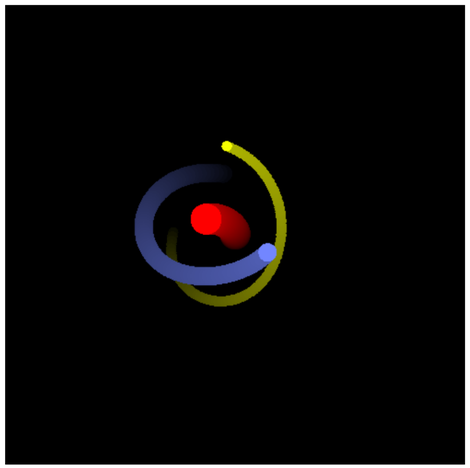
- Update node info from previous node state and aggregated “messages”

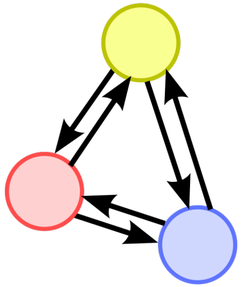
Trained to predict node states at t_1 from states at t_0

Battaglia et al., 2016, NeurIPS

Physical systems as graphs

n-body

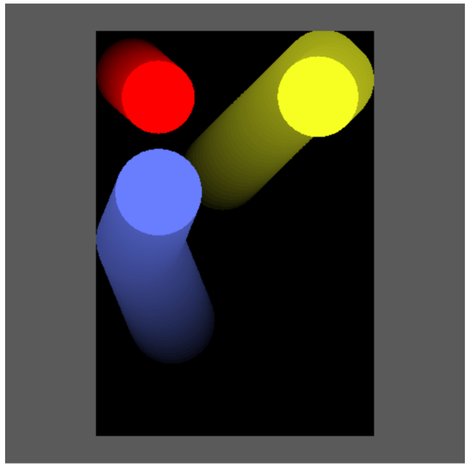


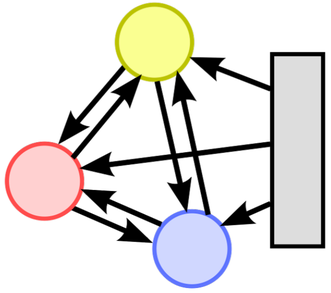


Nodes: bodies

Edges: gravitational forces

Balls

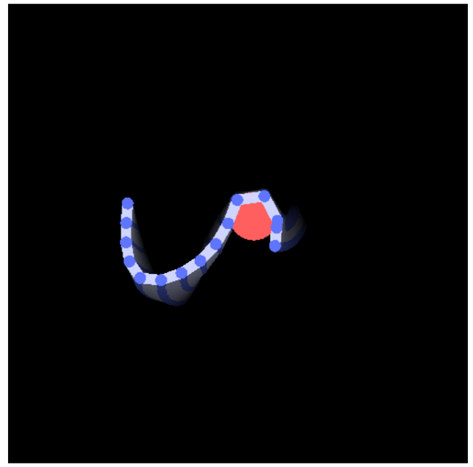


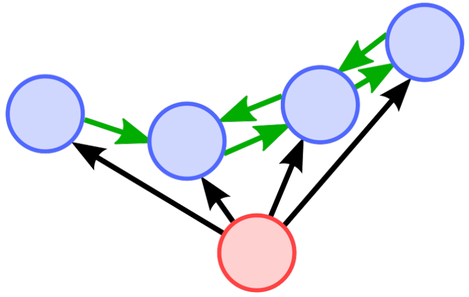


Nodes: balls

Edges: rigid collisions between balls, and walls

String



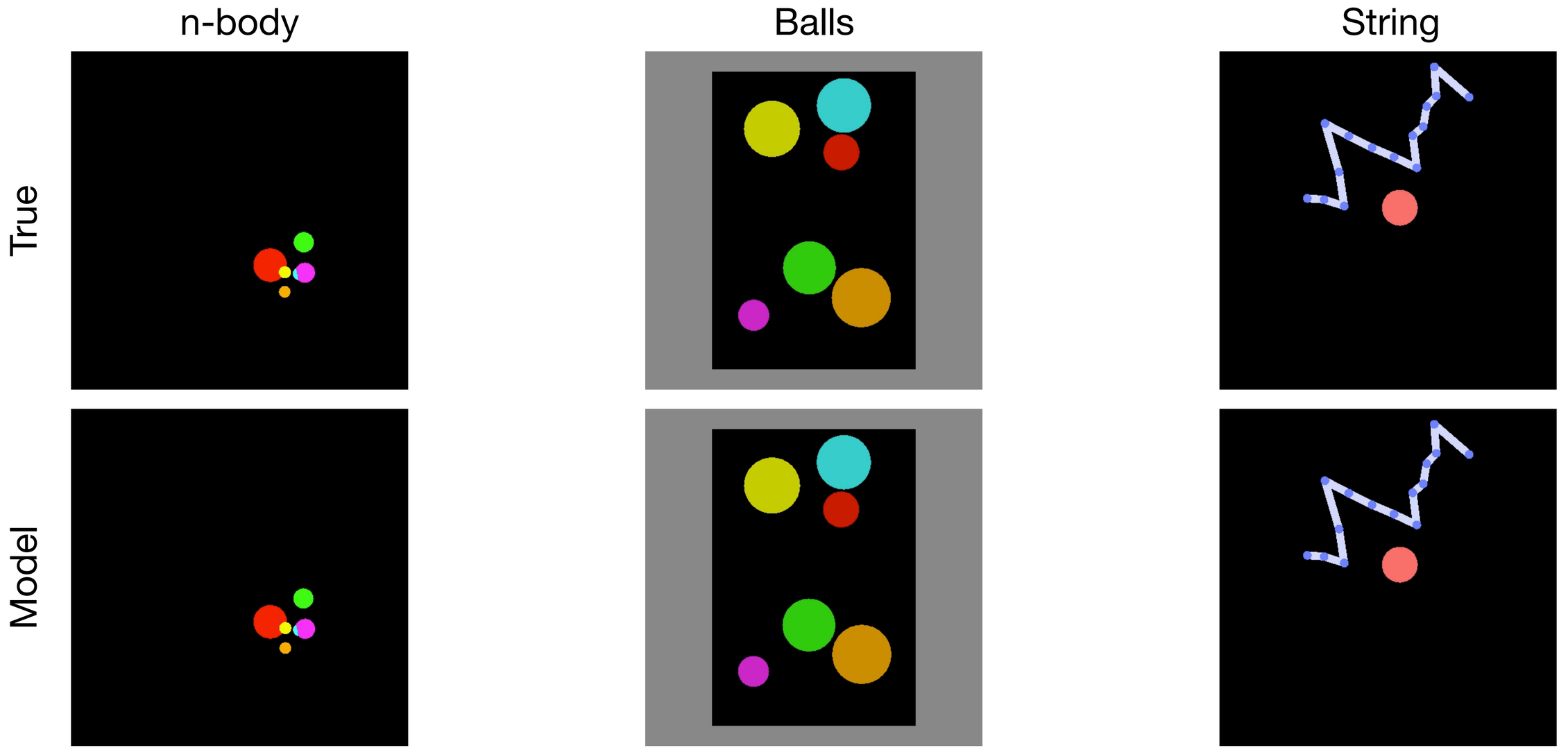


Nodes: masses

Edges: springs and rigid collisions

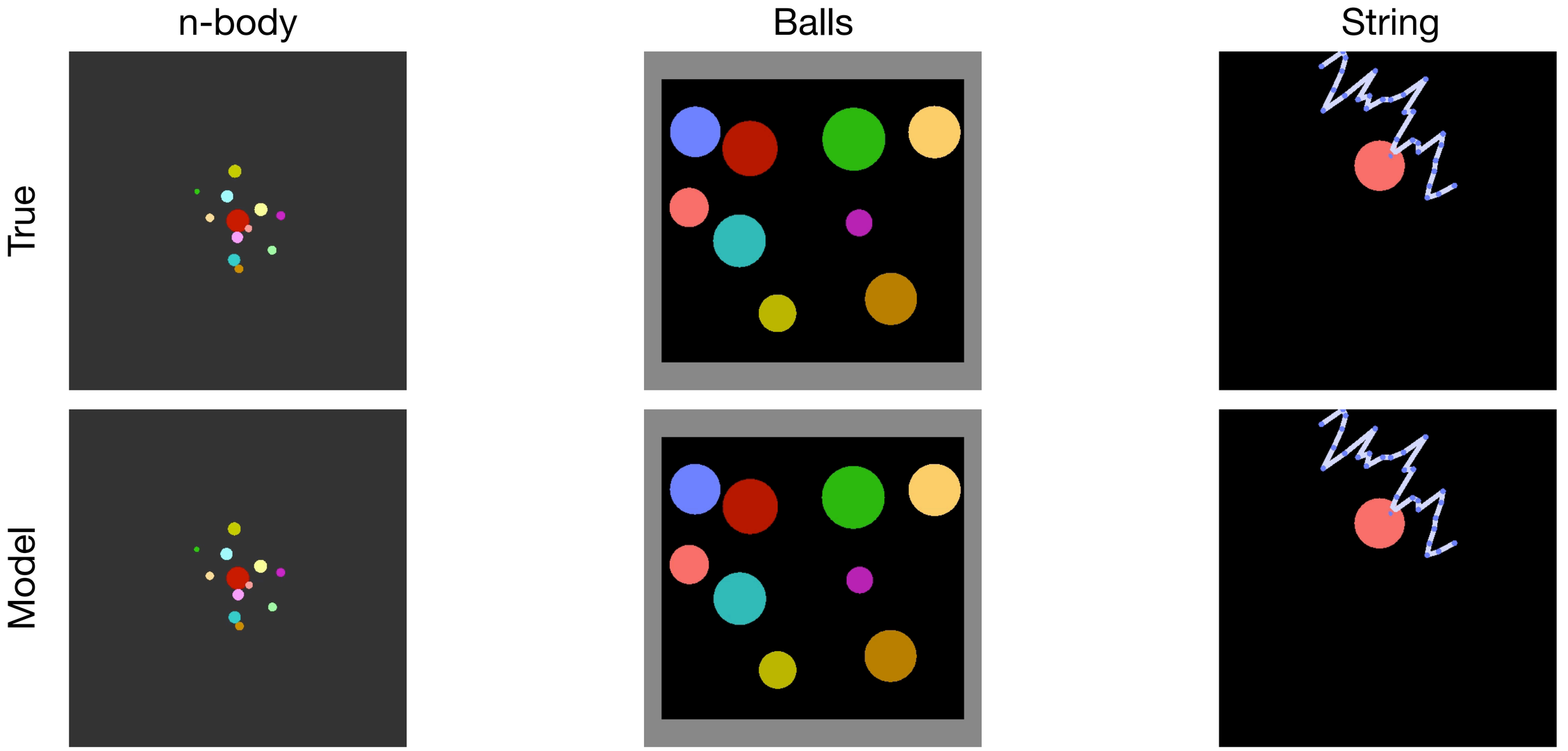
Battaglia et al., 2016, NeurIPS

1000-step rollouts of true (top row) vs predicted (bottom row)



Battaglia et al., 2016, NeurIPS

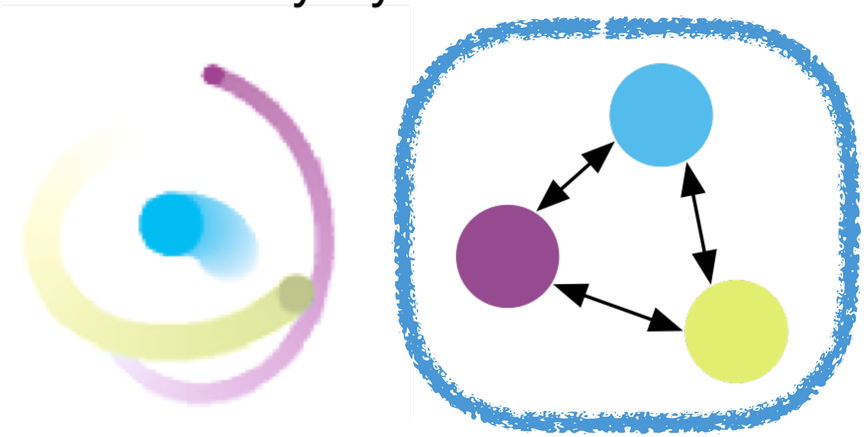
Zero shot generalisation to larger systems



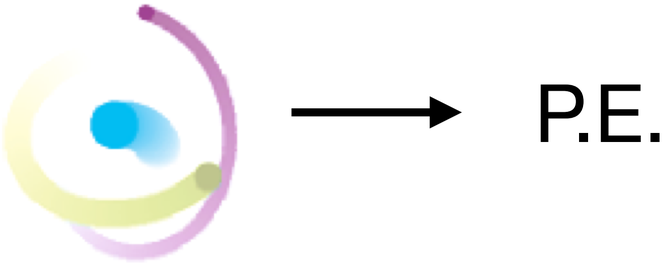
Battaglia et al., 2016, NeurIPS

Interaction Network: Predicting potential energy

n-body System



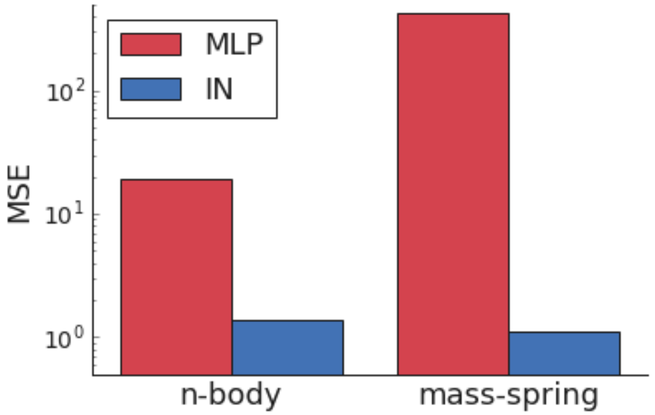
Trained to predict system's potential energy



Global function

$$\mathbf{u}' \leftarrow \phi^u(\bar{\mathbf{v}}')$$

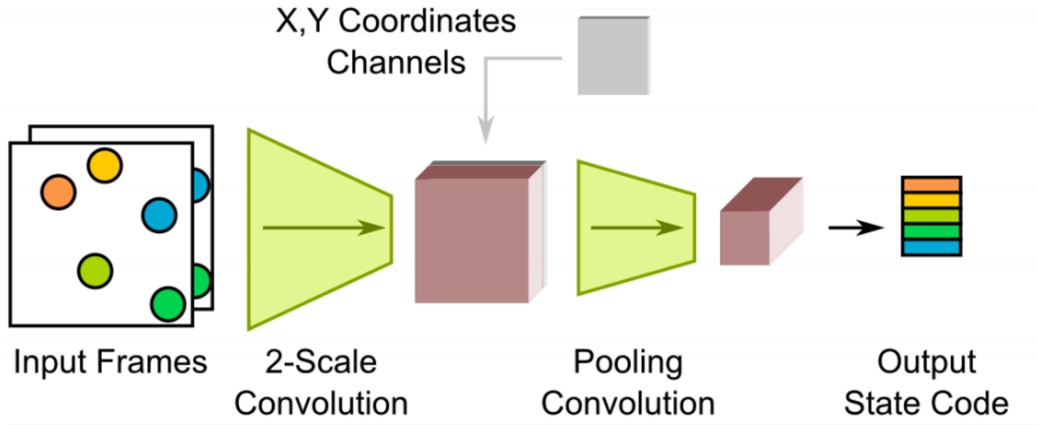
- Rather than making node-wise predictions, edge and node updates can be used to make global predictions.



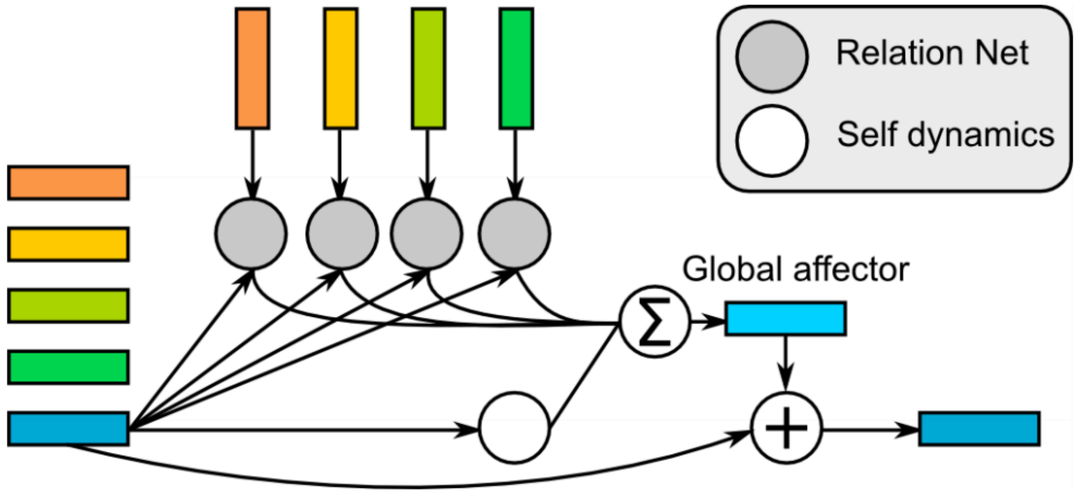
Battaglia et al., 2016, NeurIPS

Visual interaction network: Simulate from input images

Multi-frame encoder (conv net-based)



