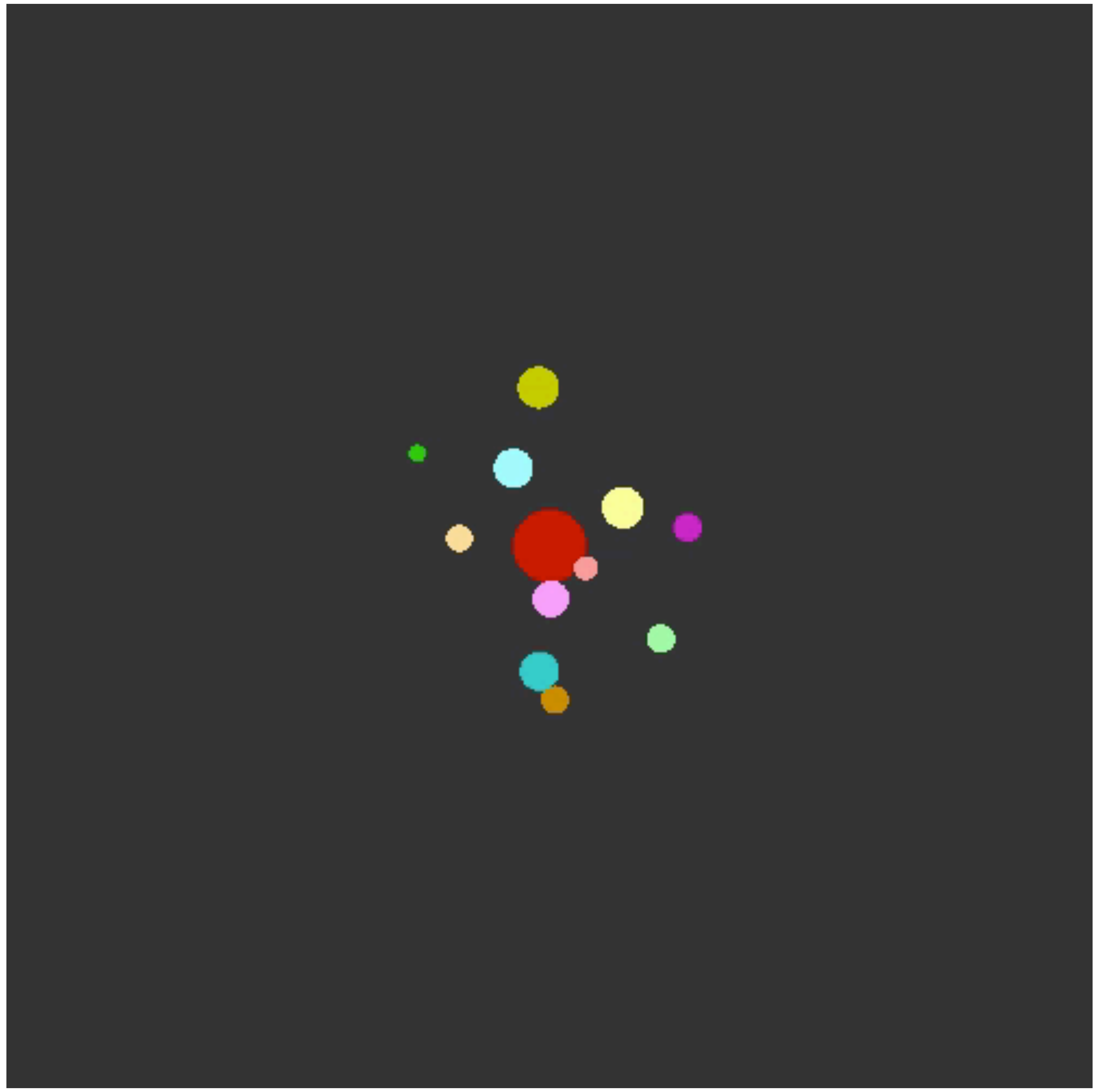
Learning Structured Models of Physics

Peter Battaglia
DeepMind

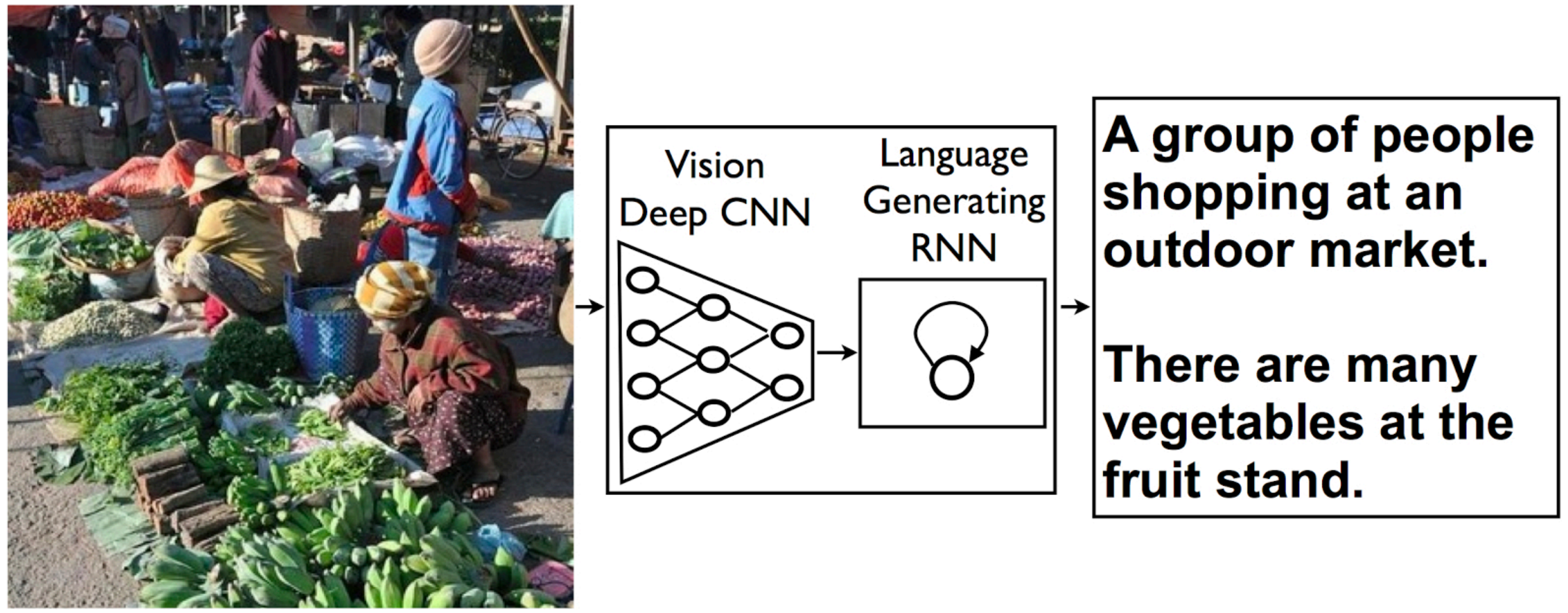
Interpretable Learning in Physical Sciences 2
IPAM, UCLA
October 16, 2019



What is deep learning good at?

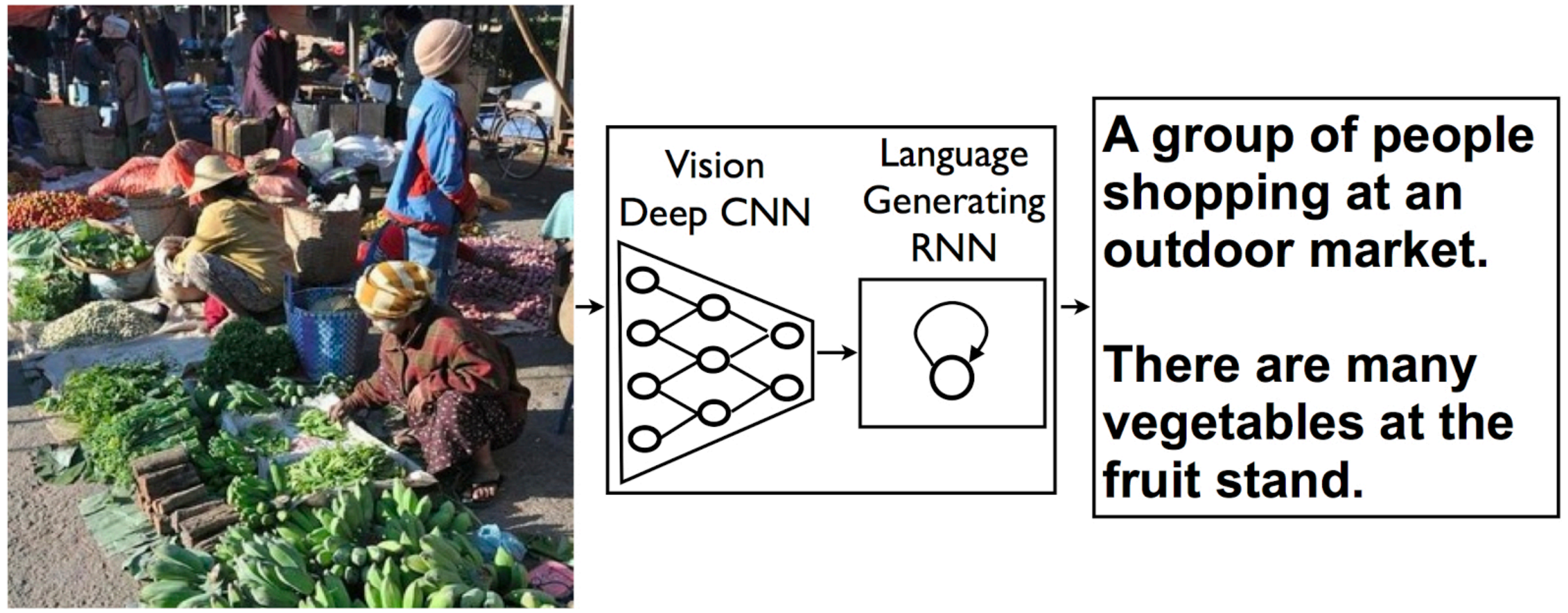
What is deep learning good at?

Image and language processing

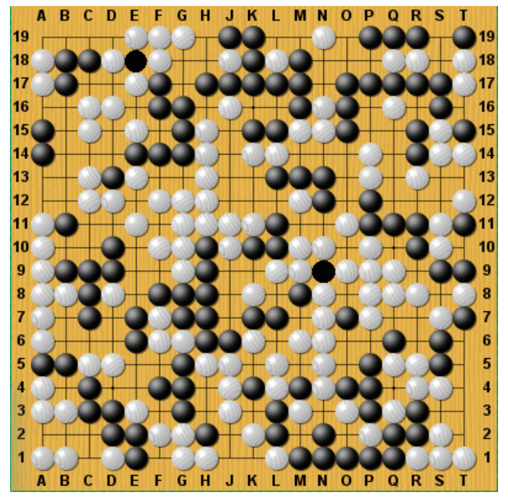


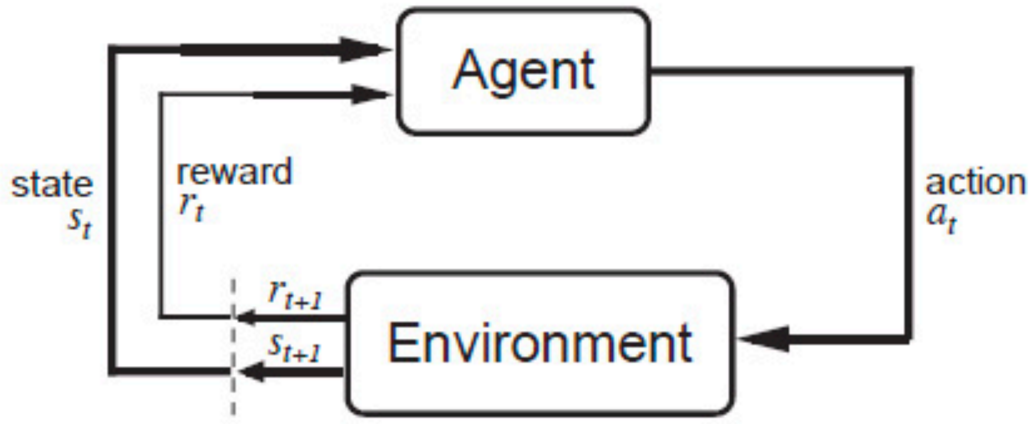
What is deep learning good at?

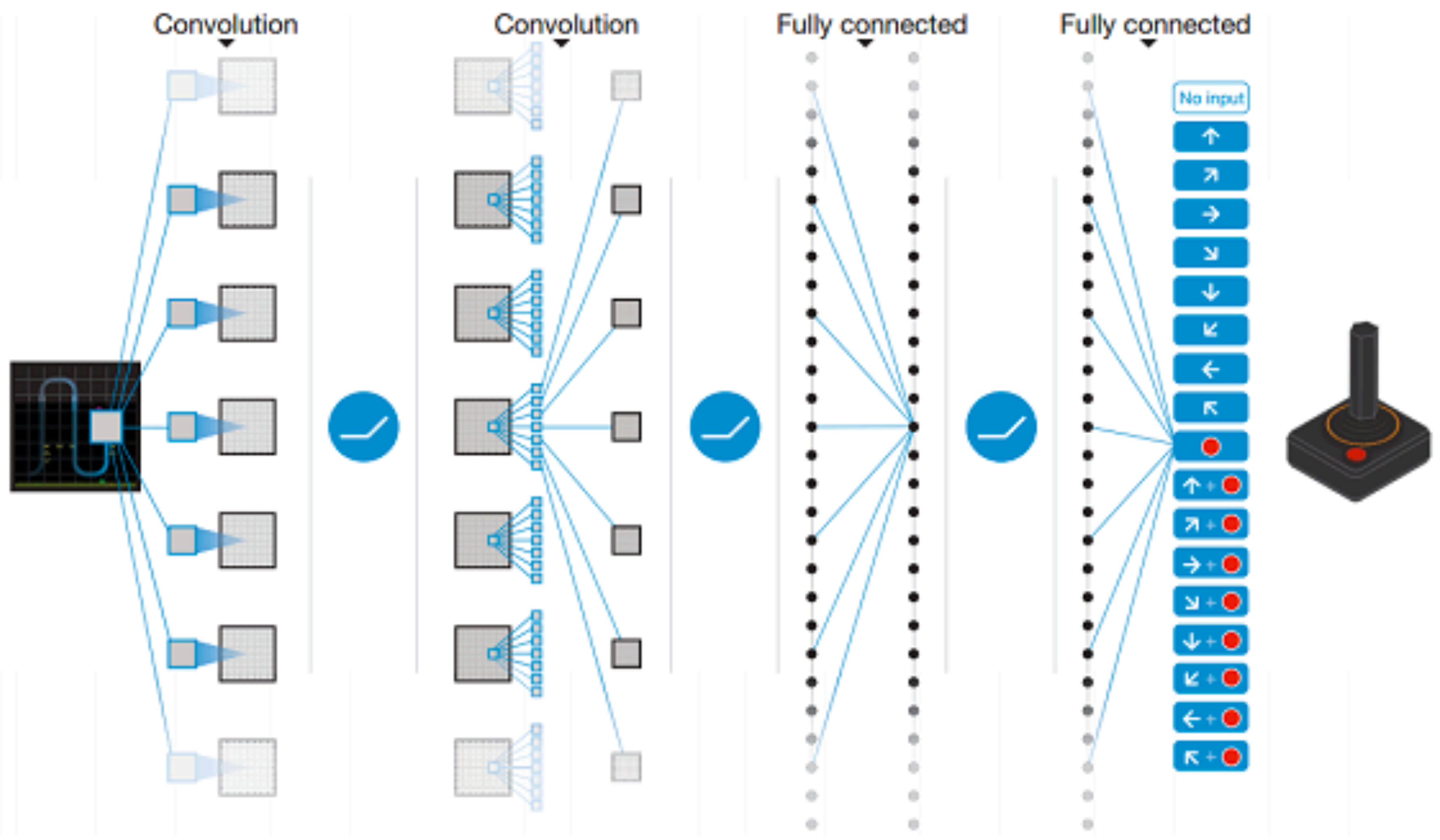
Image and language processing



Games (via deep reinforcement learning)

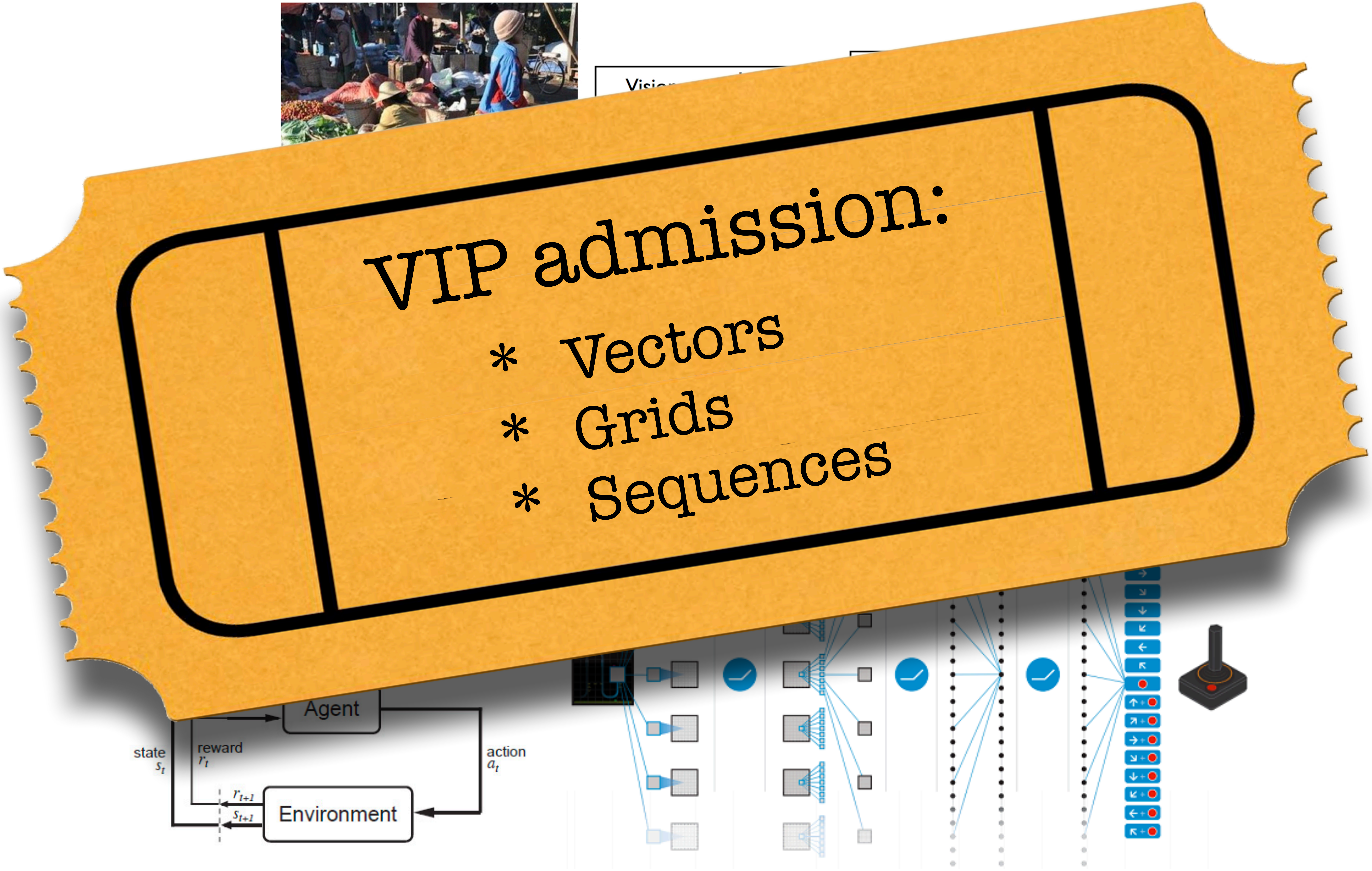






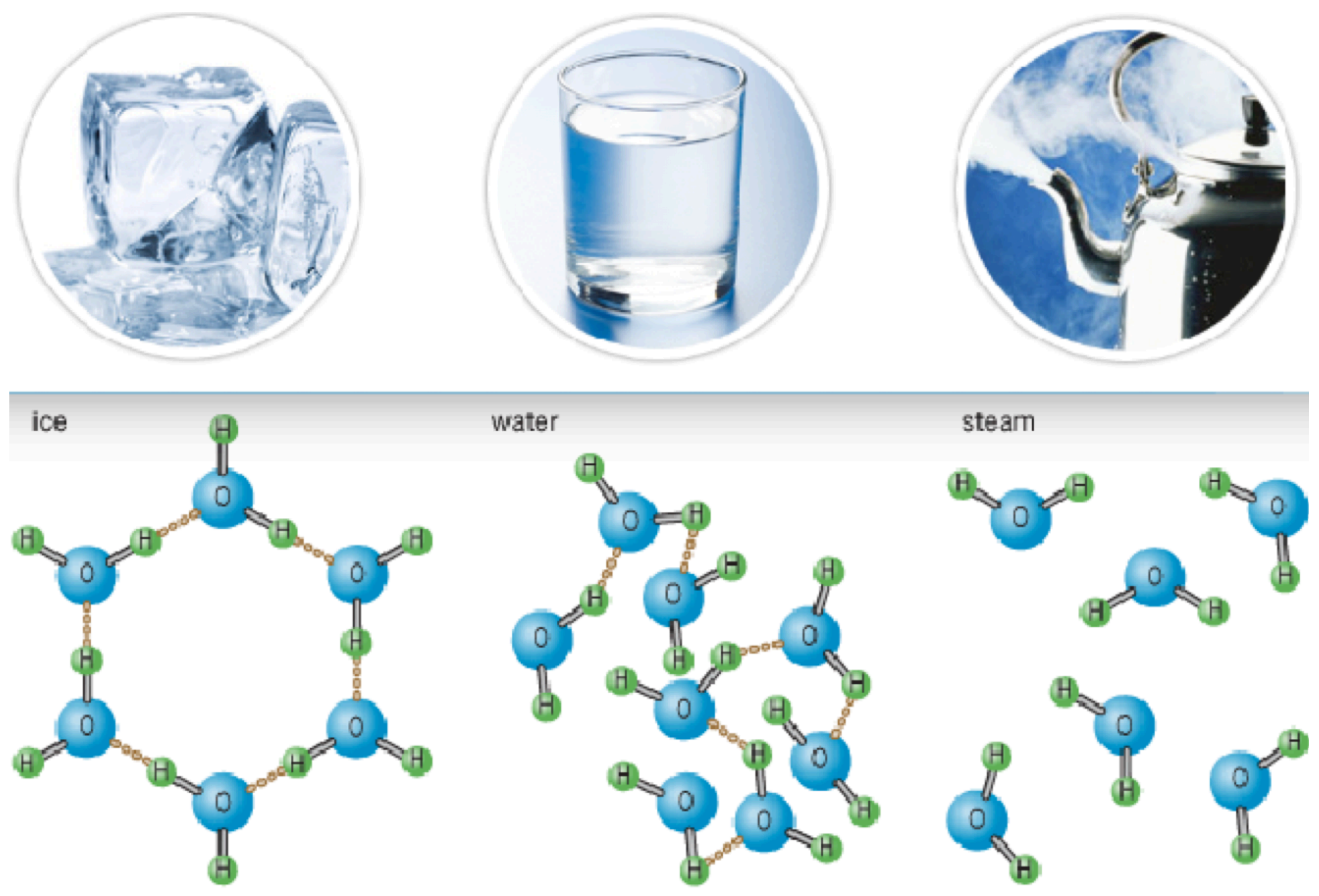
What is deep learning good at?