Interaction network



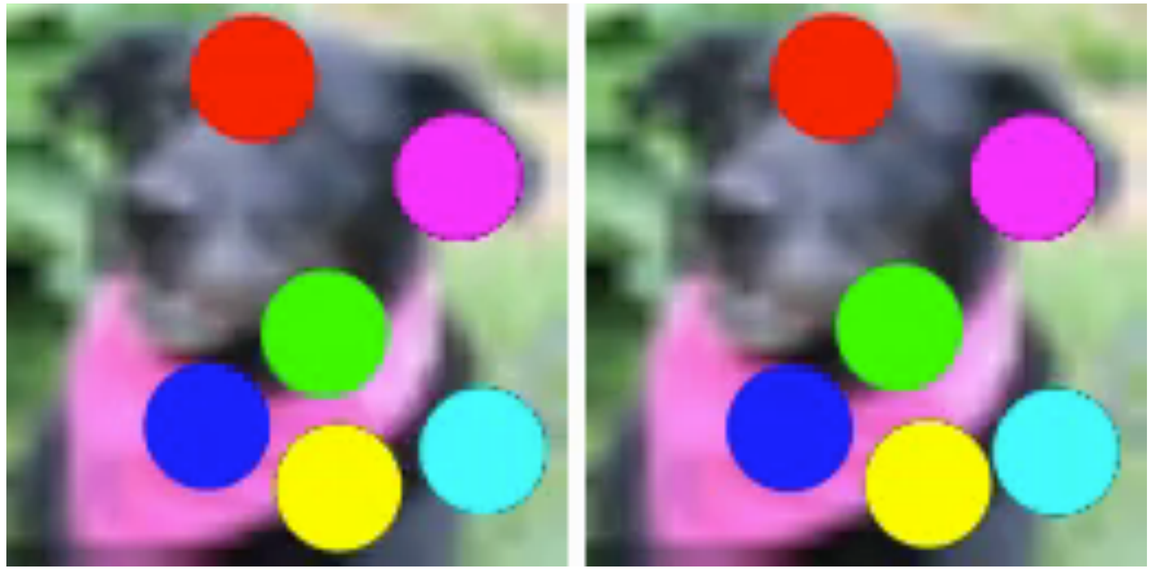
Watters et al., 2017, NeurIPS

Visual interaction network: Simulate from input images

Mass-springs

True Model

Bouncing balls



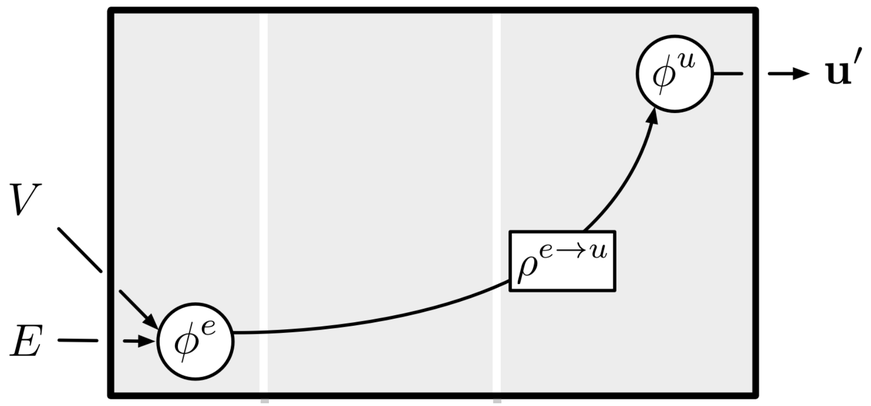
True Model

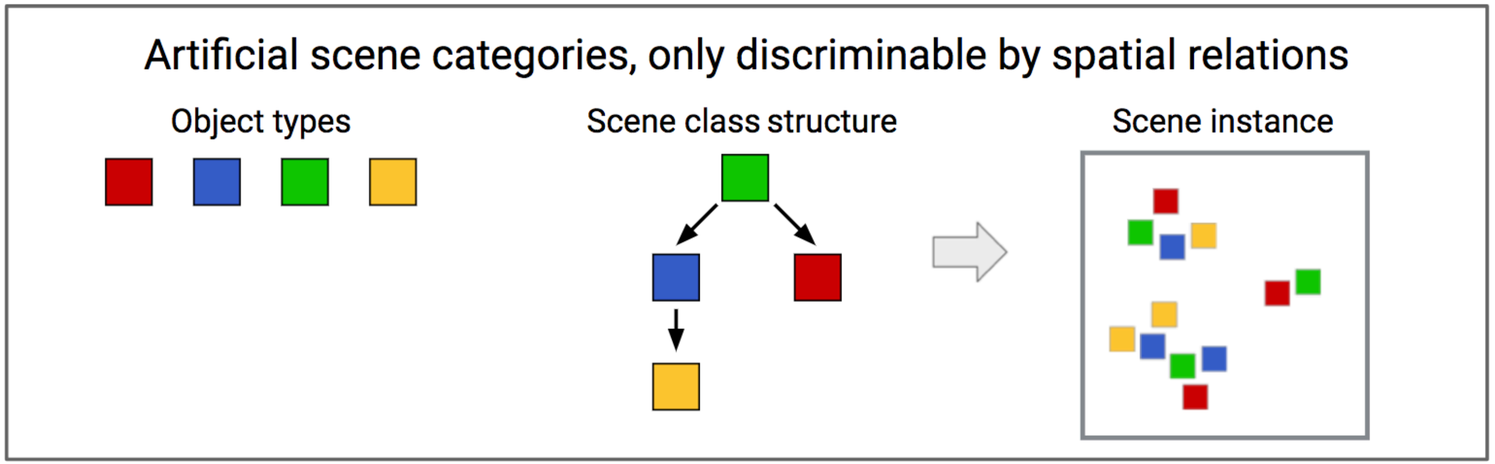
Can even predict invisible objects, inferred from how they affect visible ones

Watters et al., 2017, NeurIPS

Relation Network (RN) for scene understanding

No node updates, where instead all per-edge outputs are summed and passed to a global function



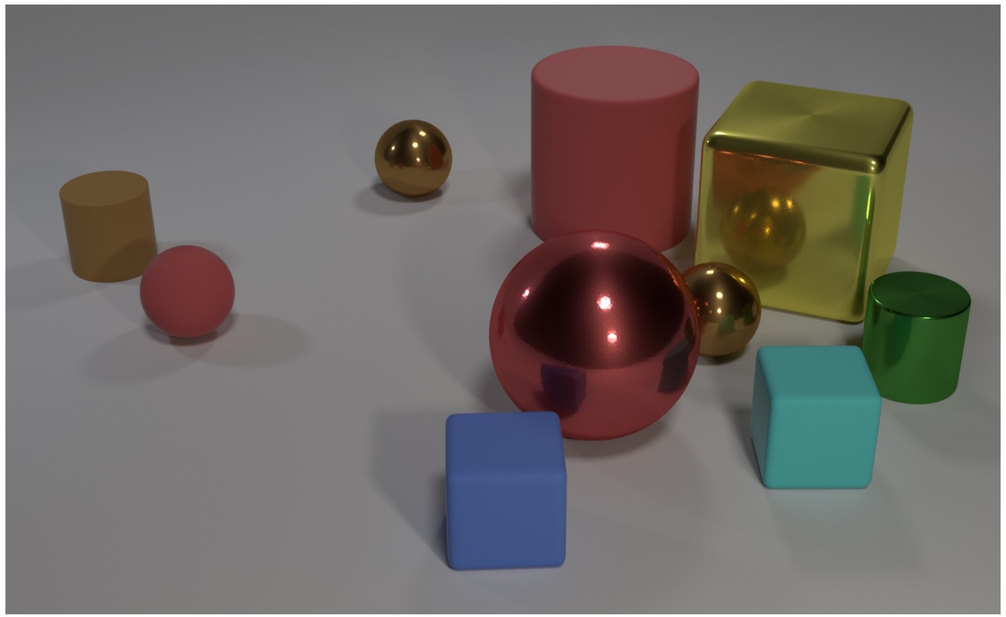


Our experiments showed that RNs can learn to:

- Classify scenes
- Infer novel scene structures
- Learn object factorizations from input states or images
- Support one-shot learning

Raposo et al., 2017, ICLR workshop

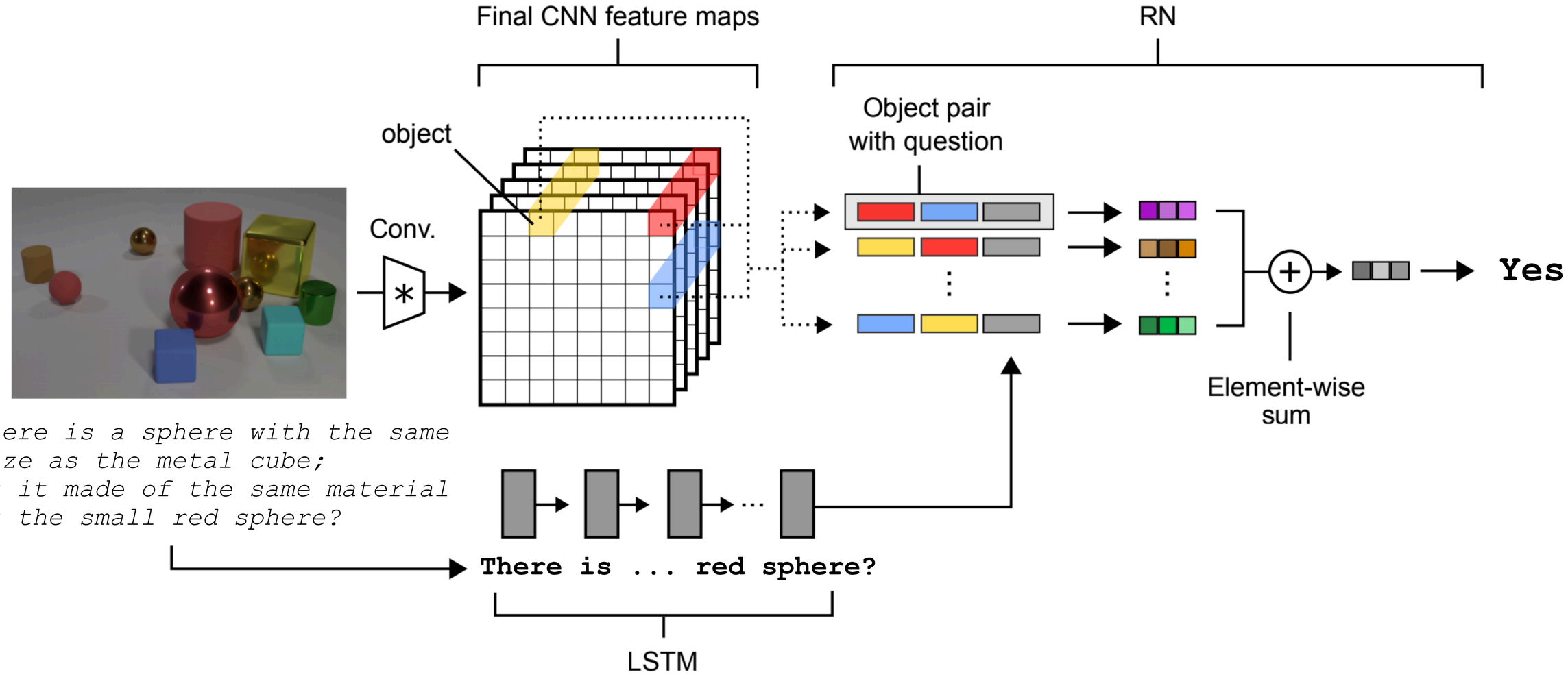
CLEVR Visual Question-Answering (VQA) dataset



There is a sphere with the same size as the metal cube; is it made of the same material as the small red sphere?

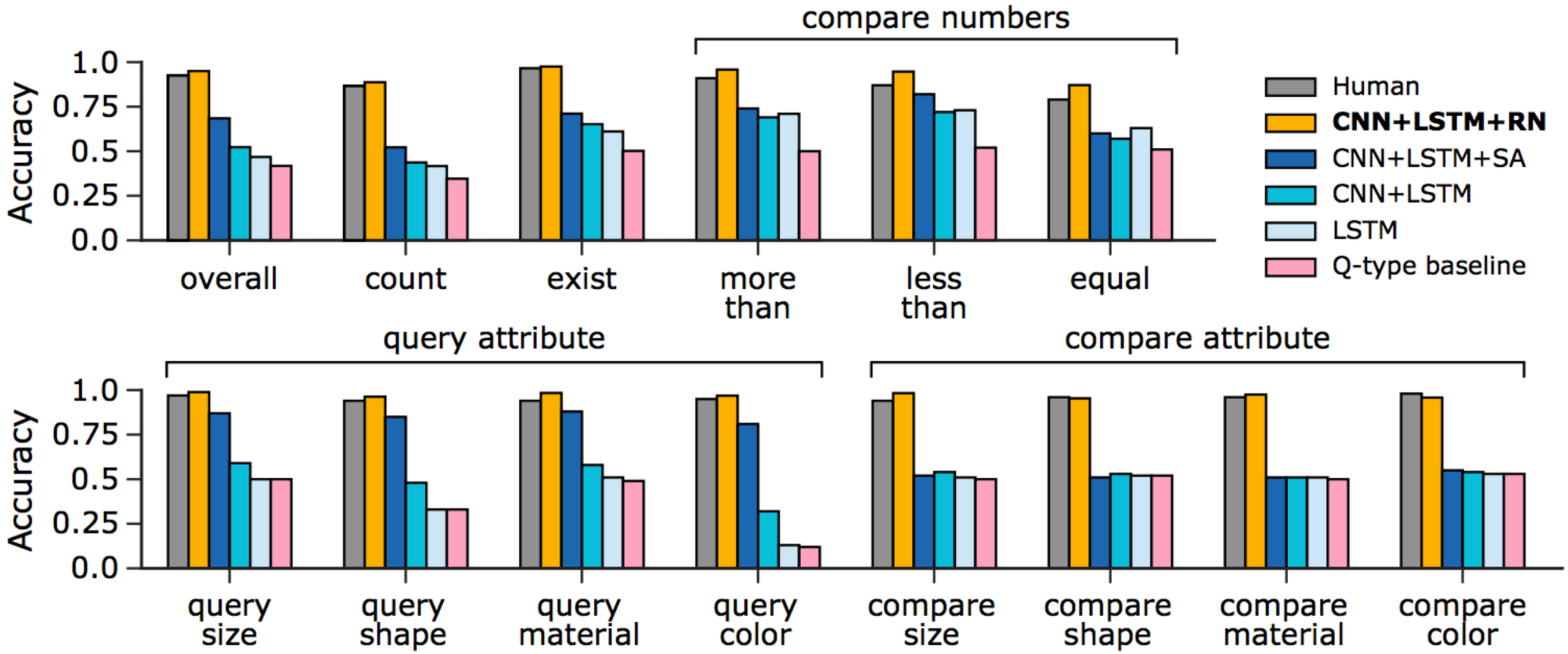
Santoro et al., 2017, NeurIPS

RN architecture for VQA



Santoro et al., 2017, NeurIPS

Results: RN applied to CLEVR



Santoro et al., 2017, NeurIPS

Results: RN applied to bAbI Q-A dataset

Sandra moved to the garden.
John moved to the office.
Sandra journeyed to the bathroom.
Mary moved to the hallway.
Daniel travelled to the office.
John went back to the garden.
John moved to the bedroom.

Where is Sandra? ***bathroom***

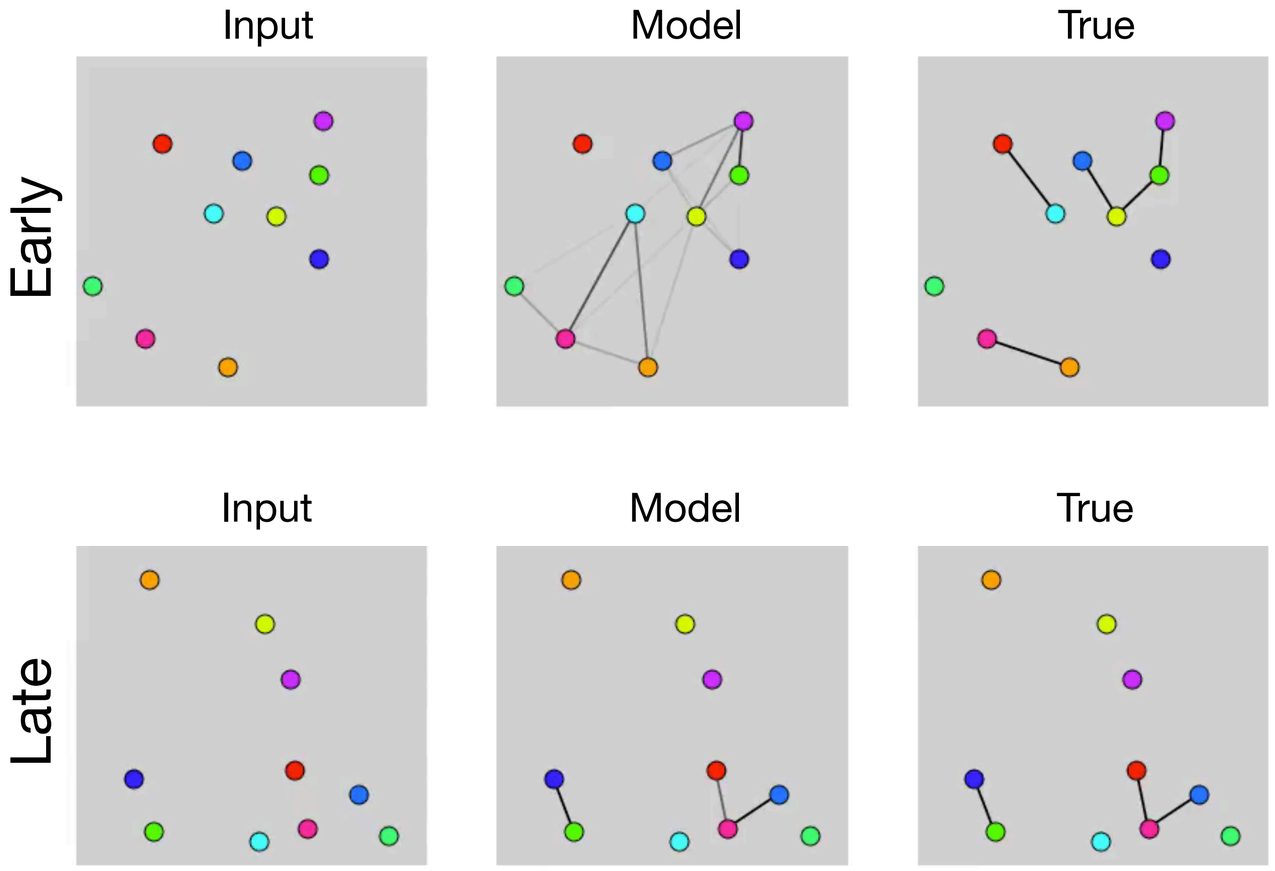
Model
- Entities are LSTM-encoded sentences
- Input to an RN

Results
- Solves 18/20 bAbI tasks

Santoro et al., 2017, NeurIPS

Results: RN can infer relations in dot motion

Trained on mass-spring systems



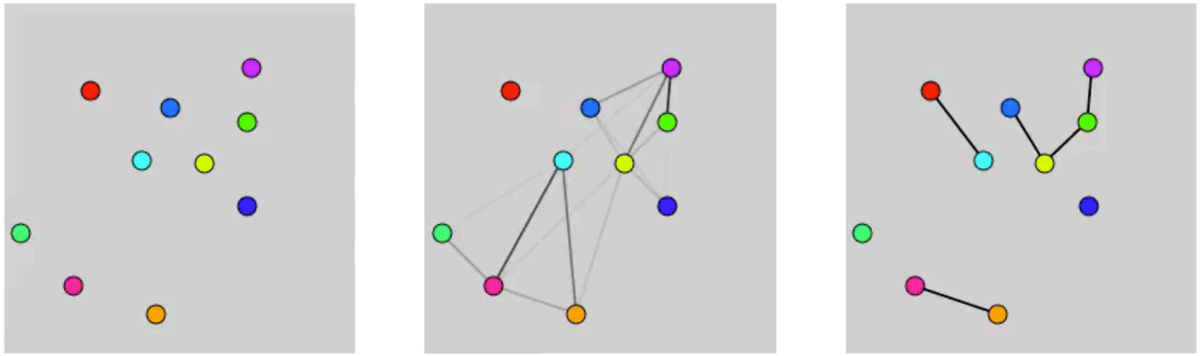
Santoro et al., 2017, NeurIPS

Results: RN can infer relations in dot motion

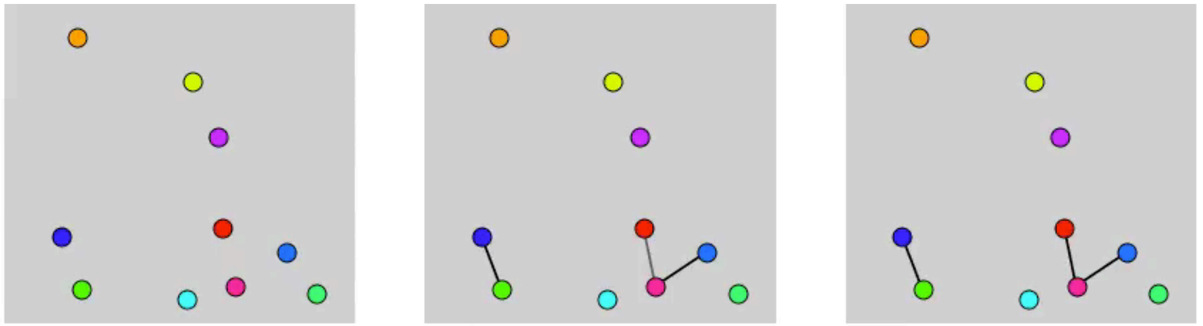
Trained on mass-spring systems

Input | Model | True

Early



Late



Generalizes to point-light walkers

Santoro et al., 2017, NeurIPS

Graph Networks (GNs)

Why do we need another Graph Neural Network variant?

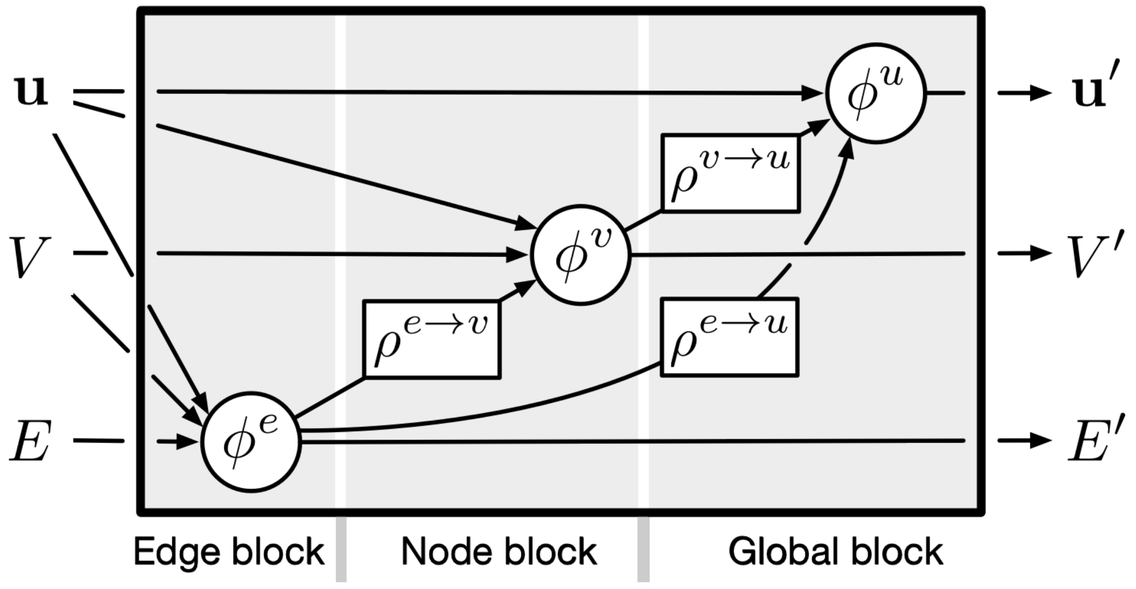
- We designed GNs to be both expressive, and easy to implement

Graph Networks (GNs)

Why do we need another Graph Neural Network variant?

- We designed GNs to be both expressive, and easy to implement
- A GN block is a “graph-to-graph” function approximator
  - The output graph’s structure (number of nodes and edge connectivity) matches the input graph’s
  - The output graph-, node-, and edge-level attributes will be functions of the input graph’s

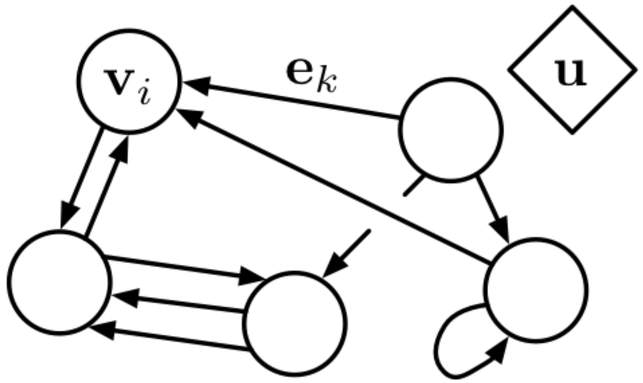
Graph Network (Battaglia et al. 2018)
(a type of Graph Neural Network (Scarselli et al. 2009))

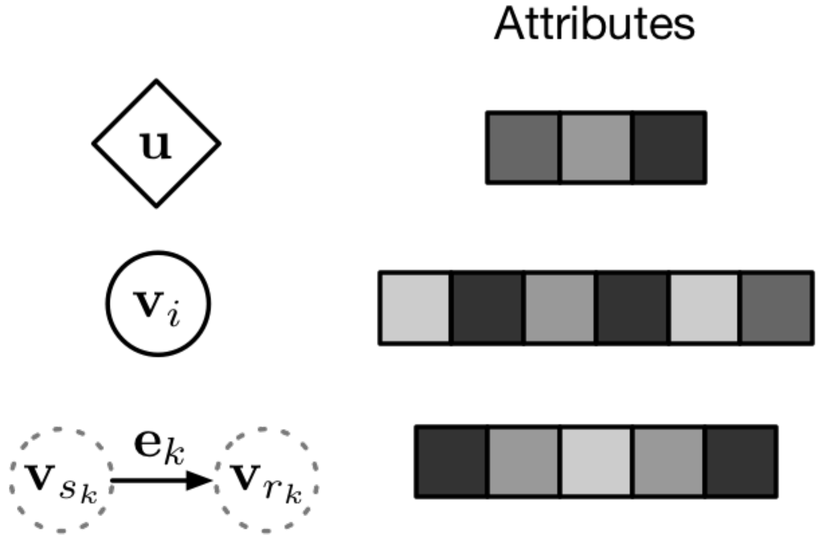


What kind of graphs do GNs operate on?

“Graph”: directed, attributed multi-graph with a global attribute

- “Directed”: one-way edges, from a “sender” node to a “receiver” node
- “Multi-graph”: there can be more than one edge between vertices, including self-edges
- “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



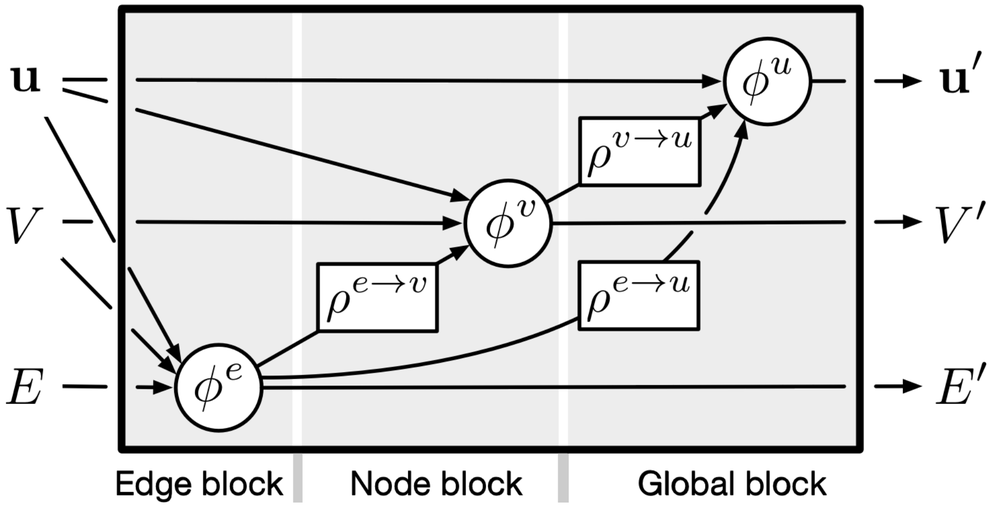


Graph: $G = (\mathbf{u}, V, E)$

- Global attribute: $\mathbf{u}$
- Node attributes: $V = \{\mathbf{v}_i\}_{i=1:N^v}$
- Edge attributes, sender node indices, receiver node indices: $E = \{(\mathbf{e}_k, s_k, r_k)\}_{k=1:N^e}$

$$s_k, r_k \in \{1, \ldots, N^v\}$$

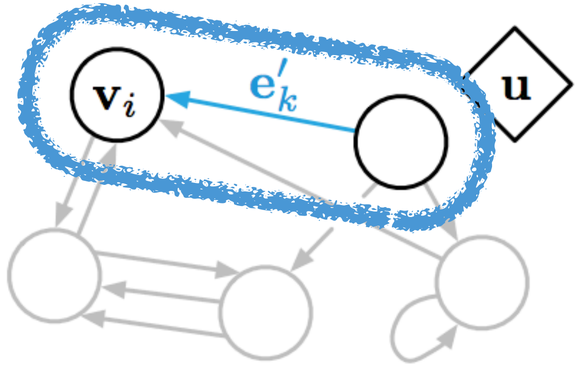
How does a GN block process a graph?



Edge block

For each edge, $\mathbf{e}_k, \mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{u}$,
are passed to an “edge-wise function”:

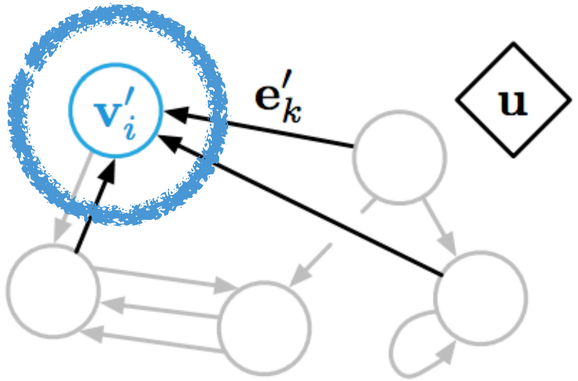
$$\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$$



Node block

For each node, $\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}$, are
passed to a “node-wise function”:

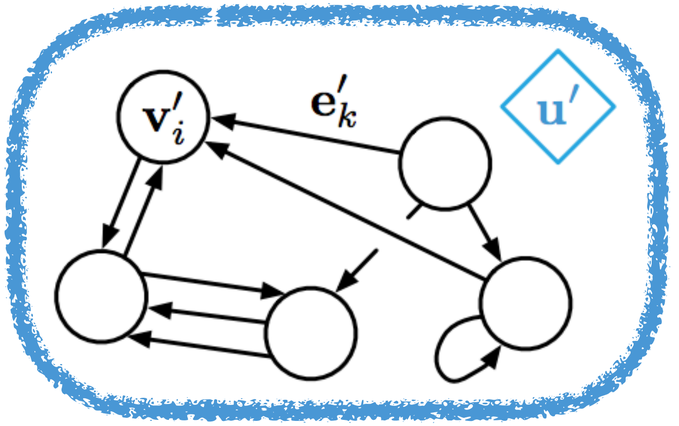
$$\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$$



Global block

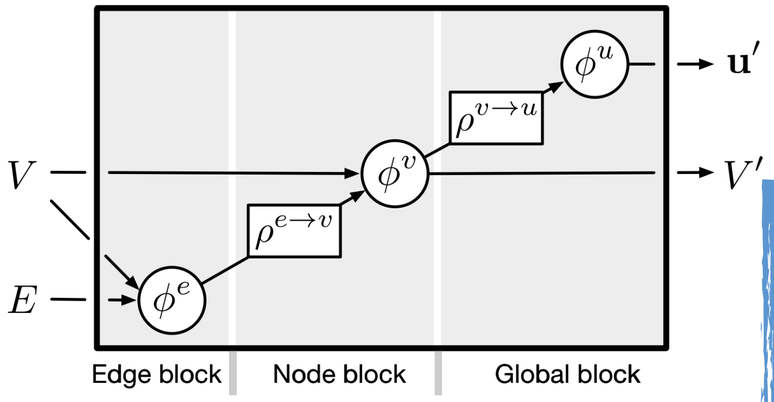
Across the graph, $\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}$, are
passed to a “global function”:

$$\mathbf{u}' \leftarrow \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$$



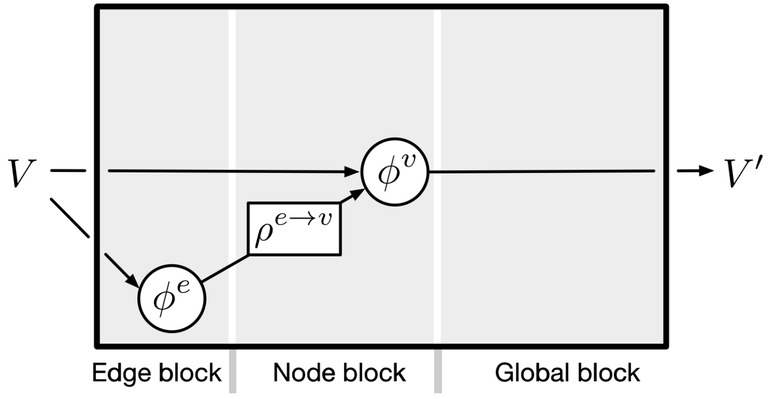
Message-Passing NN (eg. Interaction Net)

Gilmer et al. 2017



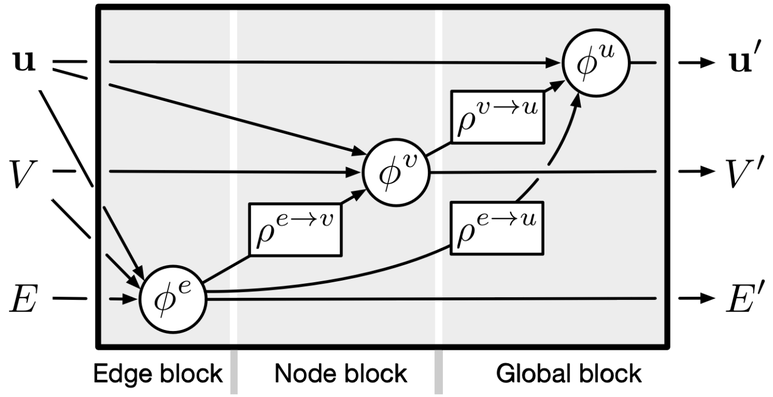
Non-Local NN (eg. Transformer)

Vaswani et al. 2017; Wang et al. 2017



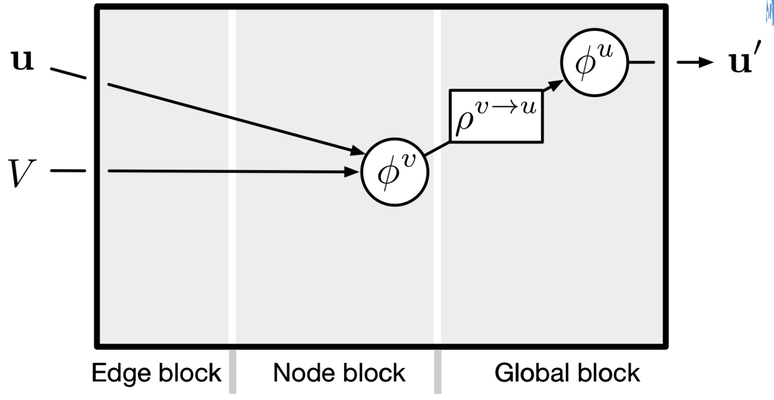
Graph Network
(a type of Graph Neural Network)

Battaglia et al. 2018



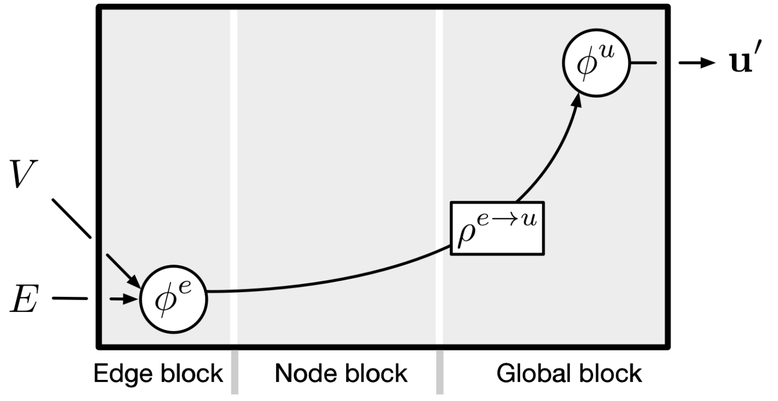
Deep Set

Zhang et al. 2017



Relation Network

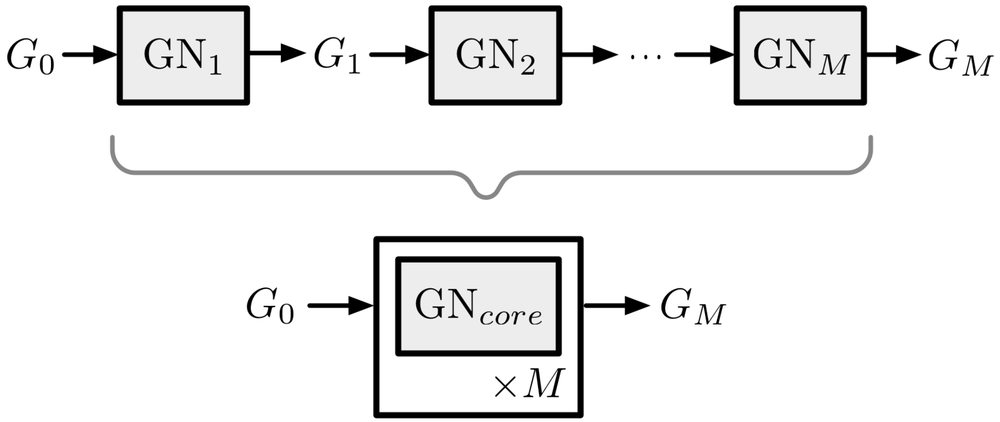
Raposo et al. 2017; Santoro et al. 2017



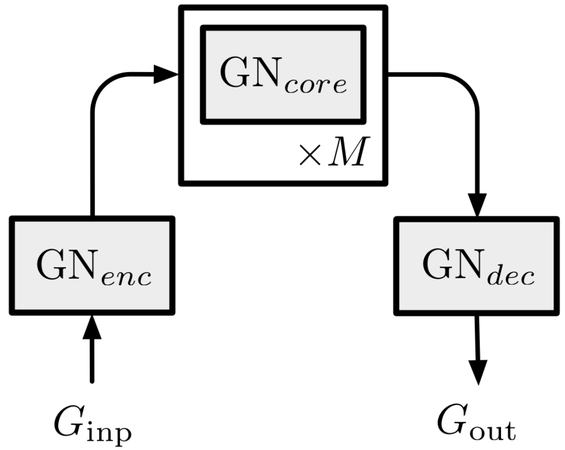
Composing GN blocks

The GN's graph-to-graph interface promotes stacking GN blocks, passing one GN's output to another GN as input