Image and language processing

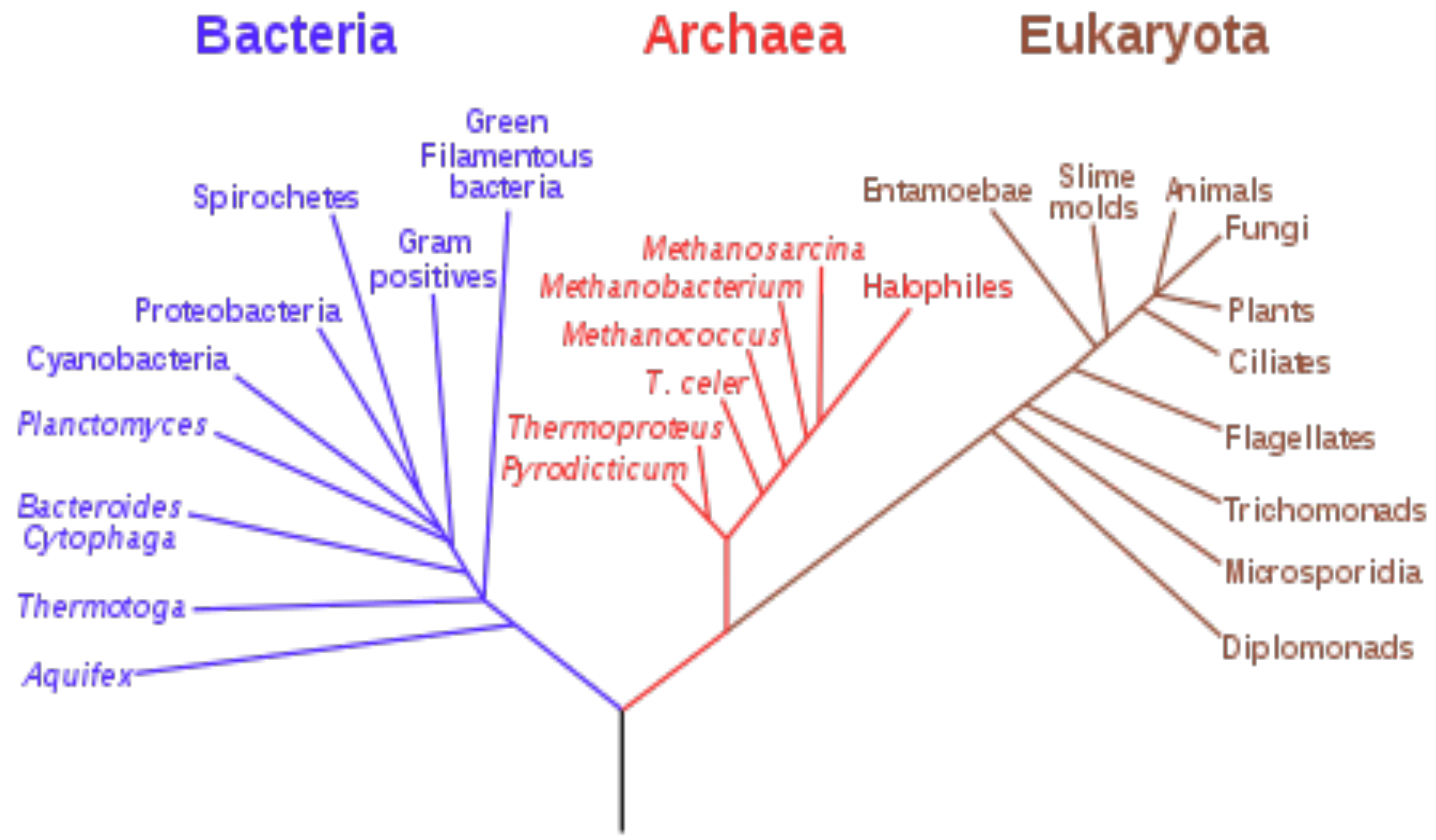


Many complex systems are structured

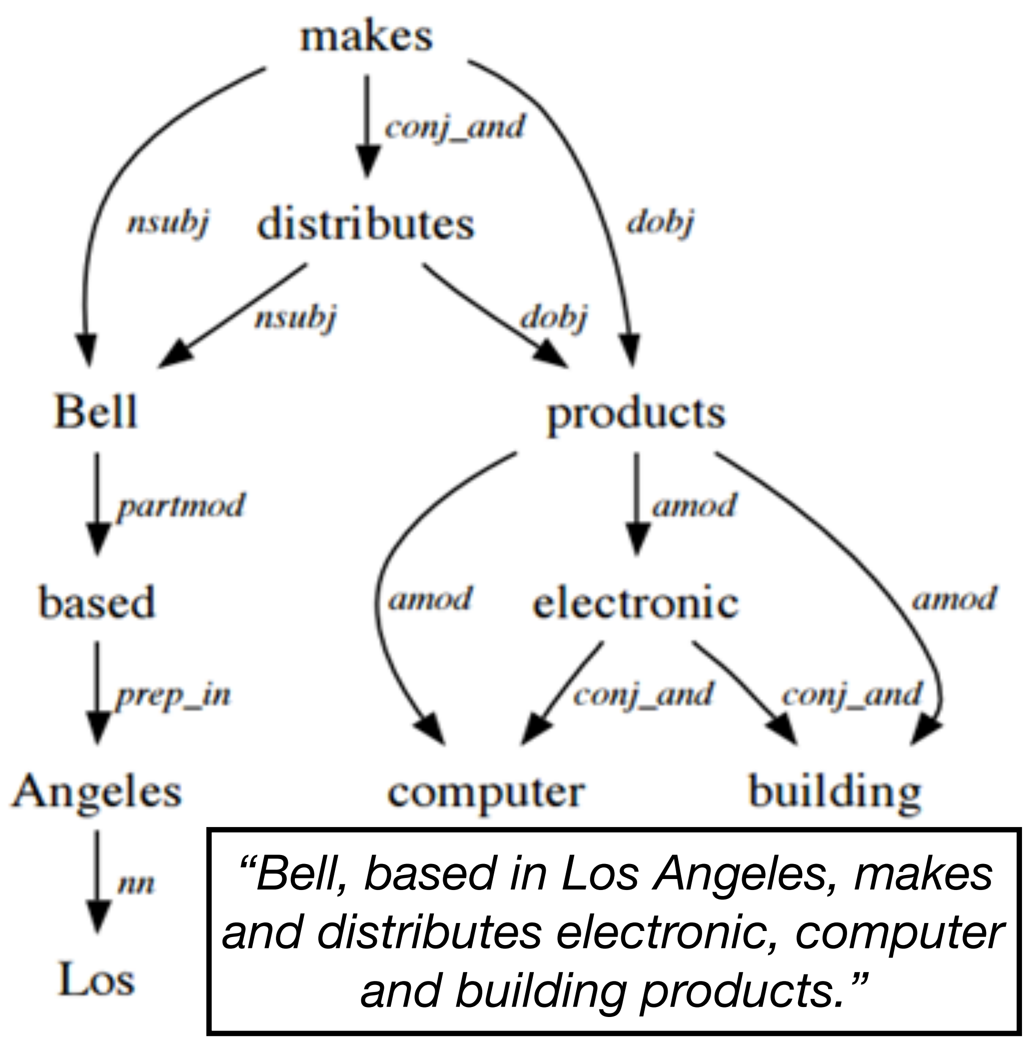
Molecules



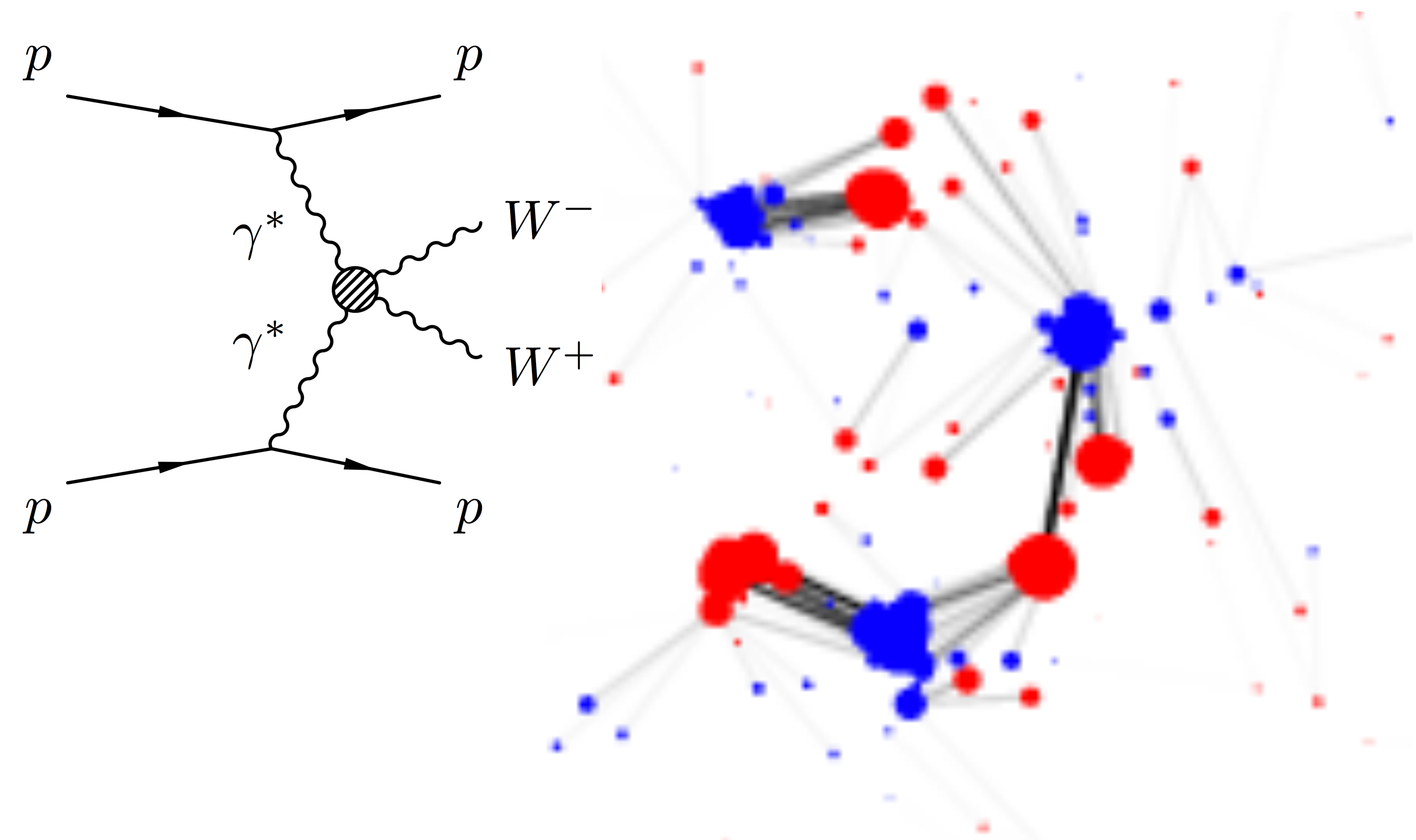
Biological species



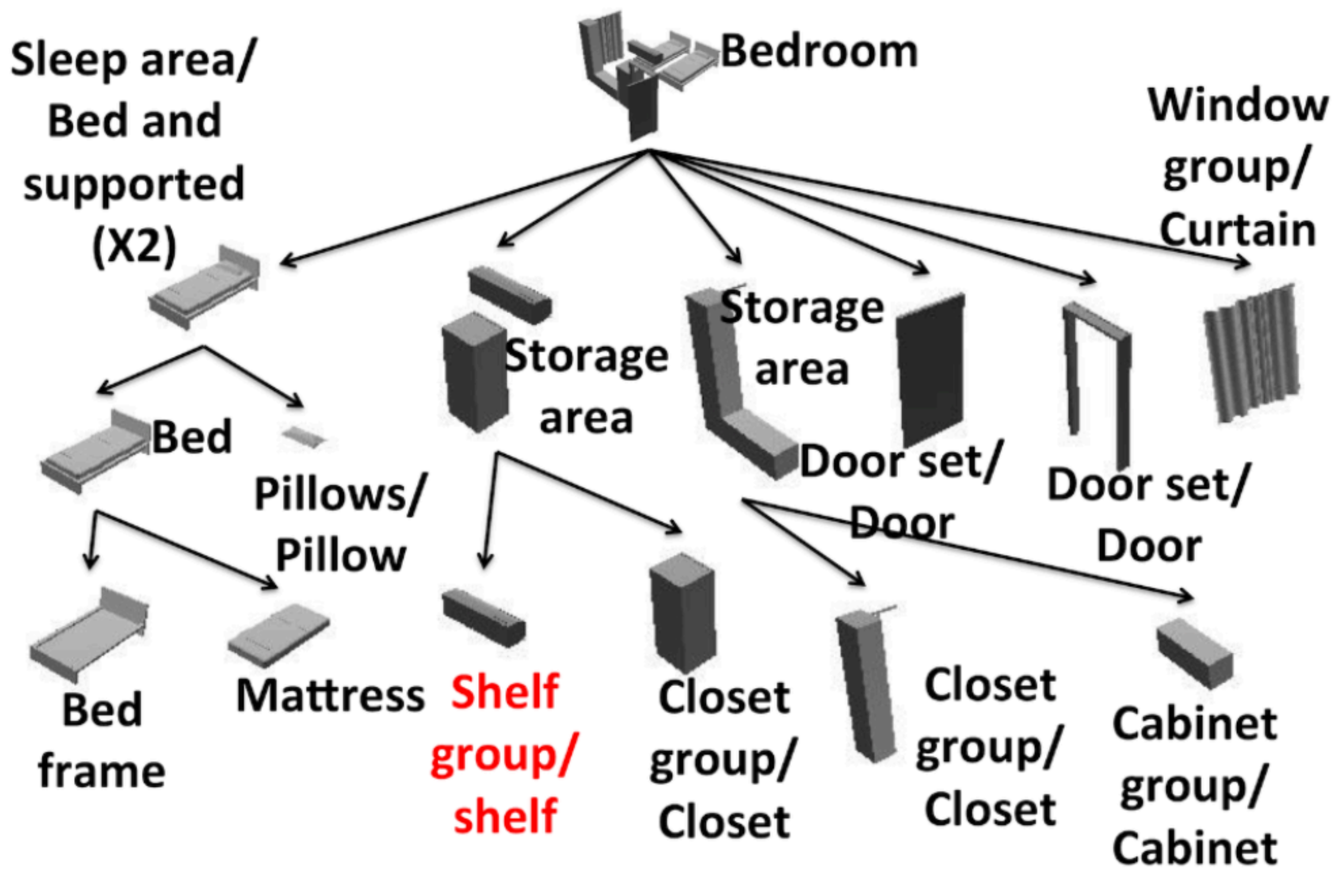
Natural language



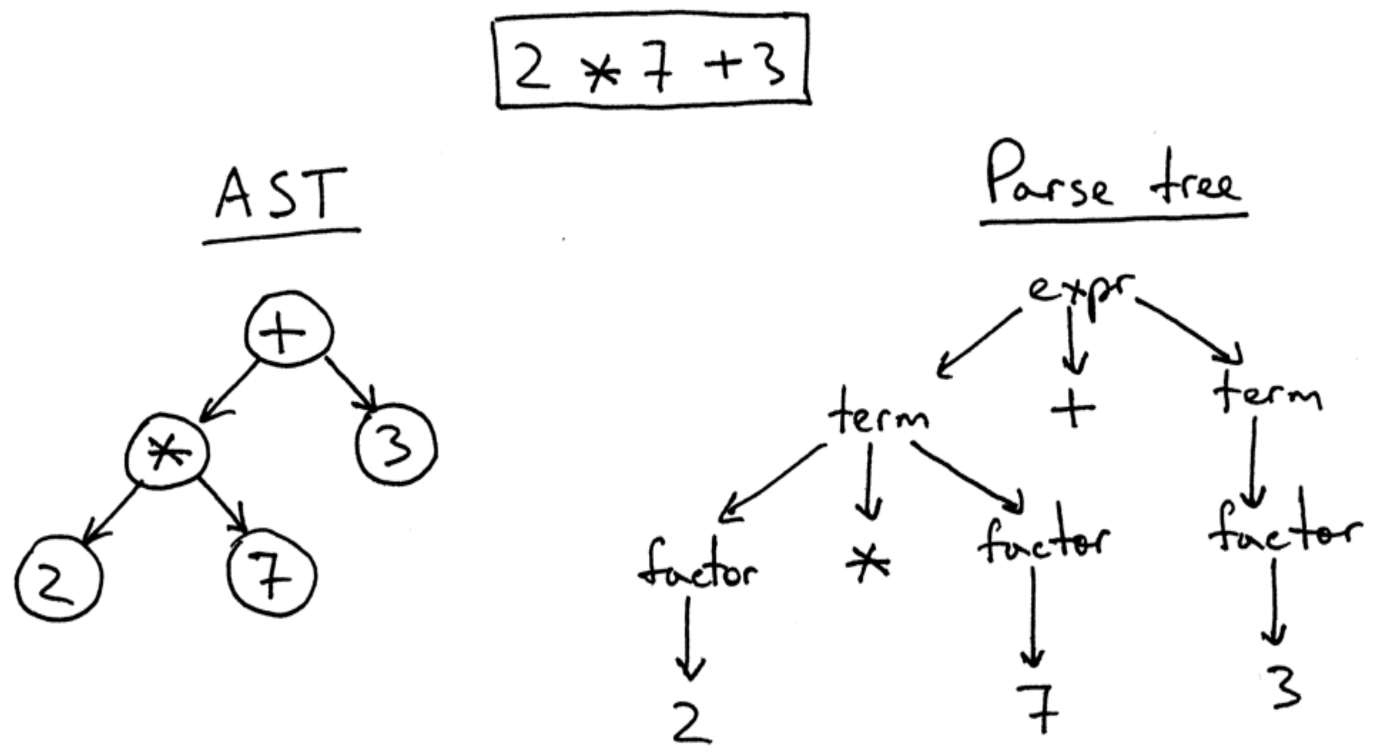
Sub-atomic particles



Everyday scenes

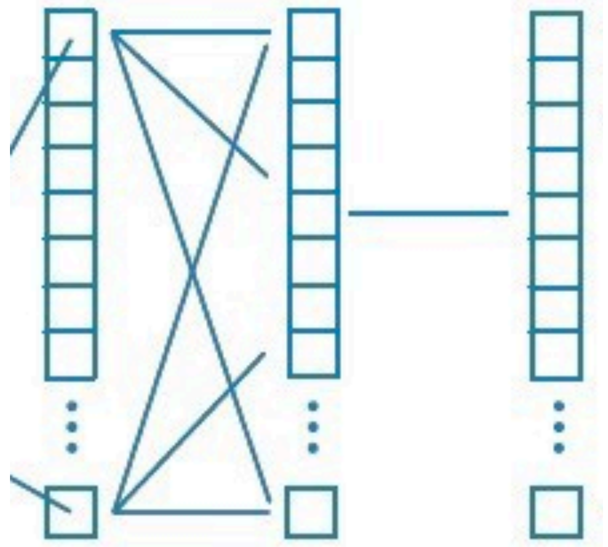


Code

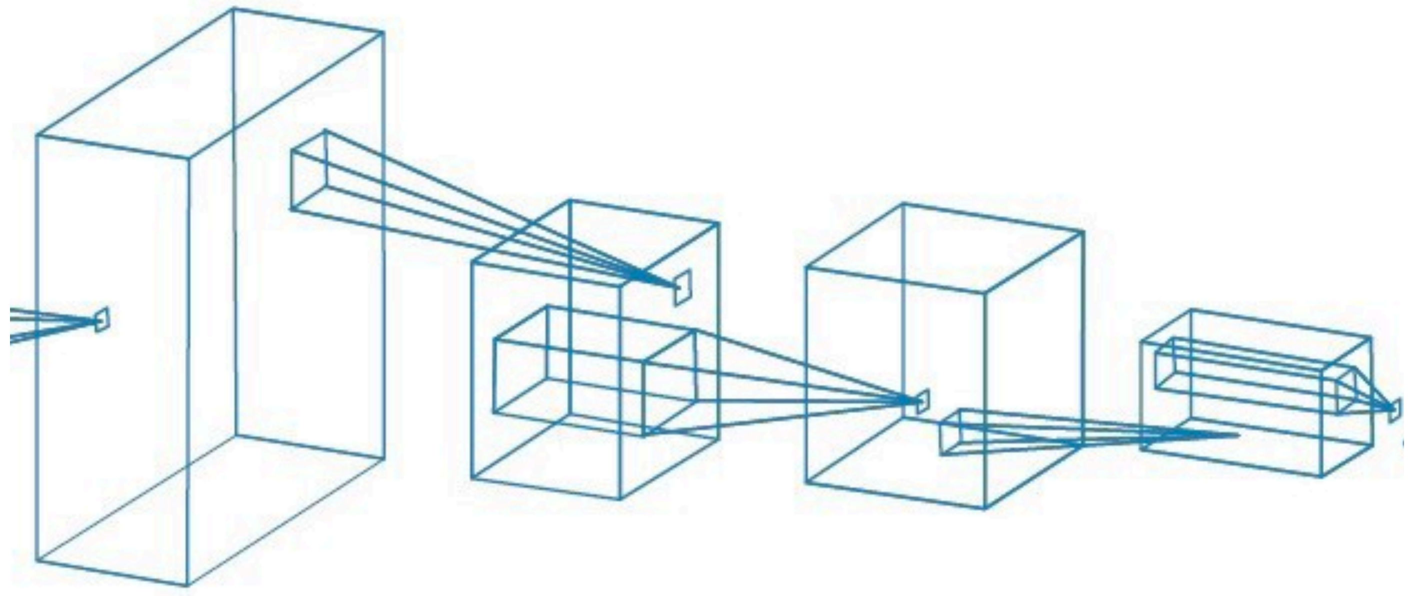


The standard deep learning toolkit…

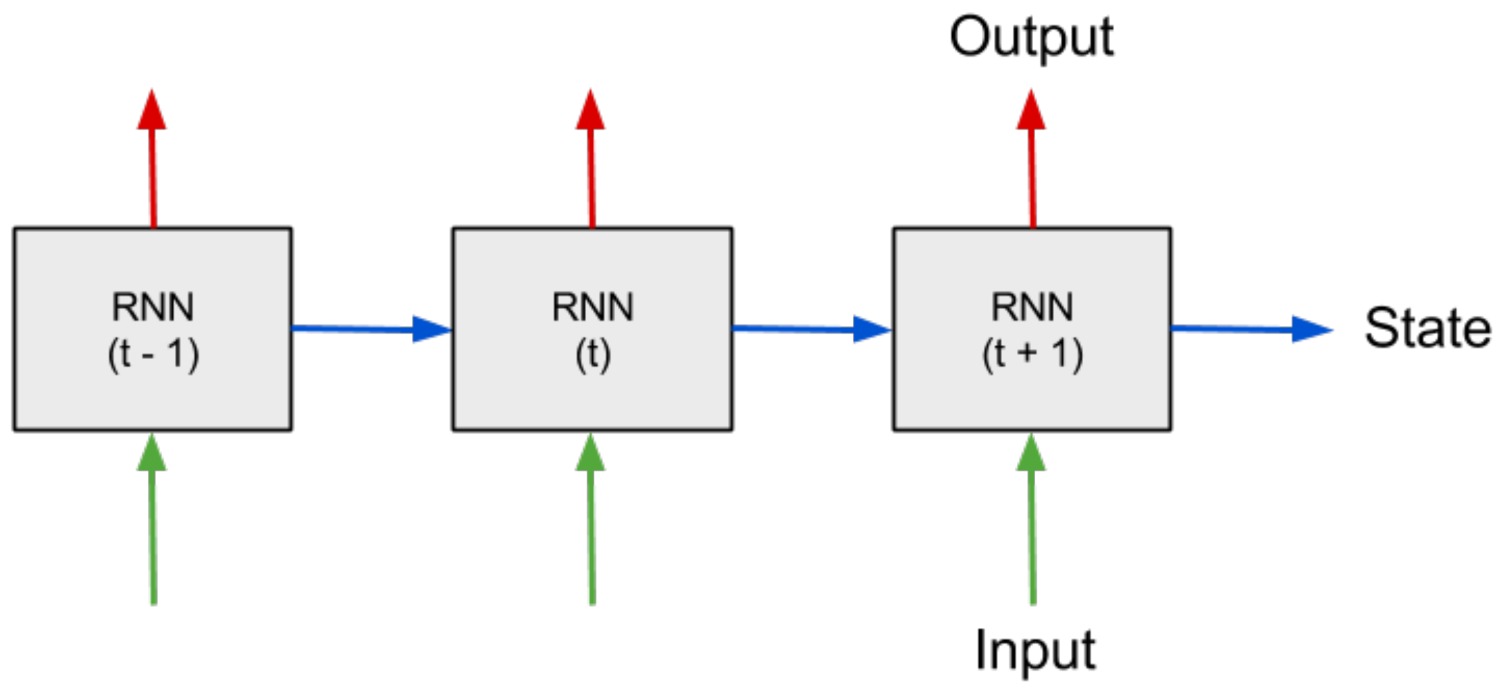
“My data is **vectors**”:
Multi-layer perceptron (MLP)



“My data is **grids**”:
Convolutional neural network (CNN)



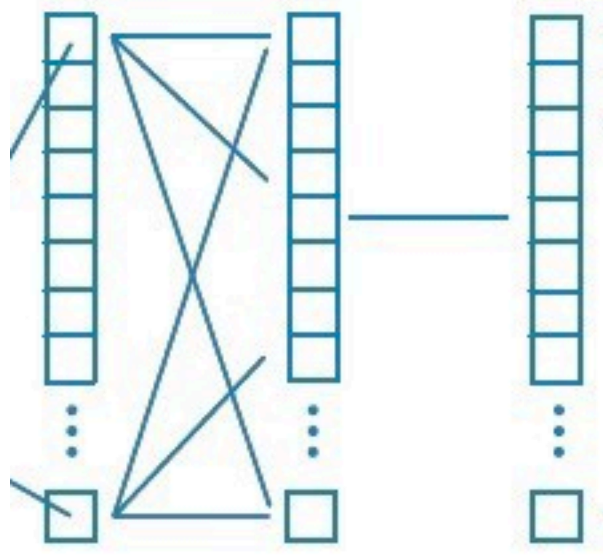
“My data is **sequences**”:
Recurrent neural network (RNN)



The standard deep learning toolkit…

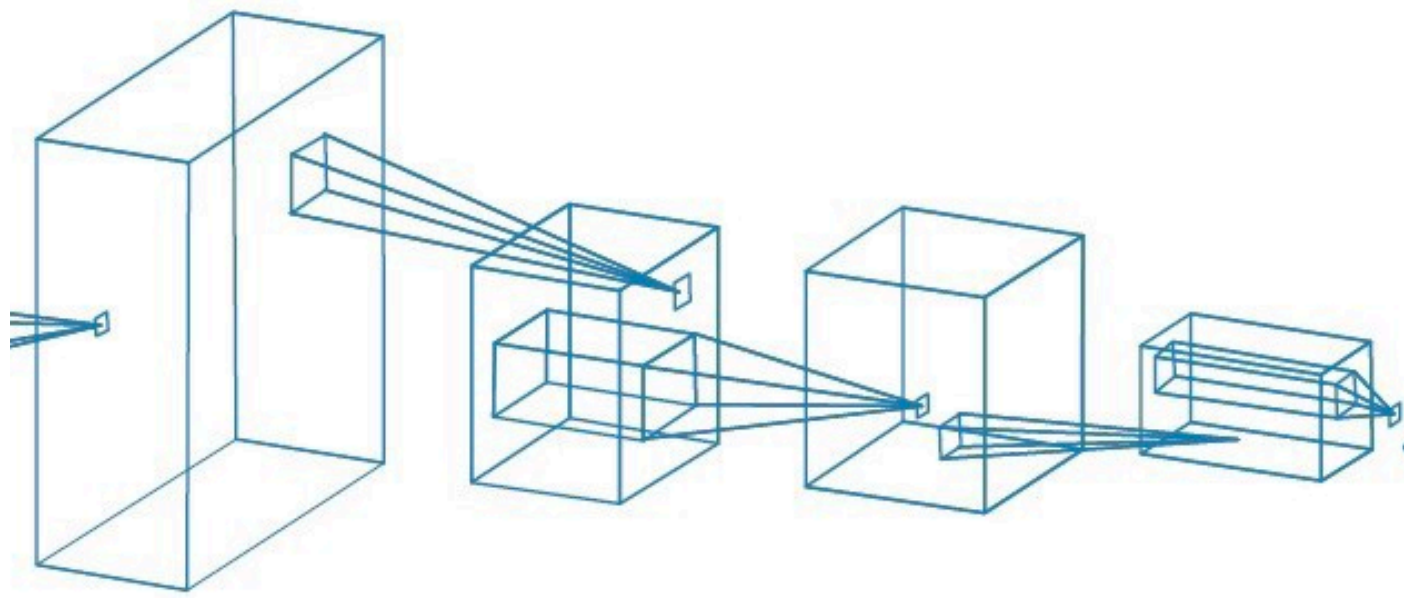
"My data is **vectors**":

Multi-layer perceptron (MLP)



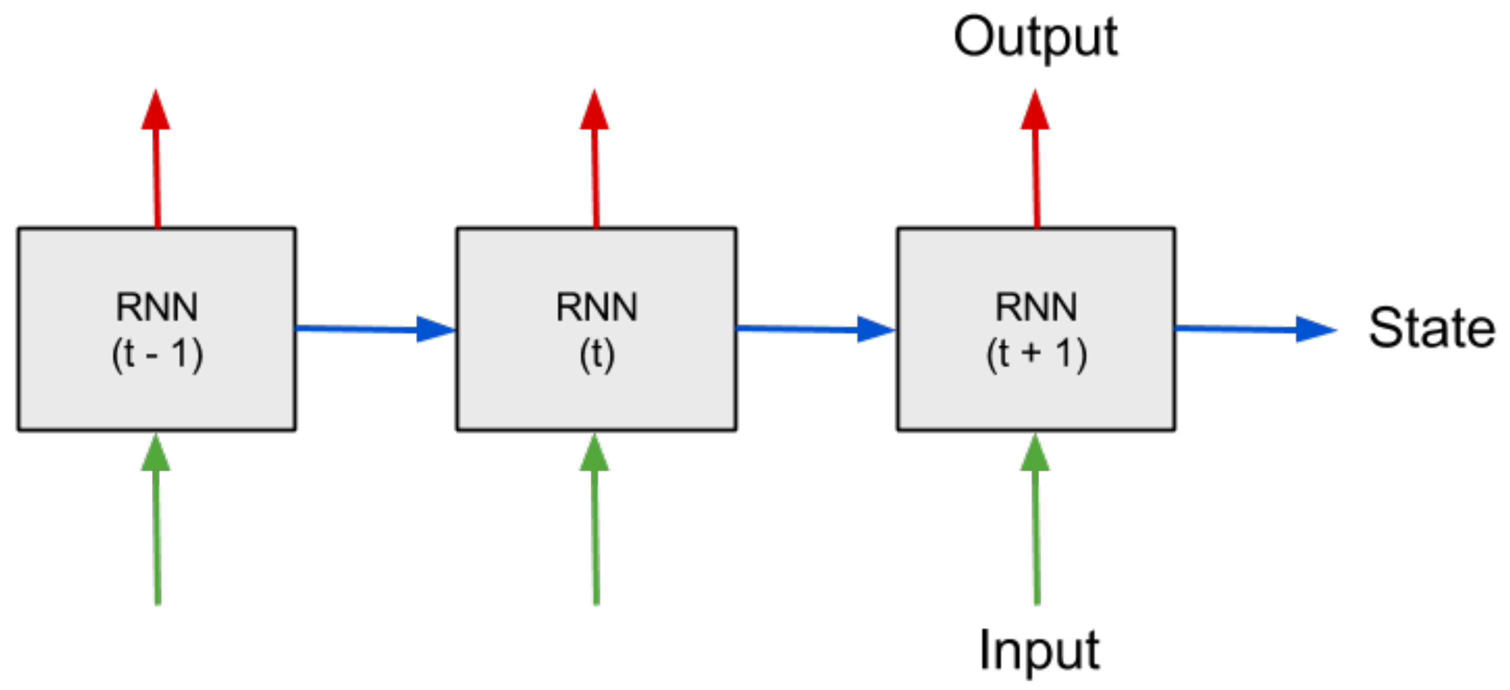
"My data is **grids**":

Convolutional neural network (CNN)



"My data is **sequences**":

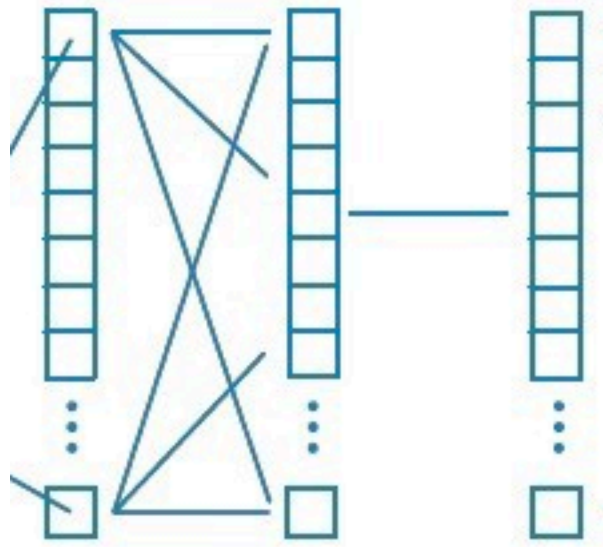
Recurrent neural network (RNN)



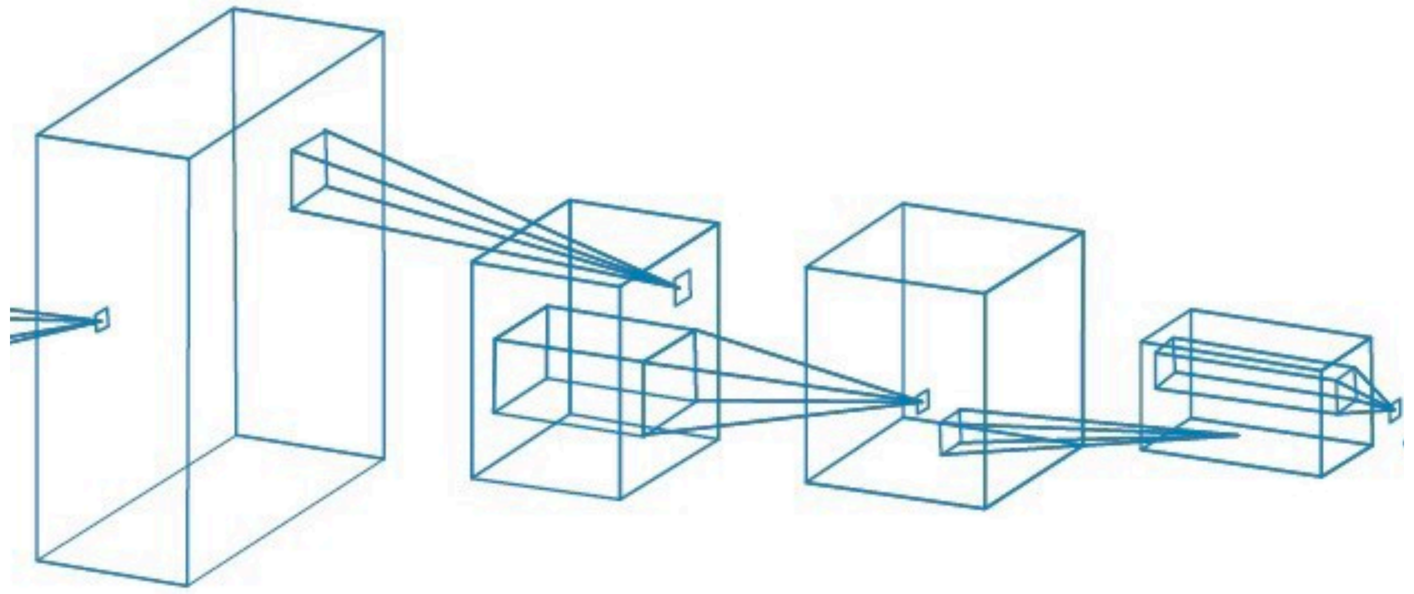
…is not well-suited to reasoning over structured representations.

The standard deep learning toolkit…

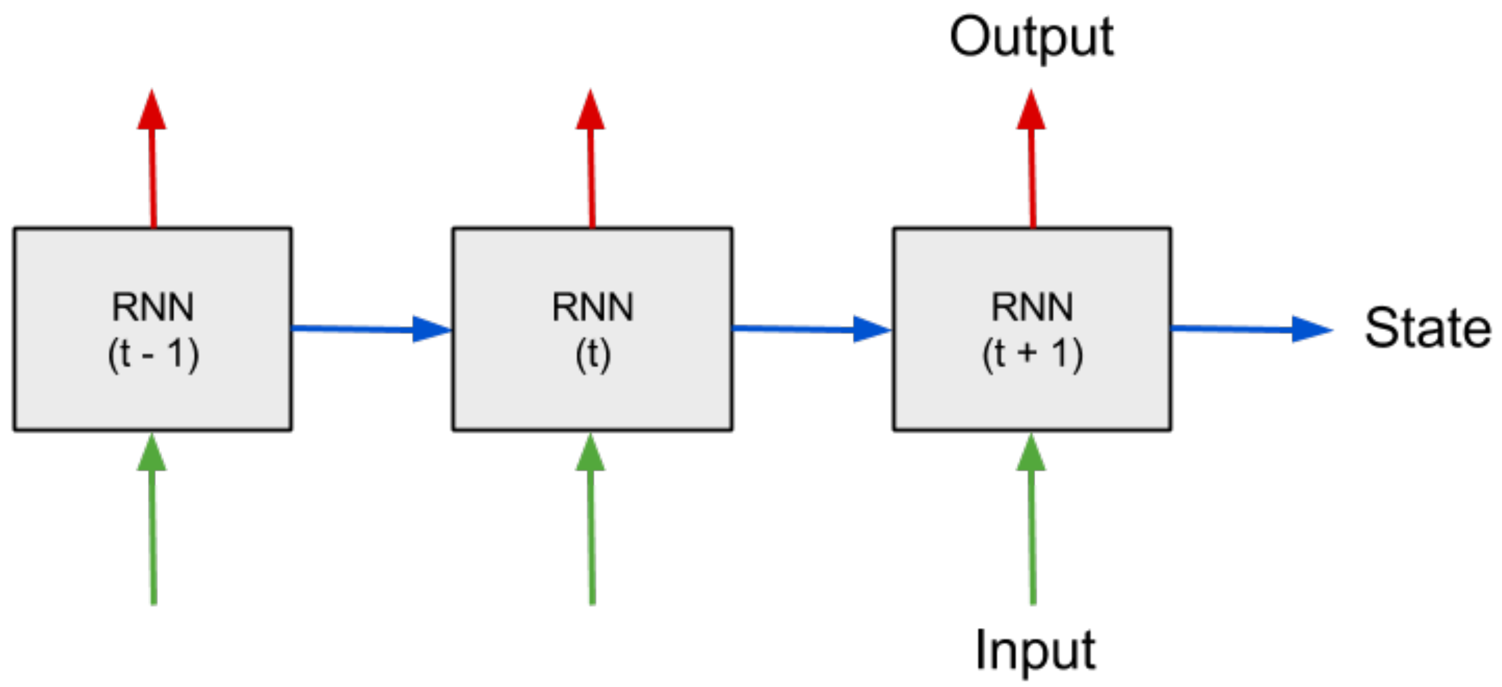
"My data is **vectors**":
Multi-layer perceptron (MLP)



"My data is **grids**":
Convolutional neural network (CNN)



"My data is **sequences**":
Recurrent neural network (RNN)



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

But neural networks that operate on **graphs** *are.*

Background: Graph Neural Networks

General idea

- Analogous to a convolutional network, but over arbitrary graphs (rather than just grids)
- Learn to reason about entities and their relations

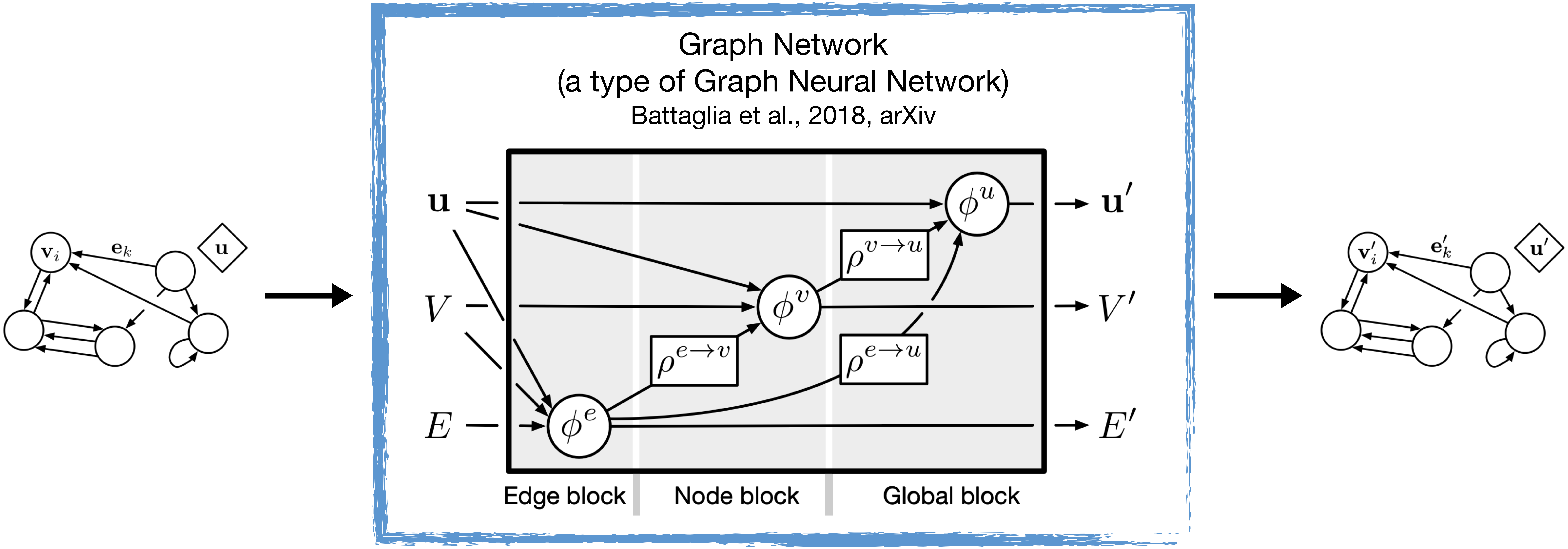
Key historical survey papers

- Scarselli et al. (2009) "The Graph Neural Network Model".
  Summarizes the initial papers on the topic from ~2005-2009. Very general formalism.
- Li et al. (2015) “Gated graph sequence neural networks”.
  Simplified the formalism, trained via backprop, used RNNs for sharing update steps across time.
- Bronstein et al. (2016) “Geometric deep learning: going beyond Euclidean data”.
  Survey of spectral and spatial approaches for deep learning on graphs.
- Gilmer et al. (2017) “Neural Message Passing for Quantum Chemistry”.
  Introduced “message-passing neural network” (MPNNs) formalism, unifying various approaches such as graph convolutional networks.
- Battaglia et al. (2018). “Relational inductive biases, deep learning, and graph networks”.
  Introduced the “graph network” (GN) formalism, extends MPNNs, unifies non-local neural networks/self-attention/Transformer.

Graph Networks (GNs)

Why do we need another graph neural network variant?