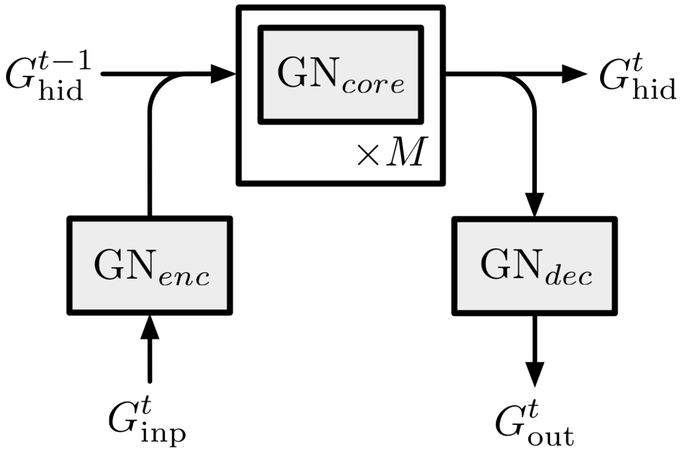
Shared GN core



Encode-process-decode

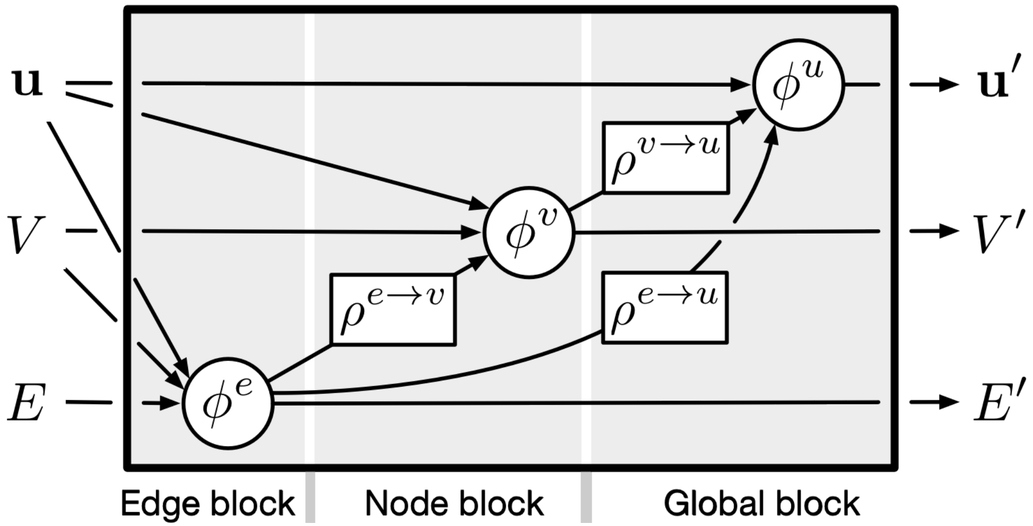


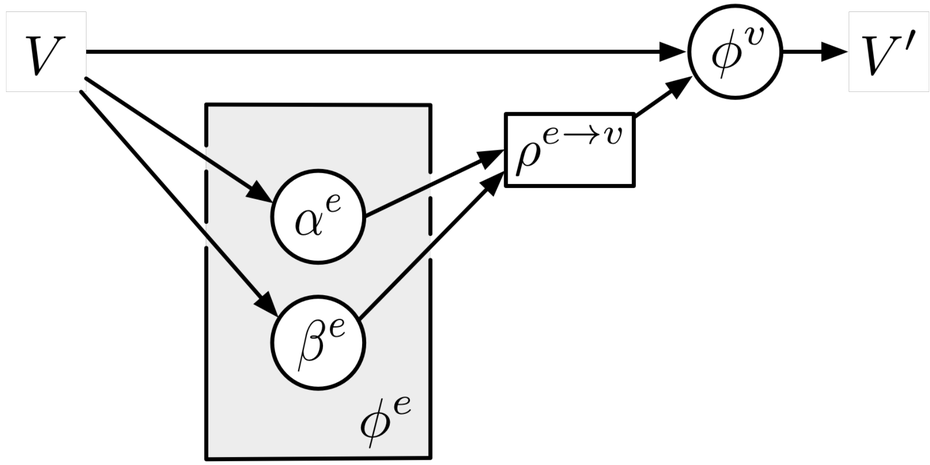
Recurrent GN architecture



Non-local Neural Nets / Self-Attention in the GN formalism

The edge function has two components—a scalar-valued pairwise interaction among nodes (α^e below), and a vector-valued function of the sender node (β^e below). The scalar component is the unnormalized attention weight, which is used to rescale the pooled edge outputs ($\rho^{e\to v}$ below)



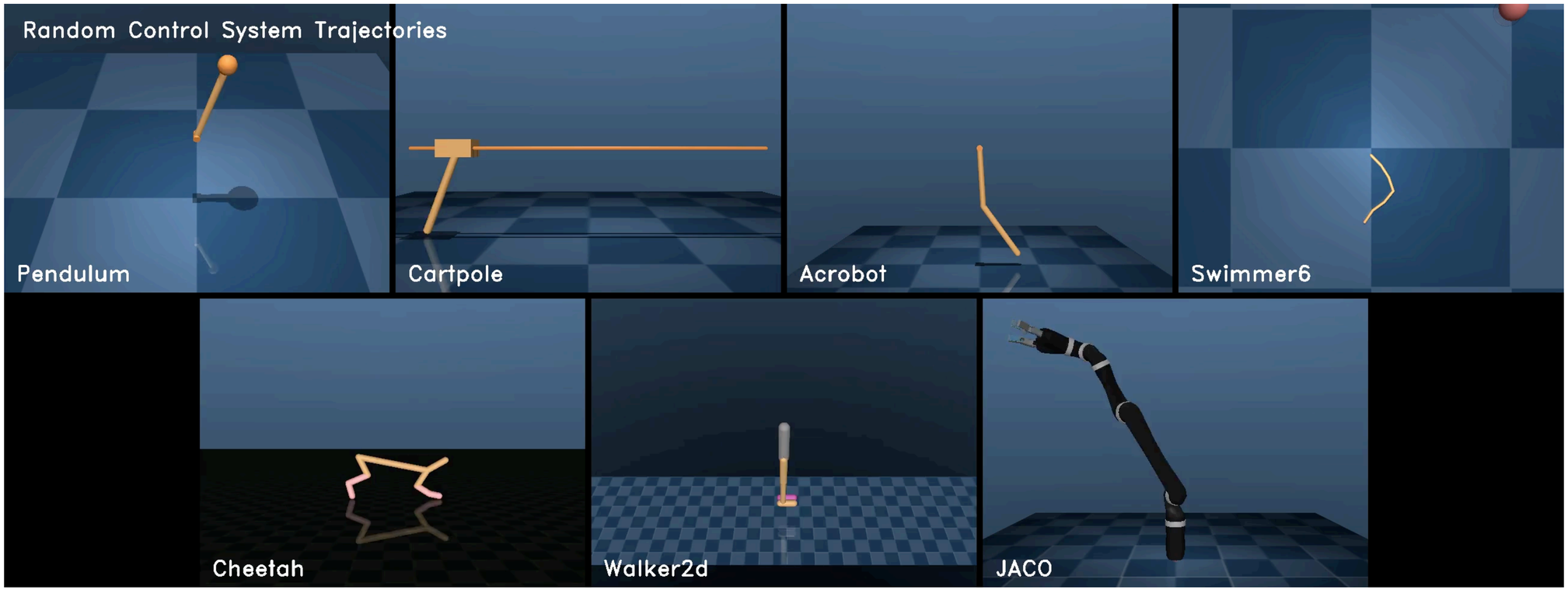


$$\phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) := f^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) \qquad = \left(\alpha^e\left(\mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right),\ \beta^e\left(\mathbf{v}_{s_k}\right)\right) = \left(a'_k, \mathbf{b}'_k\right) = \mathbf{e}'_k$$

$$\phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) := f^v(\bar{\mathbf{e}}'_i)$$

$$\rho^{e\to v}\left(E'_i\right) := \frac{1}{\sum_{\{k:\, r_k=i\}} a'_k} \sum_{\{k:\, r_k=i\}} a'_k \mathbf{b}'_k$$

Systems: "DeepMind Control Suite" (Mujoco) & real JACO



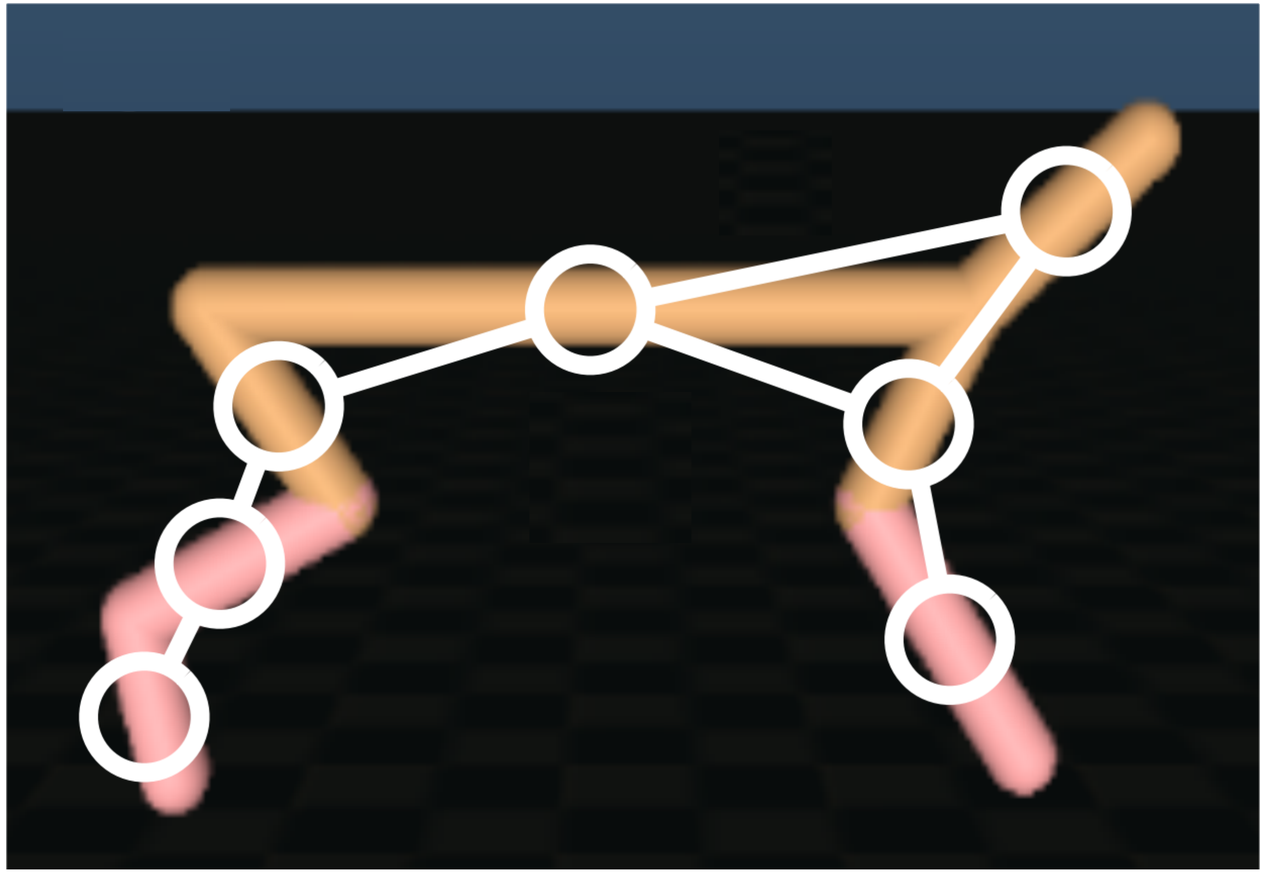
DeepMind Control Suite (Tassa et al., 2018)

JACO Arm

Kinematic tree of the actuated system as a graph

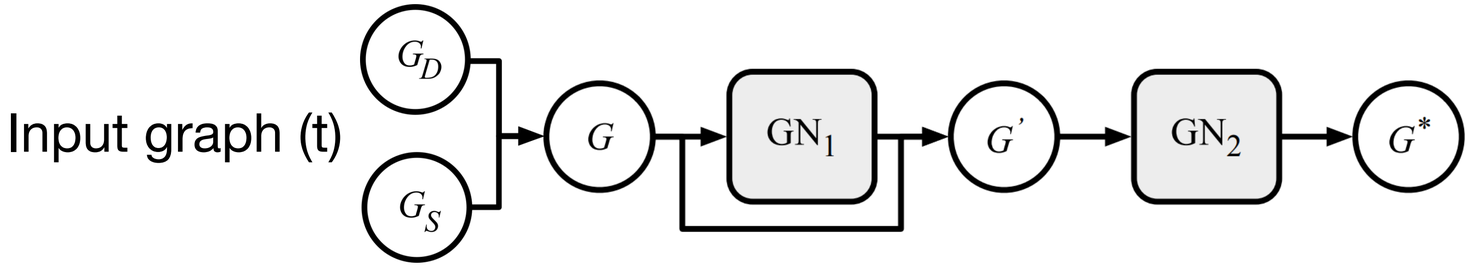
Controllable physical system as a graph:

- Bodies → Nodes
- Joints → Edges
- Global properties

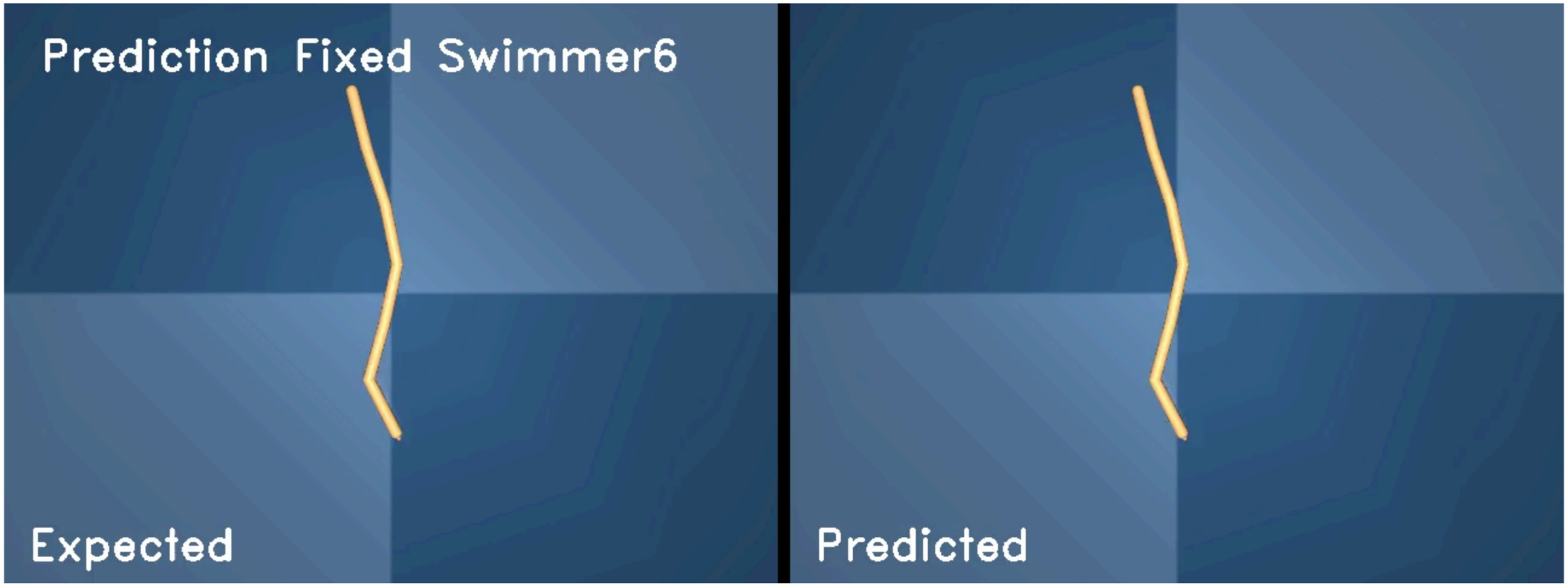


Sanchez-Gonzalez et al., 2018, ICML

Forward model: supervised, 1-step training w/ random control inputs



Chained 100-step predictions

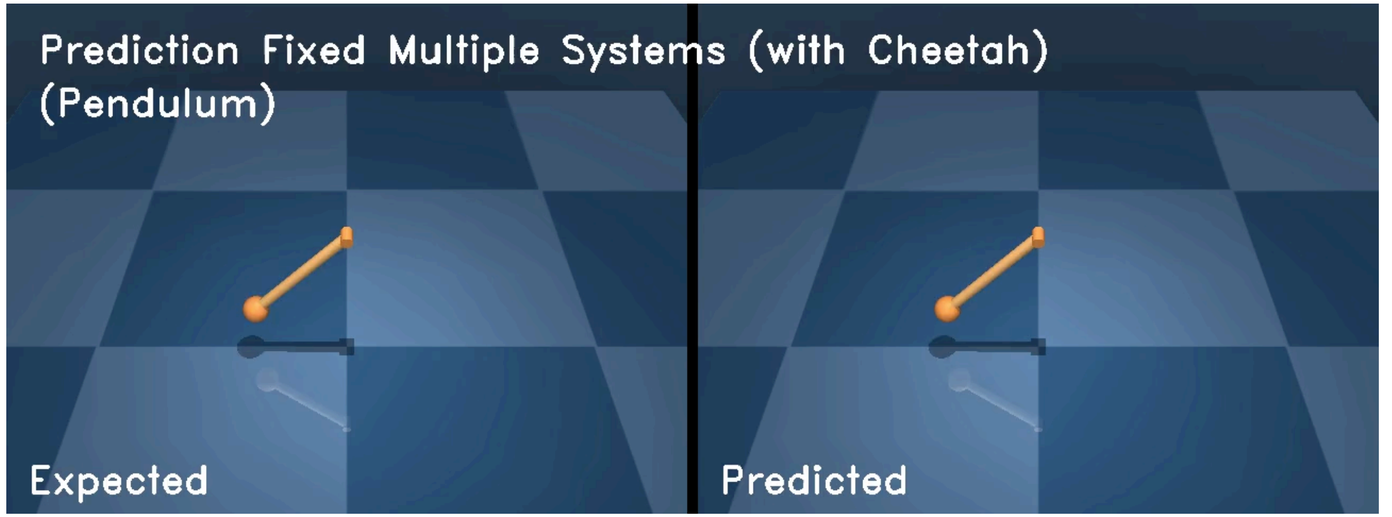


Sanchez-Gonzalez et al., 2018, ICML

GN forward model: Multiple systems & zero-shot generalization

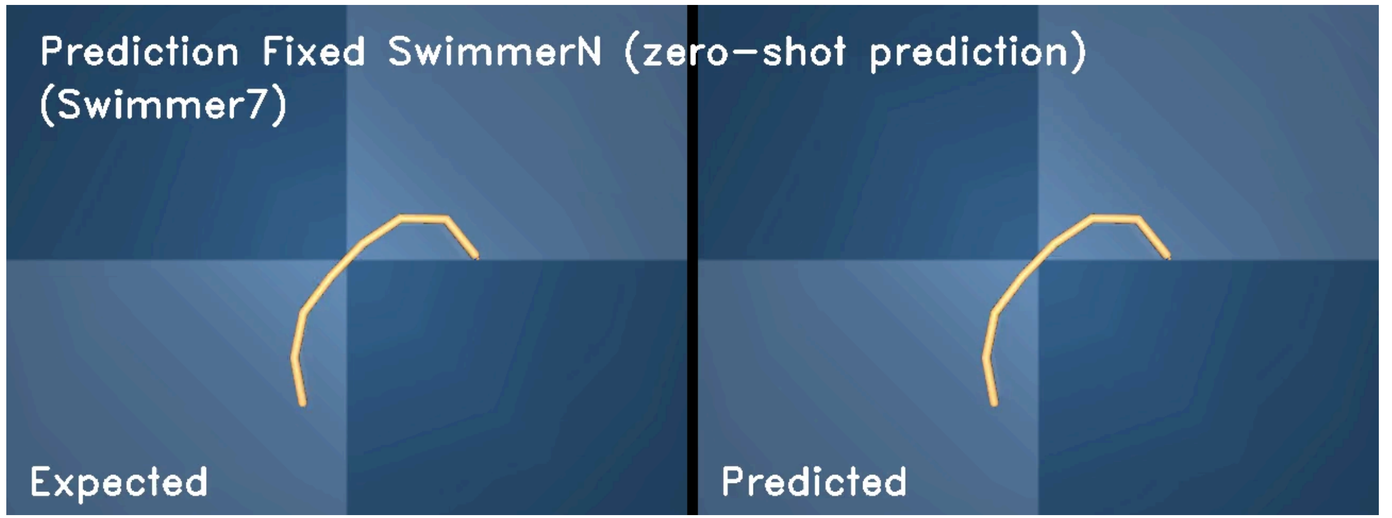
Single model trained:

- Pendulum, Cartpole, Acrobot, Swimmer6 & Cheetah



Zero-shot generalization: Swimmer

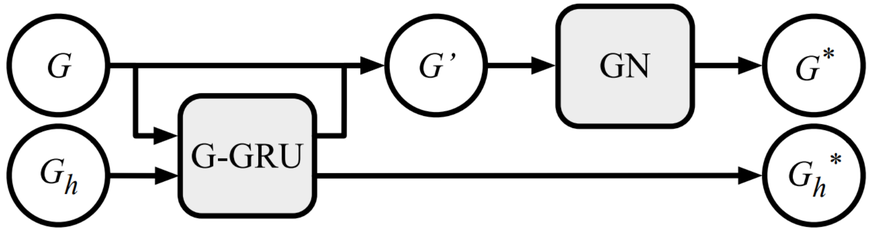
- # training links: {**3**, **4**, **5**, **6**, -, **8**, **9**, -, -, …}
- # testing links: {-, -, -, -, **7**, -, -, **10-14**}

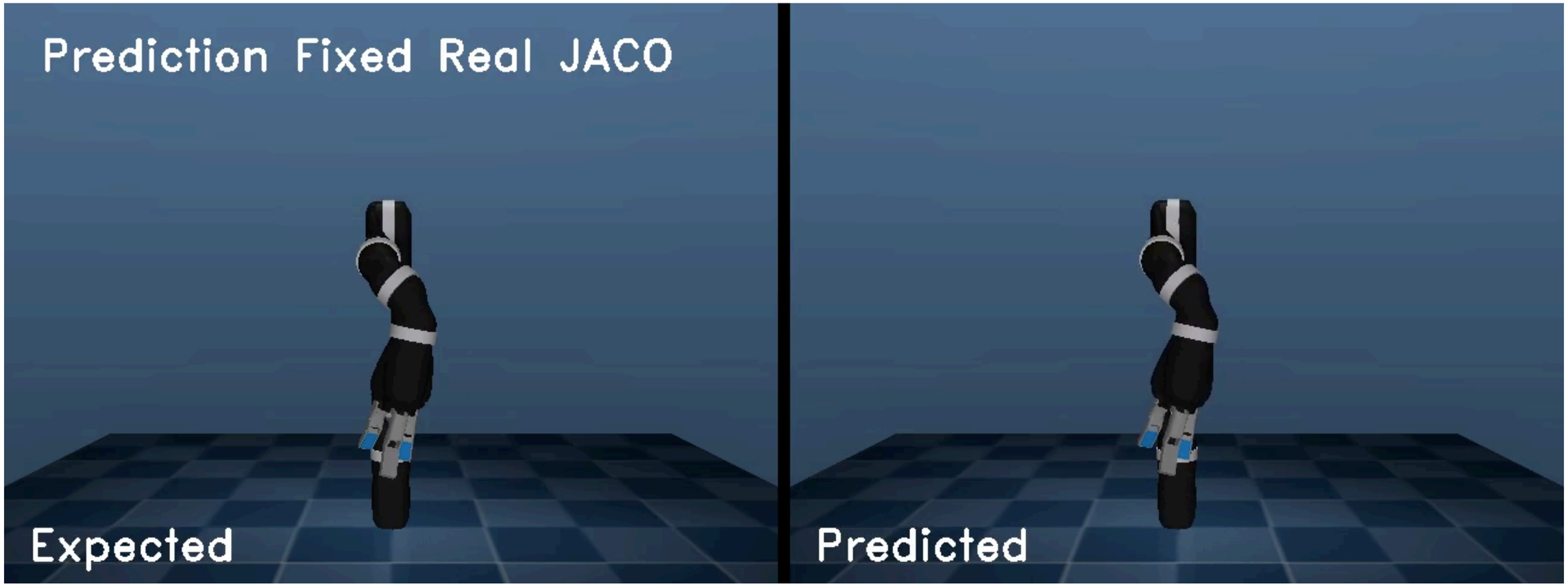


Sanchez-Gonzalez et al., 2018, ICML

GN forward model: Real JACO data

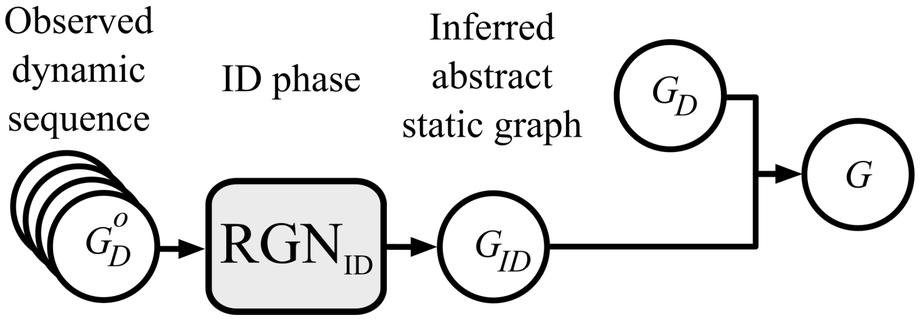
Recurrent GN



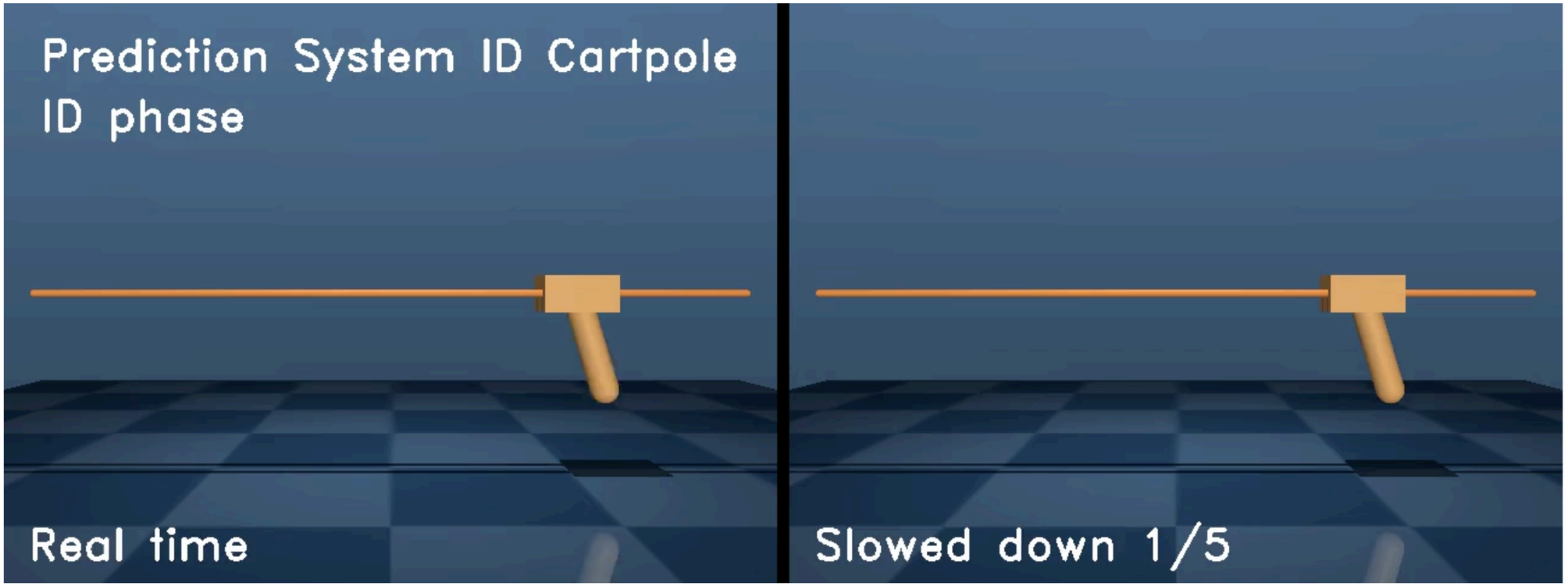


Sanchez-Gonzalez et al., 2018, ICML

System identification: GN-based inference, under diagnostic control inputs



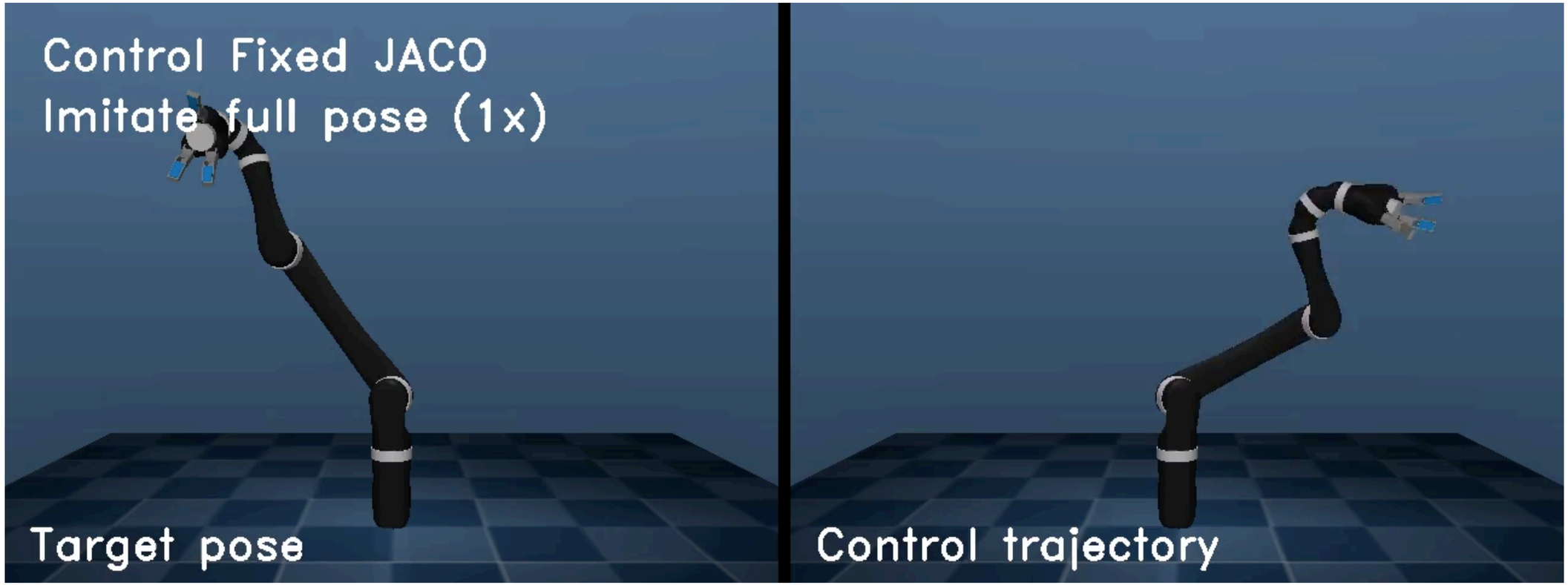
Unobserved system parameters (e.g. mass, length) are implicitly inferred



Sanchez-Gonzalez et al., 2018, ICML

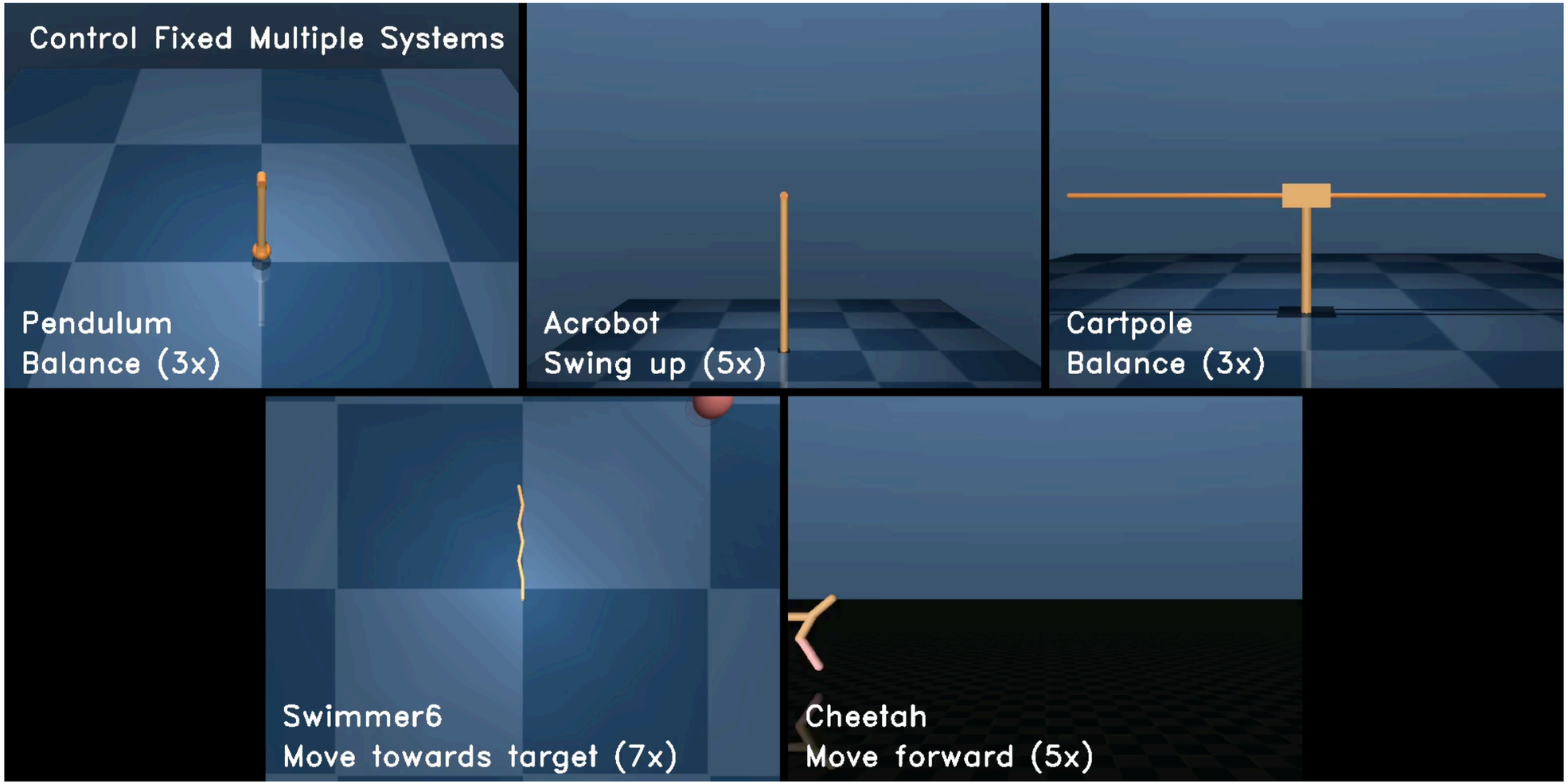
Control: Model-based planning

Trajectory optimization: the GN-based forward model is differentiable, so we can backpropagate through it, and find a sequence of actions that maximize reward