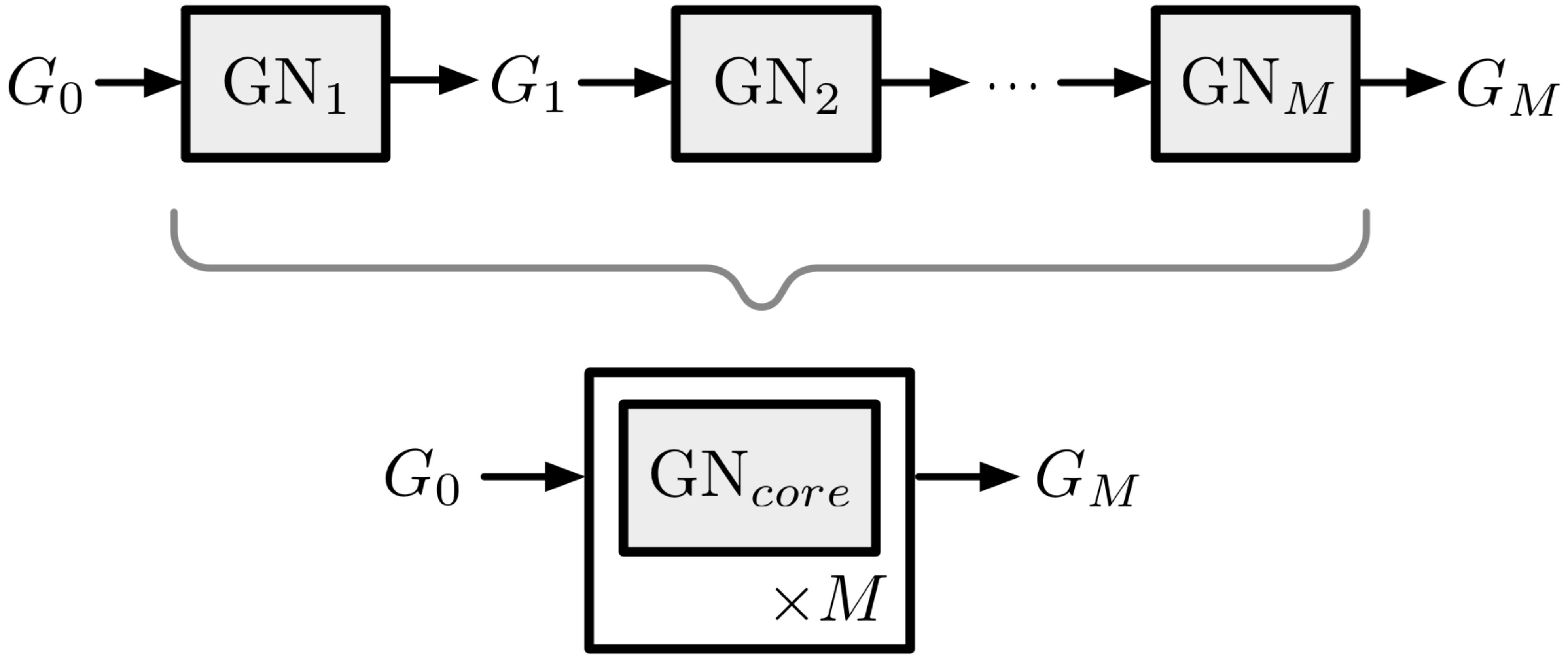
- We designed GNs to be both expressive, and easy to implement
- A GN block is a "graph-to-graph" function approximator
  - The output graph's structure (number of nodes and edge connectivity) matches the input graph's
  - The output graph-, node-, and edge-level attributes will be functions of the input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



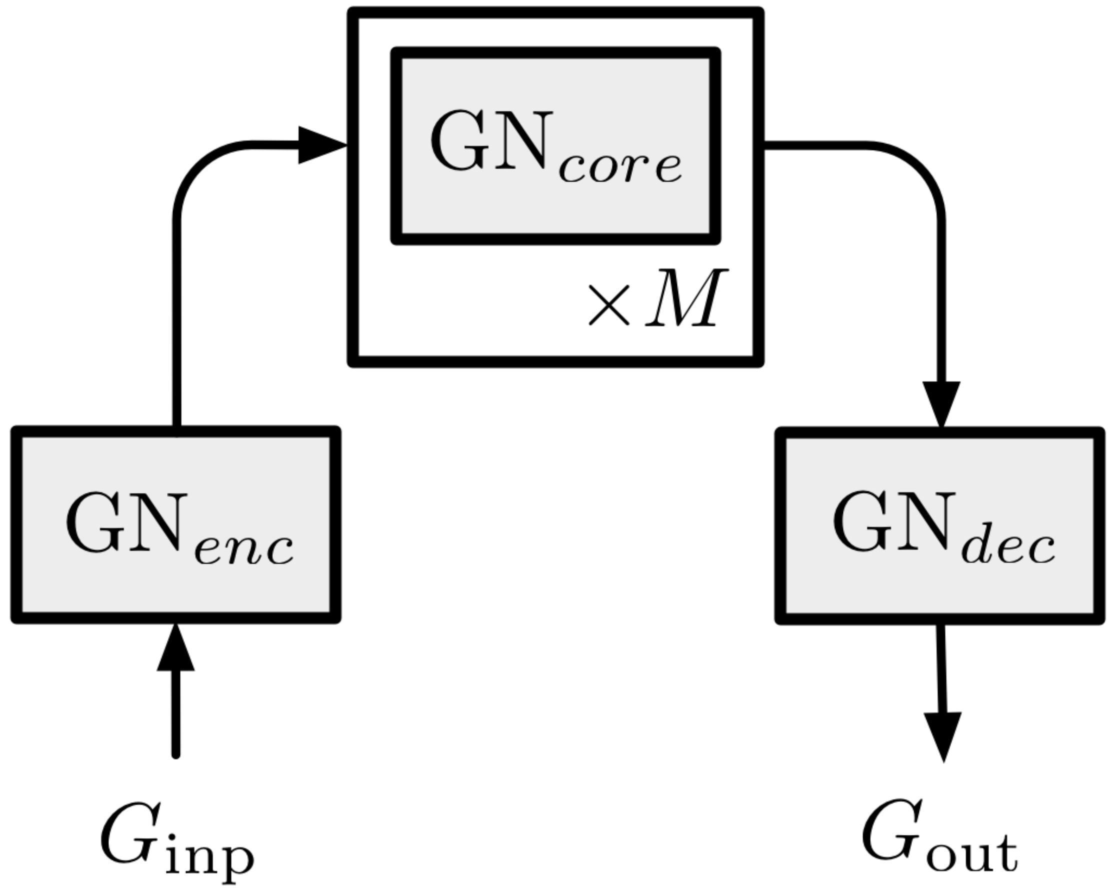
Composing GN blocks

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

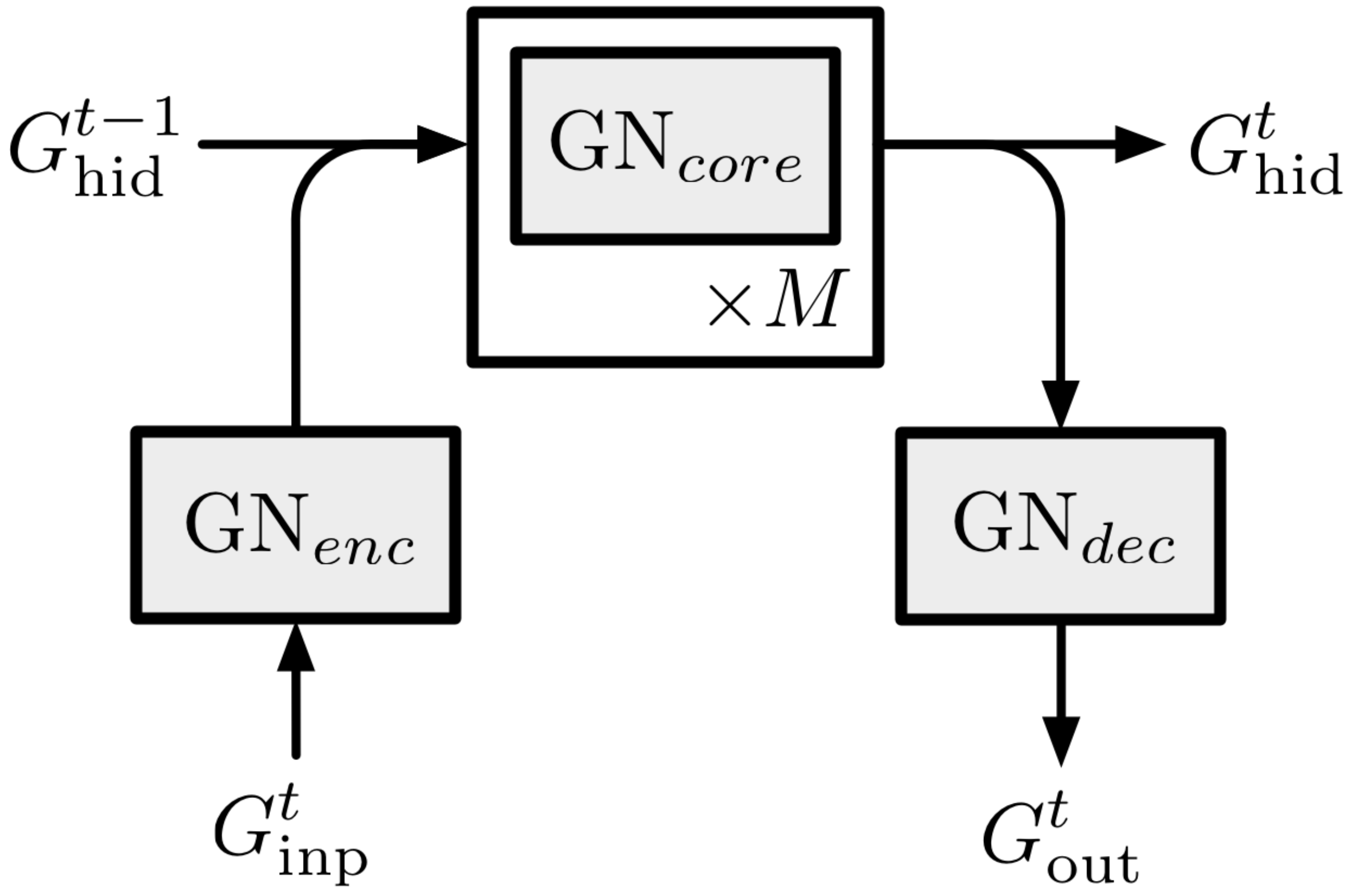
Shared GN core



Encode-process-decode



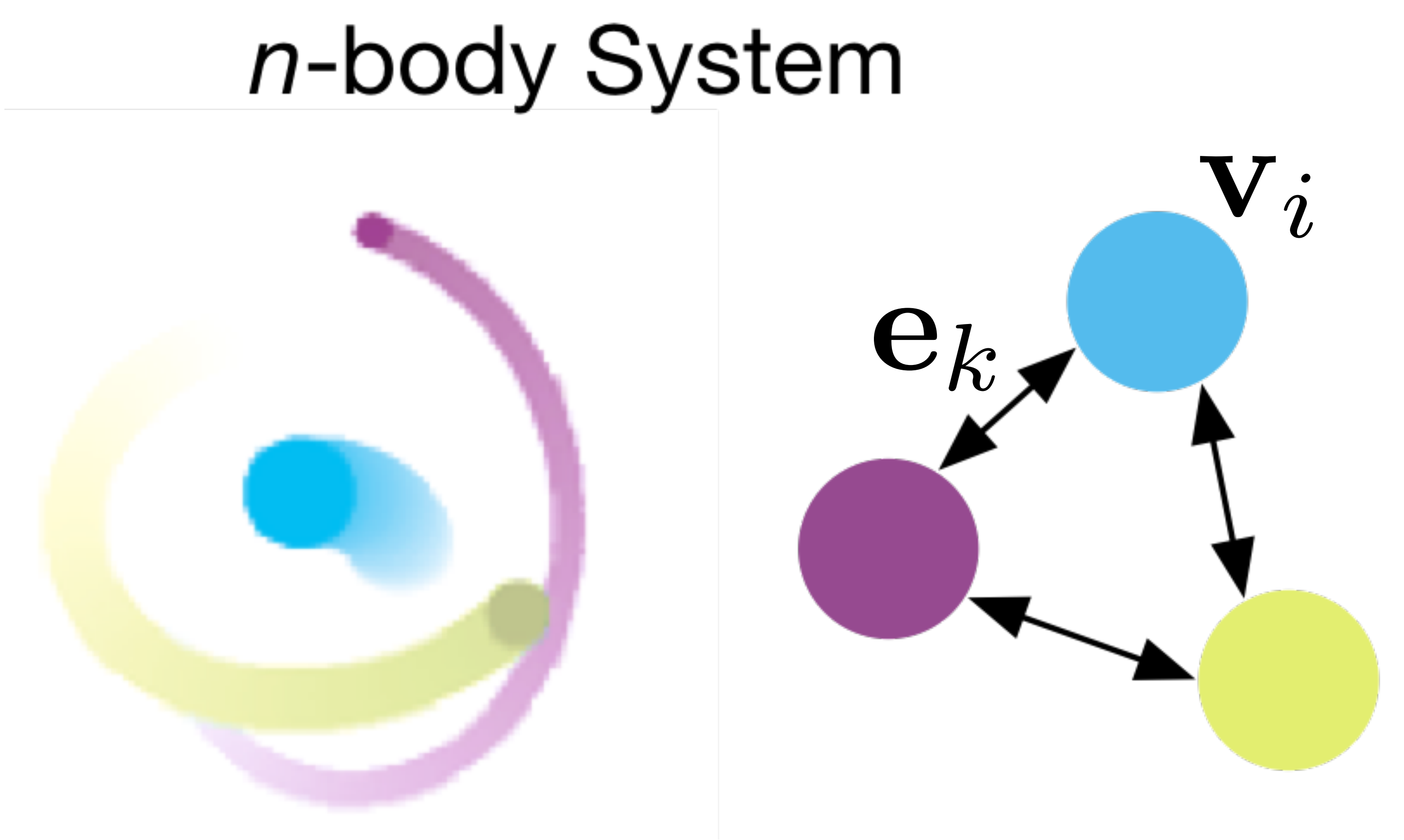
Recurrent GN architecture



Battaglia et al., 2018, arXiv

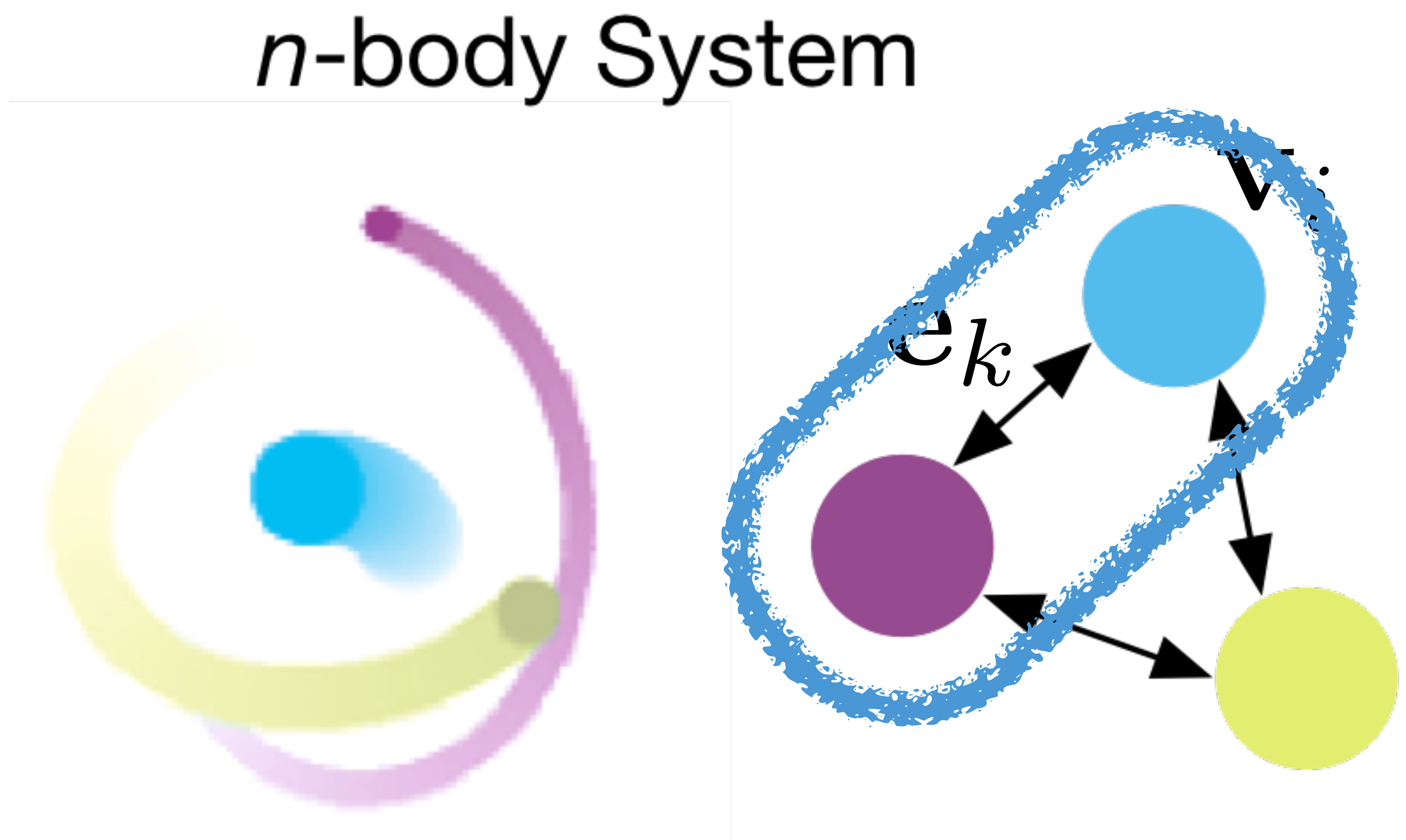
Battaglia et al., 2018, arXiv

Interaction Network: Learning simulation as message-passing



Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



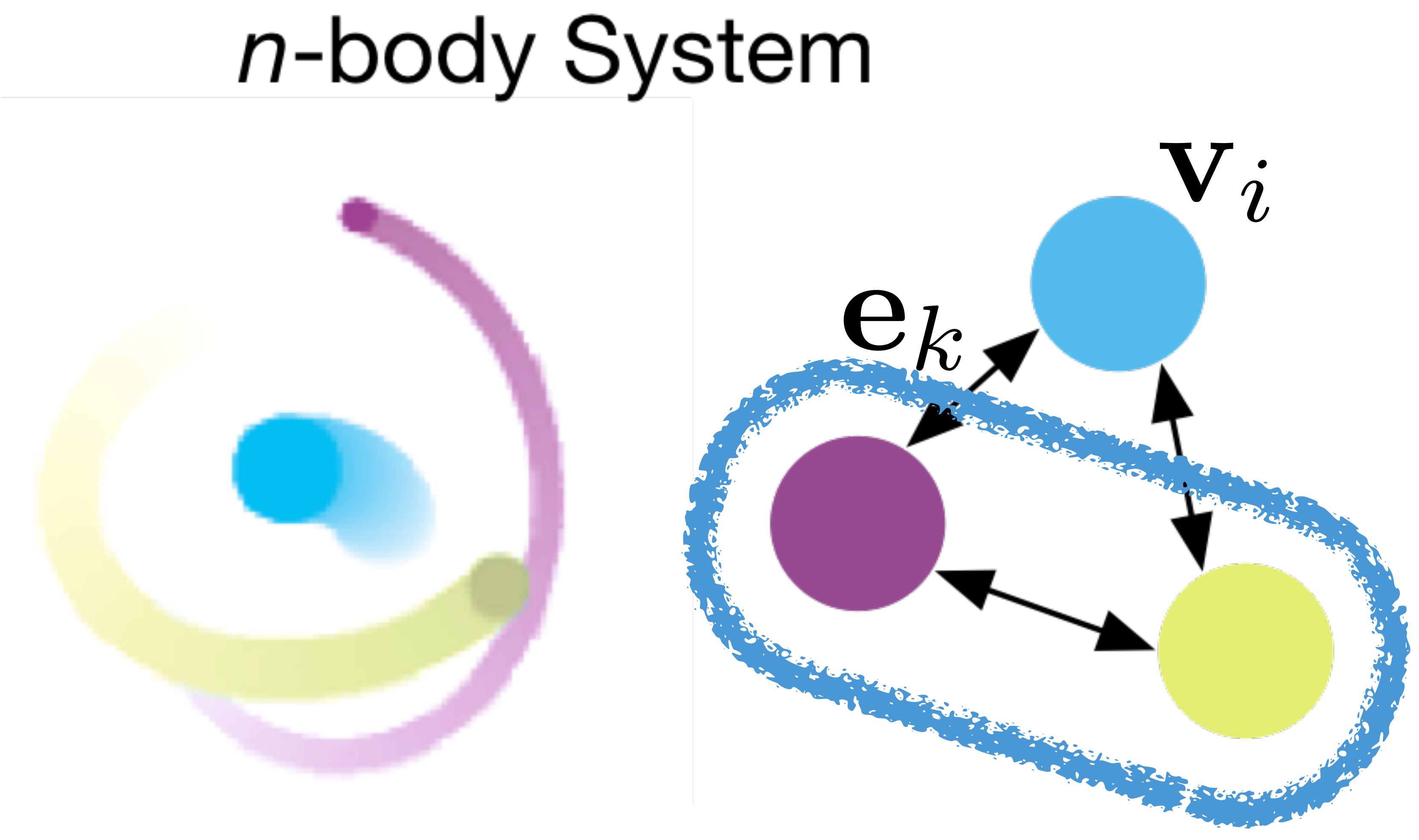
Edge function

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

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

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



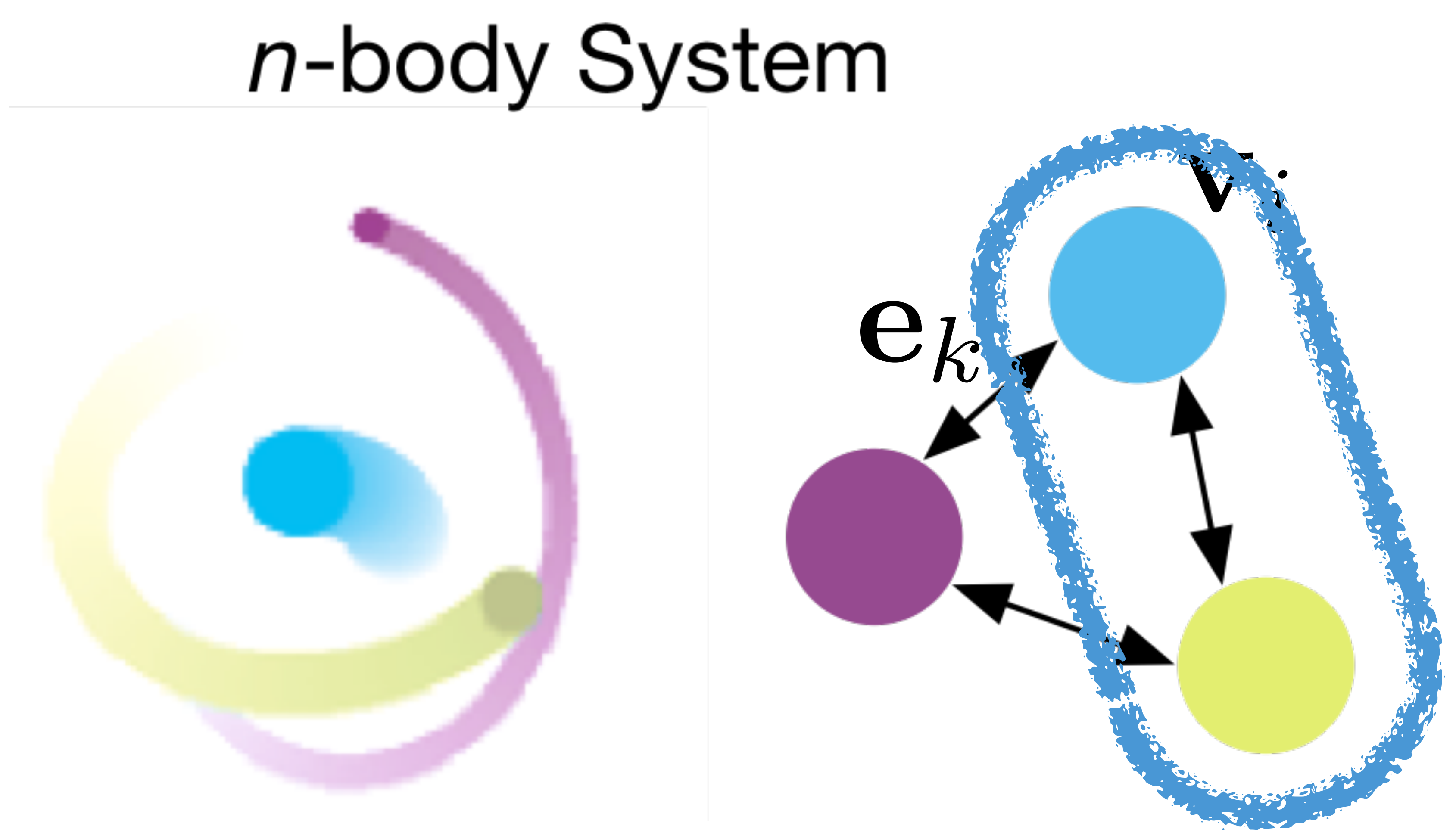
Edge function

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

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

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



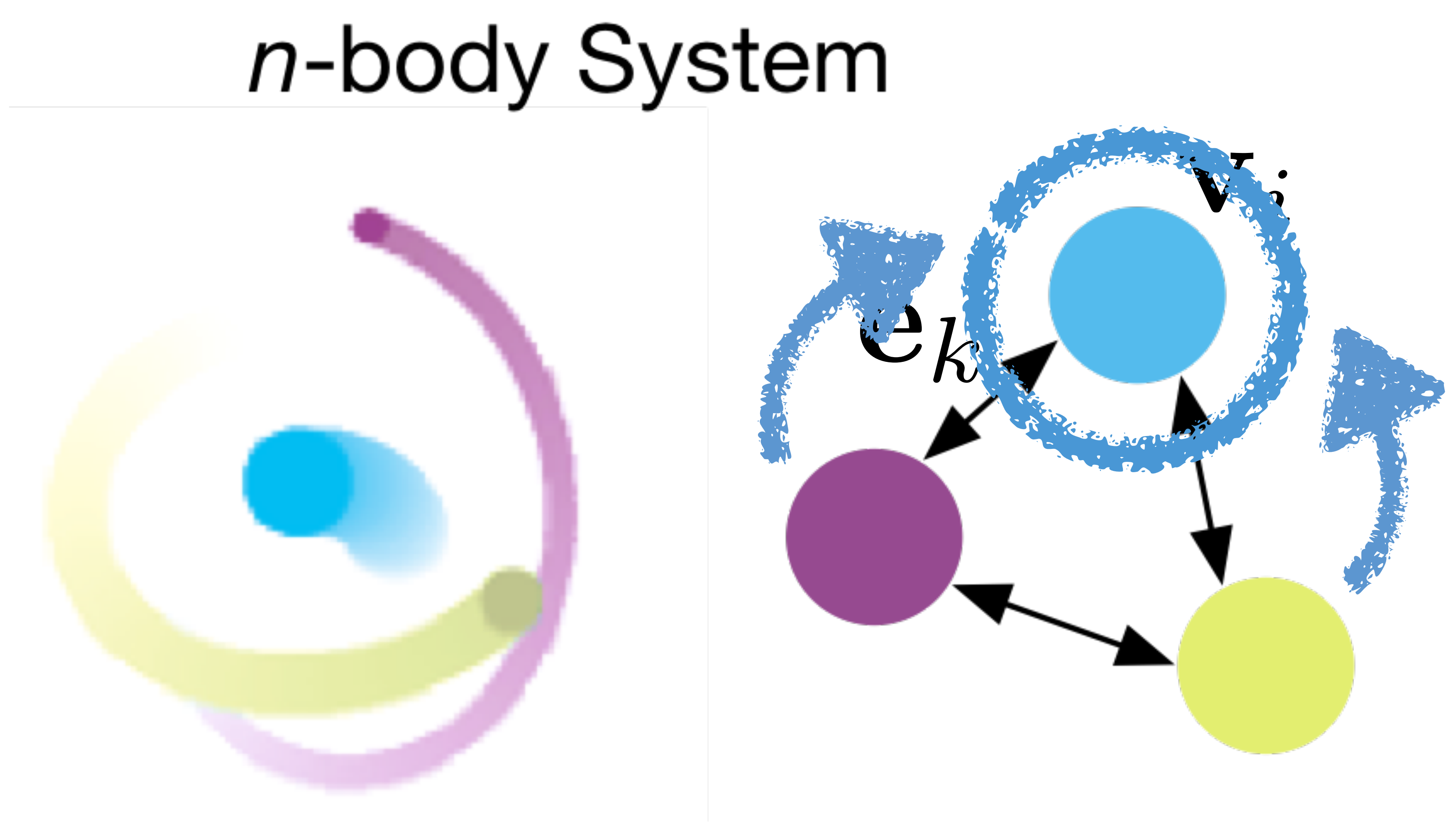
Edge function

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

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

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



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

Message aggregation

$$\bar{\mathbf{e}}'_i \leftarrow \sum_{r_k=i} \mathbf{e}'_k$$

Node function

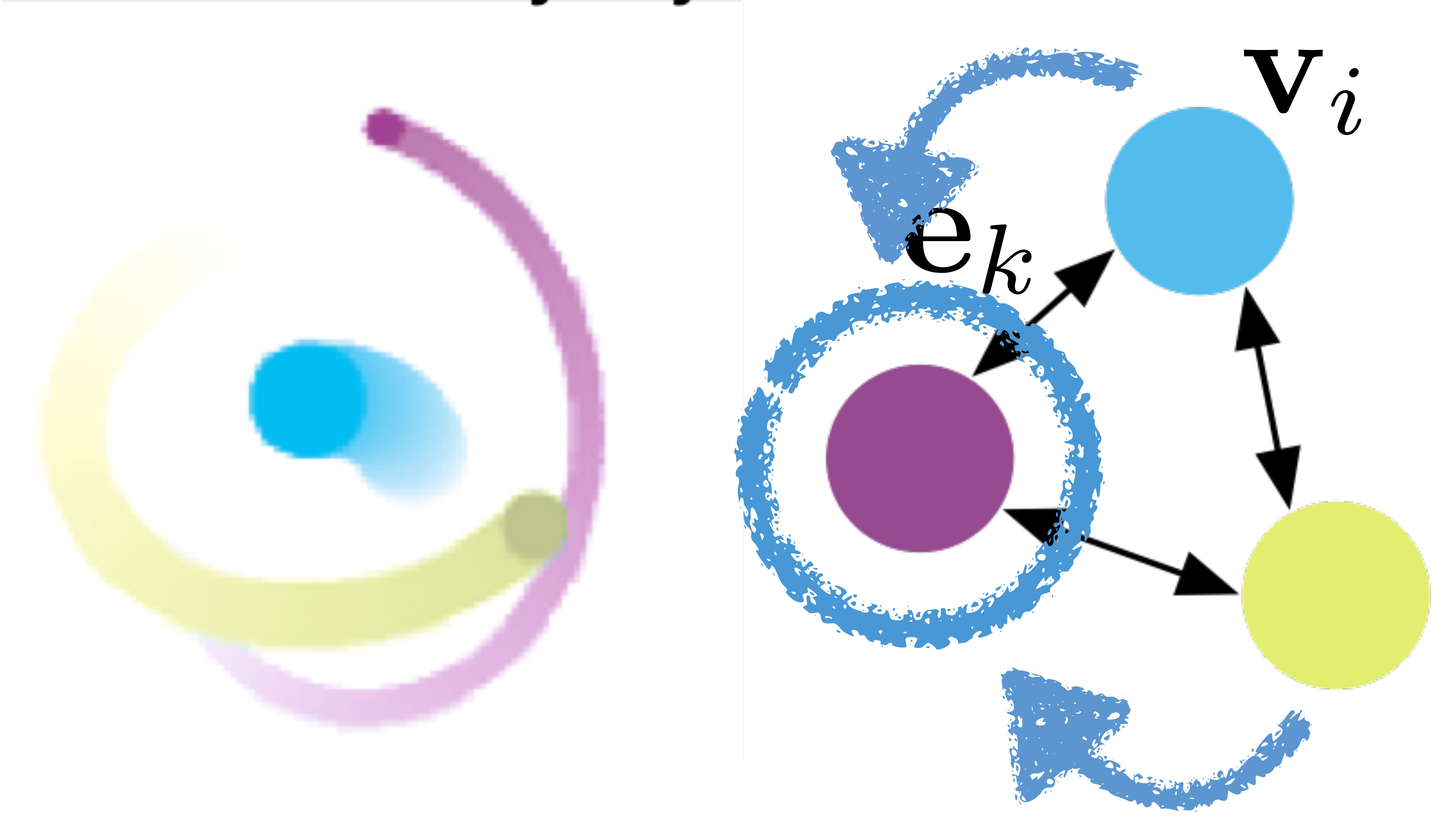
$$\mathbf{v}'_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$$

- 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

Message aggregation

$$\bar{\mathbf{e}}'_i \leftarrow \sum_{r_k=i} \mathbf{e}'_k$$

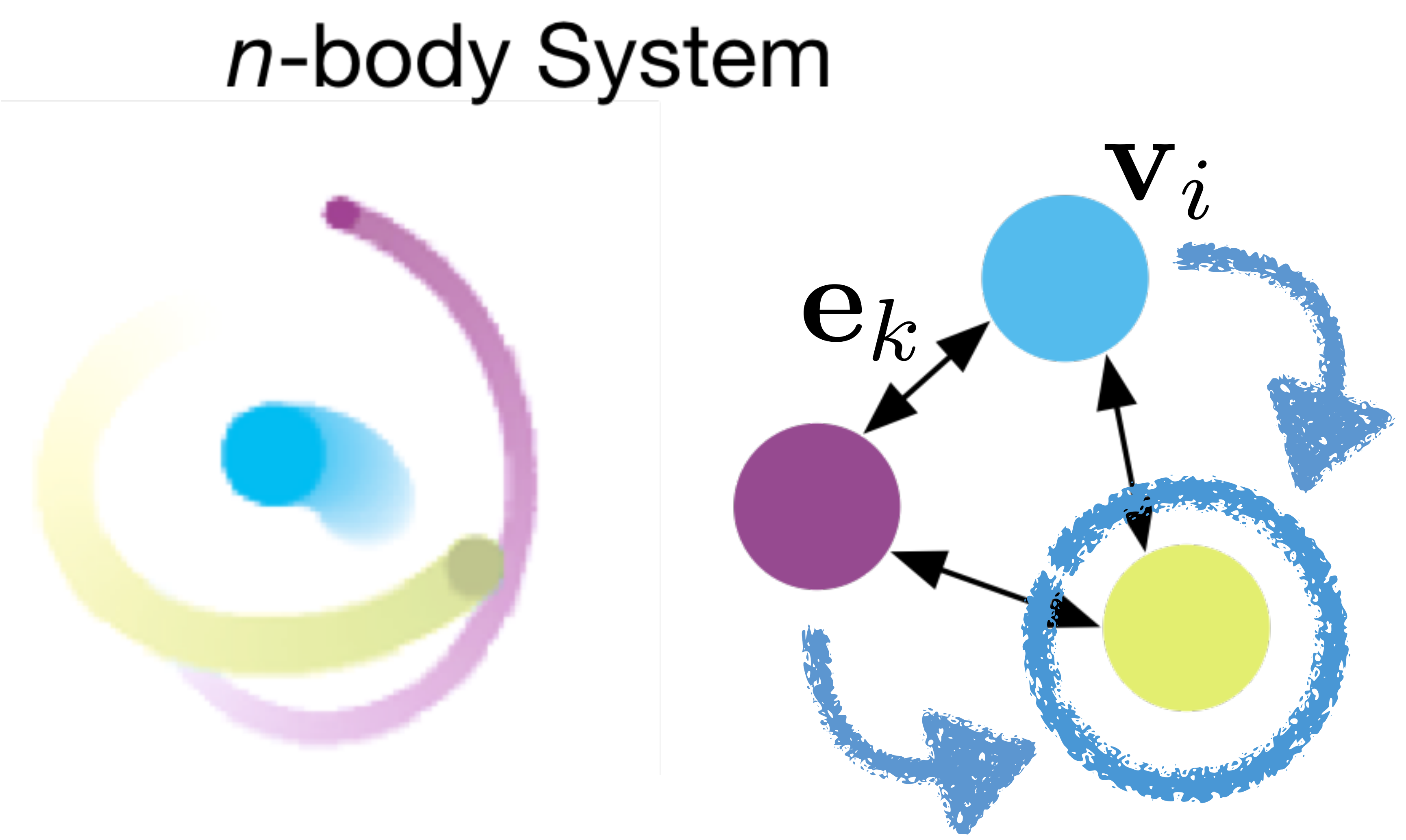
Node function

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

- Update node info from previous node state and aggregated "messages"