Sanchez-Gonzalez et al., 2018, ICML

Control: Multiple systems via a single model

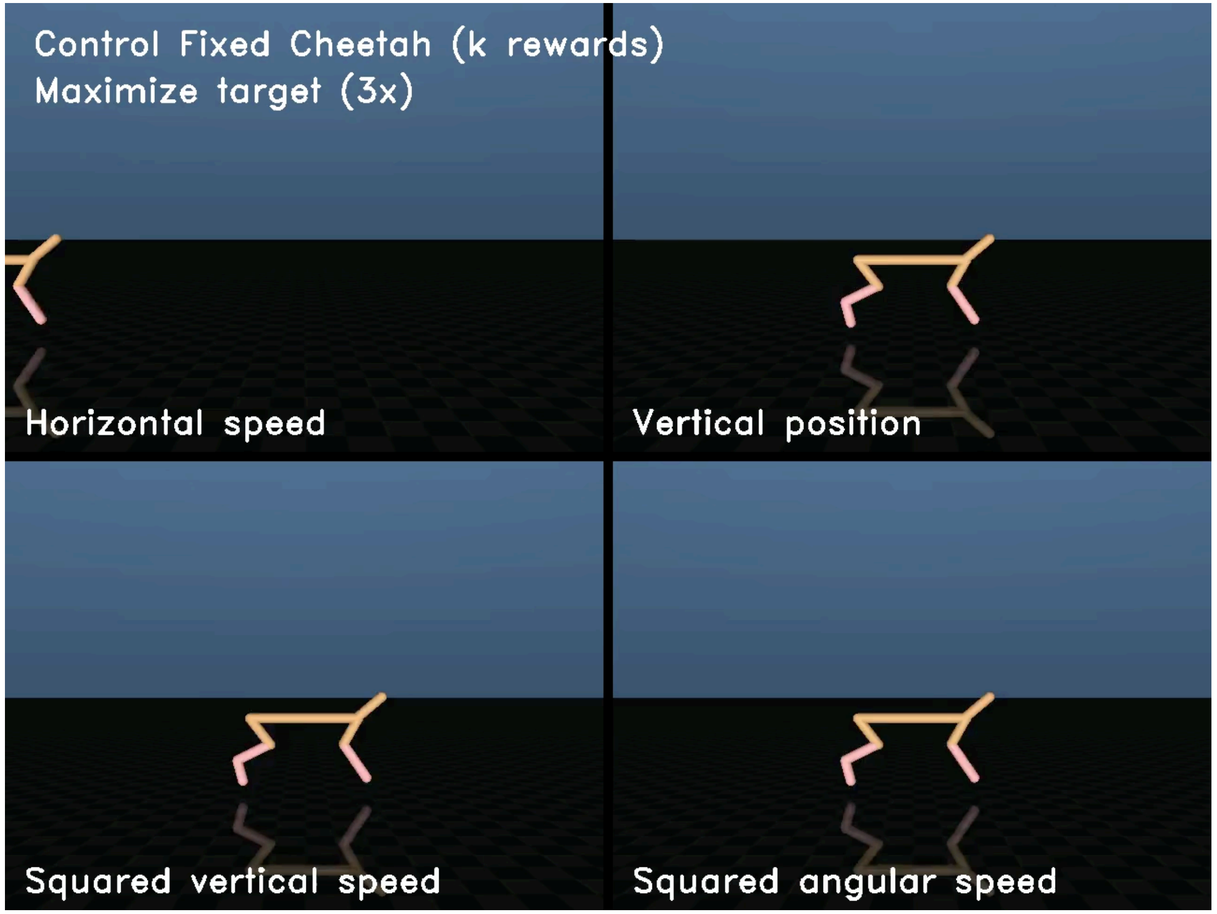


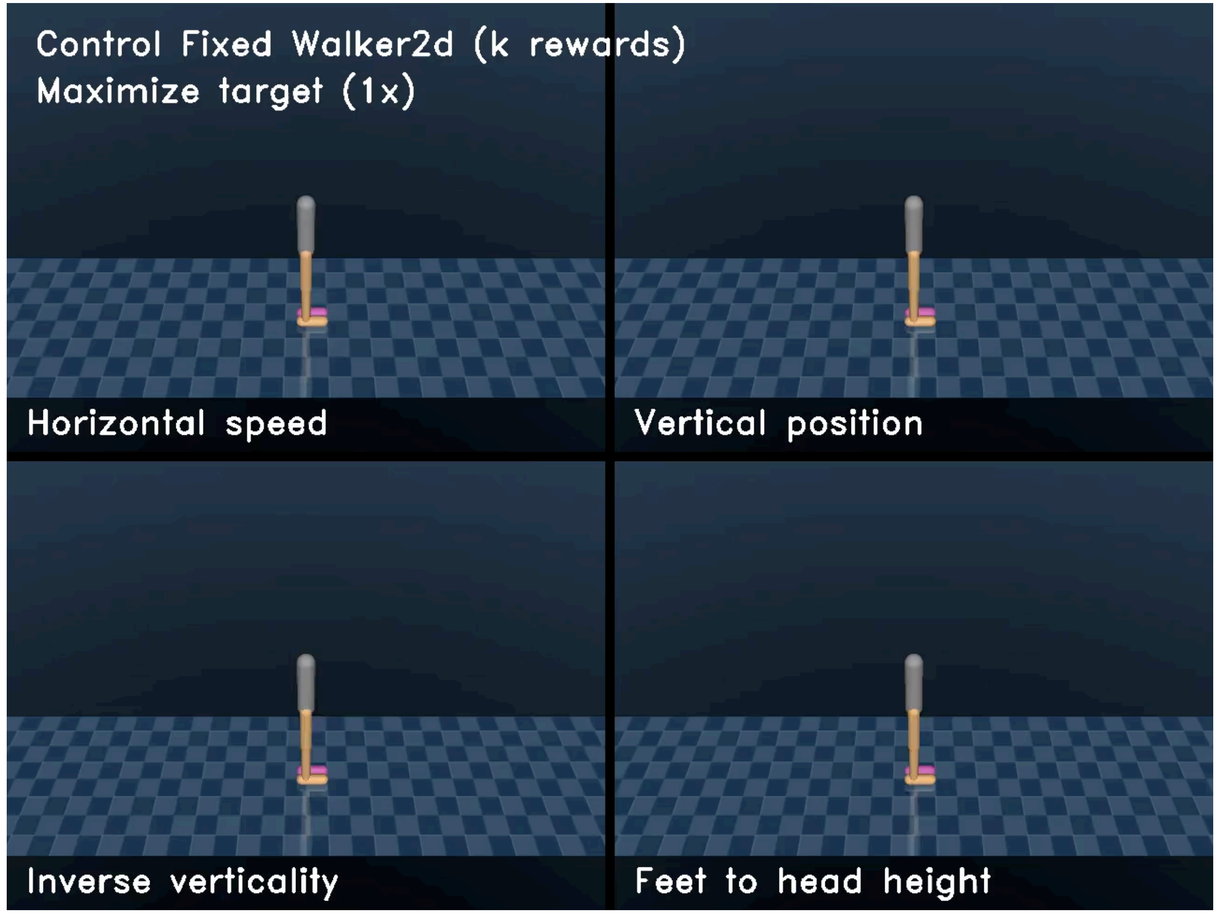
Sanchez-Gonzalez et al., 2018, ICML

Control: Zero-shot control

Sanchez-Gonzalez et al., 2018, ICML

Control: Multiple reward functions

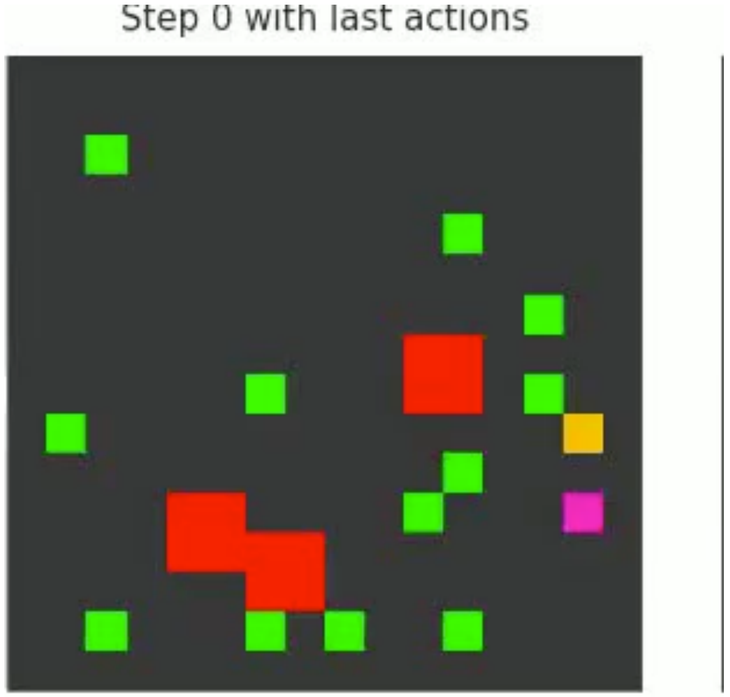




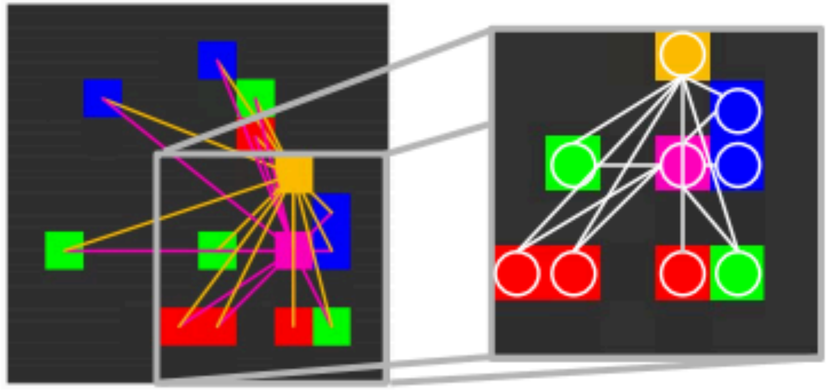
Sanchez-Gonzalez et al., 2018, ICML

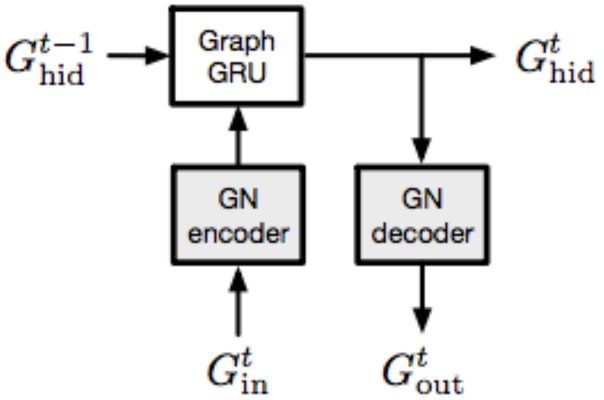
Relational forward models for multi-agent RL

Stag hunt

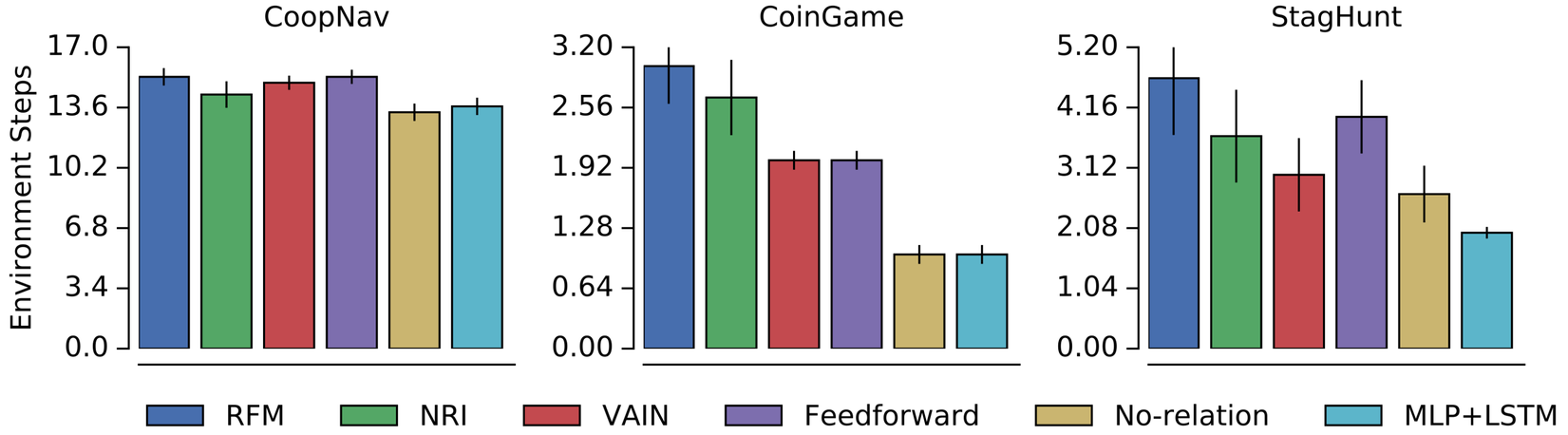


Graph representation



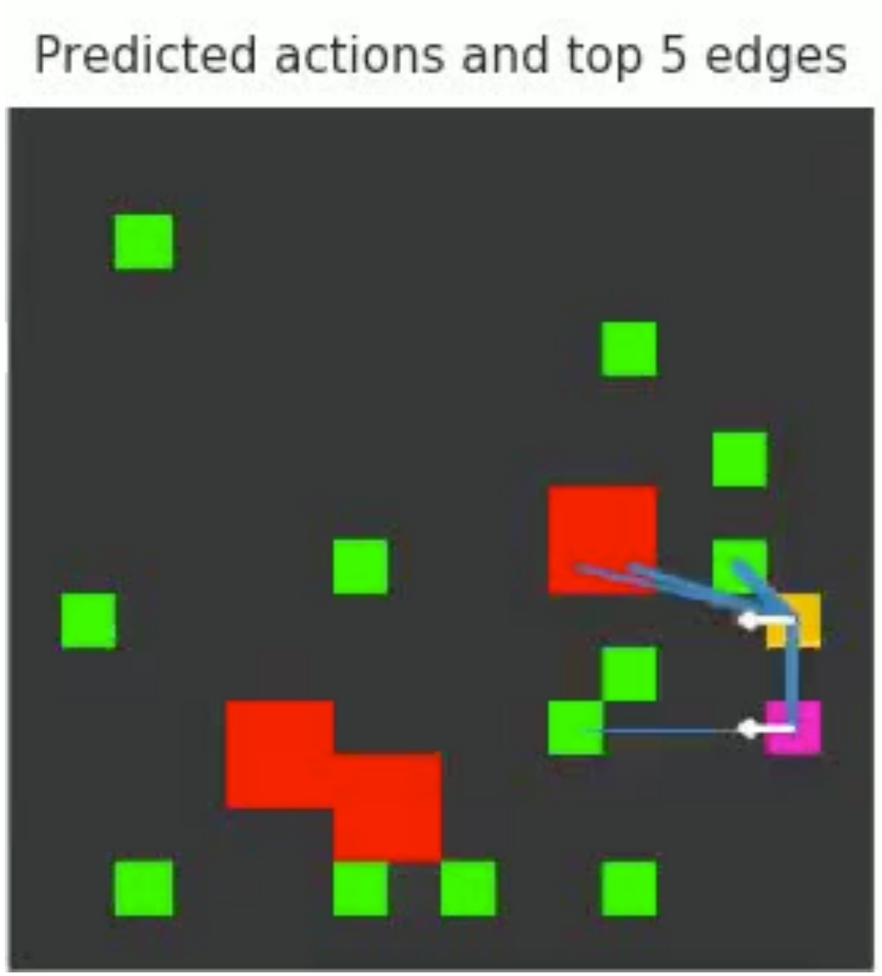


Forward prediction performance

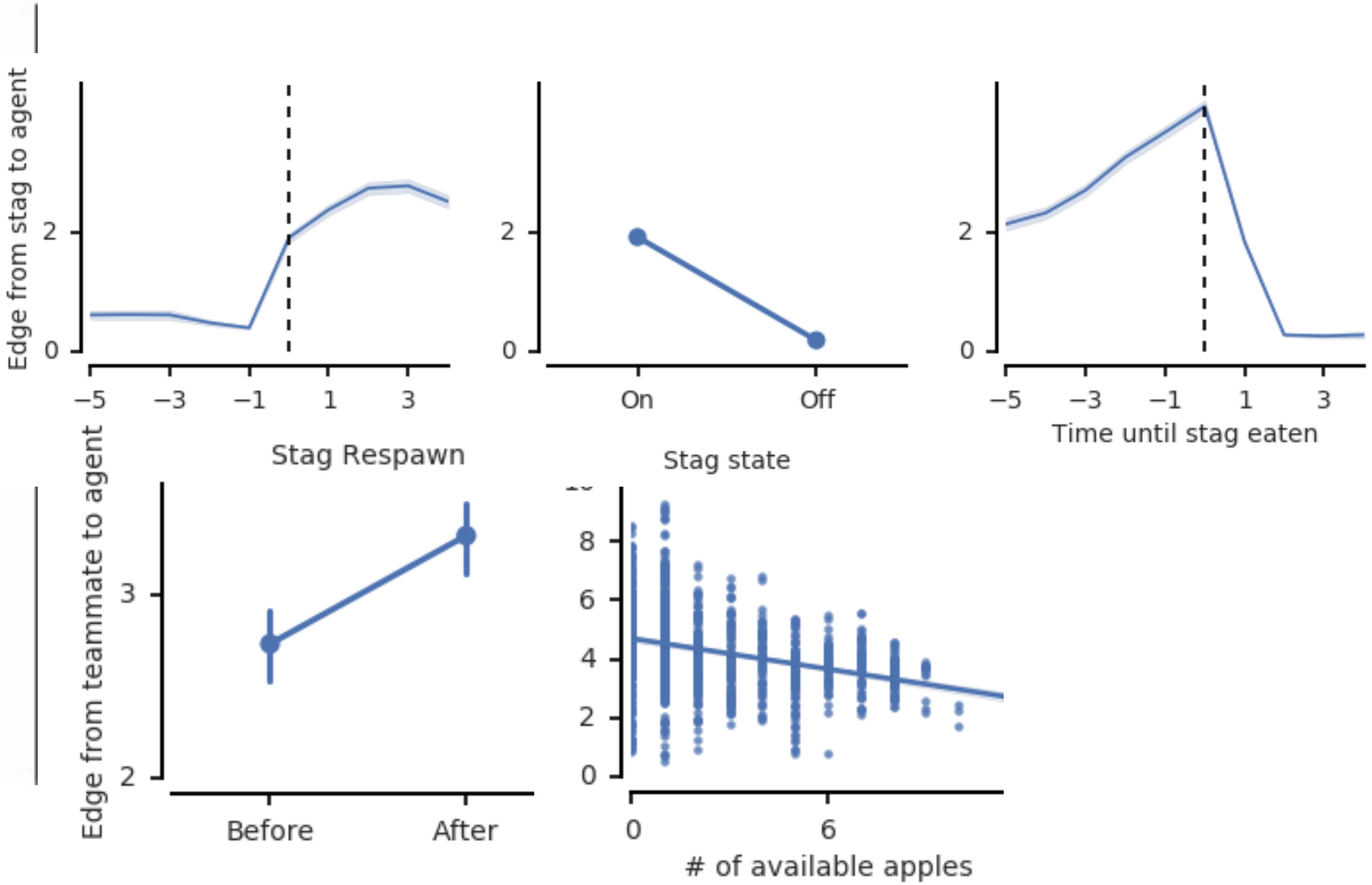


Tacchetti et al., 2019, ICLR

Interpretable learned representations



Magnitude of edge/“message” vector is meaningful

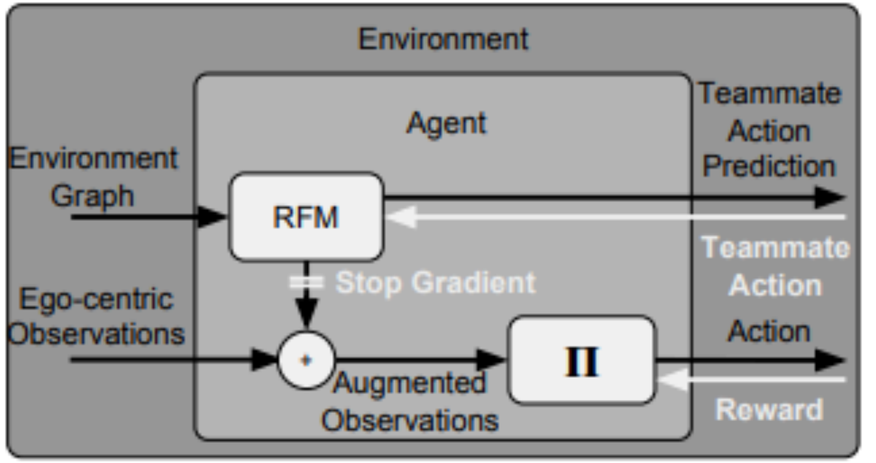


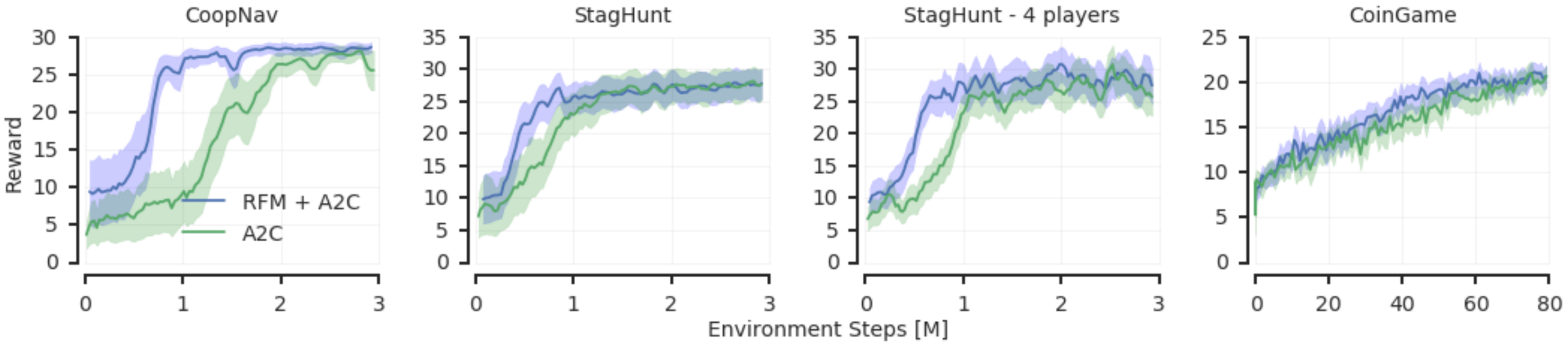
Tacchetti et al., 2019, ICLR

Agents learn faster with model-augmented observations

1. Train a set of agents to perform a game.
2. Train an RFM to predict the agents' future actions.
3. Train a new agent, whose observations are augmented with the RFM's message magnitudes.

The new agent (blue curve) trains faster in all environments.