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



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

Message aggregation

$$\bar{\mathbf{e}}'_i \leftarrow \sum_{r_k=i} \mathbf{e}'_k$$

Node function

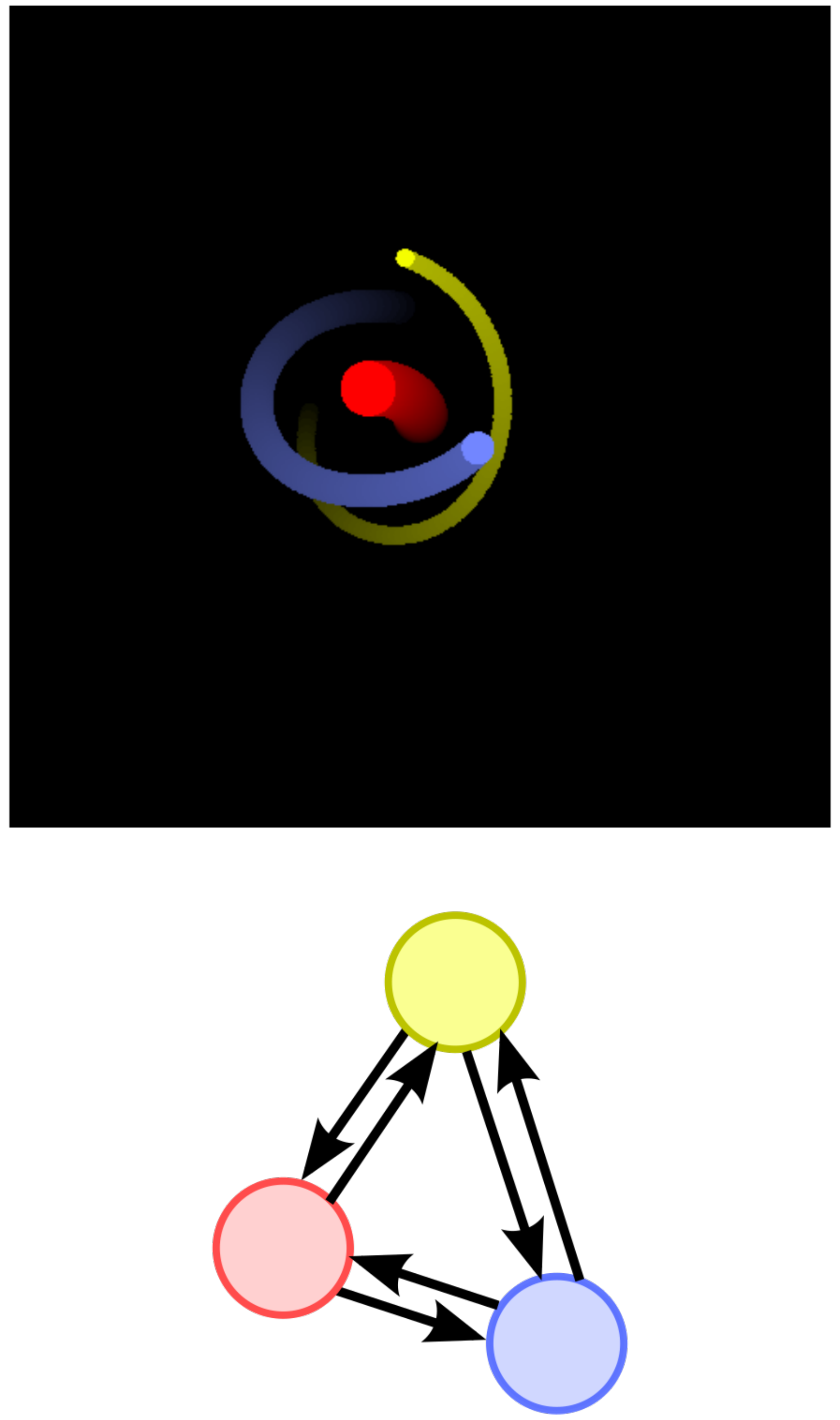
$$\mathbf{v}'_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$$

- Update node info from previous node state and aggregated "messages"

Battaglia et al., 2016, NeurIPS

Physical systems as graphs

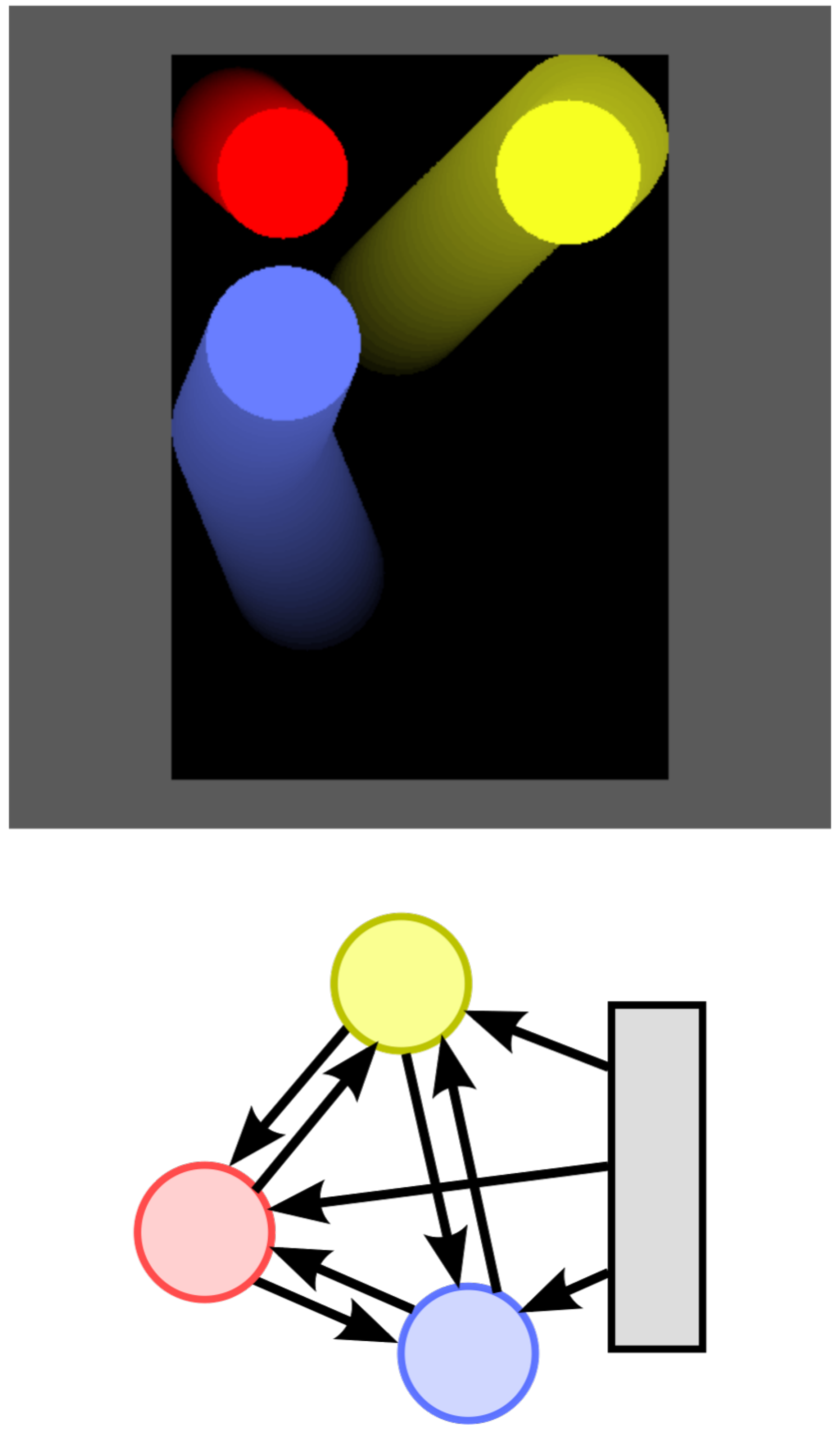
n-body



Nodes: bodies

Edges: gravitational forces

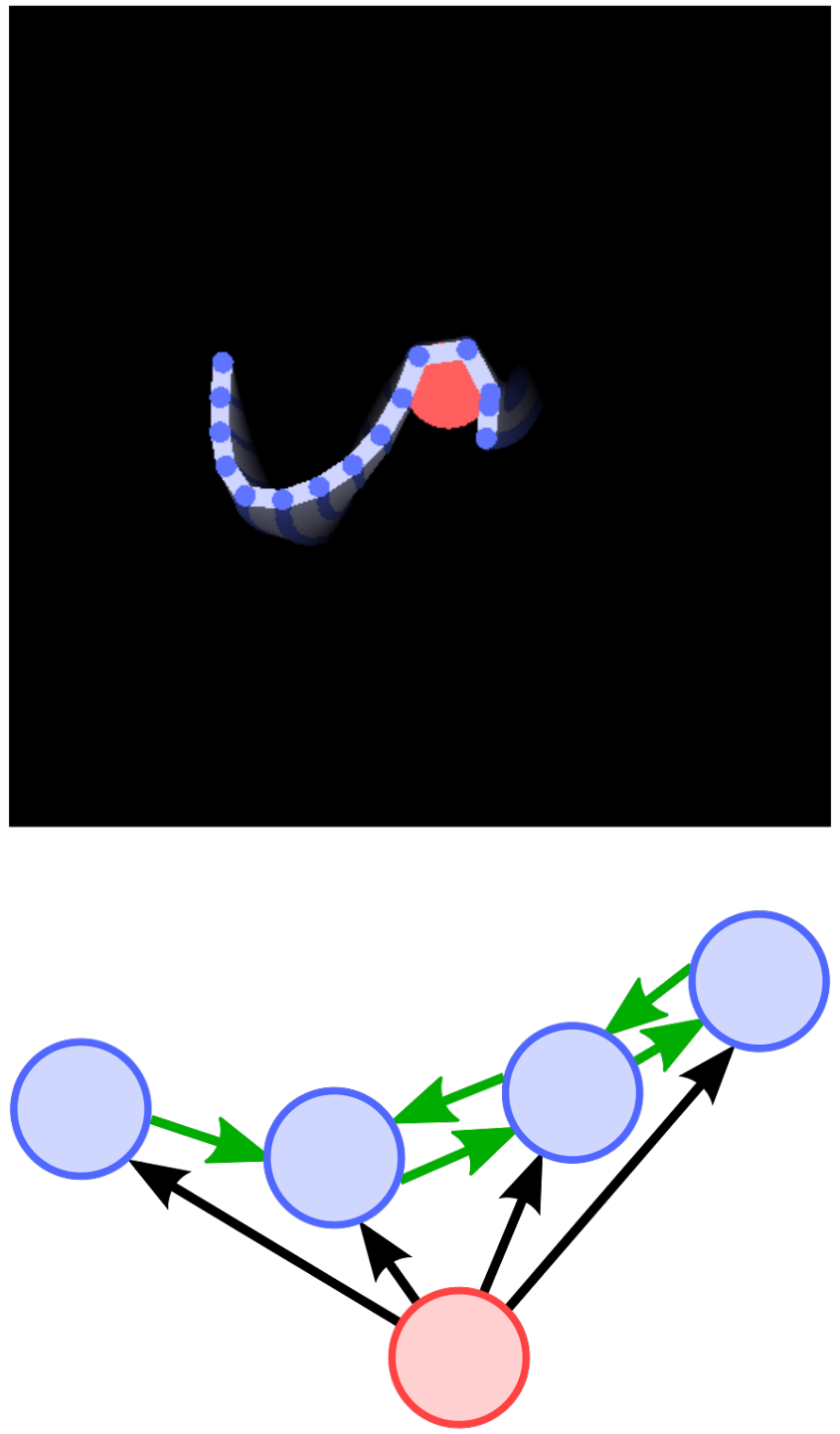
Balls



Nodes: balls

Edges: rigid collisions between balls, and walls

String

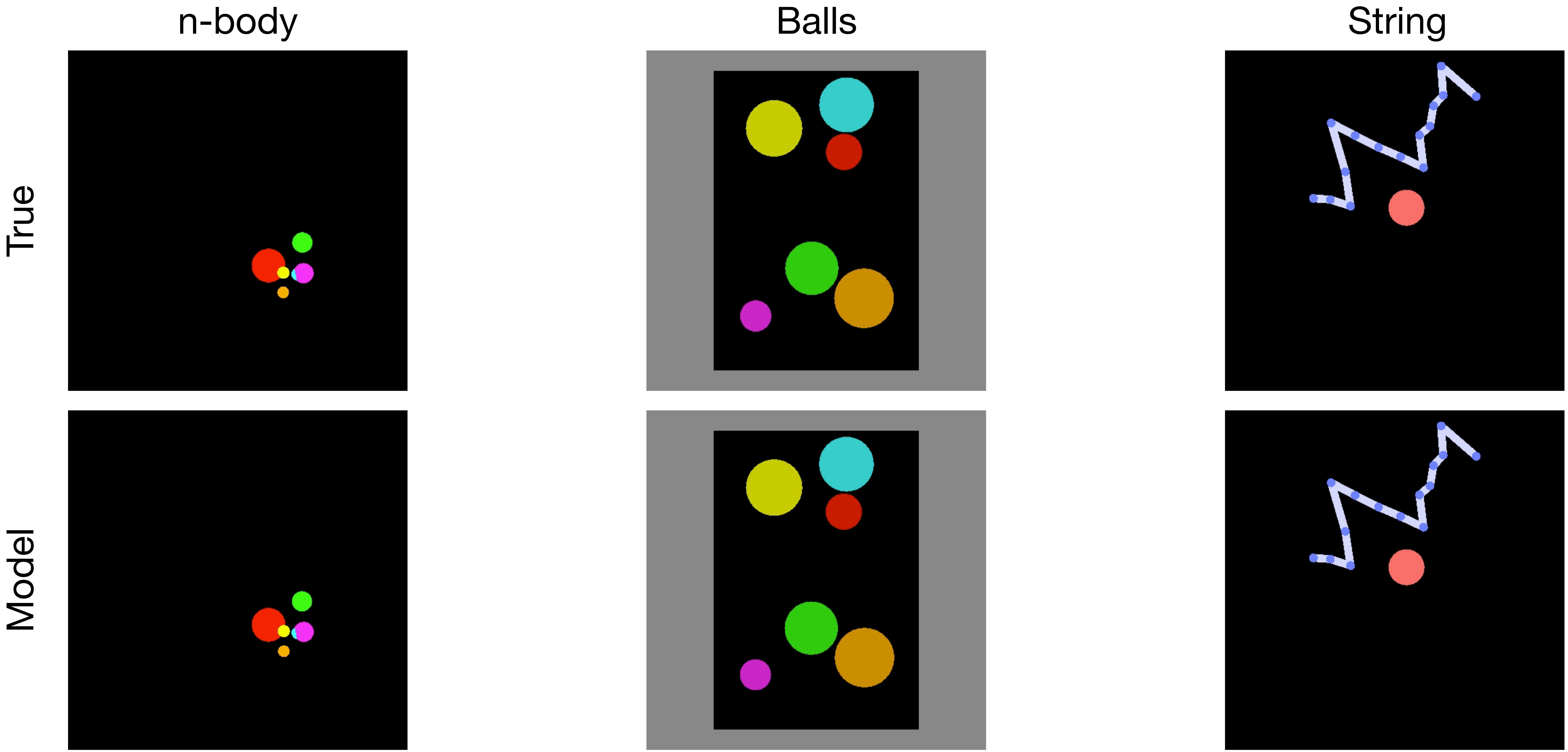


Nodes: masses

Edges: springs and rigid collisions

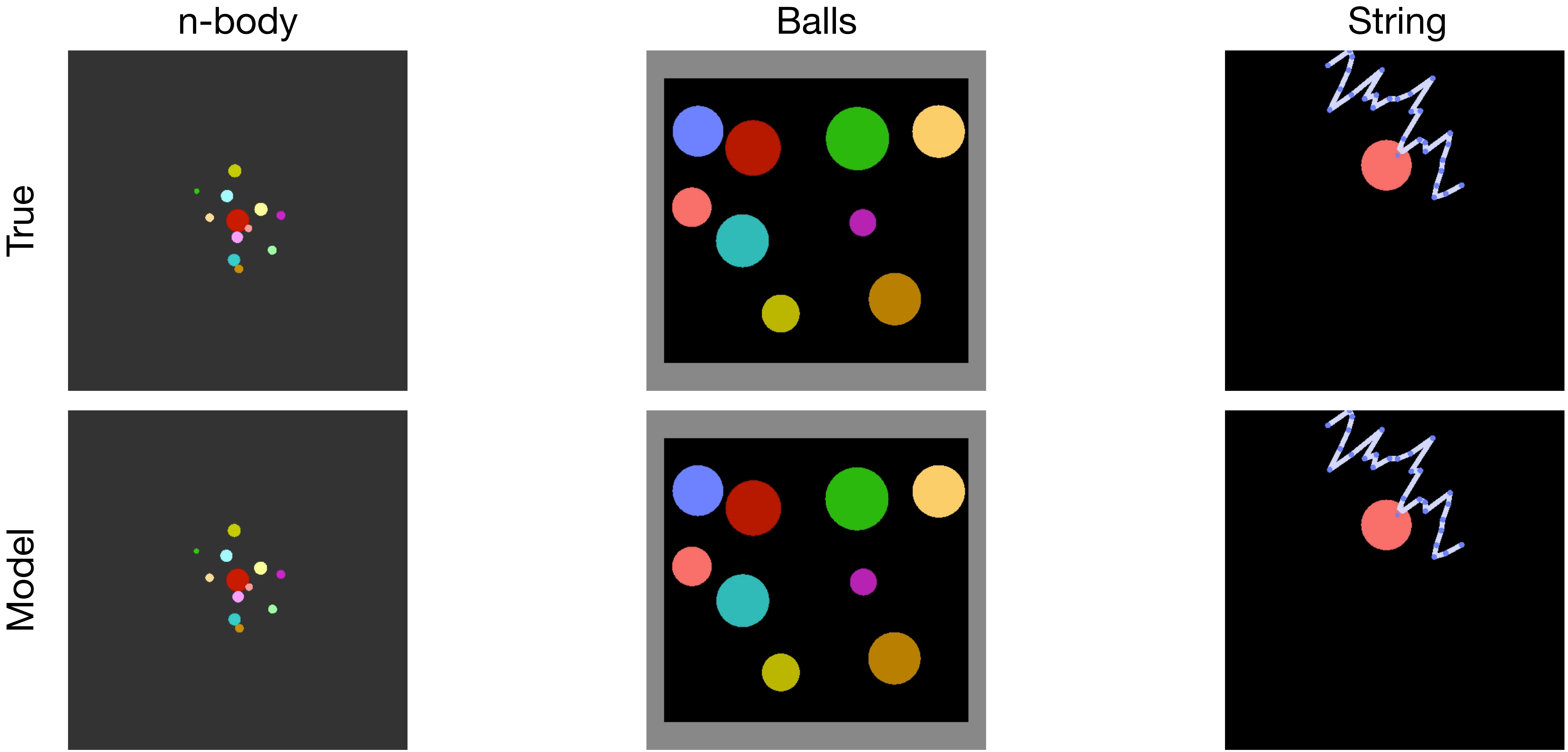
Battaglia et al., 2016, NeurIPS

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



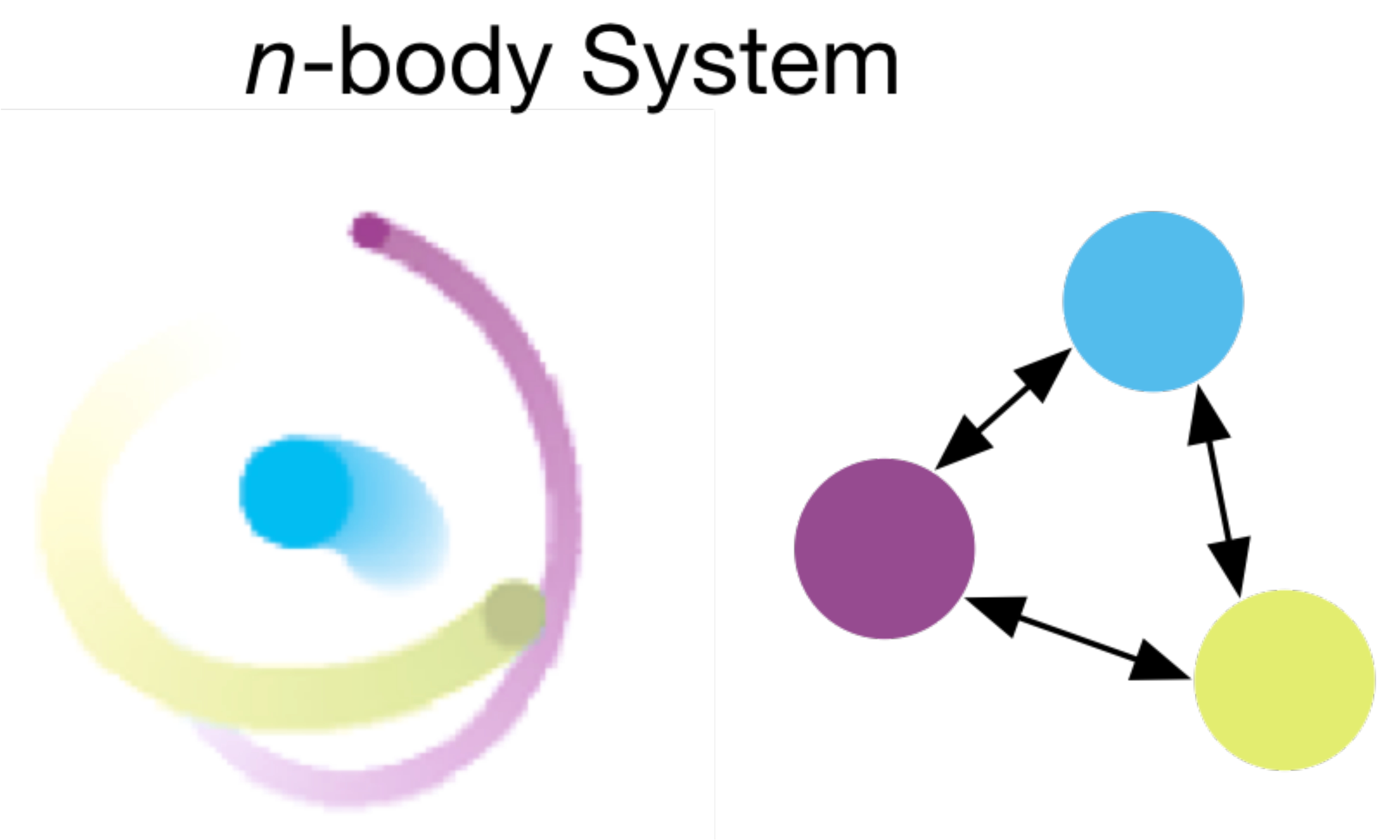
Battaglia et al., 2016, NeurIPS

Zero-shot generalisation to larger systems



Battaglia et al., 2016, NeurIPS

Interaction Network: Predicting potential energy



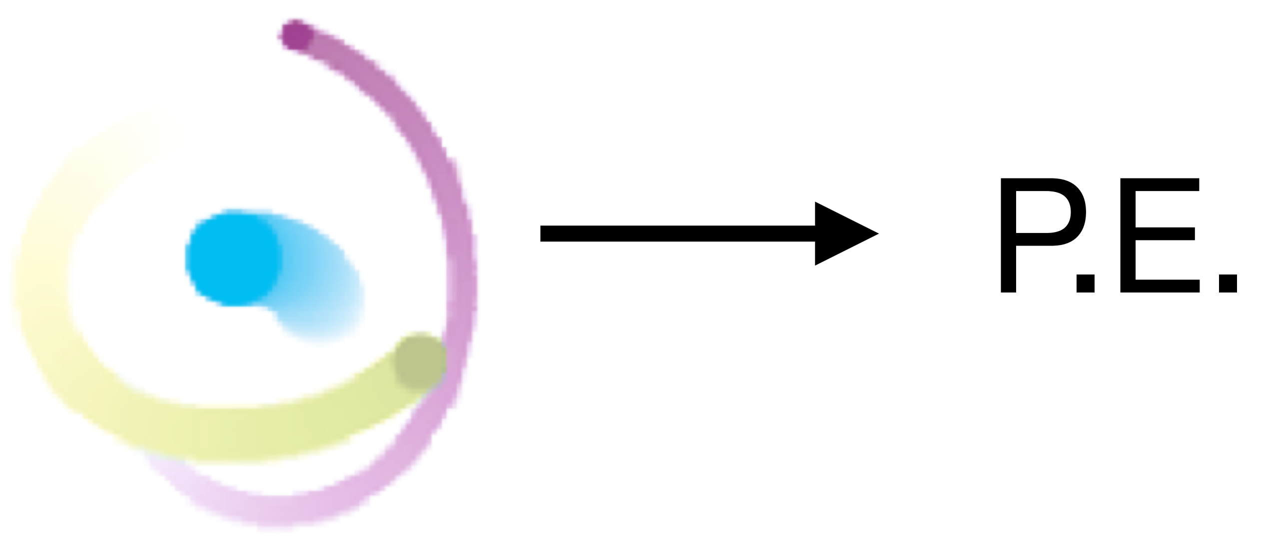
Node aggregation and global function

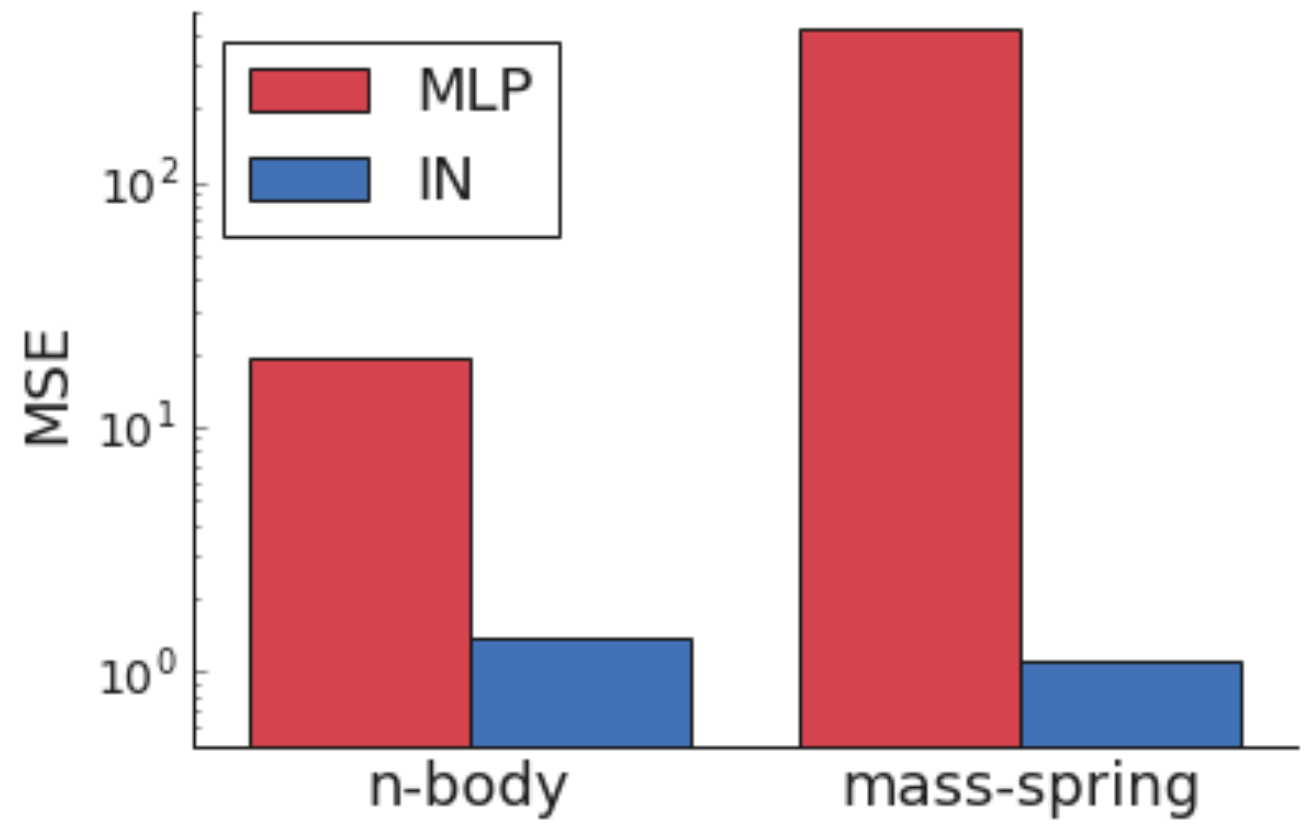
$$\bar{\mathbf{v}}' \leftarrow \sum_i \mathbf{v}'_i$$

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

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

Trained to predict system's potential energy

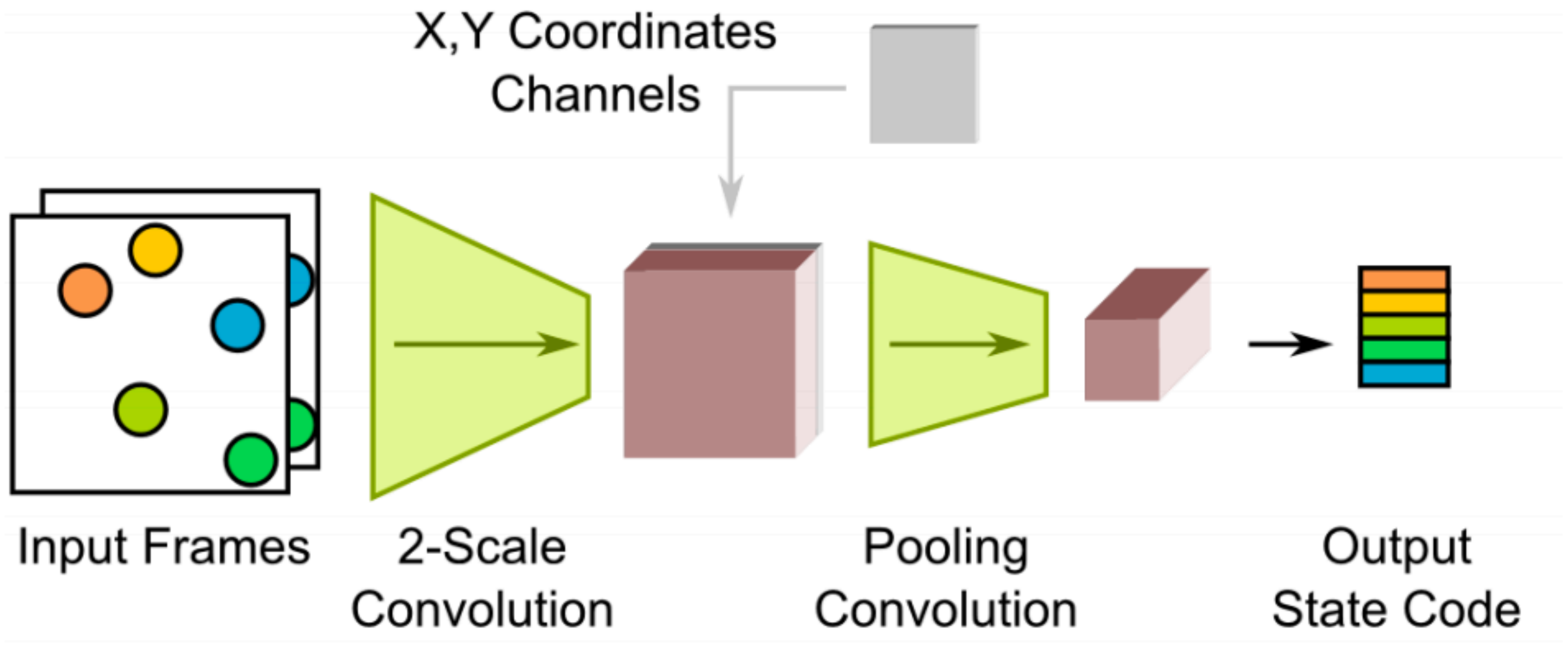




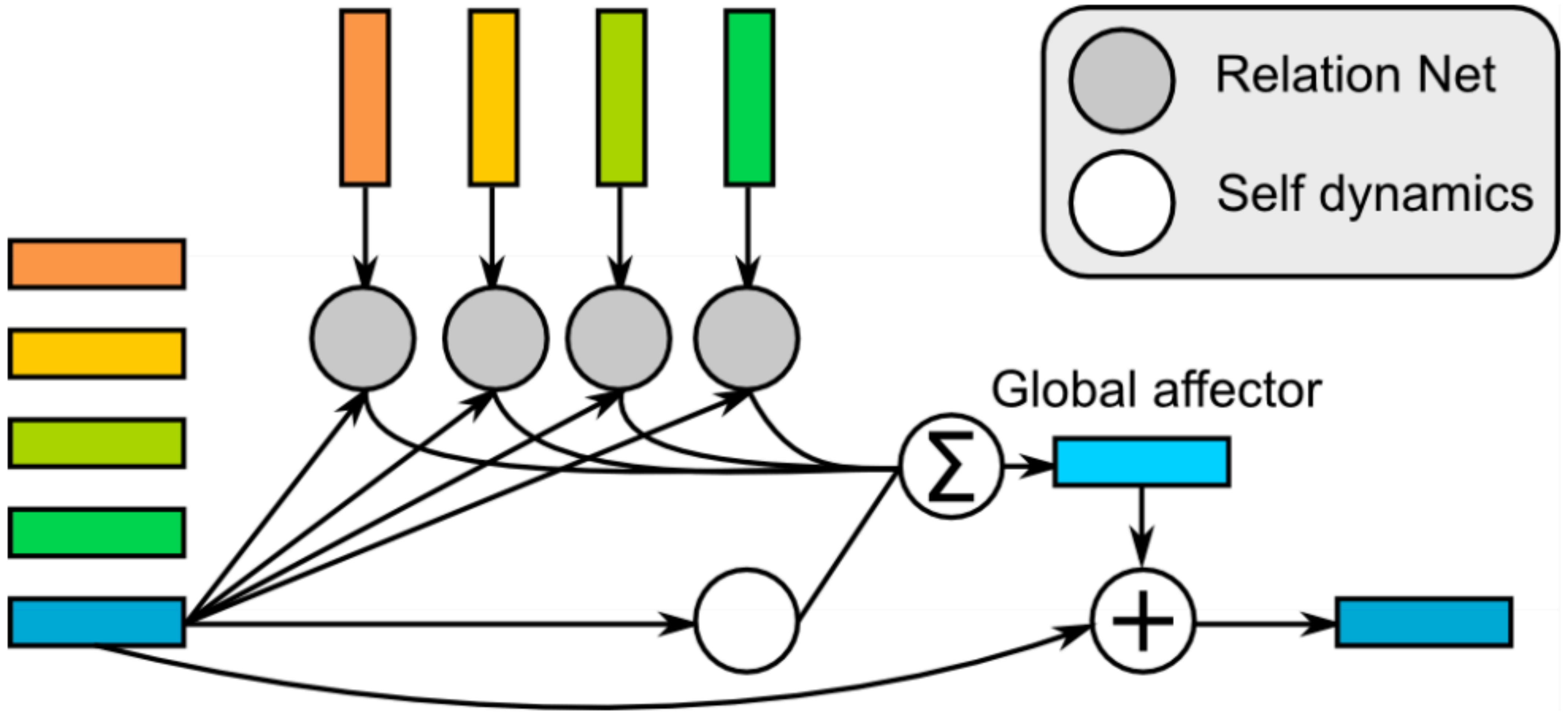
Battaglia et al., 2016, NeurIPS

Visual interaction network: Simulate from input images

Multi-frame encoder (conv net-based)