Tacchetti et al., 2019, ICLR

Humans are a “construction species”

Humans are a “construction species”

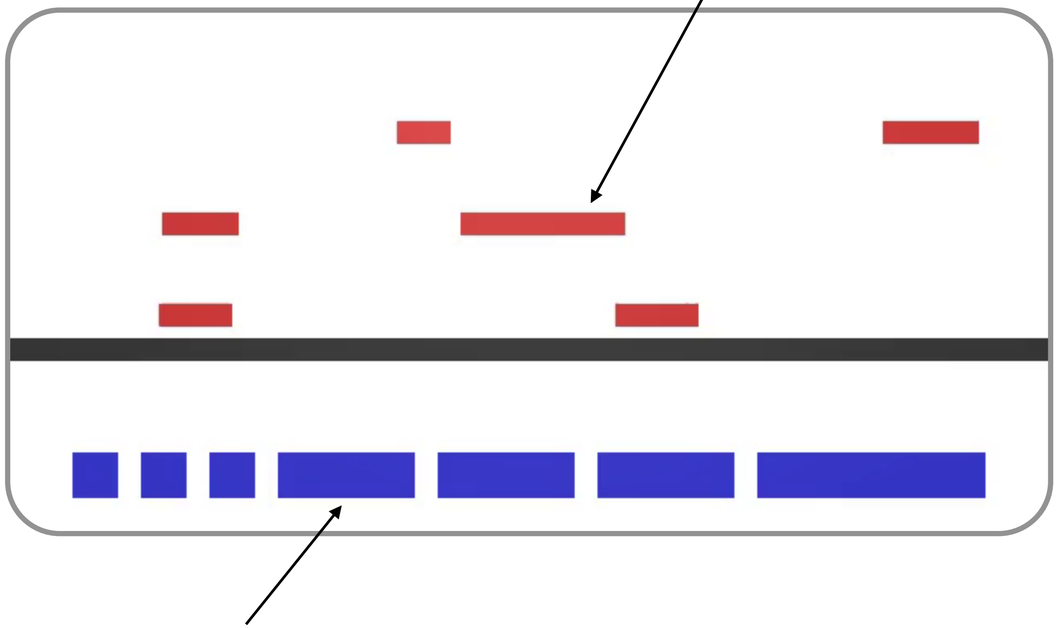
Our construction behaviors are:

Combinatorial
\+ Structured
\+ Use rich physical knowledge

Graph networks

Structured agents for physical construction

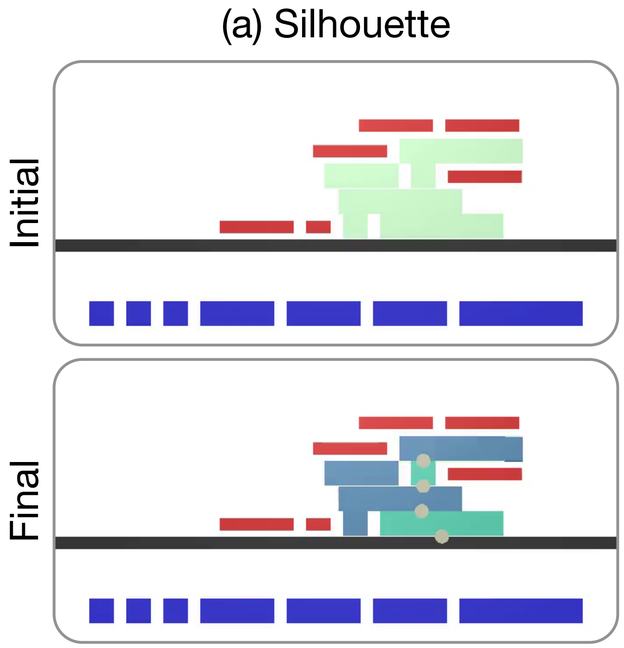
Avoid touching **obstacles**



Pick up **blocks** and place them in the scene
(and optionally make them **sticky**)

Bapst et al., 2019, ICML

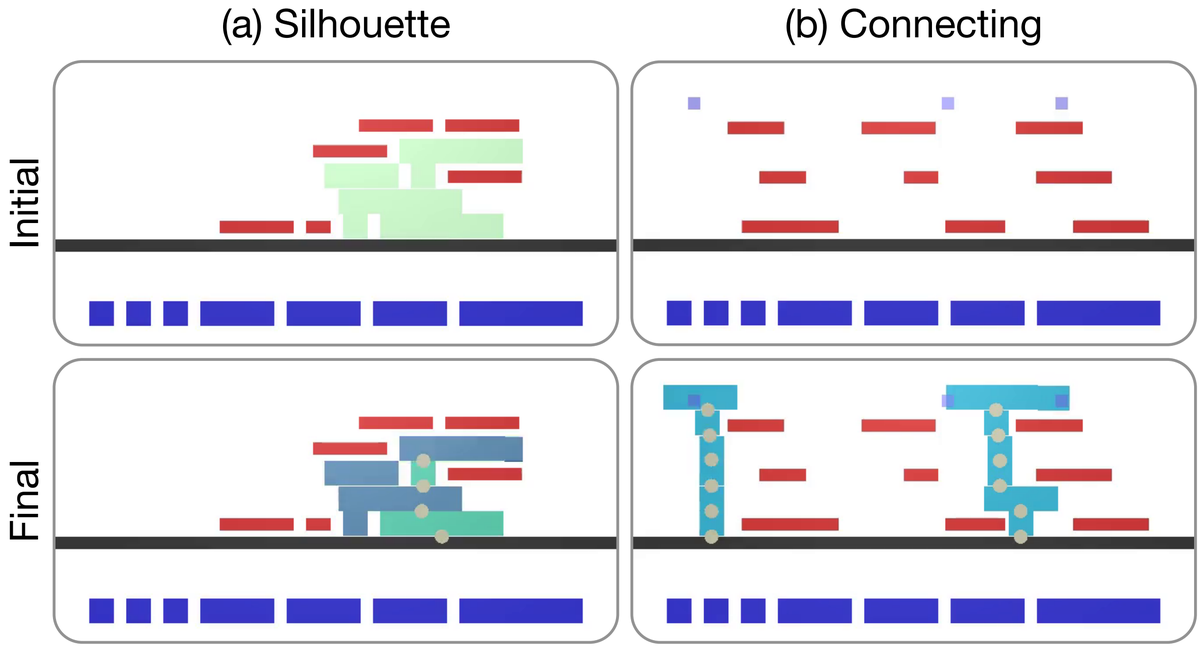
Structured agents for physical construction



+1 per target
-0.5 per sticky block

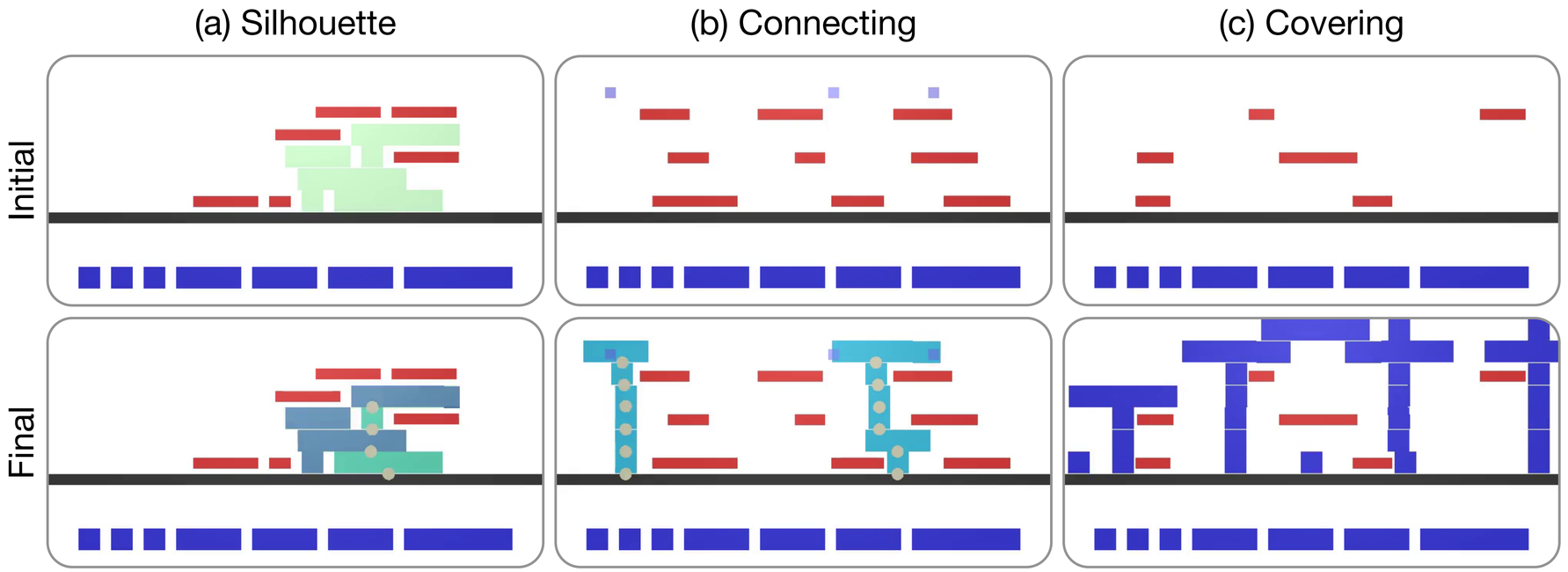
Bapst et al., 2019, ICML

Structured agents for physical construction



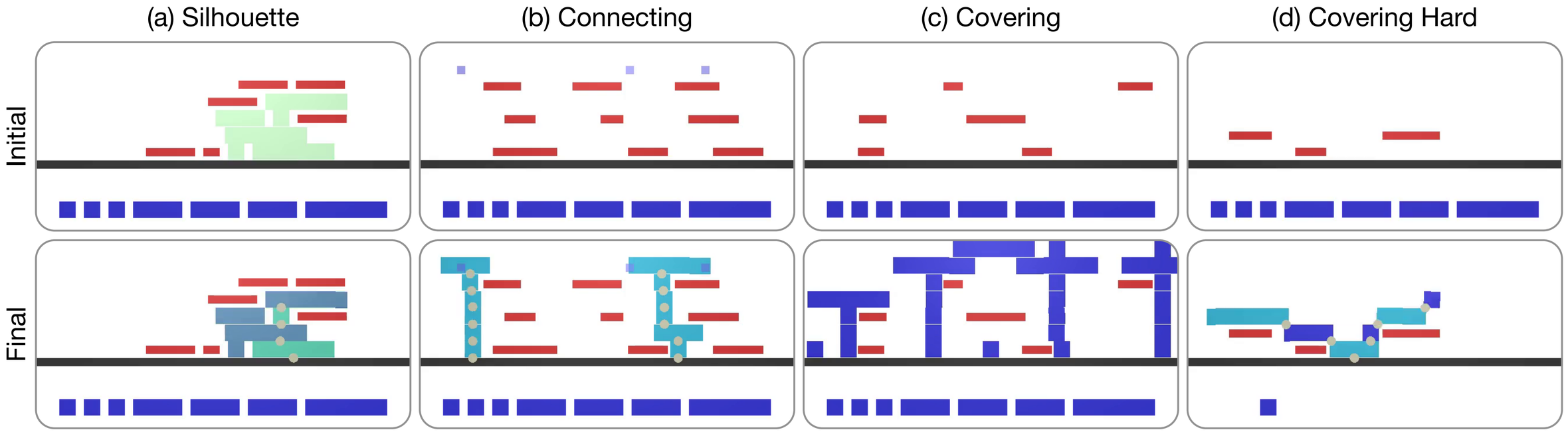
Bapst et al., 2019, ICML

Structured agents for physical construction



Bapst et al., 2019, ICML

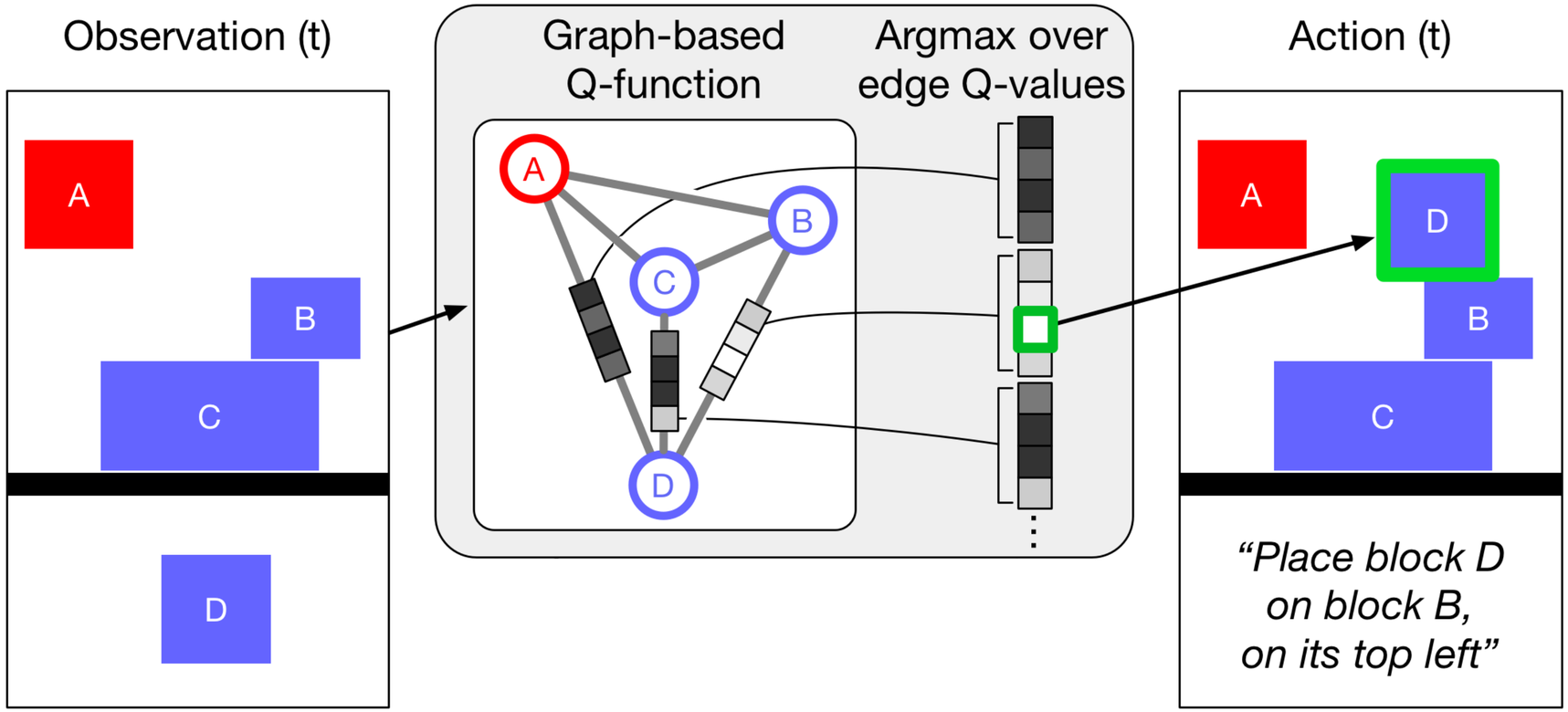
Structured agents for physical construction



Bapst et al., 2019, ICML

Graph net-based agent: model-free

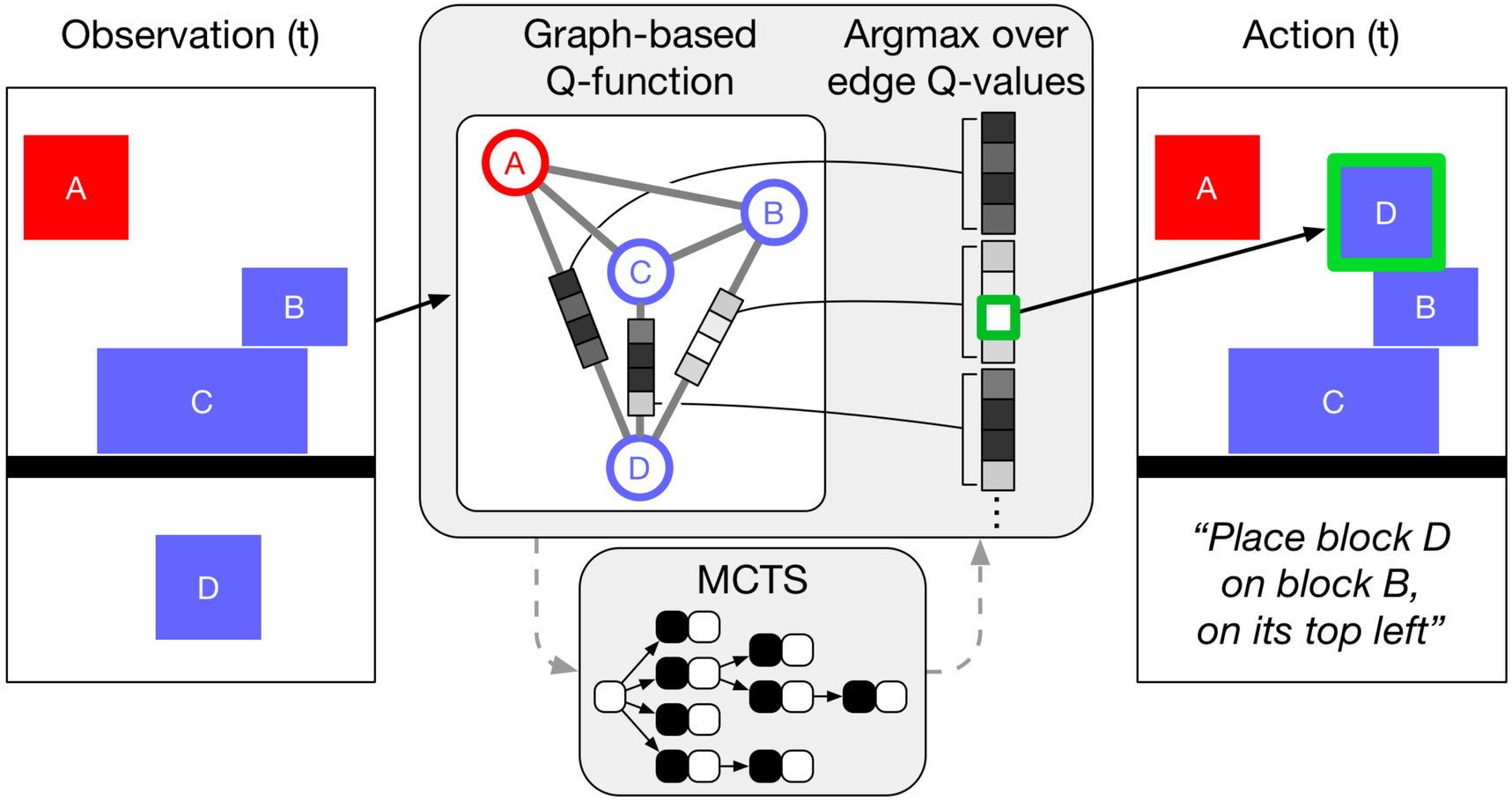
Can be thought of as “graph building” agent