Interaction network



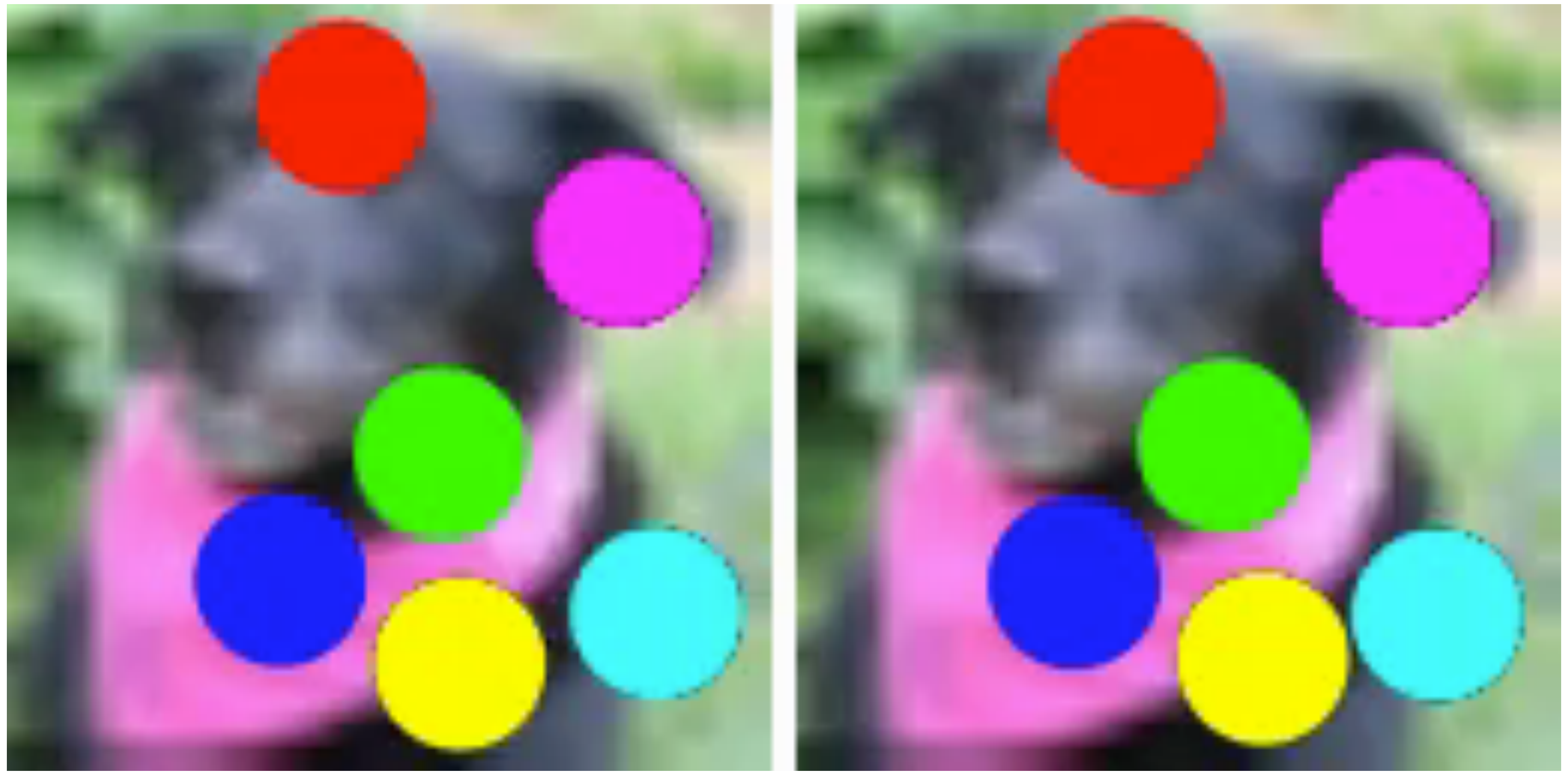
Watters et al., 2017, NeurIPS

Visual interaction network: Simulate from input images

Mass-springs

True Model

Bouncing balls

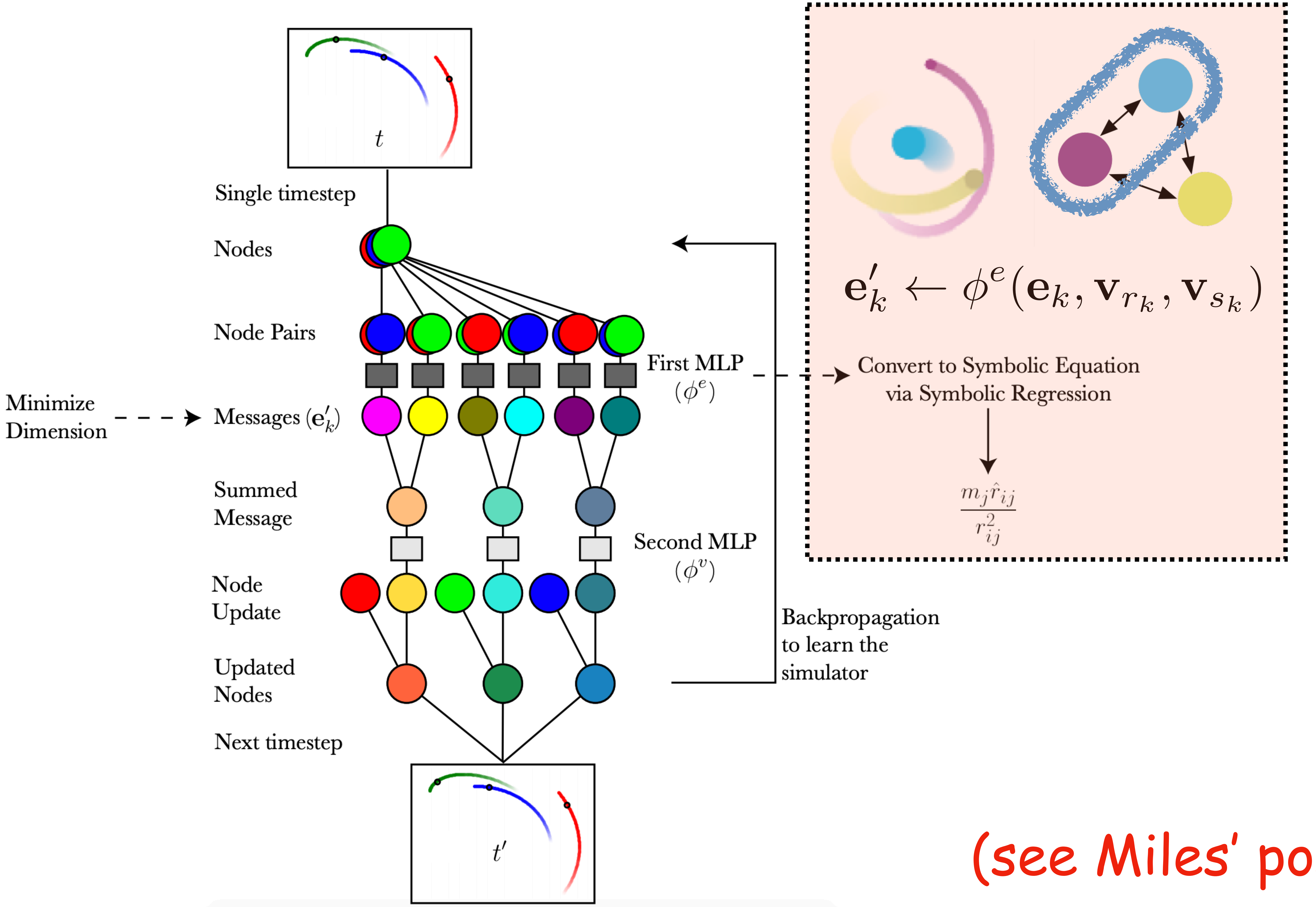


True Model

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

Watters et al., 2017, NeurIPS

Learning symbolic physics with graph networks



(see Miles' poster in the atrium)

Cranmer et al., 2019, arXiv/NeurIPS 2019 workshop

Learning symbolic physics with graph networks

<u>Experiments</u>

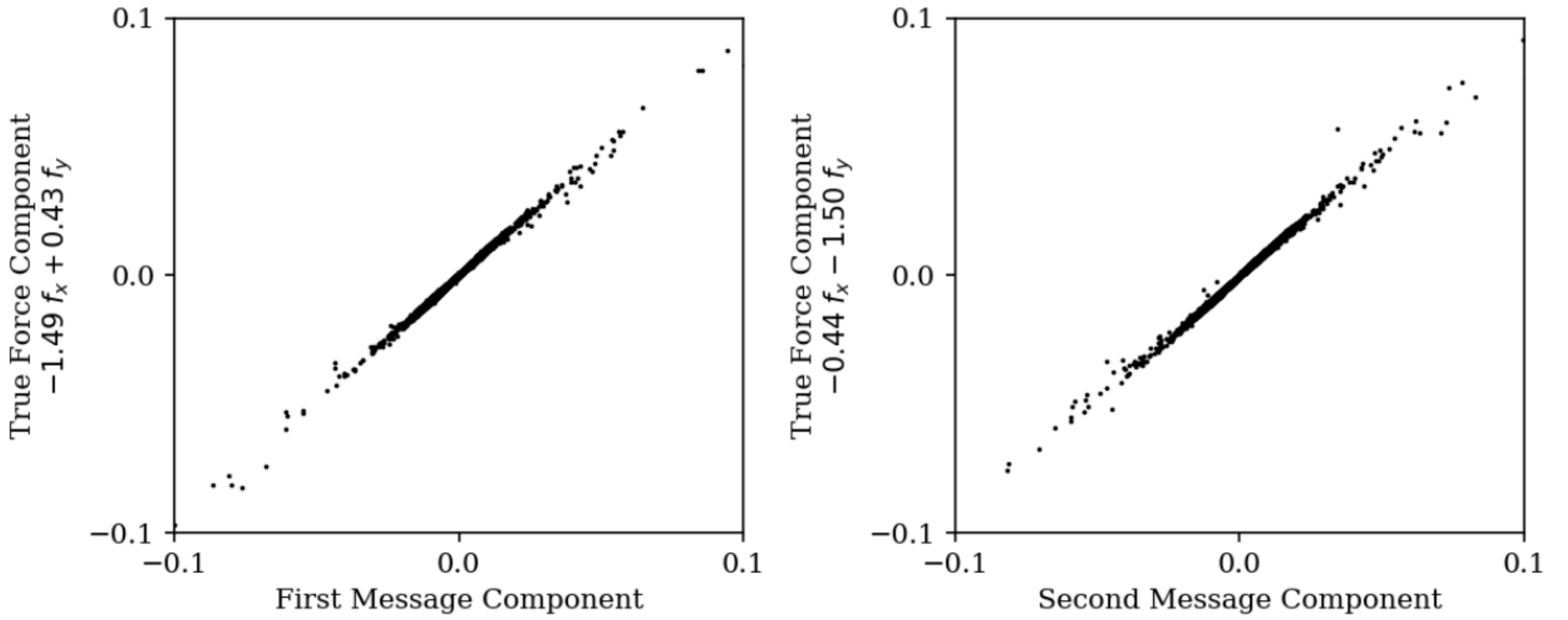
- 2D and 3D n-body (1/r and $1/r^2$ force laws)
- Mass-spring system

<u>Architecture</u>

Interaction network with message vectors constrained to 2 or 3 dimensions

<u>Results</u>

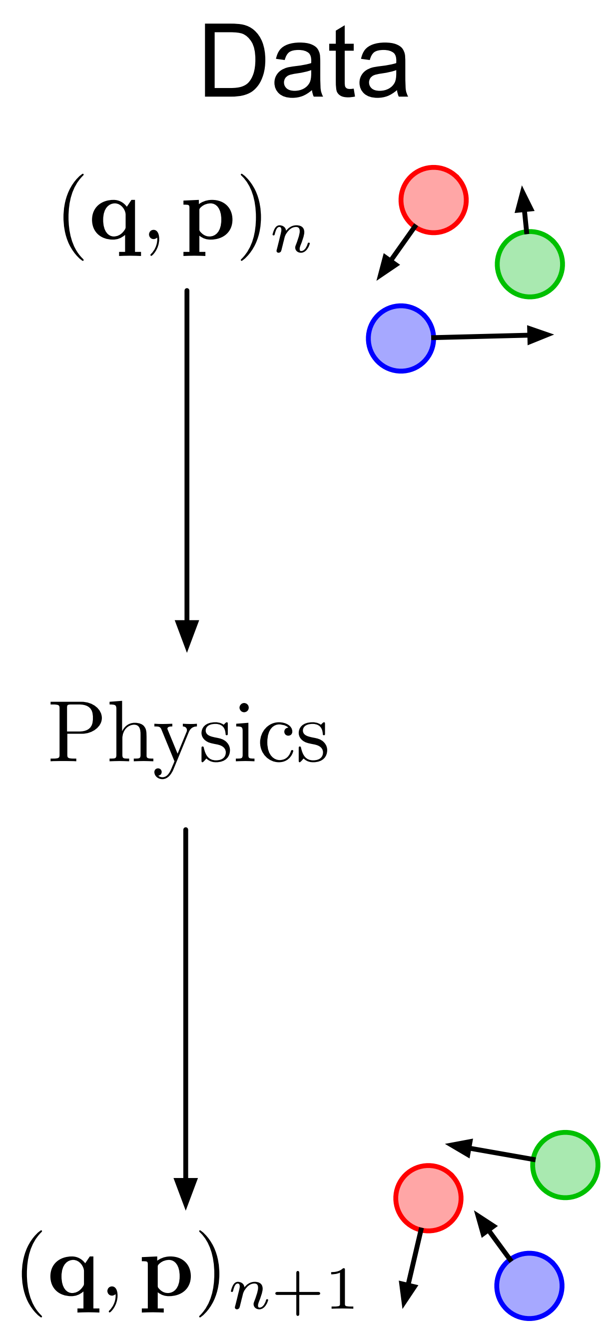
- After training, message vectors are linear transforms of the true forces
- Symbolic regression of the message function's formula reveals the analytical form of the true force laws



(see Miles' poster in the atrium)

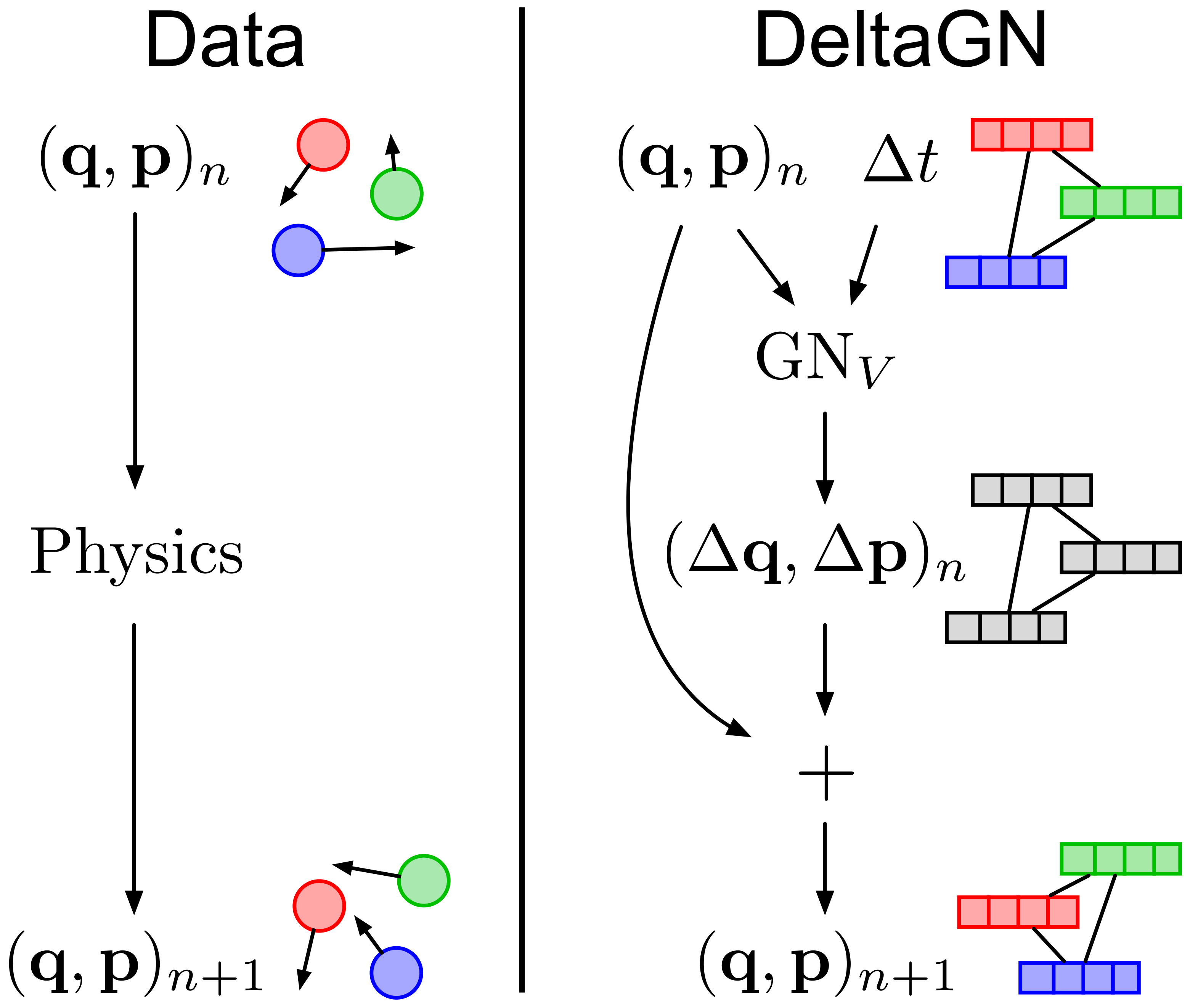
Cranmer et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



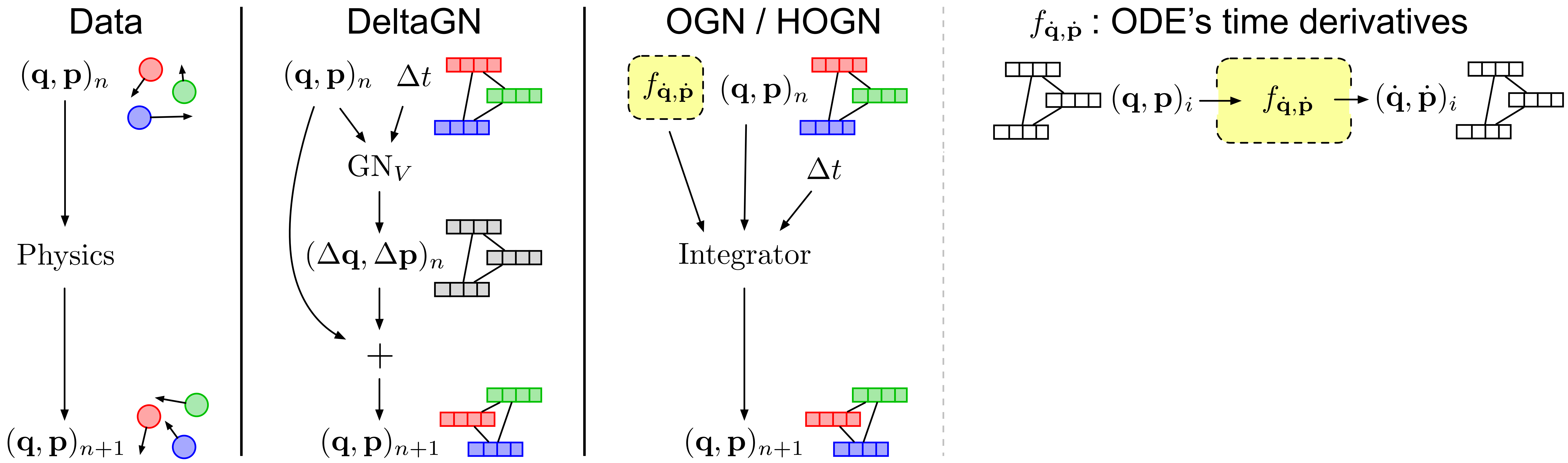
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



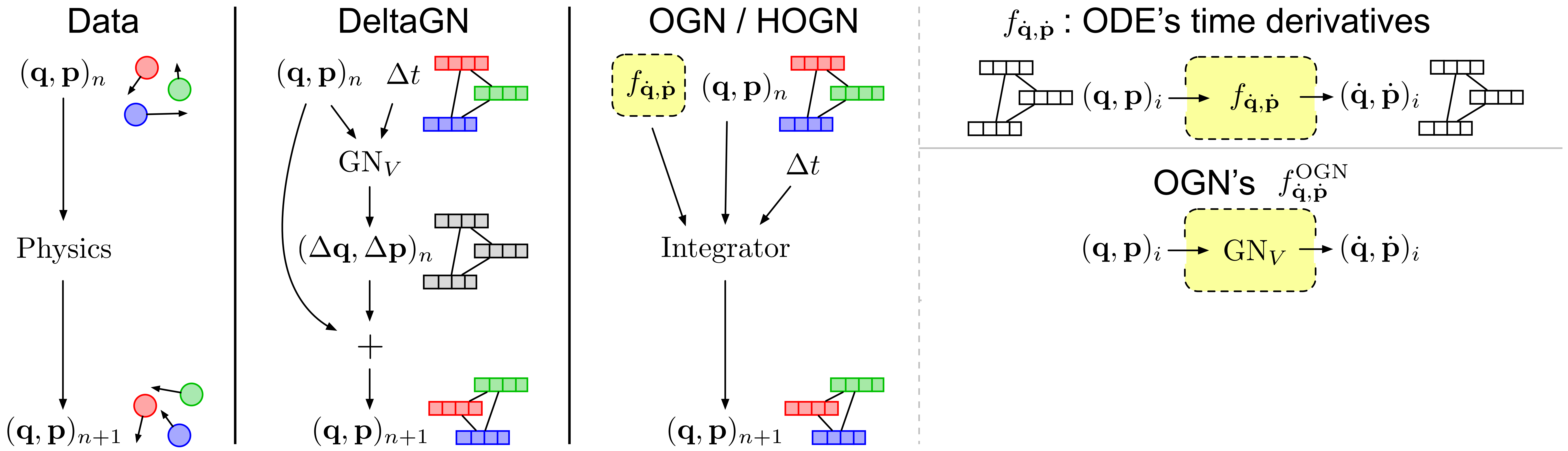
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



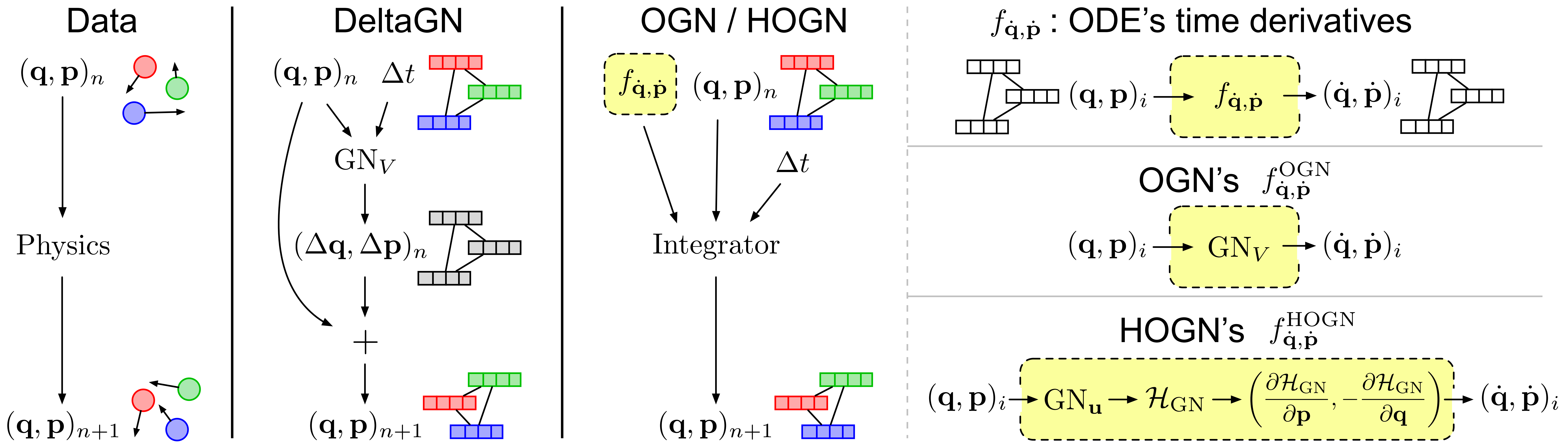
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



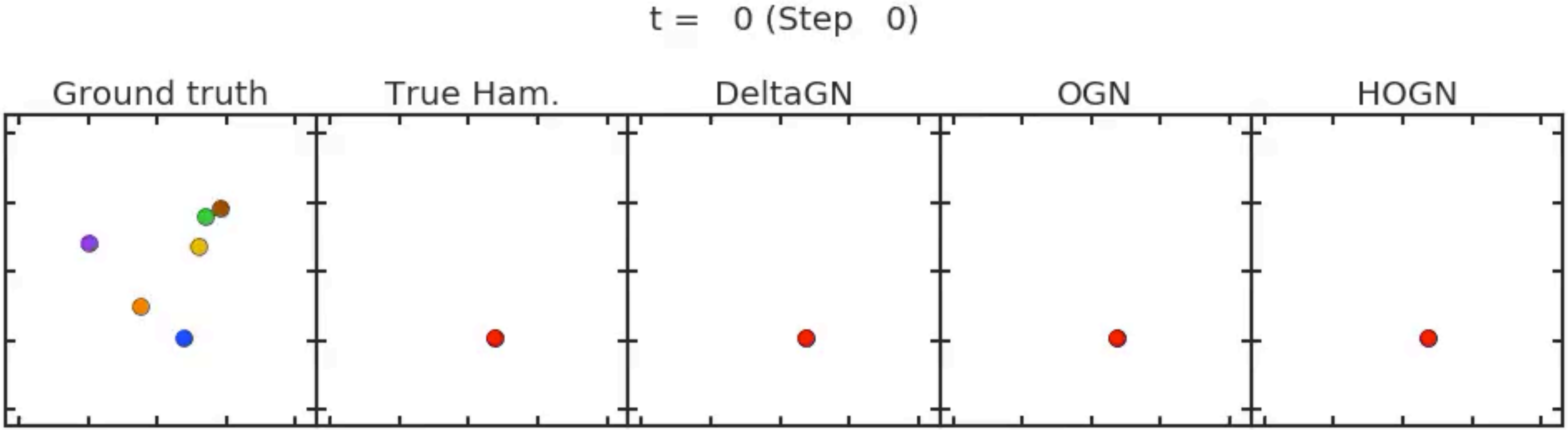
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

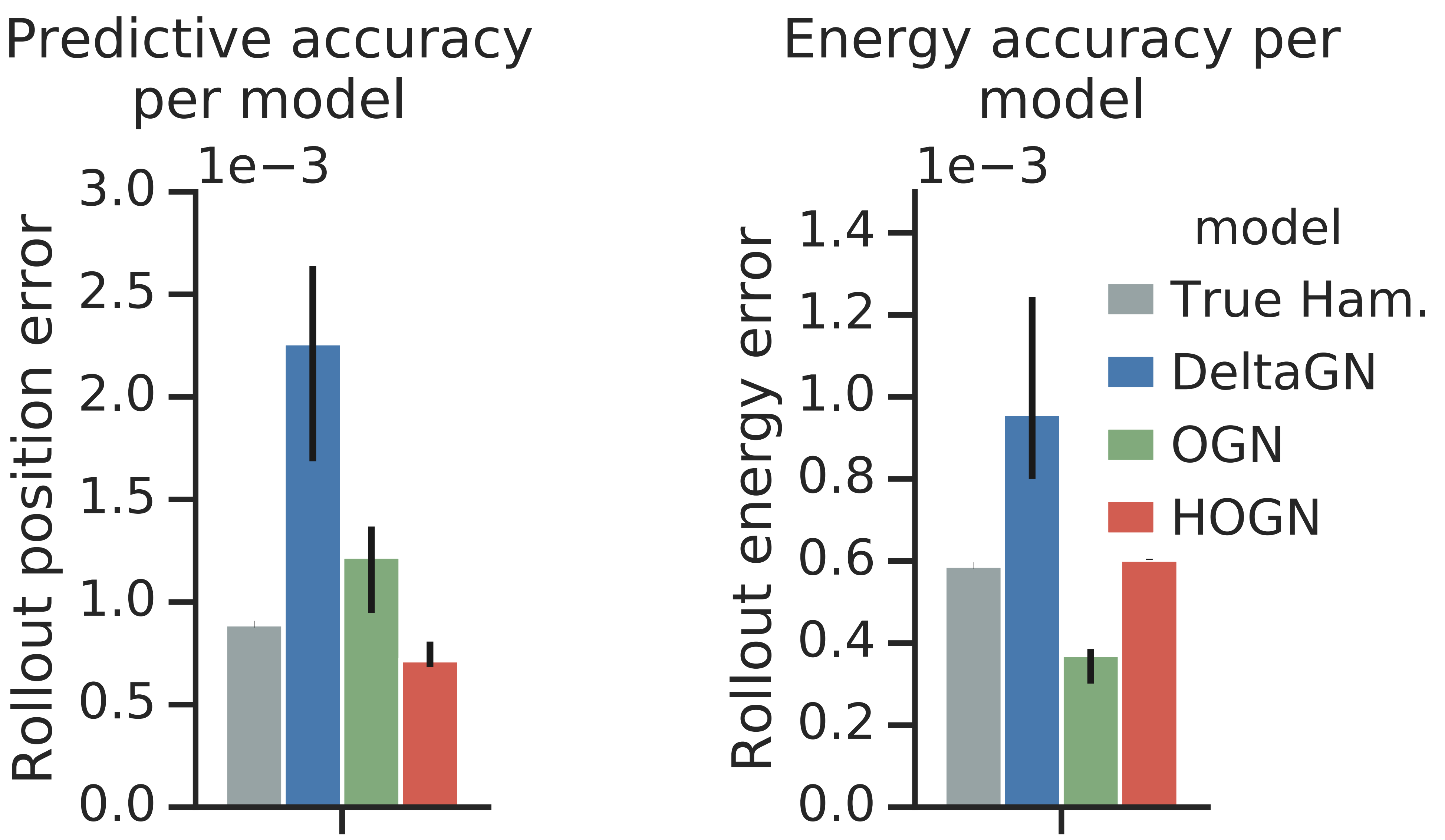
Hamiltonian ODE Graph Network



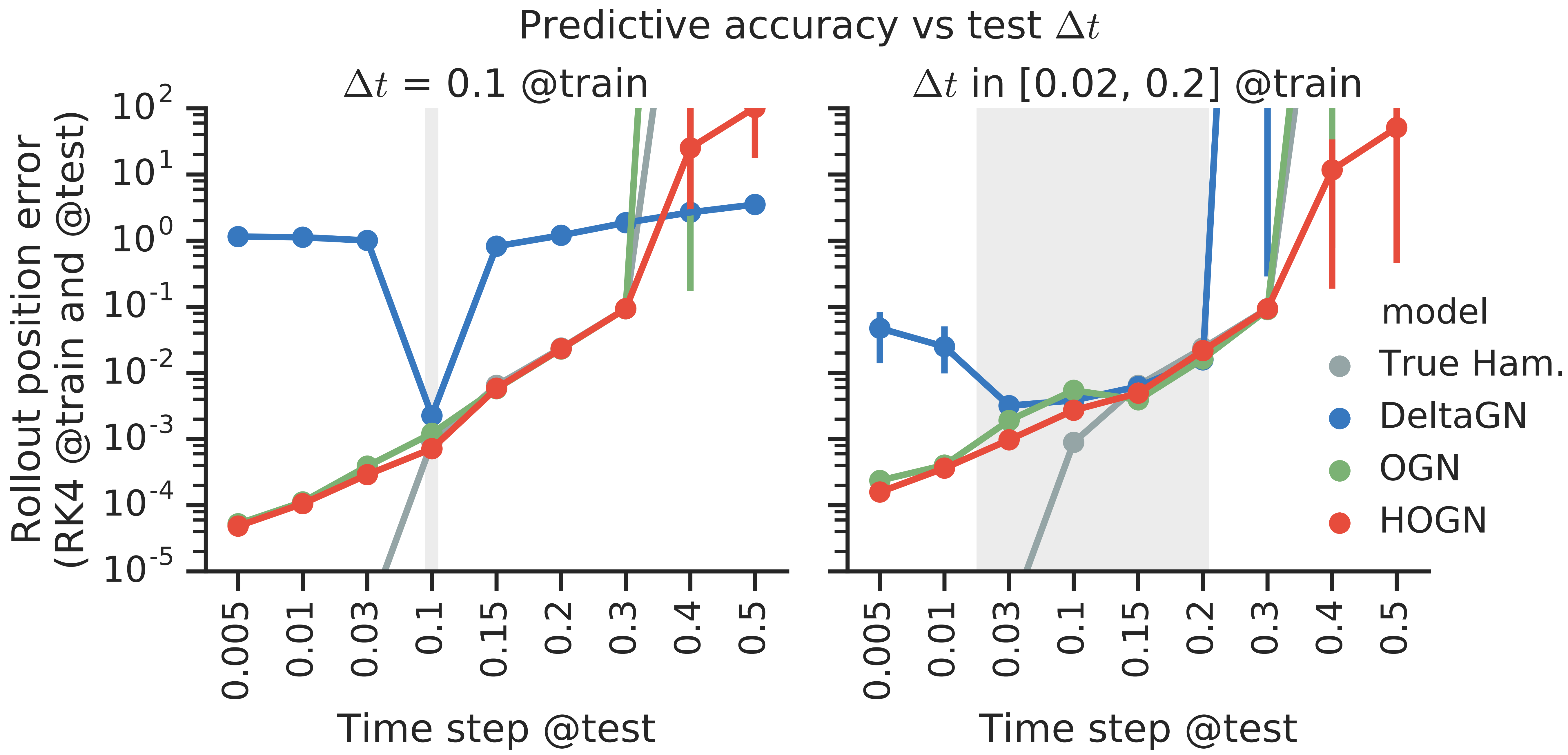
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network

Performance



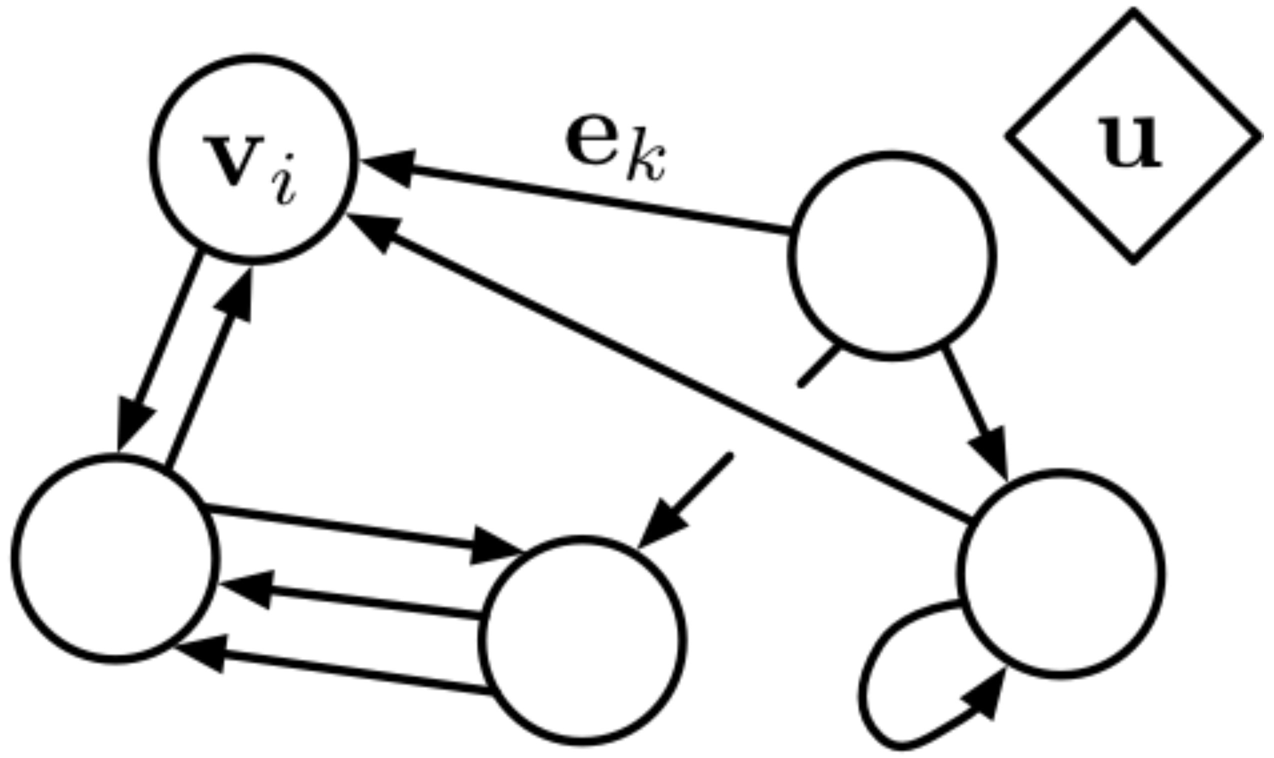
Generalization to untrained time steps

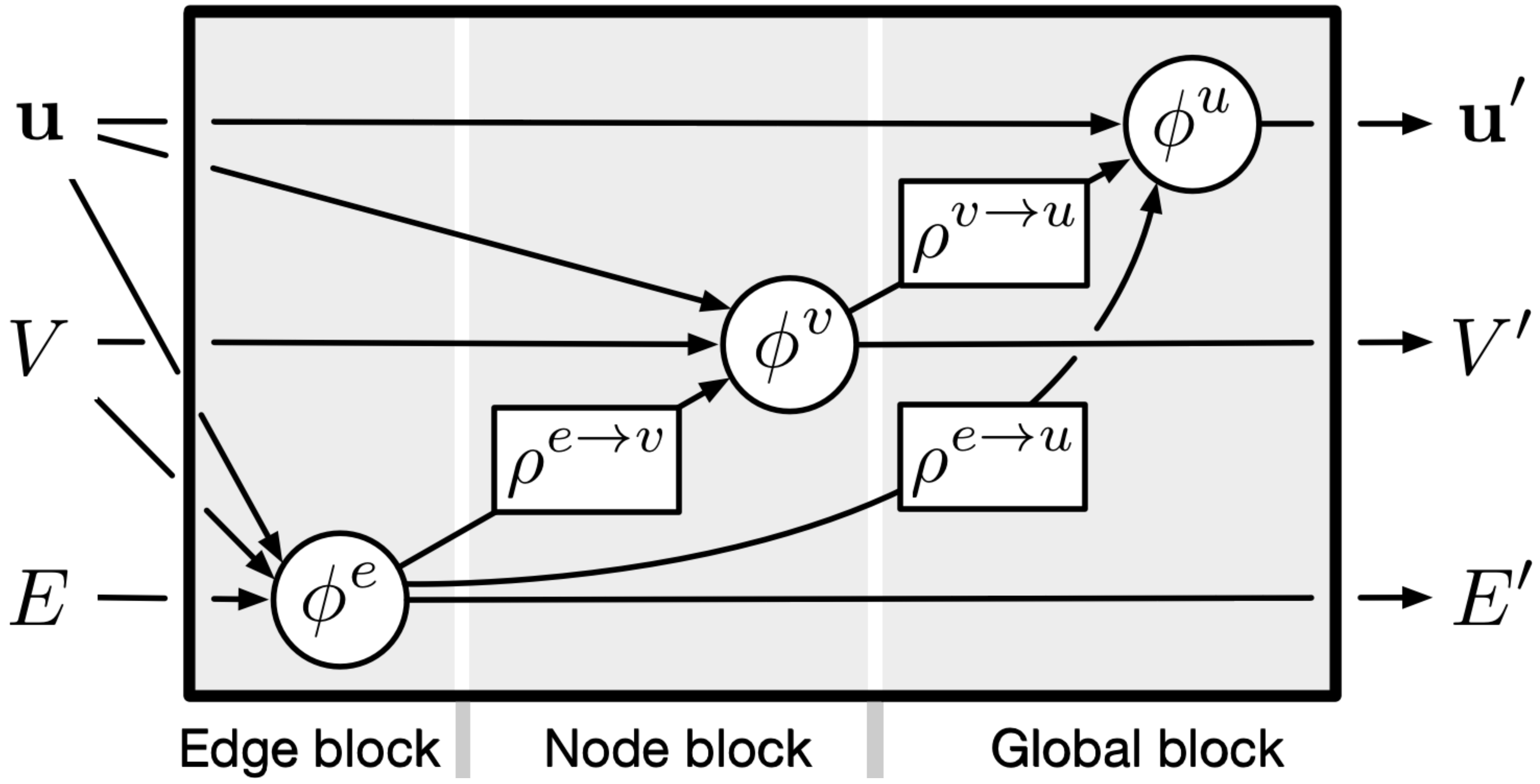


- OGN and HOGN used RK4 integrator (we also tested lower order RK integrators)
- We also tested symplectic integrators, and found HOGN has better energy accuracy/conservation

Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

General graph network processing pipeline

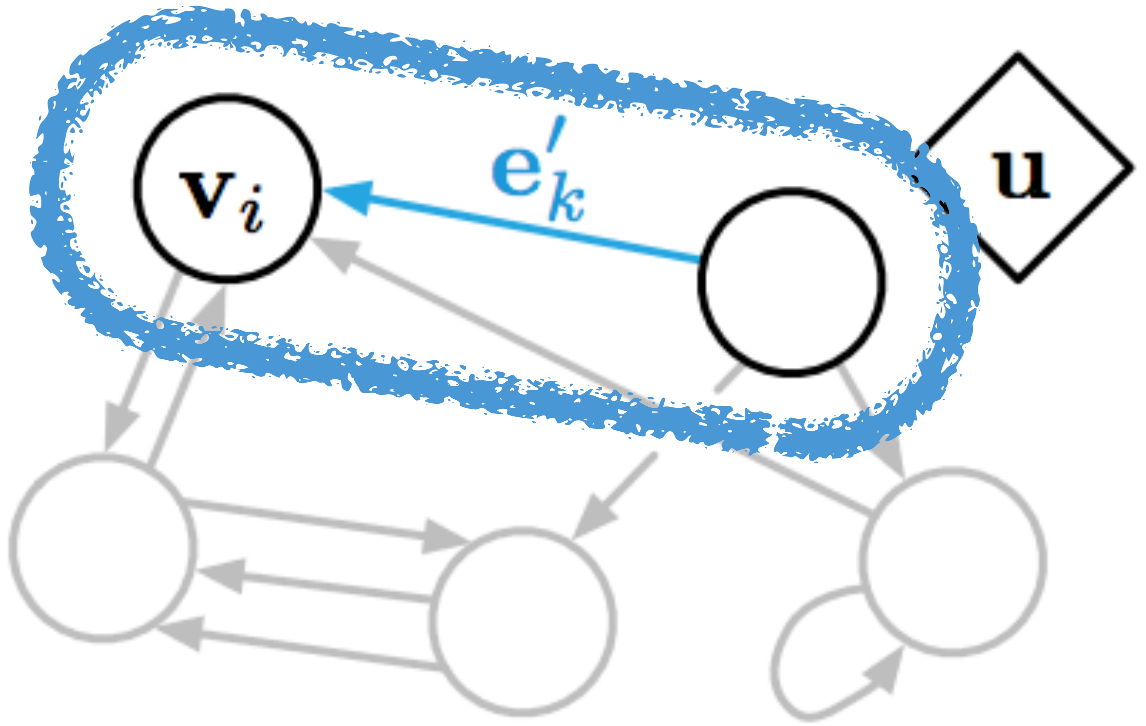




Edge block

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

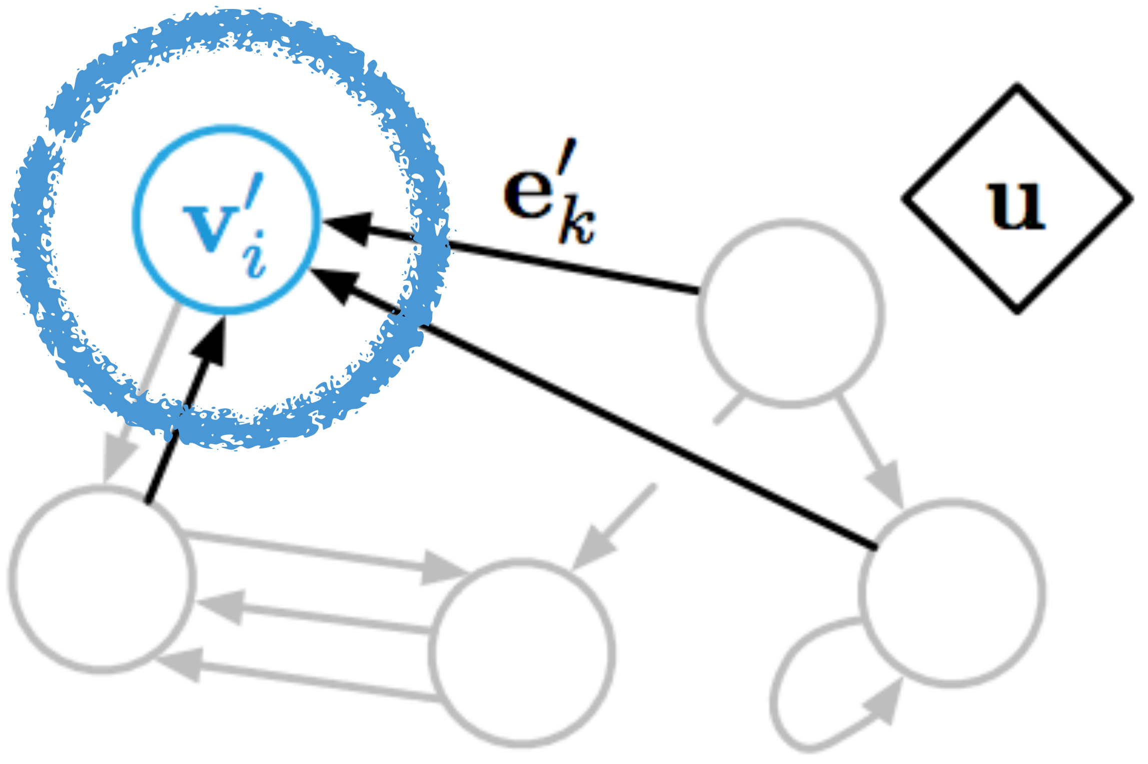
$$\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$$



Node block

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

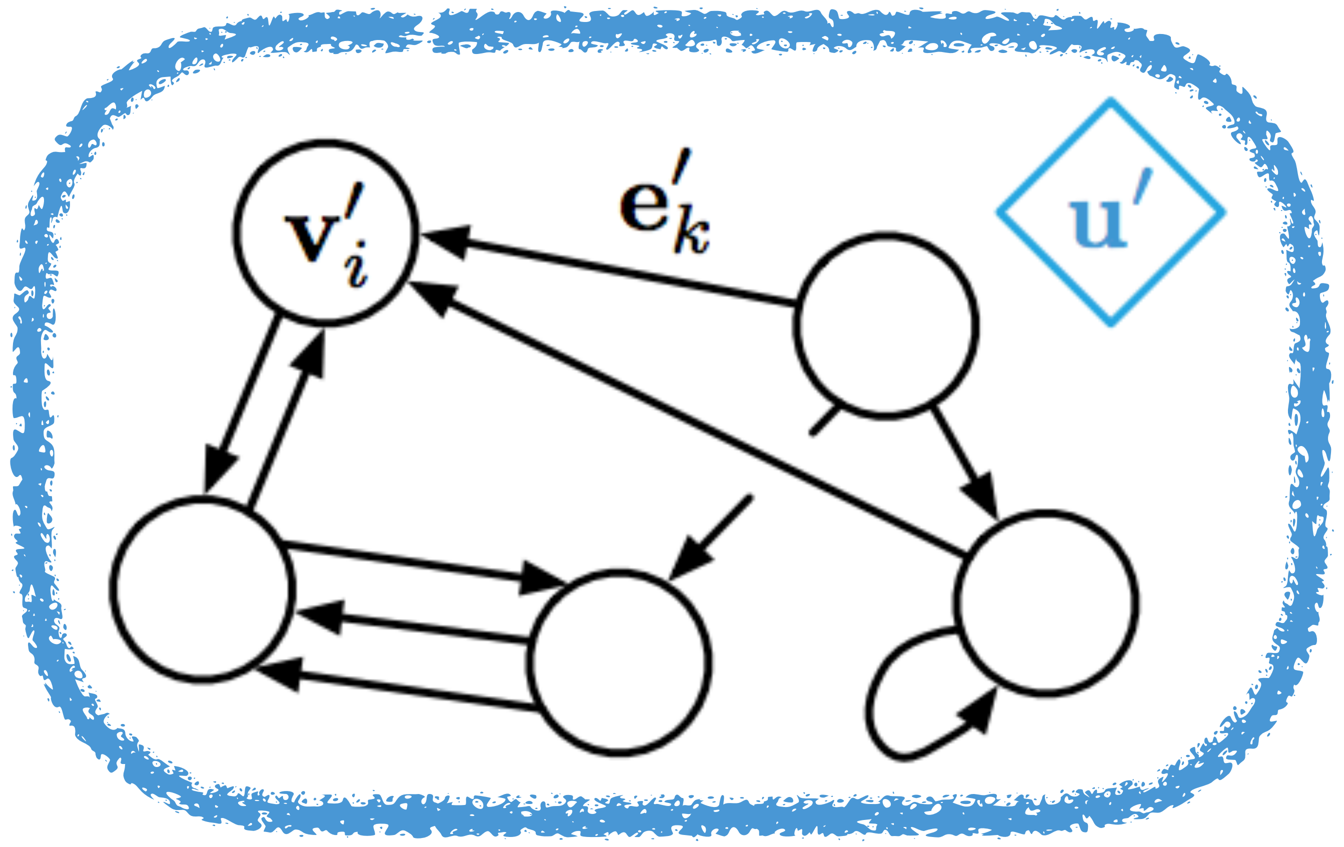
$$\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$$



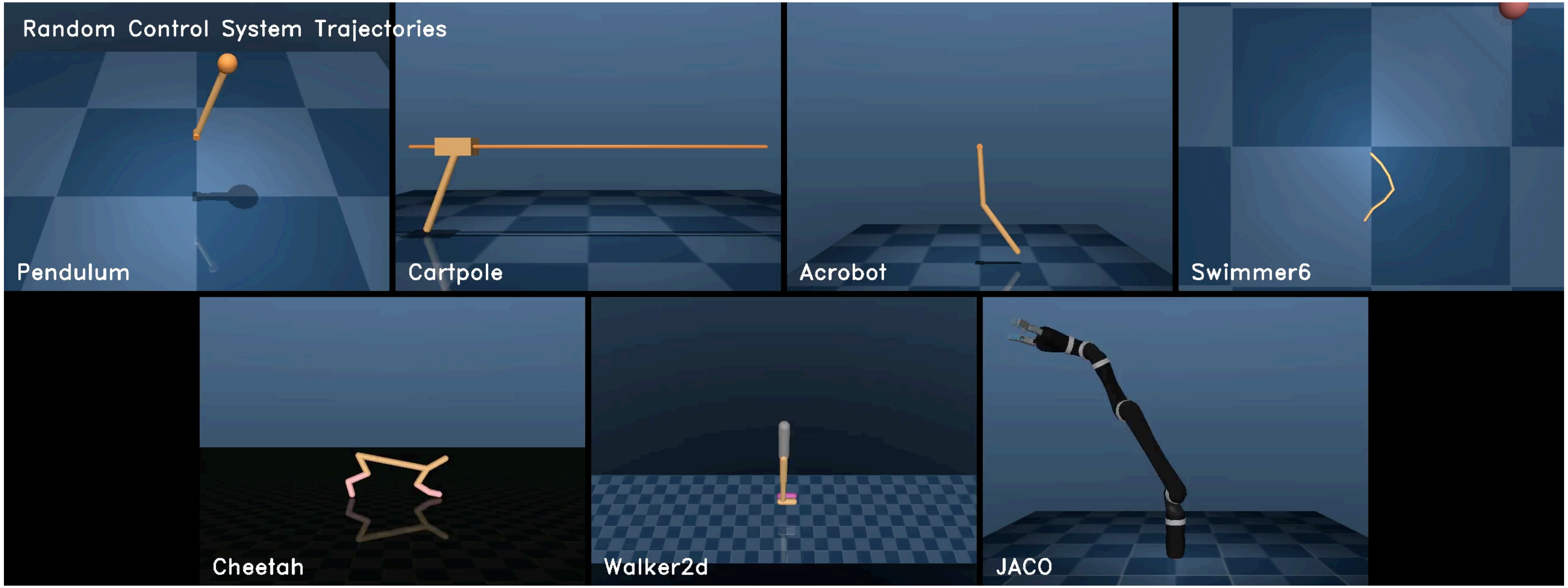
Global block

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

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



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



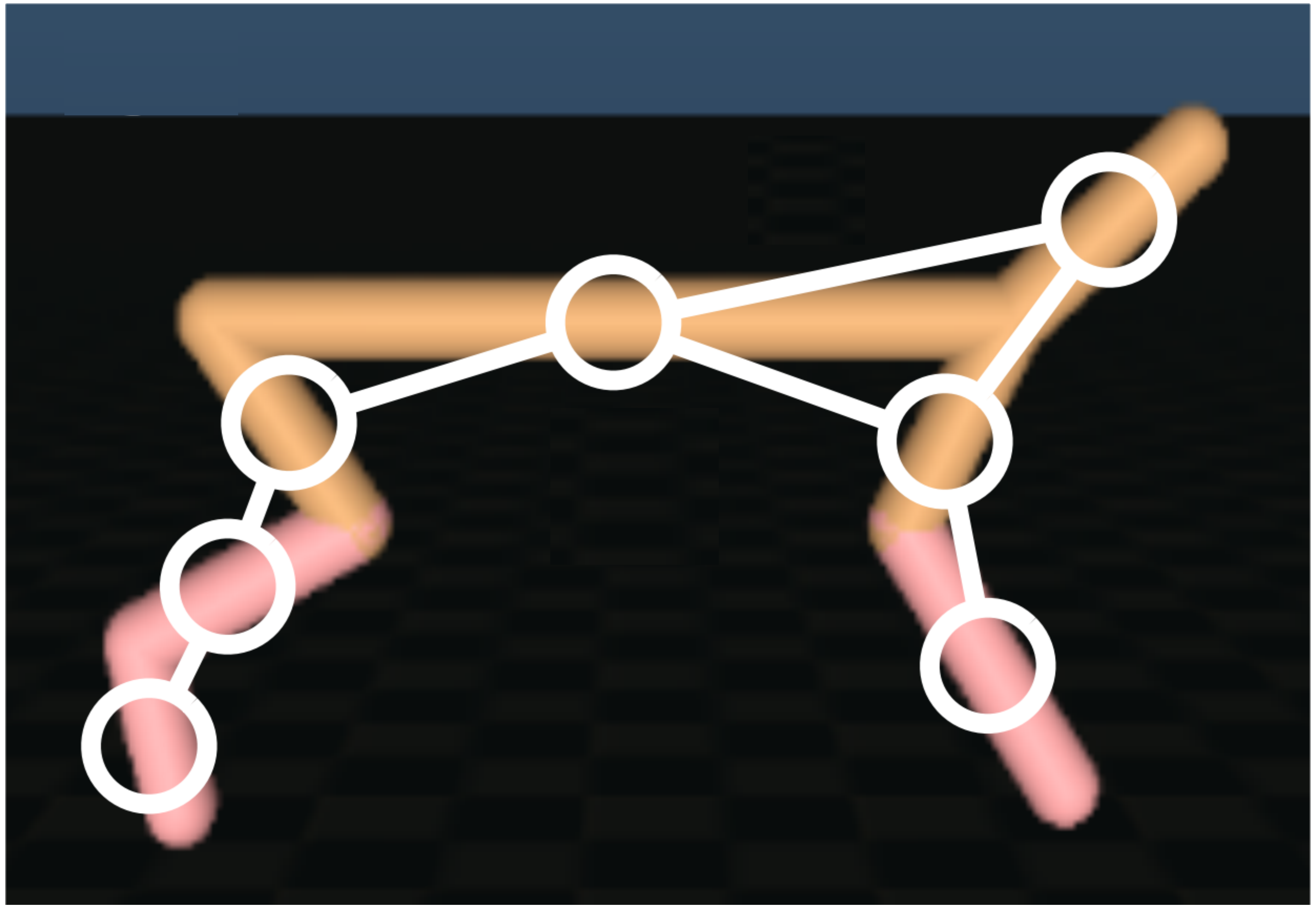
DeepMind Control Suite (Tassa et al., 2018)

JACO Arm

Kinematic tree of the actuated system as a graph

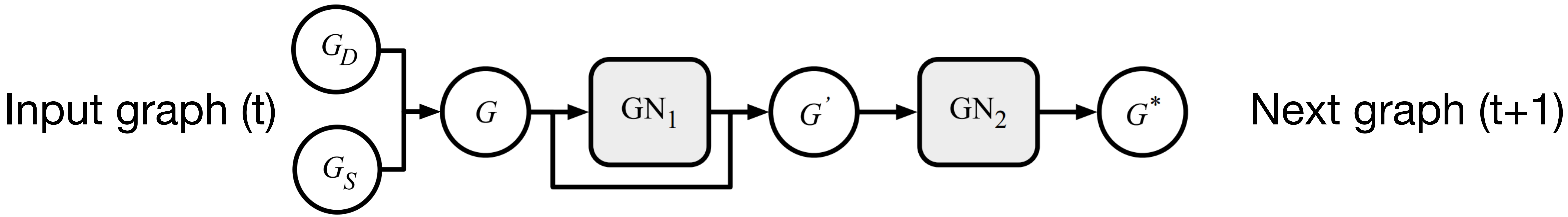
Controllable physical system as a graph:

- Bodies → Nodes
- Joints → Edges
- Global properties

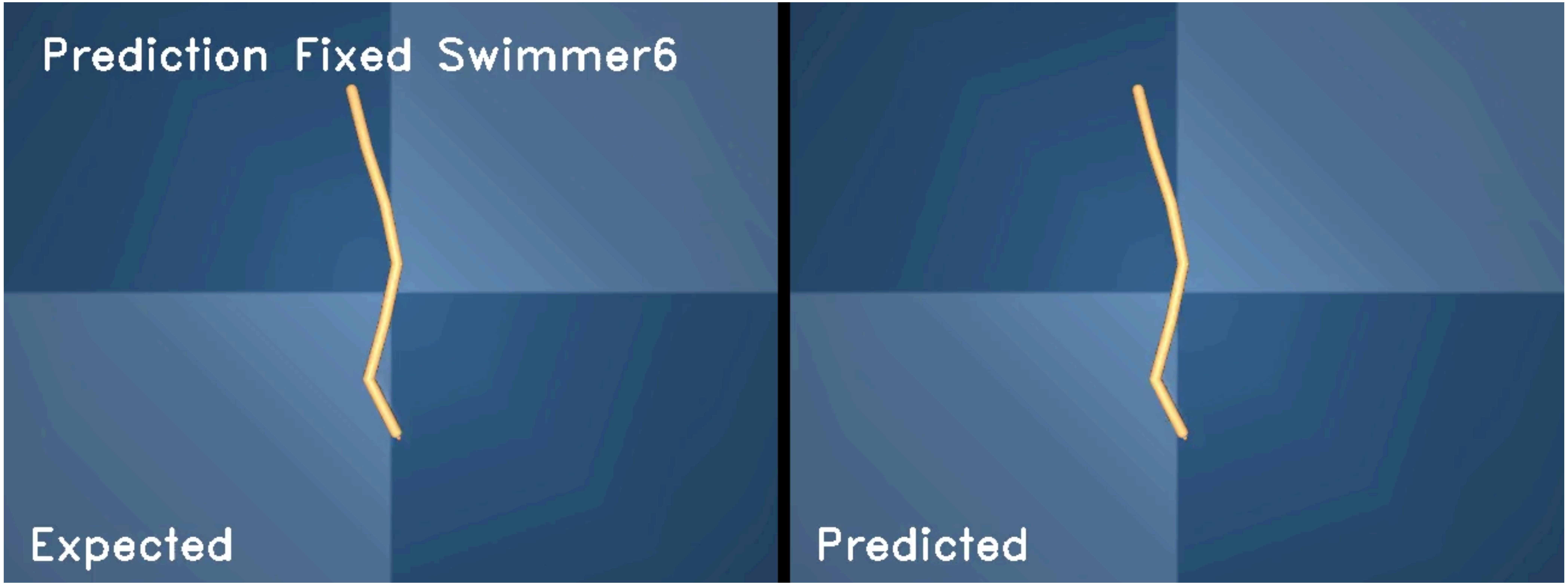


Sanchez-Gonzalez et al., 2018, ICML

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



Chained 100-step predictions

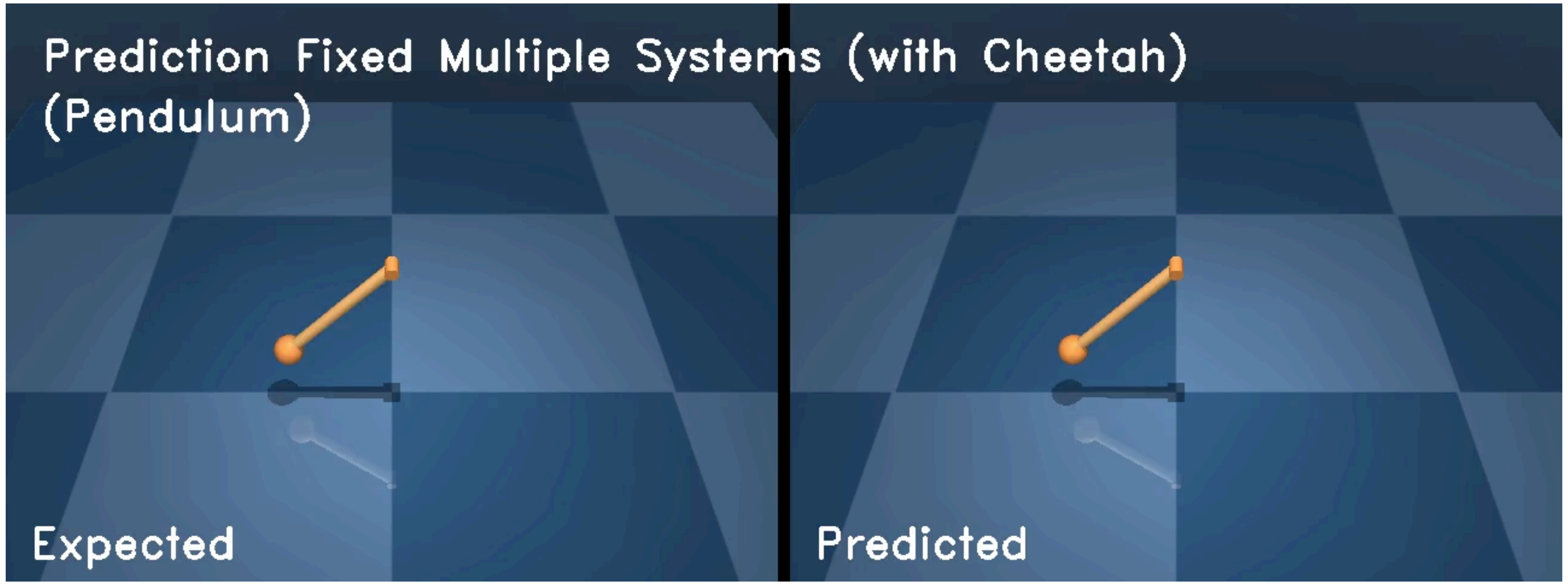


Sanchez-Gonzalez et al., 2018, ICML

Forward model: Multiple systems & zero-shot generalization

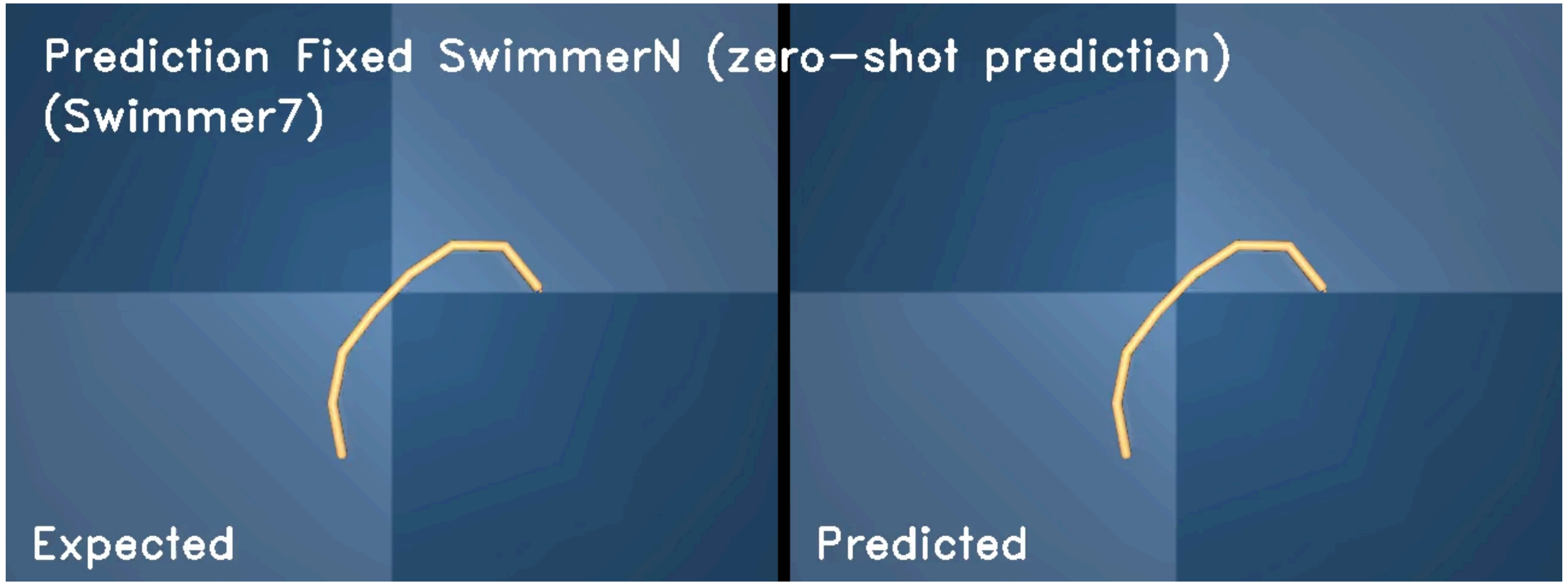
Single model trained:

- Pendulum, Cartpole, Acrobot, Swimmer6 & Cheetah



Zero-shot generalization: Swimmer

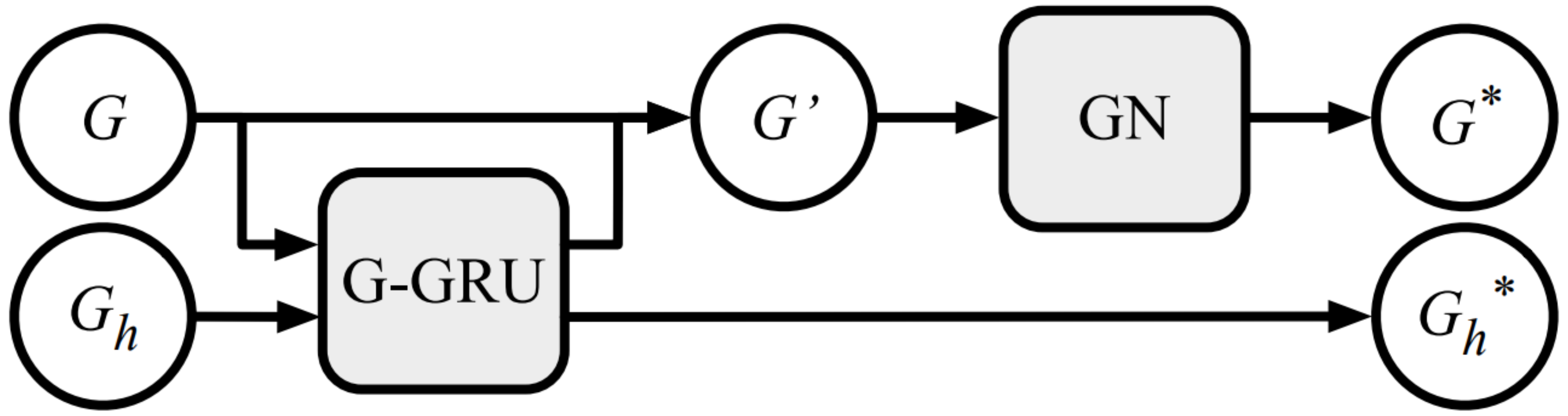
- # training links: {**3**, **4**, **5**, **6**, -, **8**, **9**, -, -, …}
- # testing links: {-, -, -, -, **7**, -, -, **10-14**}