Bapst et al., 2019, ICML

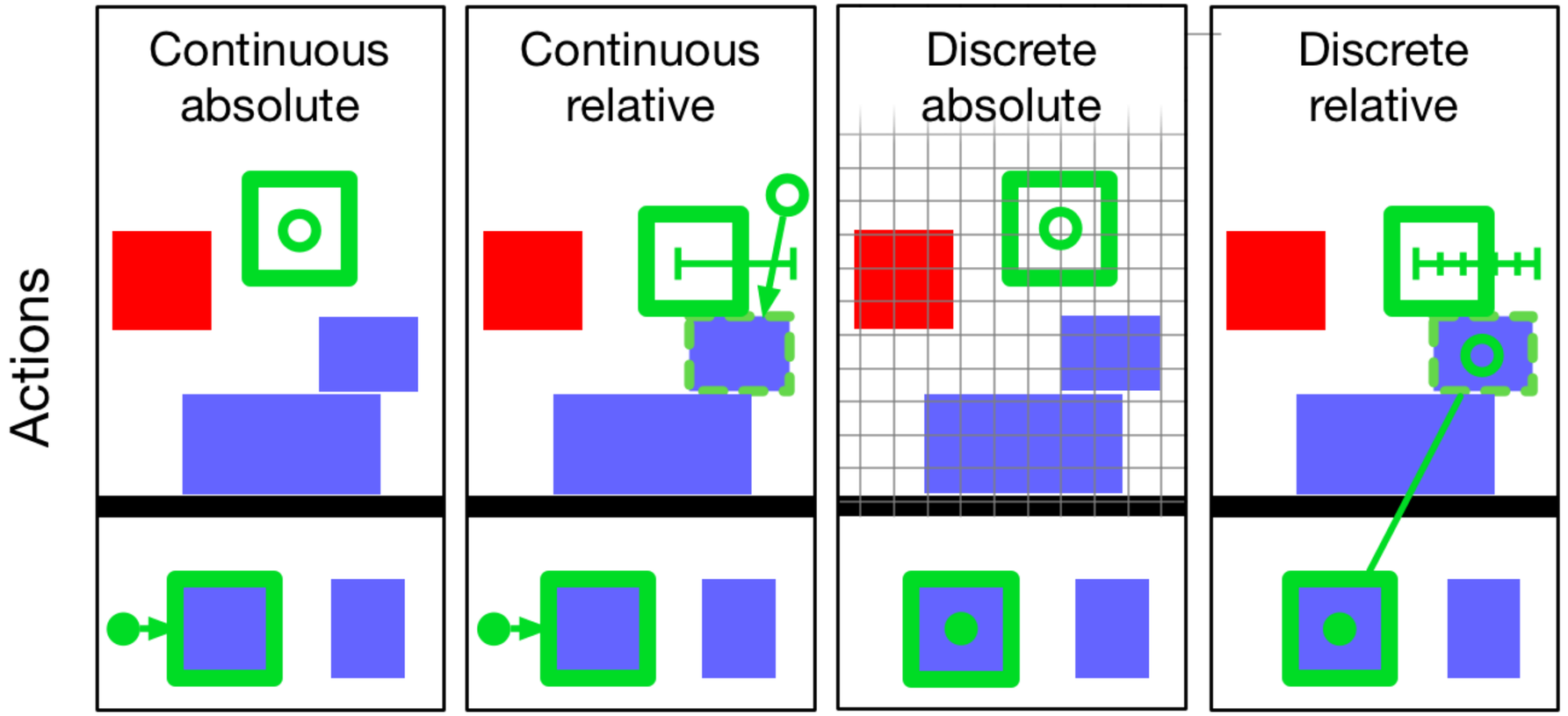
Graph net-based agent: model-based

Can be thought of as “graph building” agent



Bapst et al., 2019, ICML

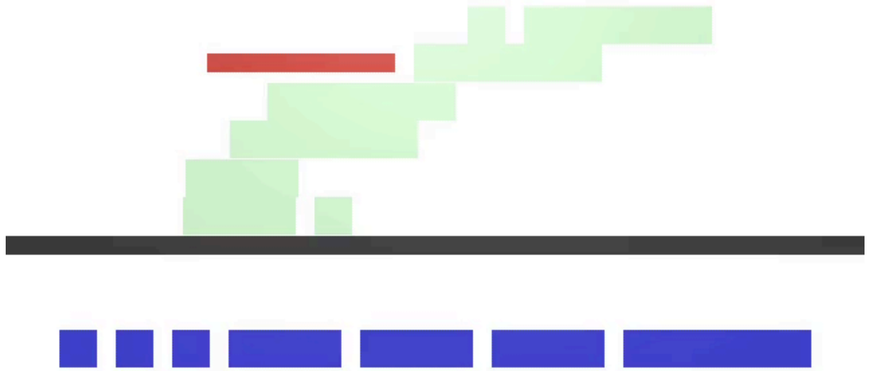
Absolute vs relative actions



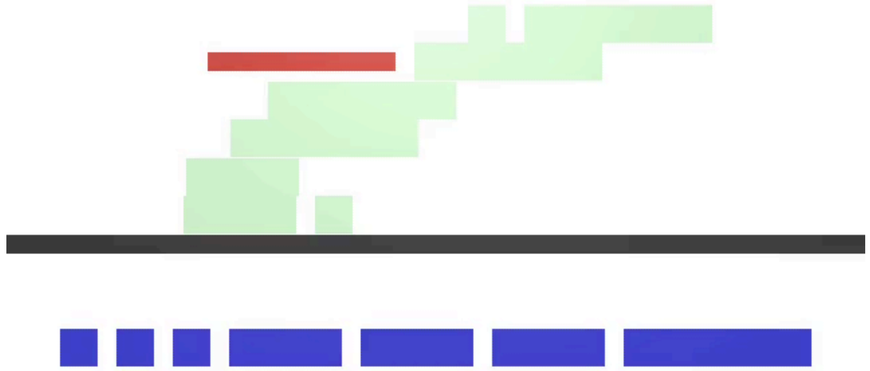
Bapst et al., 2019, ICML

Results: “Silhouette” task

Absolute Actions



Object-Centric Actions

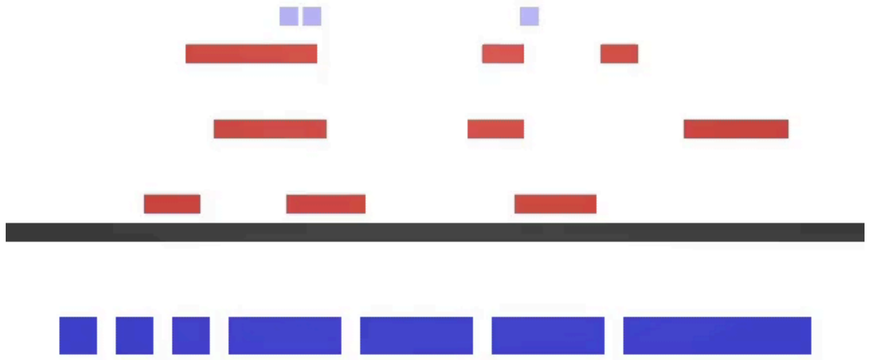


Reward: +1 per target, -0.5 per sticky block

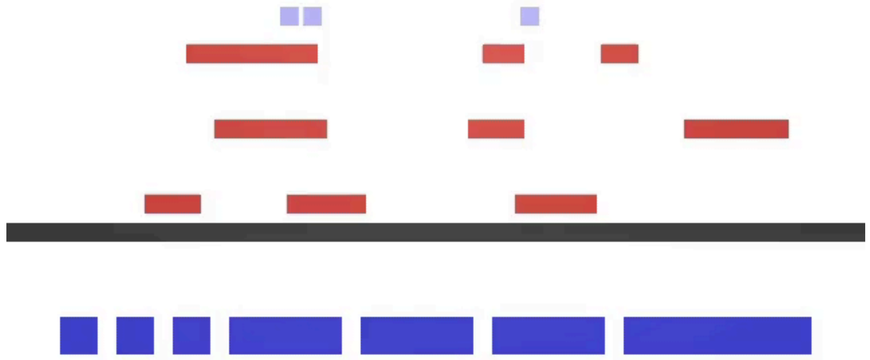
Bapst et al., 2019, ICML

Results: “Connecting” task

Absolute Actions



Object-Centric Actions

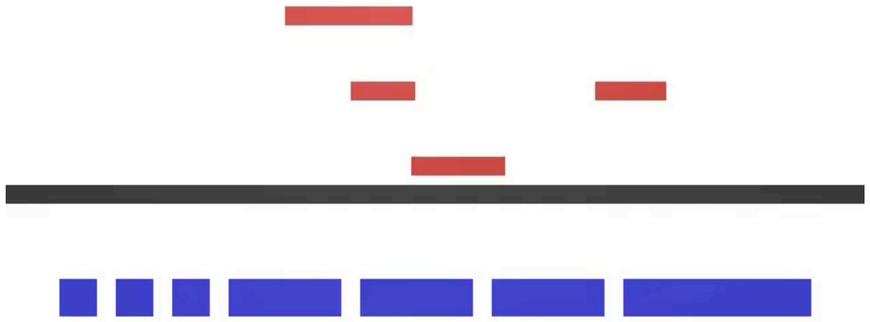


Reward: +1 per target, free sticky blocks

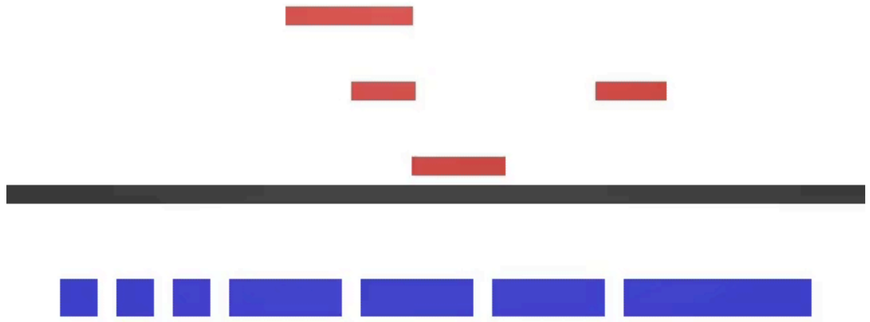
Bapst et al., 2019, ICML

Results: “Covering” task

Absolute Actions



Object-Centric Actions

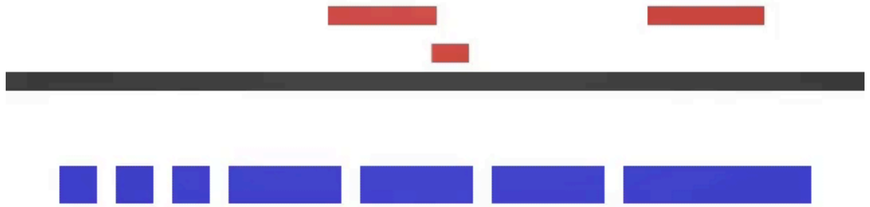


Reward: proportional to length covered, -2 per sticky block

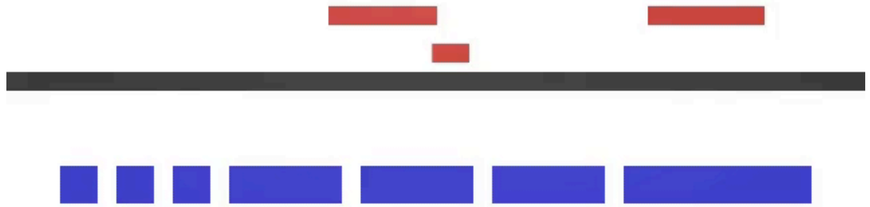
Bapst et al., 2019, ICML

Results: “Covering hard” task

Absolute Actions



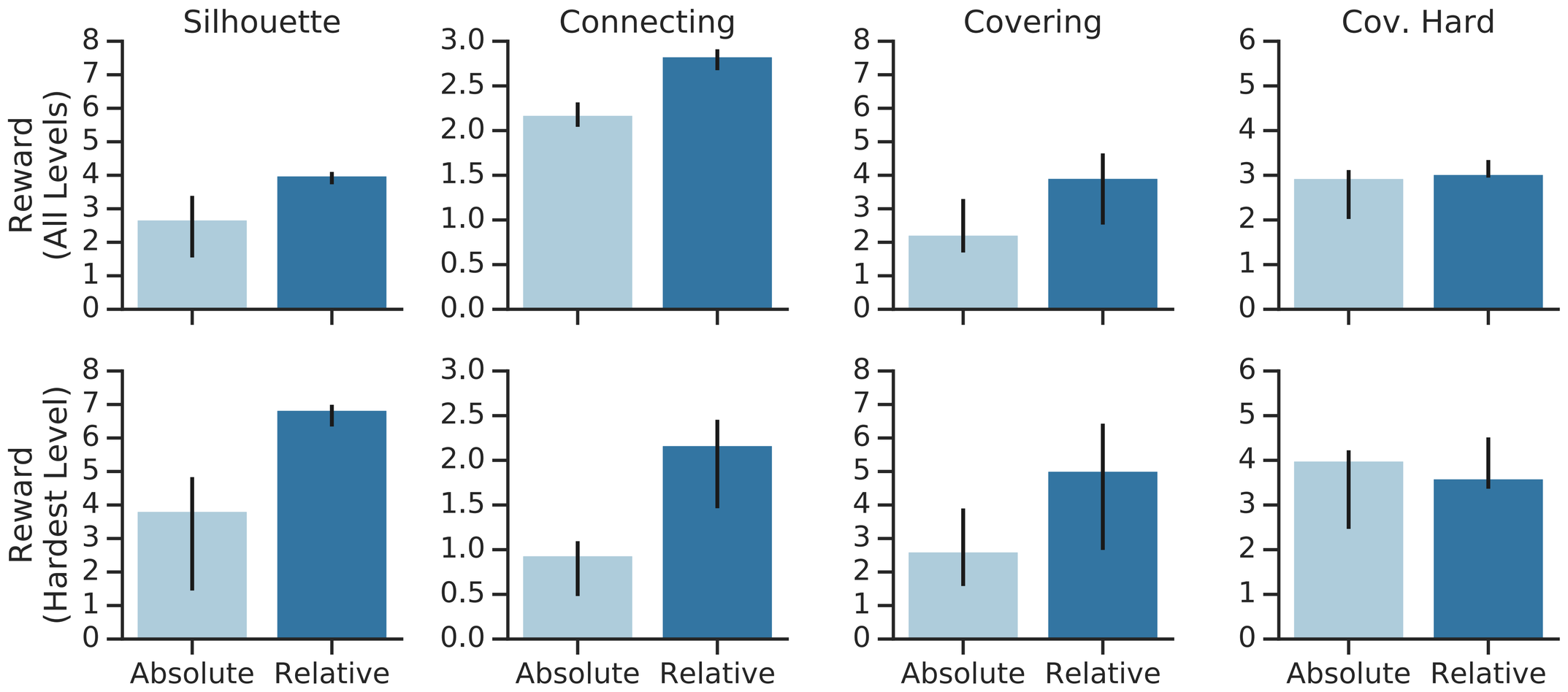
Object-Centric Actions



Reward: proportional to length covered, -0.5 per sticky block

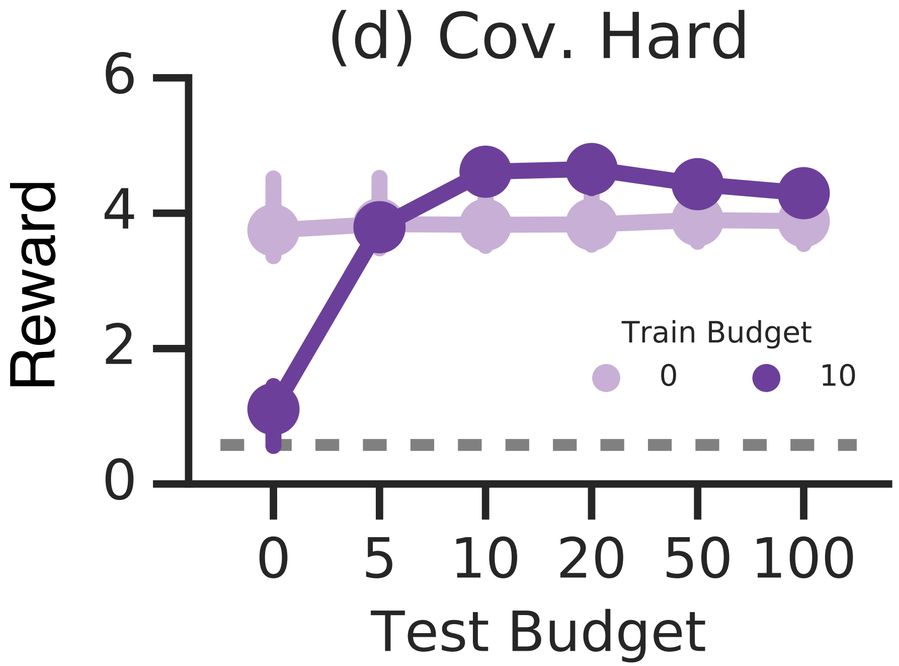
Bapst et al., 2019, ICML

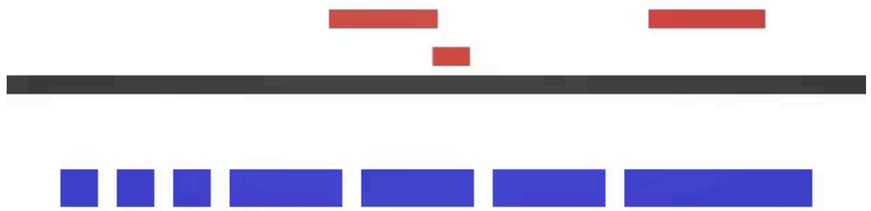
Results: Absolute vs relative actions



Bapst et al., 2019, ICML

Results: Planning agent (using MCTS)





Bapst et al., 2019, ICML

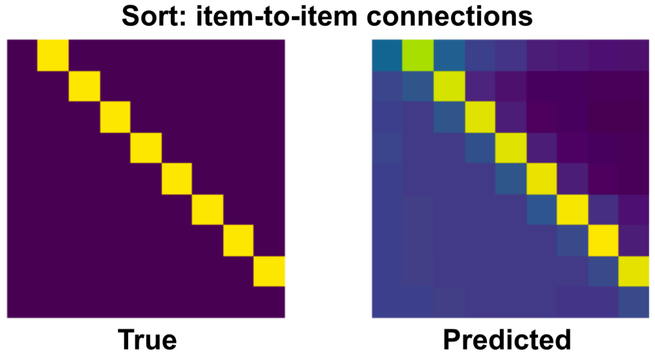
Build Graph Nets in Tensorflow

github.com/deepmind/graph_nets

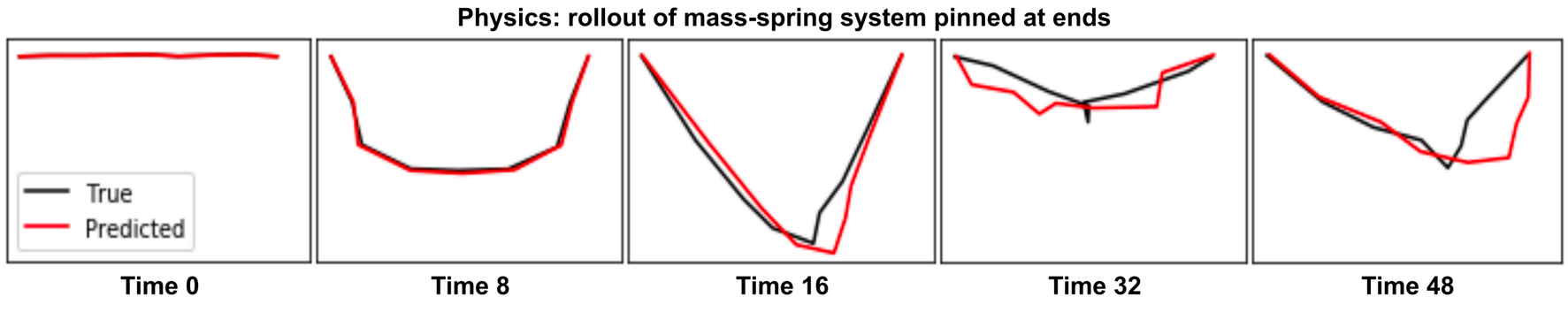
IPython Notebook demos
(All use same architecture)

Shortest path:

Sorting:



Predicting physics:



Build Graph Nets in Tensorflow

github.com/deepmind/graph_nets

Example code for getting started:

```
# Provide your own functions to generate graph-structured data.
input_graphs = get_graphs()

# Create the graph network.
graph_net_module = gn.modules.GraphNetwork(
    edge_model_fn=lambda: snt.nets.MLP([32, 32]),
    node_model_fn=lambda: snt.nets.MLP([32, 32]),
    global_model_fn=lambda: snt.nets.MLP([32, 32]))

# Pass the input graphs to the graph network, and return the output graphs.
output_graphs = graph_net_module(input_graphs)
```

For GNN libraries in PyTorch, check out:

- pytorch_geometric: github.com/rusty1s/pytorch_geometric (for a GN analog, see `MetaLayer`)
- Deep Graph Library: github.com/dmlc/dgl

Conclusions

- Graph neural networks: a first-class member of the deep learning toolkit.

Conclusions

- Graph neural networks: a first-class member of the deep learning toolkit.

- Learned message-passing in graphs can support classification/regression, simulation, but also more sophisticated patterns of reasoning.

Conclusions

- Graph neural networks: a first-class member of the deep learning toolkit.
- Learned message-passing in graphs can support classification/regression, simulation, but also more sophisticated patterns of reasoning.
- RL agents with graph-structured inputs, outputs, and/or latent representations are effective in domains for which less structured methods struggle.

Conclusions

- Graph neural networks: a first-class member of the deep learning toolkit.
- Learned message-passing in graphs can support classification/regression, simulation, but also more sophisticated patterns of reasoning.
- RL agents with graph-structured inputs, outputs, and/or latent representations are effective in domains for which less structured methods struggle.
- Build Graph Nets in Tensorflow: `github.com/deepmind/graph_nets`

Key collaborators

Razvan Pascanu
Jess Hamrick
Alvaro Sanchez-Gonzalez
Victor Bapst
Kim Stachenfeld
Carl Doersch
Nick Watters
Andrea Tacchetti
Theophane Weber
Daniel Zoran
Mateusz Malinowski
David Raposo
Adam Santoro
Nicholas Heess
Koray Kavukcuoglu

References

Battaglia et al., 2016, NIPS
Watters et al., 2017, NIPS
Raposo et al., 2017, ICLR workshop
Santoro et al., 2017, NIPS
Battaglia et al. 2018 arXiv
Sanchez-Gonzalez et al., 2018, ICML
Tacchetti et al., 2019, ICLR
Bapst et al., 2019, ICML