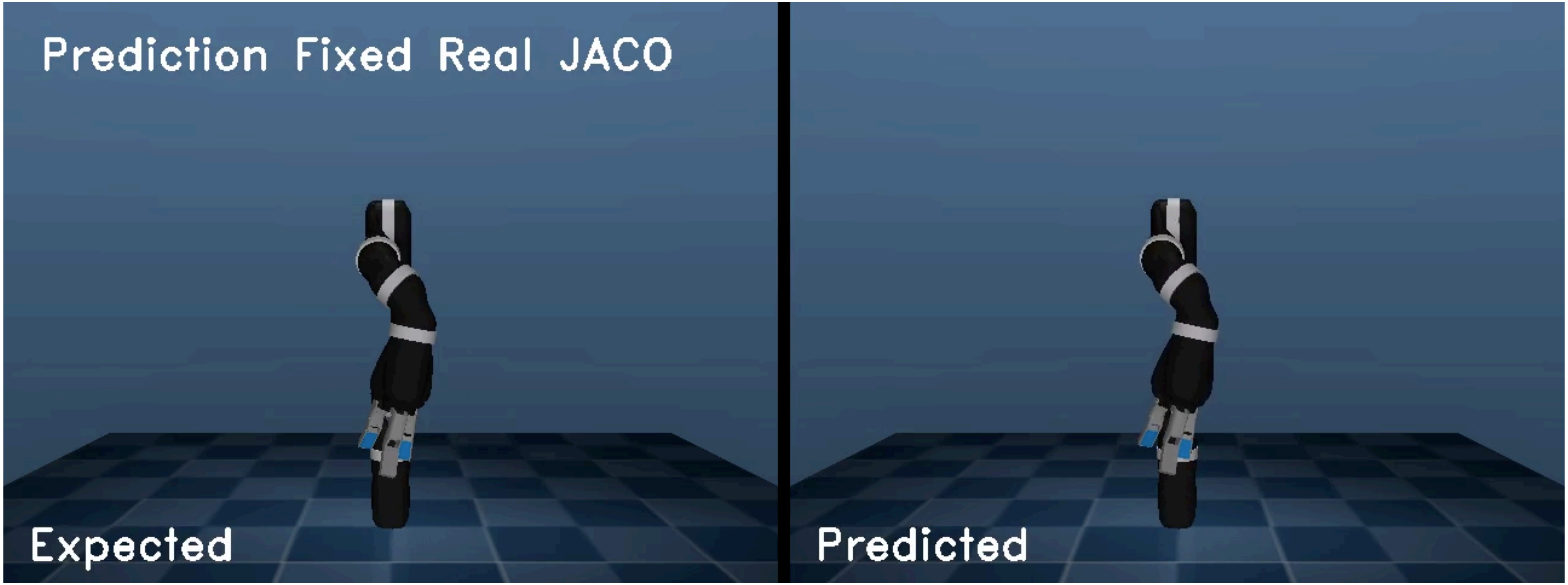


Sanchez-Gonzalez et al., 2018, ICML

Forward model: Real JACO data

Recurrent GN

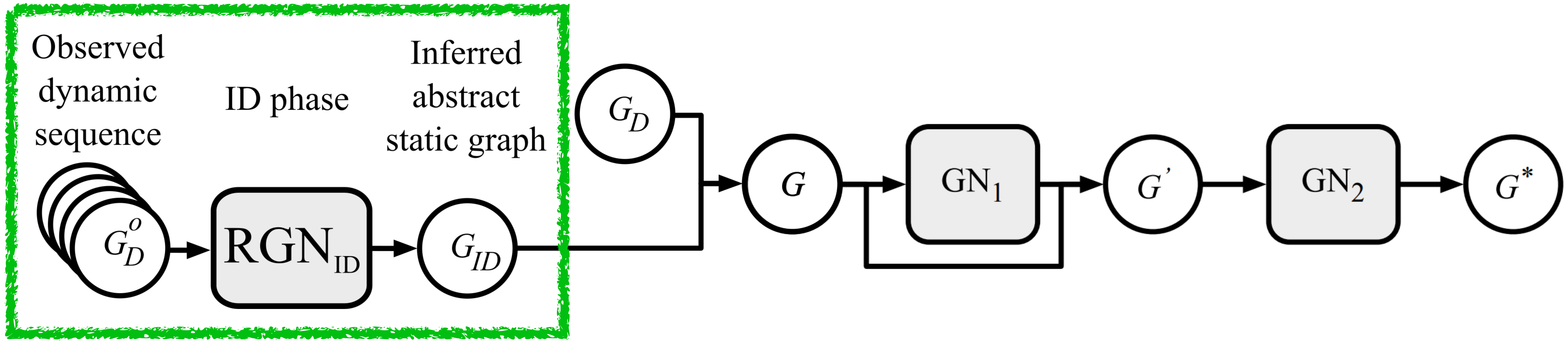


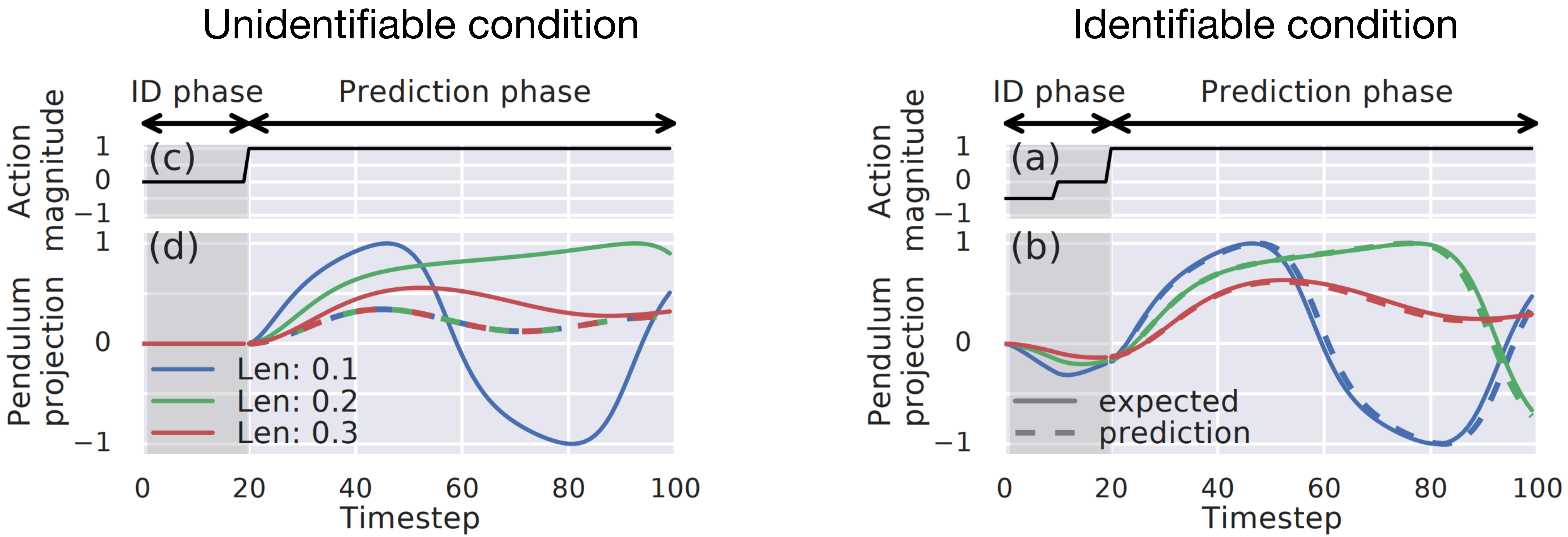


Sanchez-Gonzalez et al., 2018, ICML

Inference: GN-based system identification

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

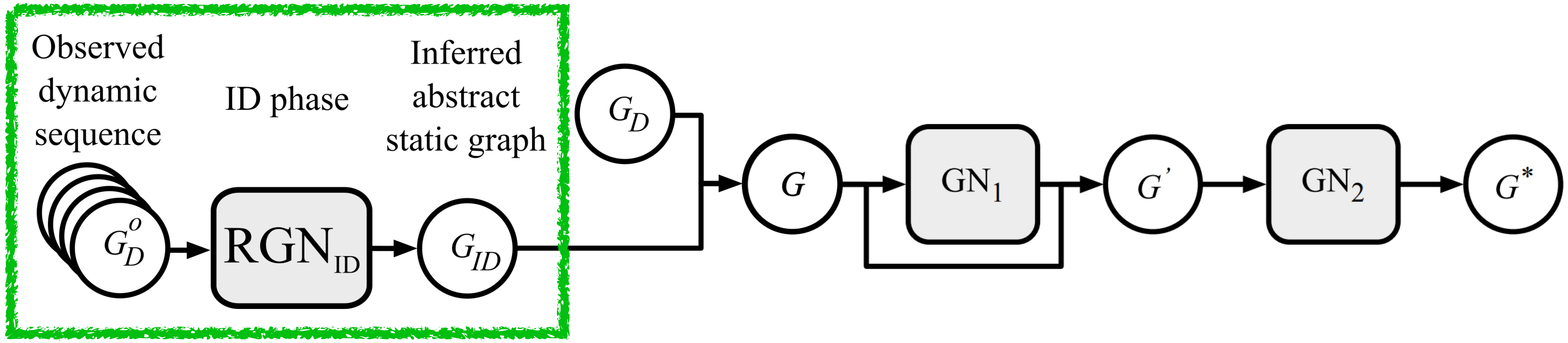


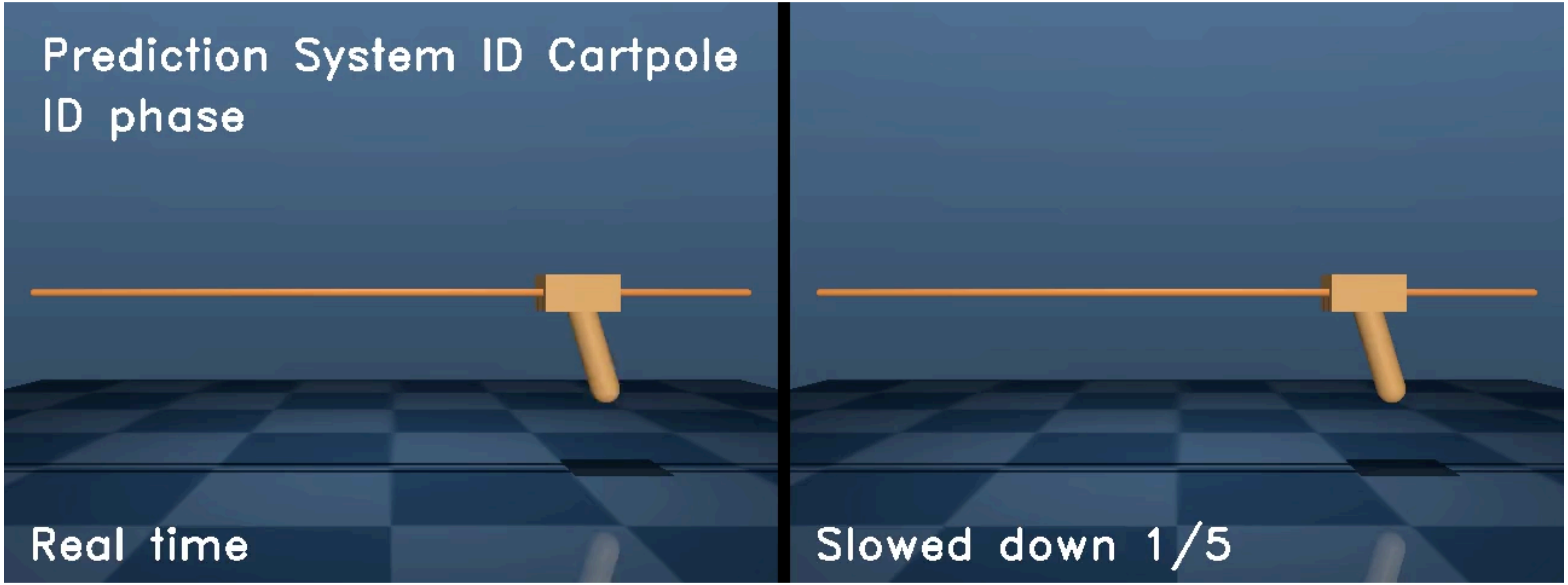


Sanchez-Gonzalez et al., 2018, ICML

Inference: GN-based system identification

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

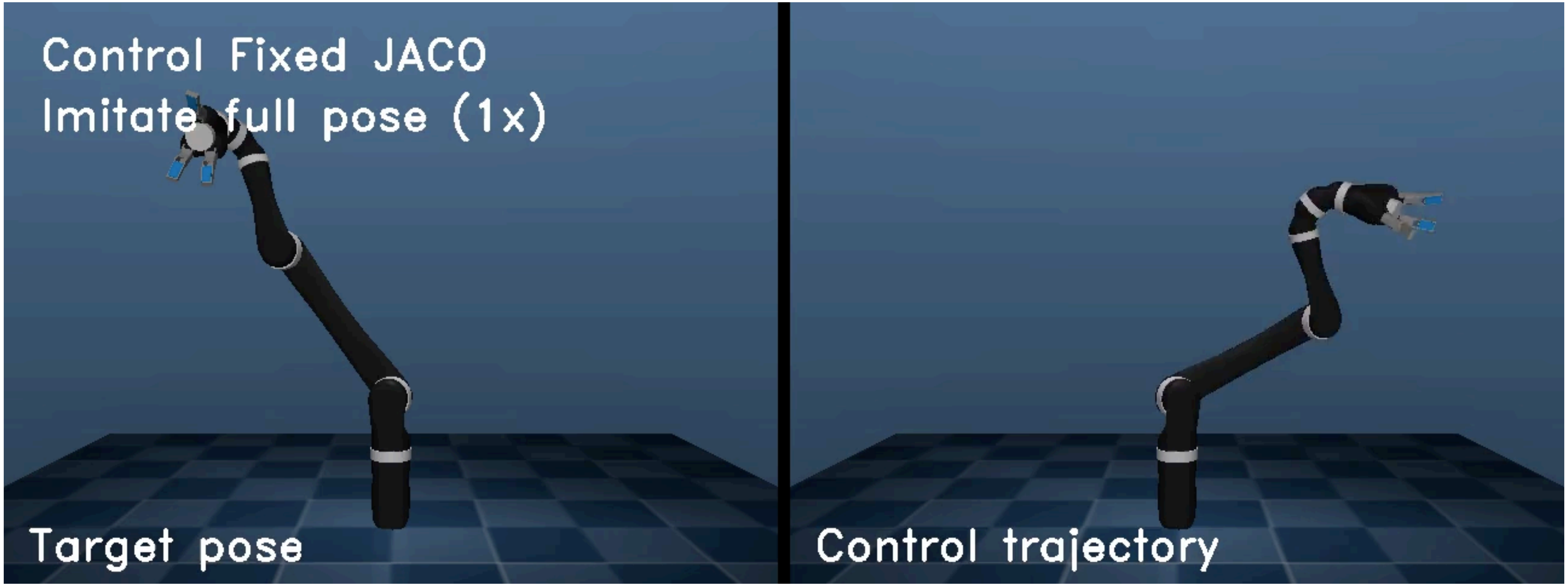




Sanchez-Gonzalez et al., 2018, ICML

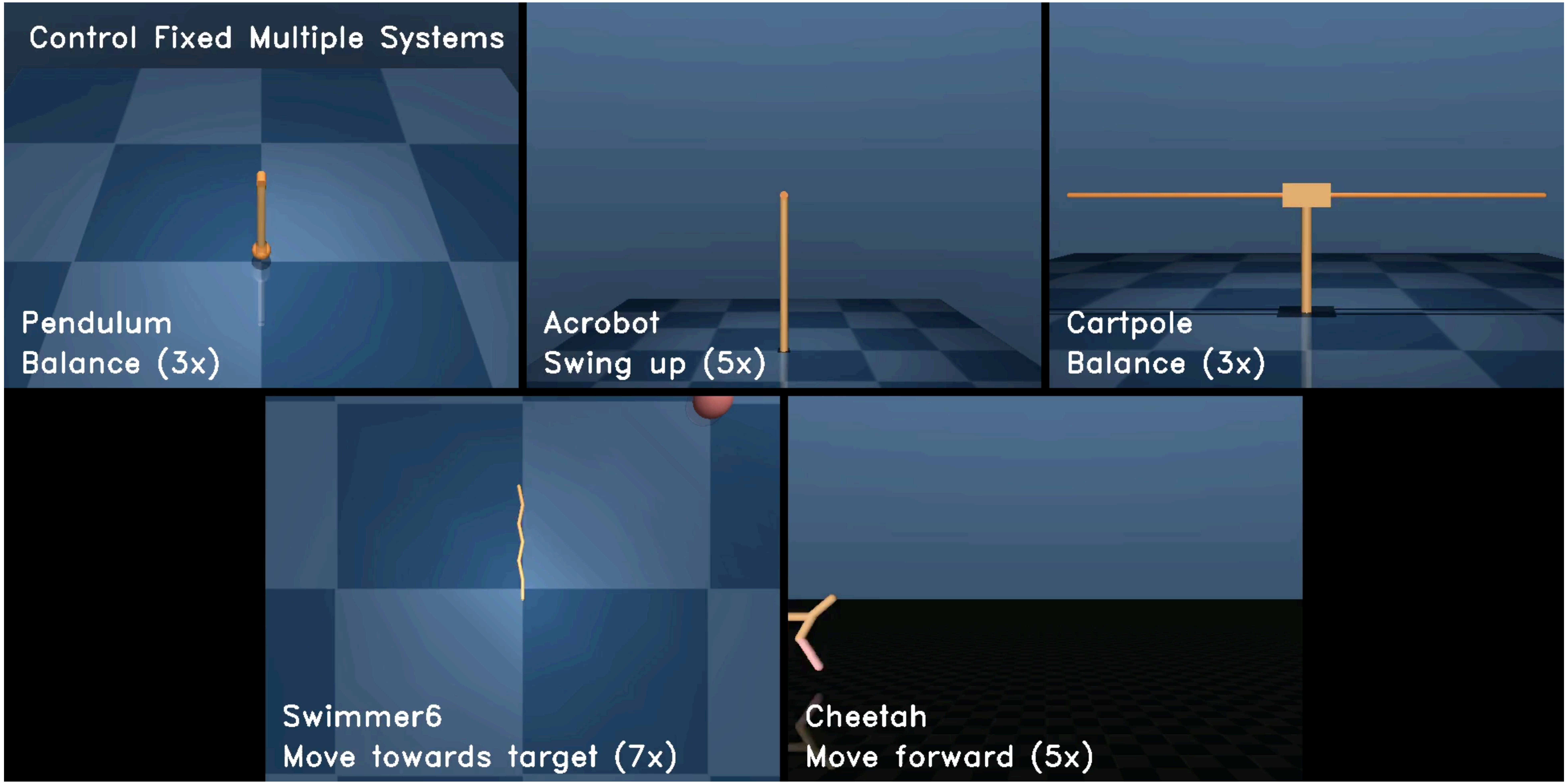
Control: Model-based planning

The GN-based forward model is differentiable, so we can backpropagate through it to search for a sequence of actions that maximize reward.



Sanchez-Gonzalez et al., 2018, ICML

Control: Multiple systems via a single model

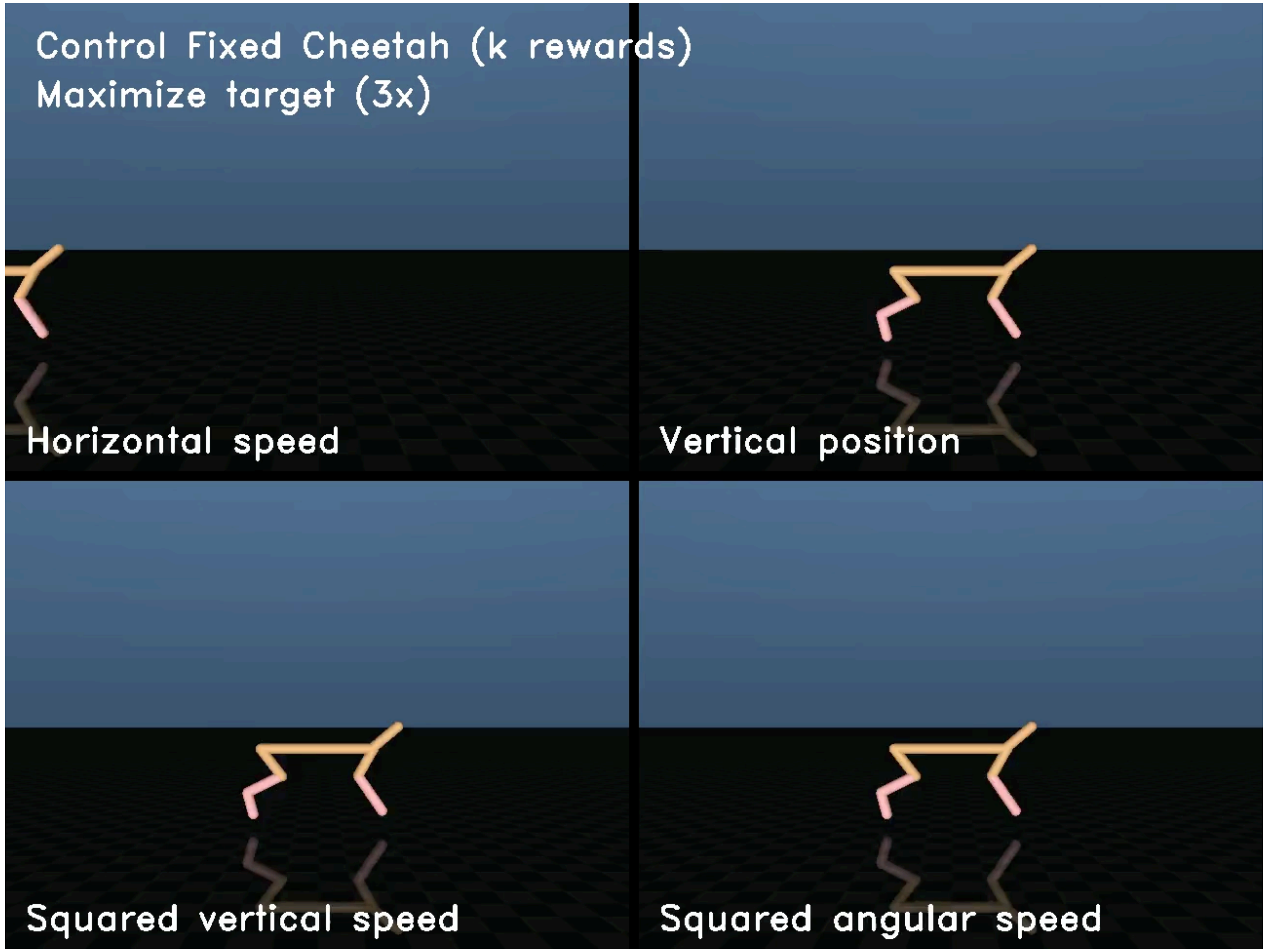


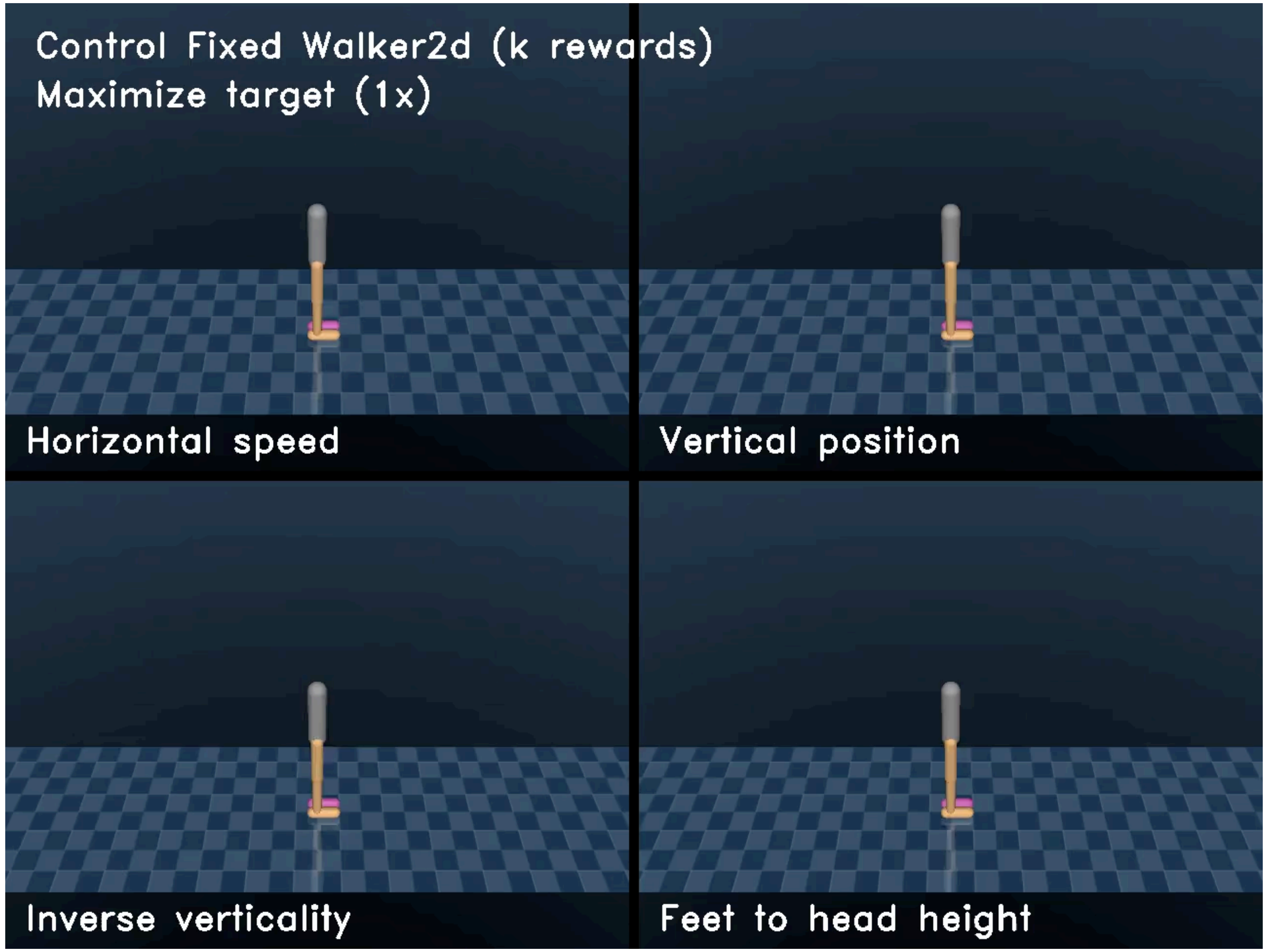
Sanchez-Gonzalez et al., 2018, ICML

Control: Zero-shot control

Sanchez-Gonzalez et al., 2018, ICML

Control: Multiple reward functions

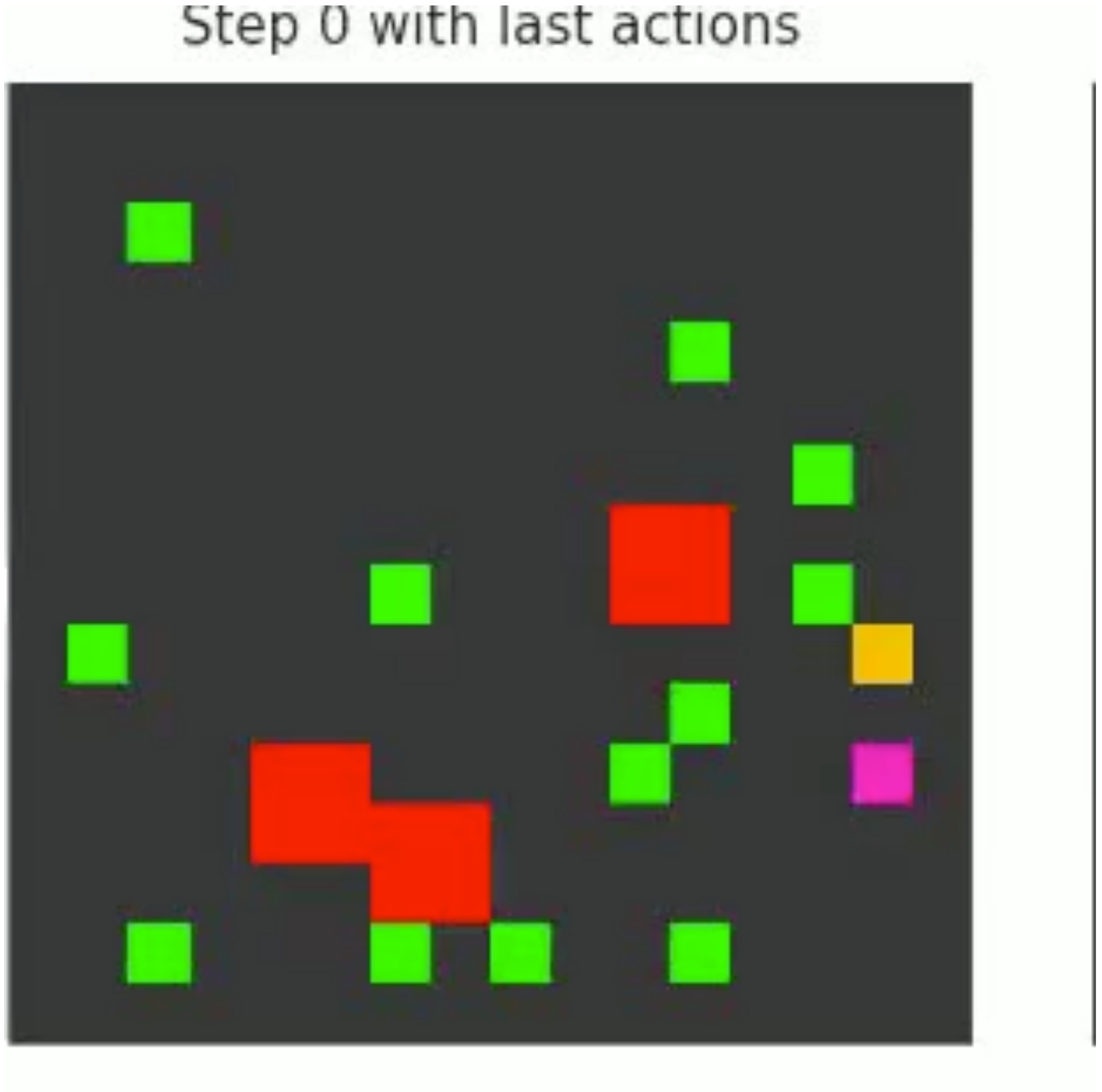




Sanchez-Gonzalez et al., 2018, ICML

Graph networks as forward models of multi-agent RL systems

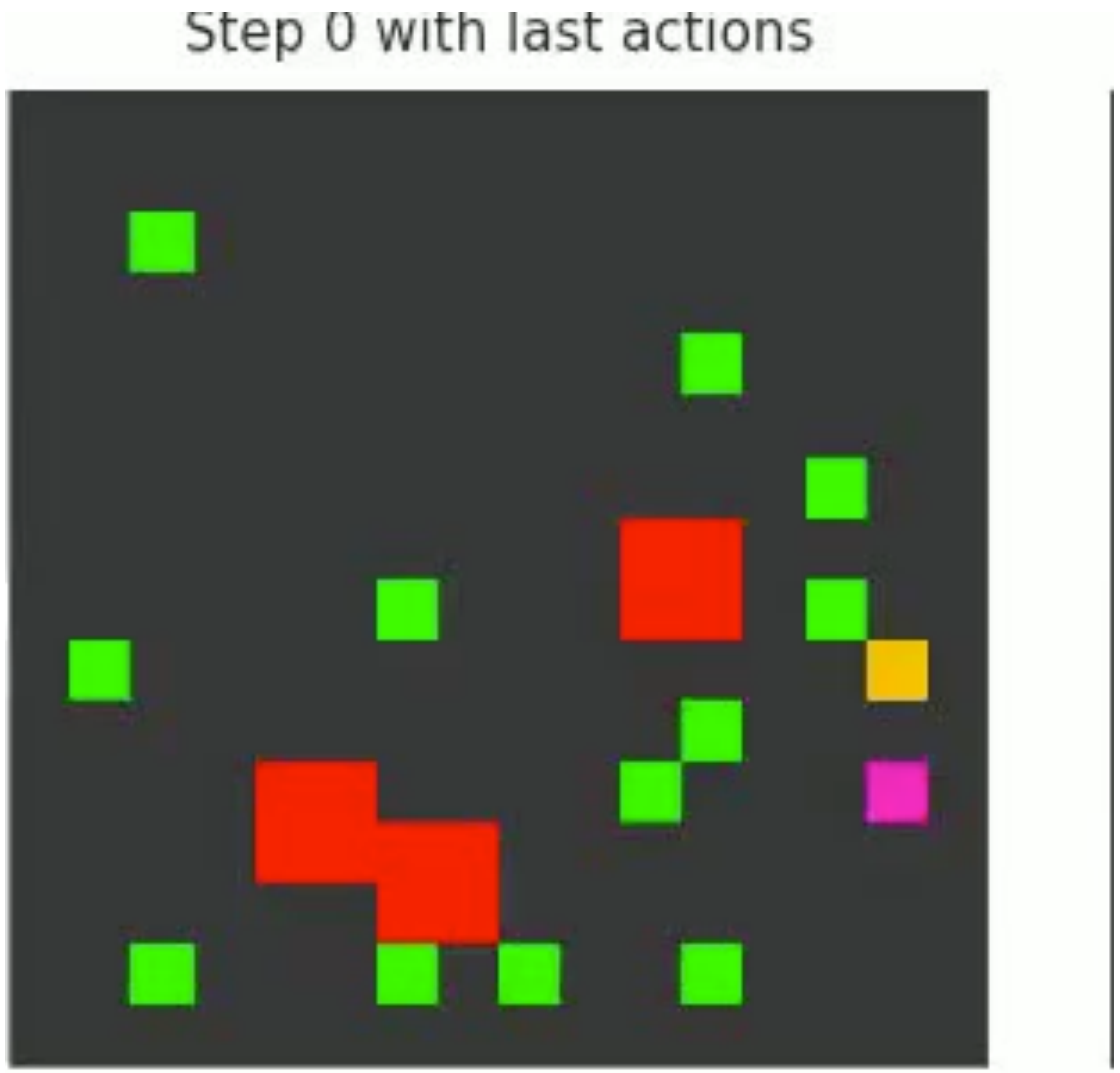
"Stag hunt"



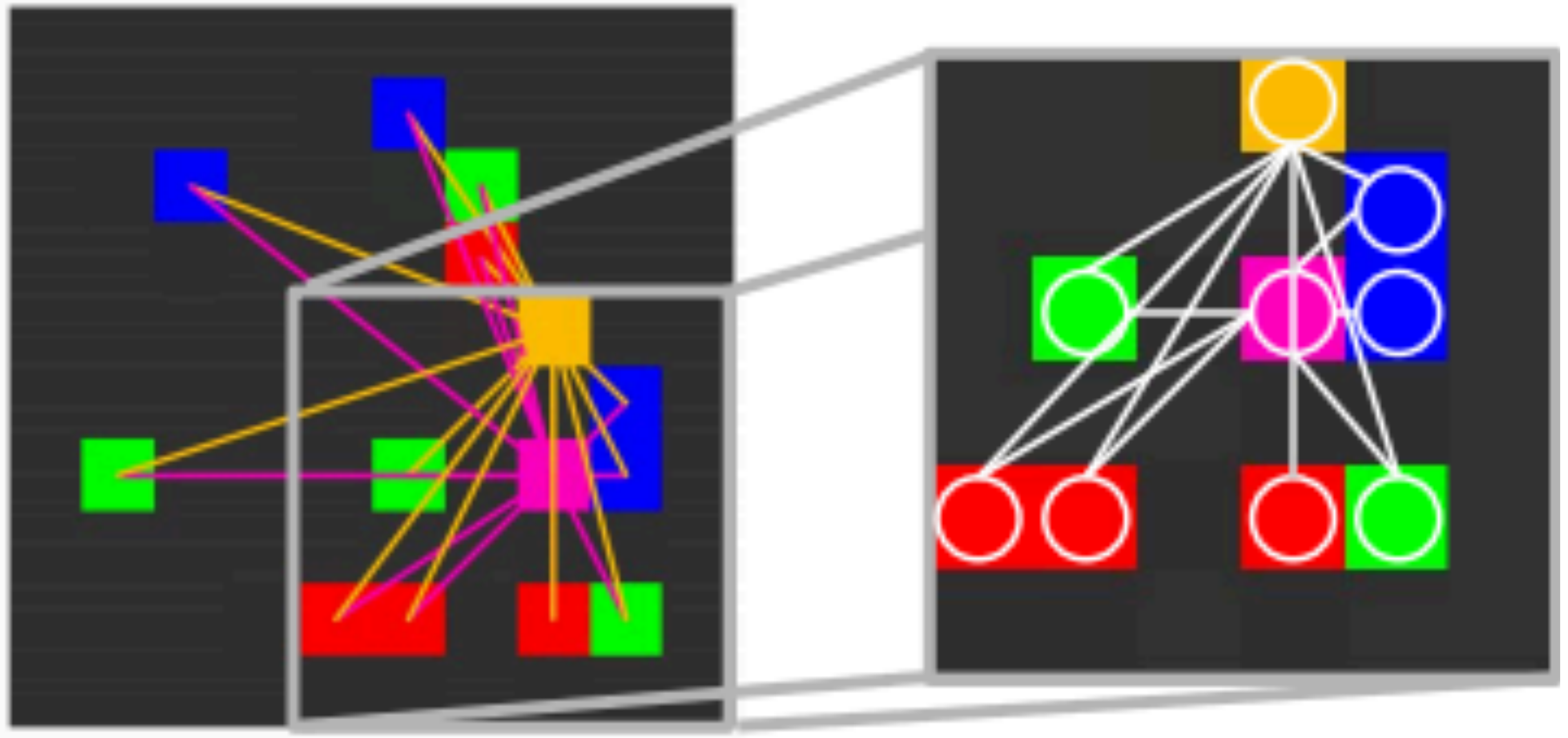
Tacchetti et al., 2019, ICLR

Graph networks as forward models of multi-agent RL systems

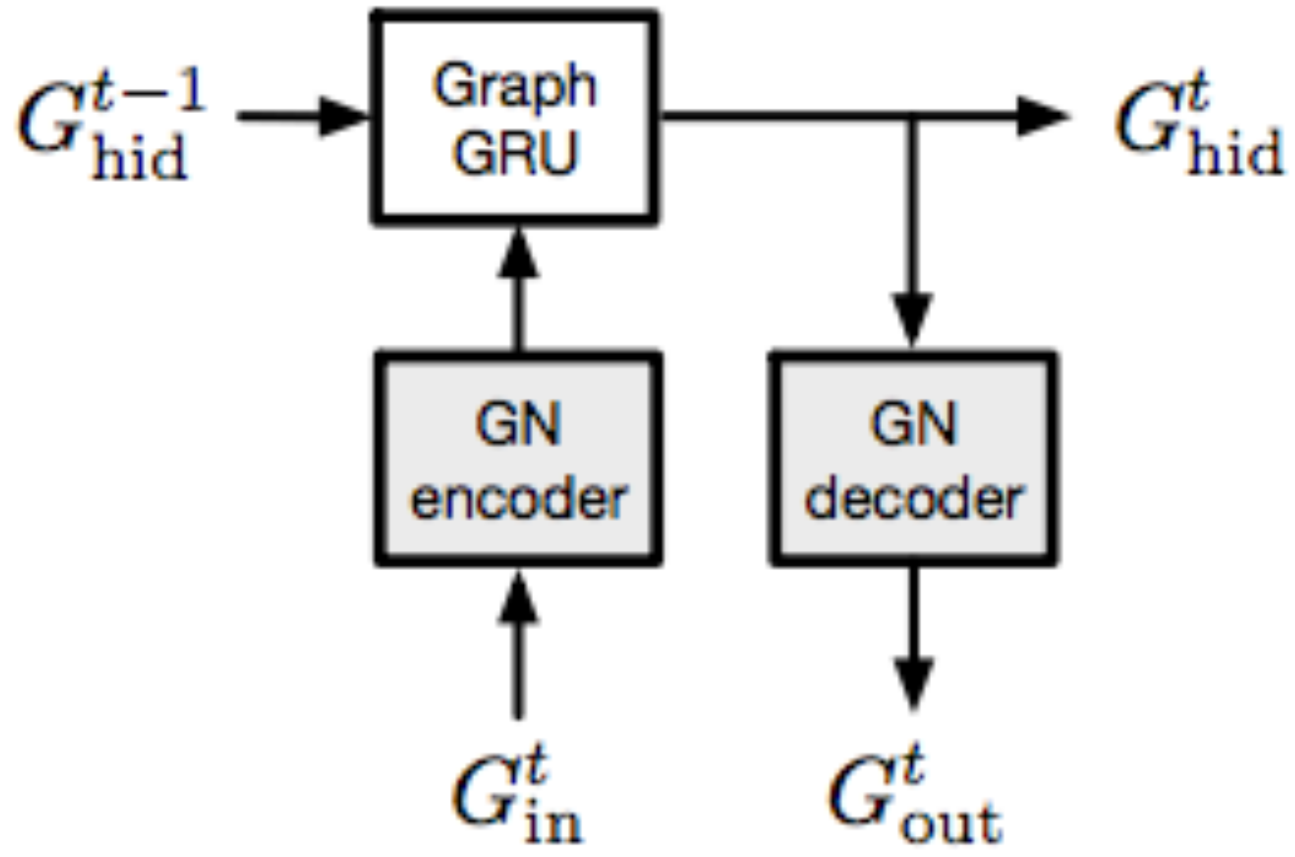
"Stag hunt"



Graph representation



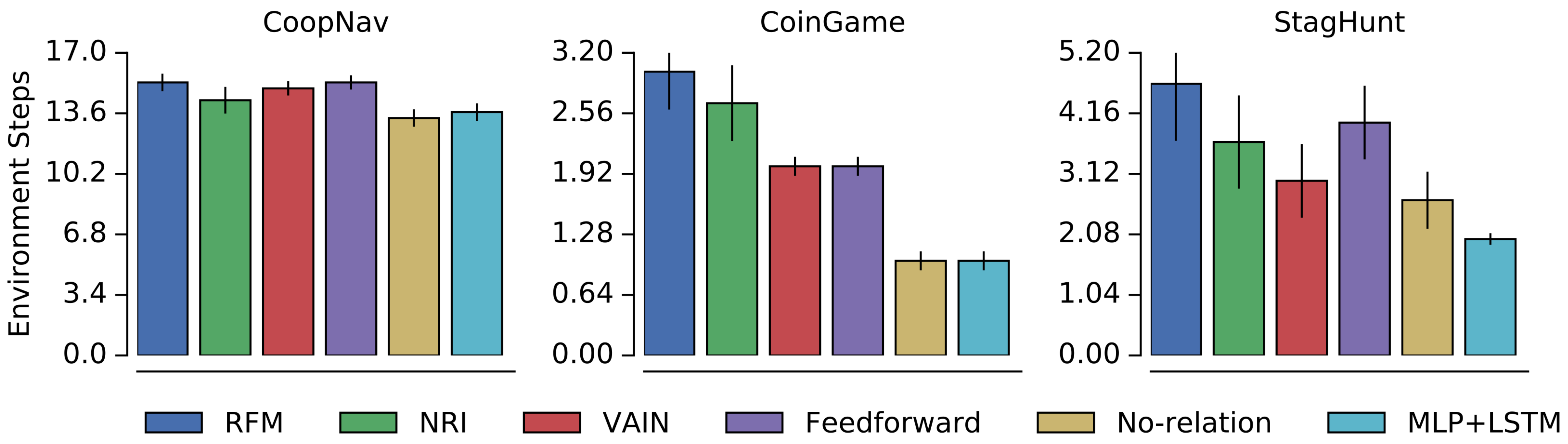
Recurrent graph net architecture



Tacchetti et al., 2019, ICLR

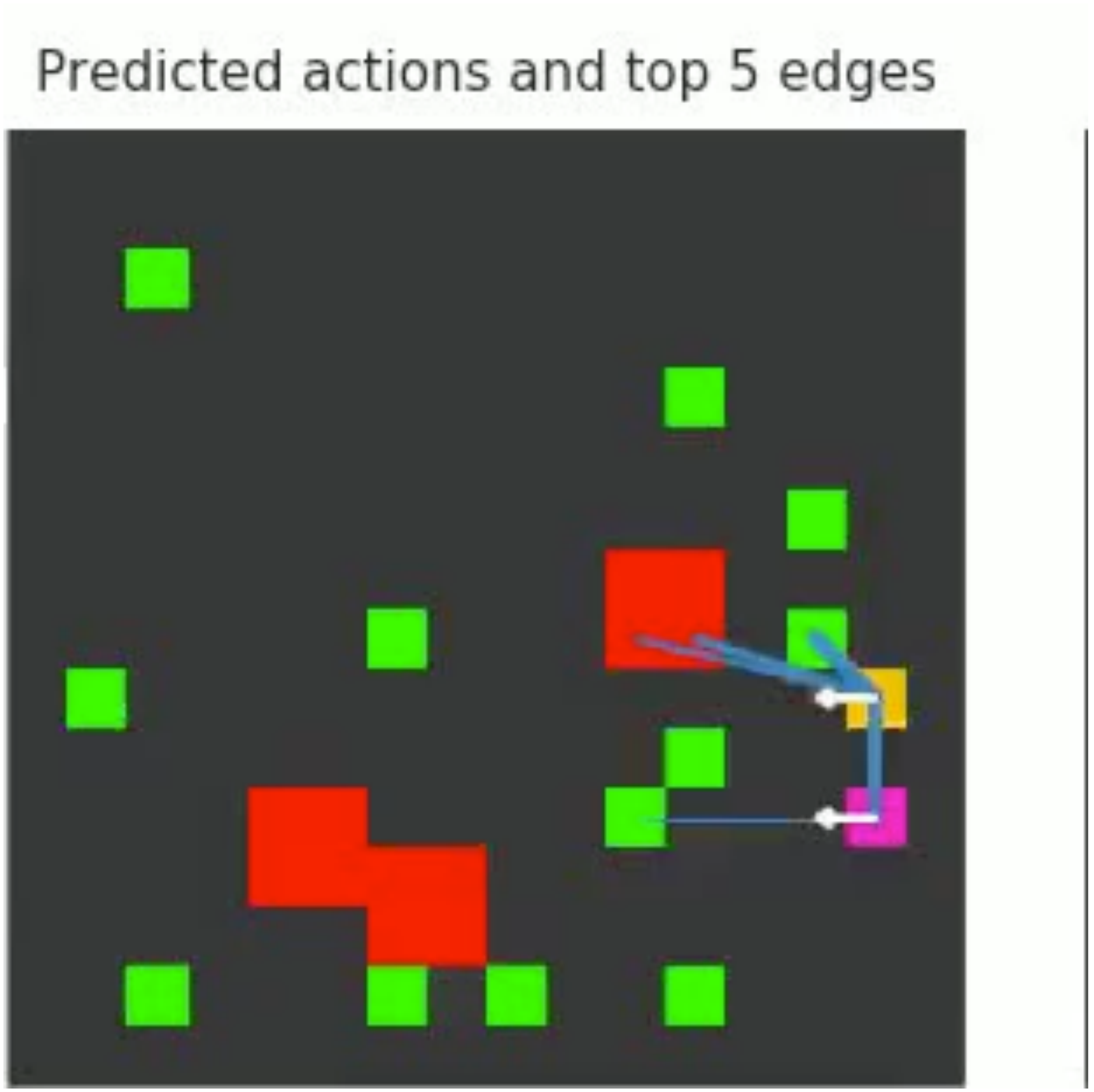
Graph networks as forward models of multi-agent RL systems

Trained model's predictive accuracy:
medium number of correctly predicted future steps (random ~= 0.1)

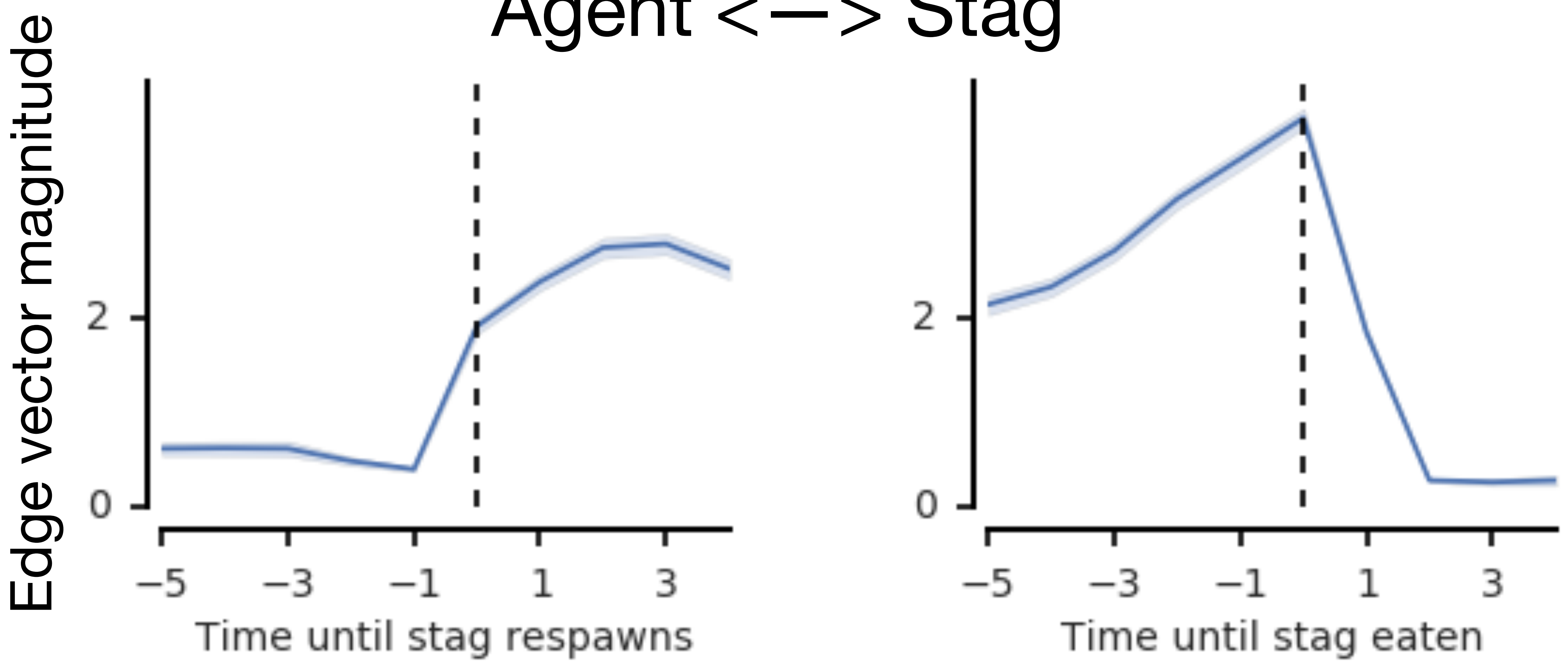


Tacchetti et al., 2019, ICLR

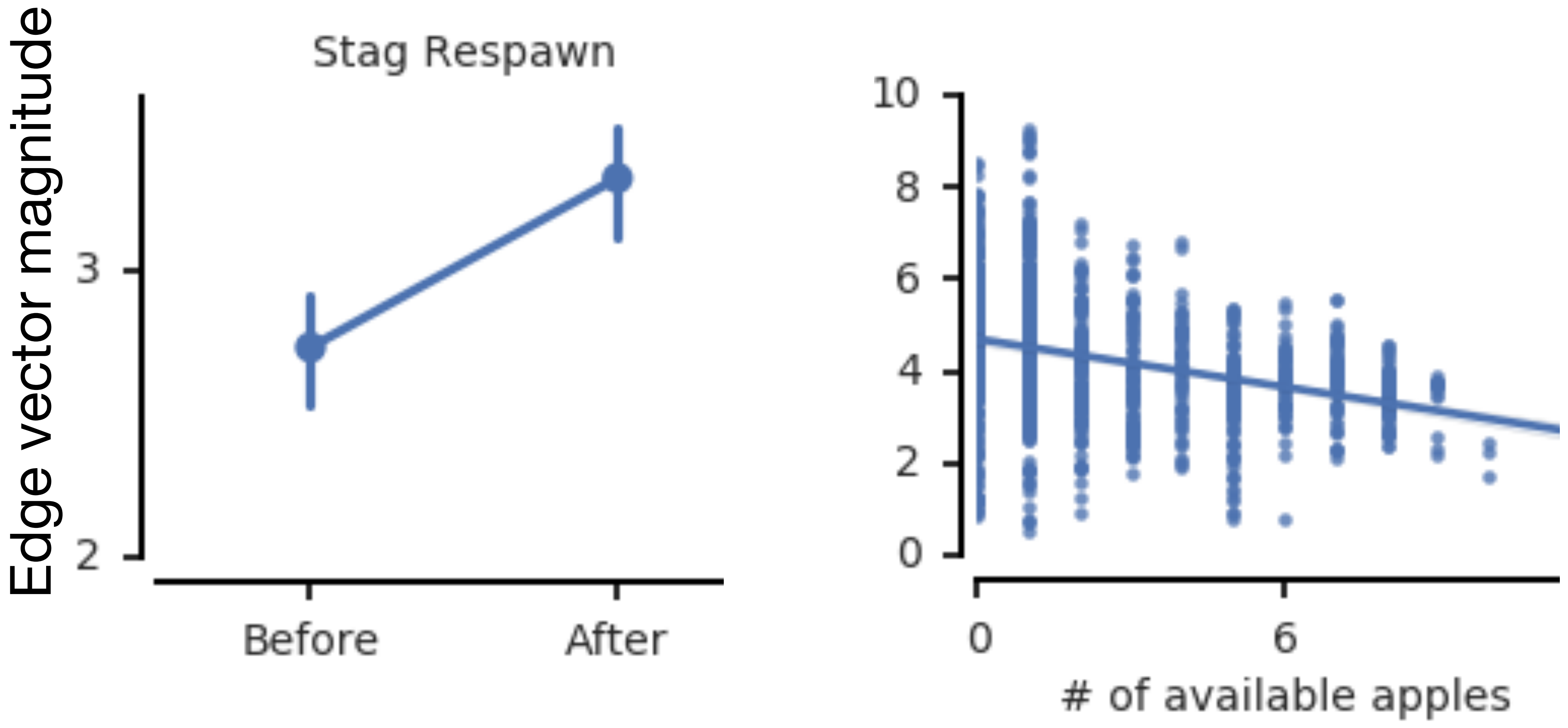
Interpretable learned representations



Agent <—> Stag



Agent <—> Other agent

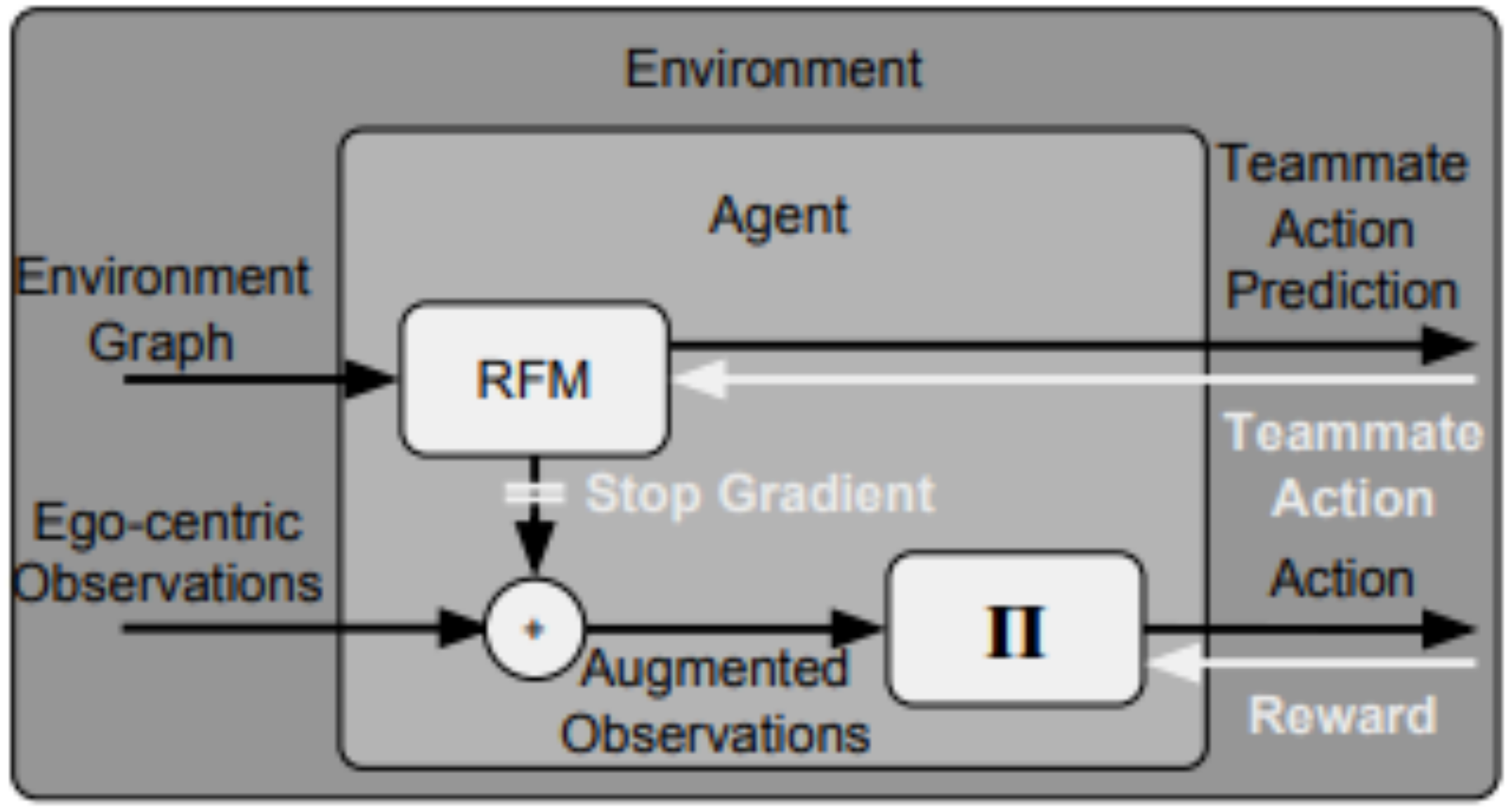


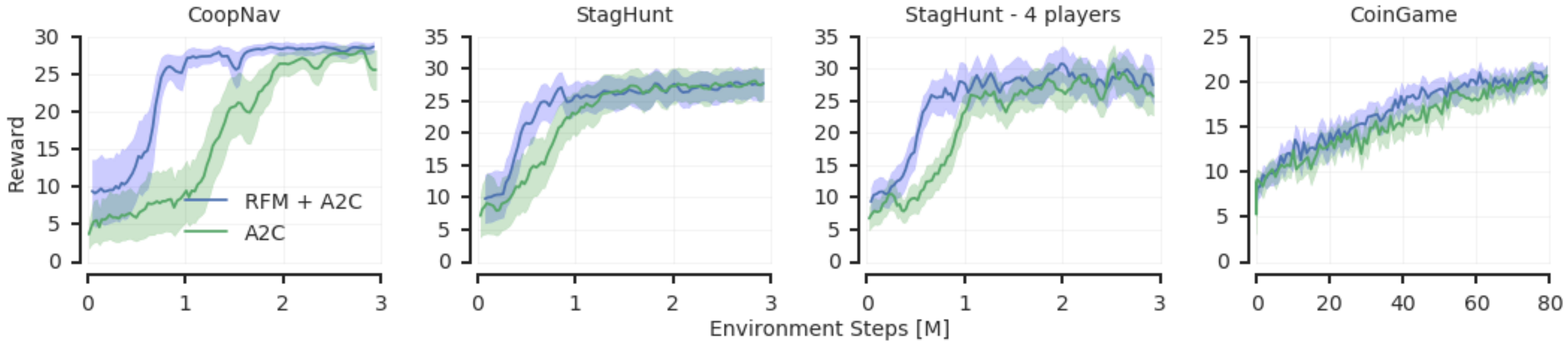
Tacchetti et al., 2019, ICLR

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, untrained agent, whose observations are augmented with the RFM's message magnitudes.

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

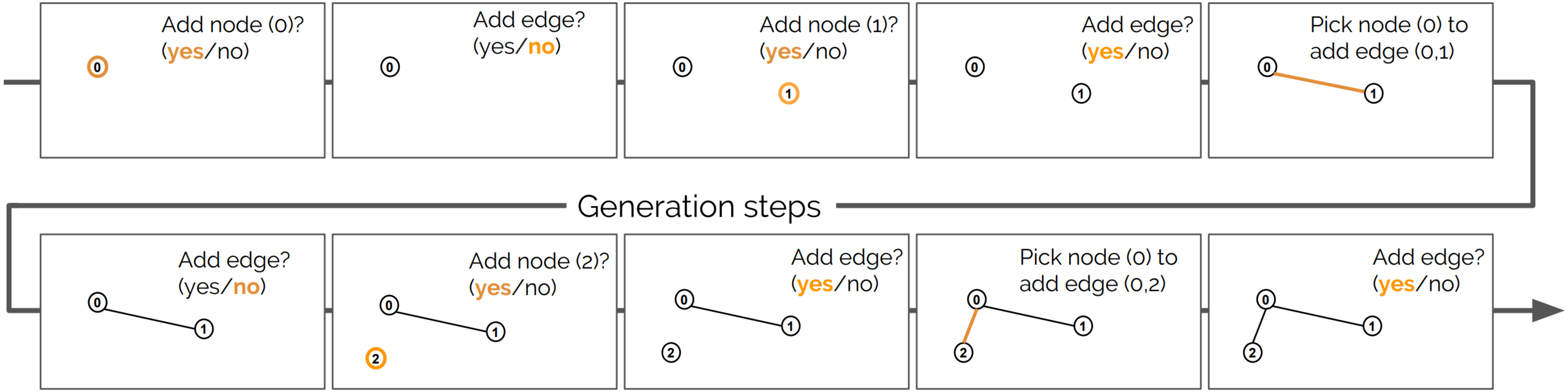




Tacchetti et al., 2019, ICLR

Learning deep generative models of chemical graphs

- Generative model defines joint distribution over graph-generating decisions (structure and order).
- Analogous to a decision tree, where decisions are selected by a GNN:
  1. *Add node?* If NO, terminate.
  2. If YES, *Add edge?* If NO, goto (1).
  3. If YES, *Pick node to add edge to.* Goto (2).
- Training optimizes the joint log-likelihood of structure and order, with Monte Carlo integration over permutations.



Li et al., 2018, arXiv

Learning deep generative models of chemical graphs

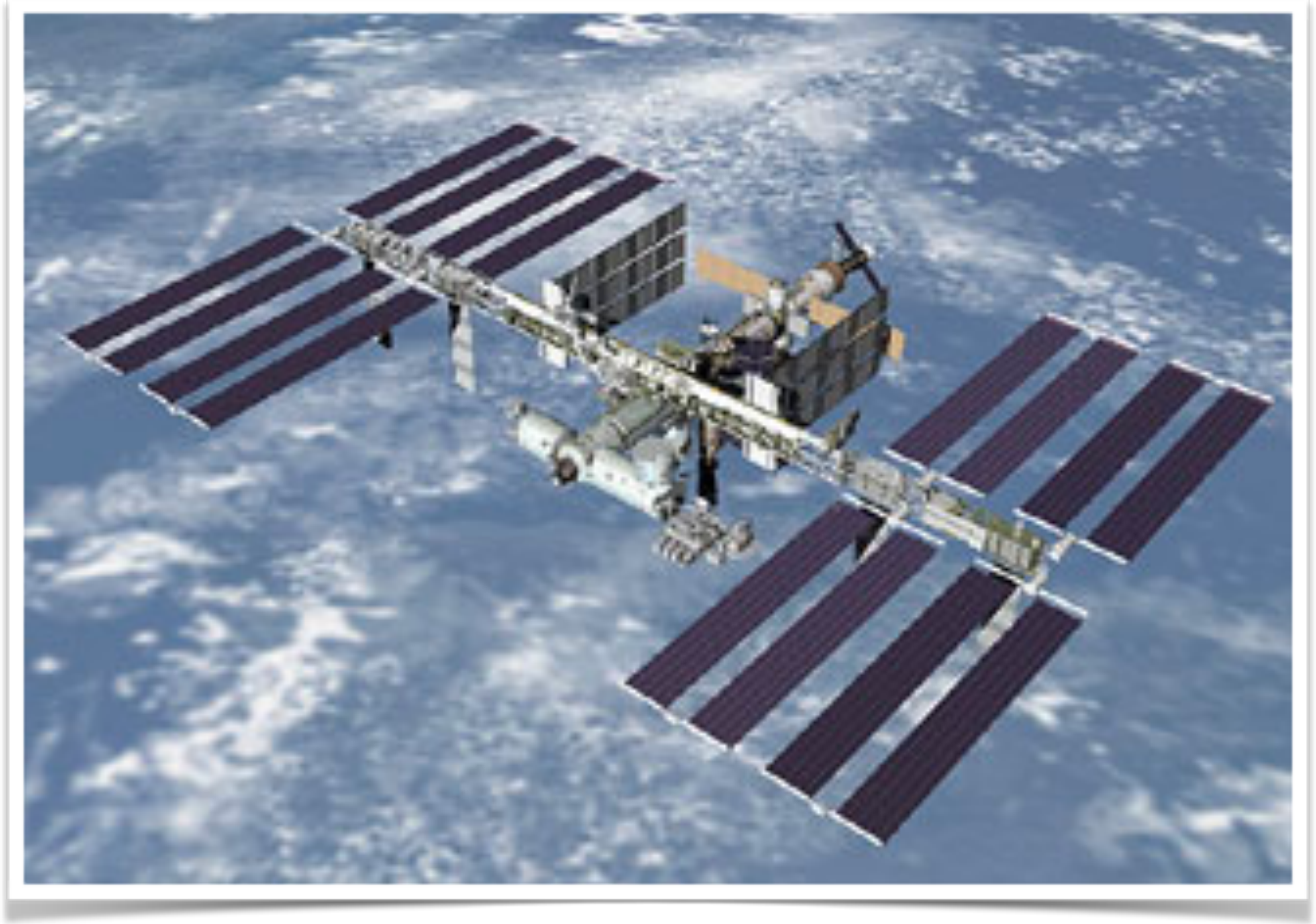
- GrammarVAE (Kusner et al., 2017) has qualitatively poorer samples from the prior.

- Our model learns a more accurate model than LSTMs, and can generate more novel molecules.

Arch	Grammar	Ordering	N	NLL	%valid	%novel
LSTM	Graph	Fixed	1	22.06	85.16	80.14
LSTM	Graph	Random	$O(n!)$	63.25	91.44	91.26
Graph	Graph	Fixed	1	**20.55**	**97.52**	90.01
Graph	Graph	Random	$O(n!)$	58.36	95.98	**95.54**

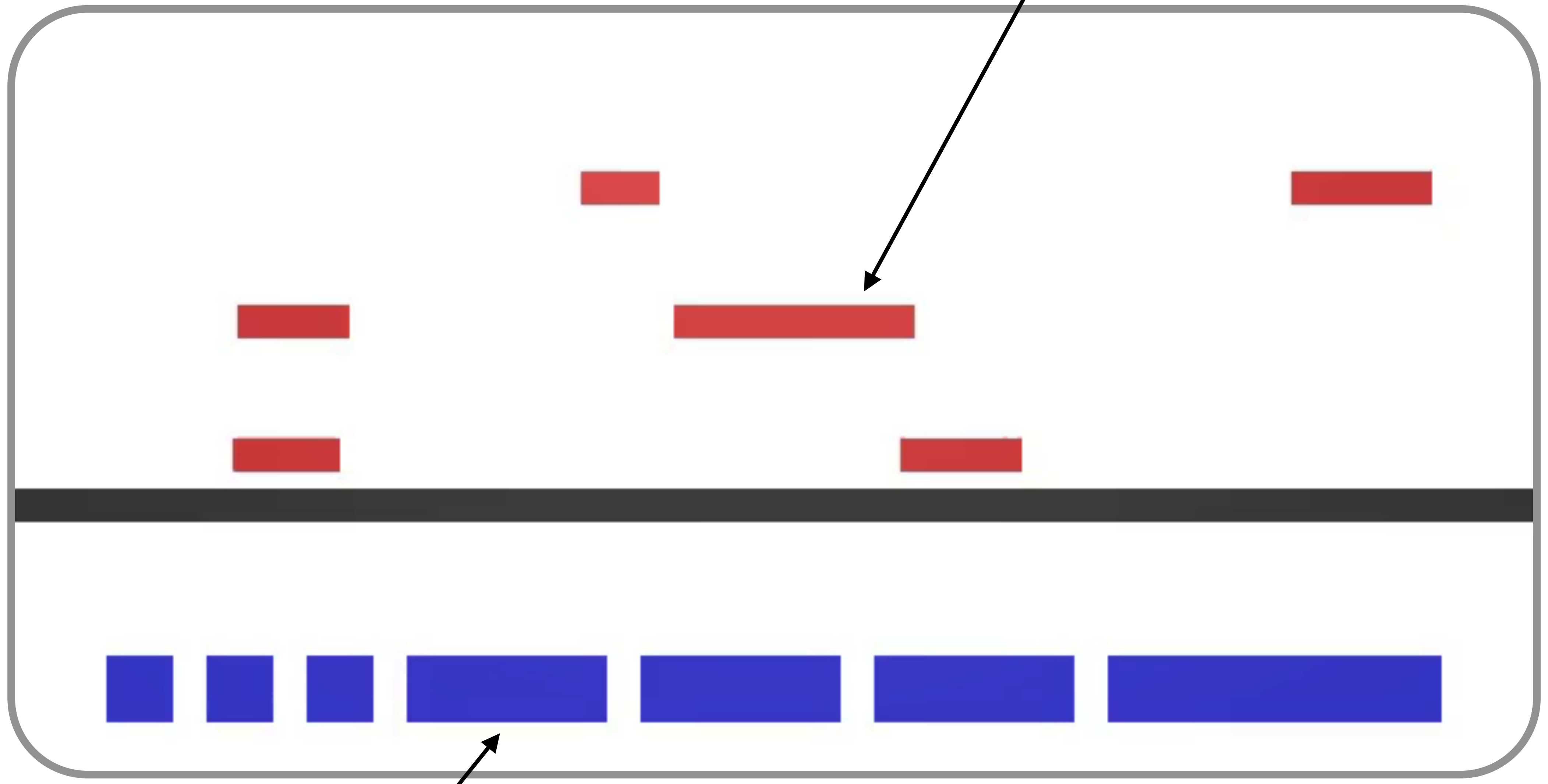
Li et al., 2018, arXiv

Humans are a “construction species”



Structured agents for physical construction

Avoid touching **obstacles**

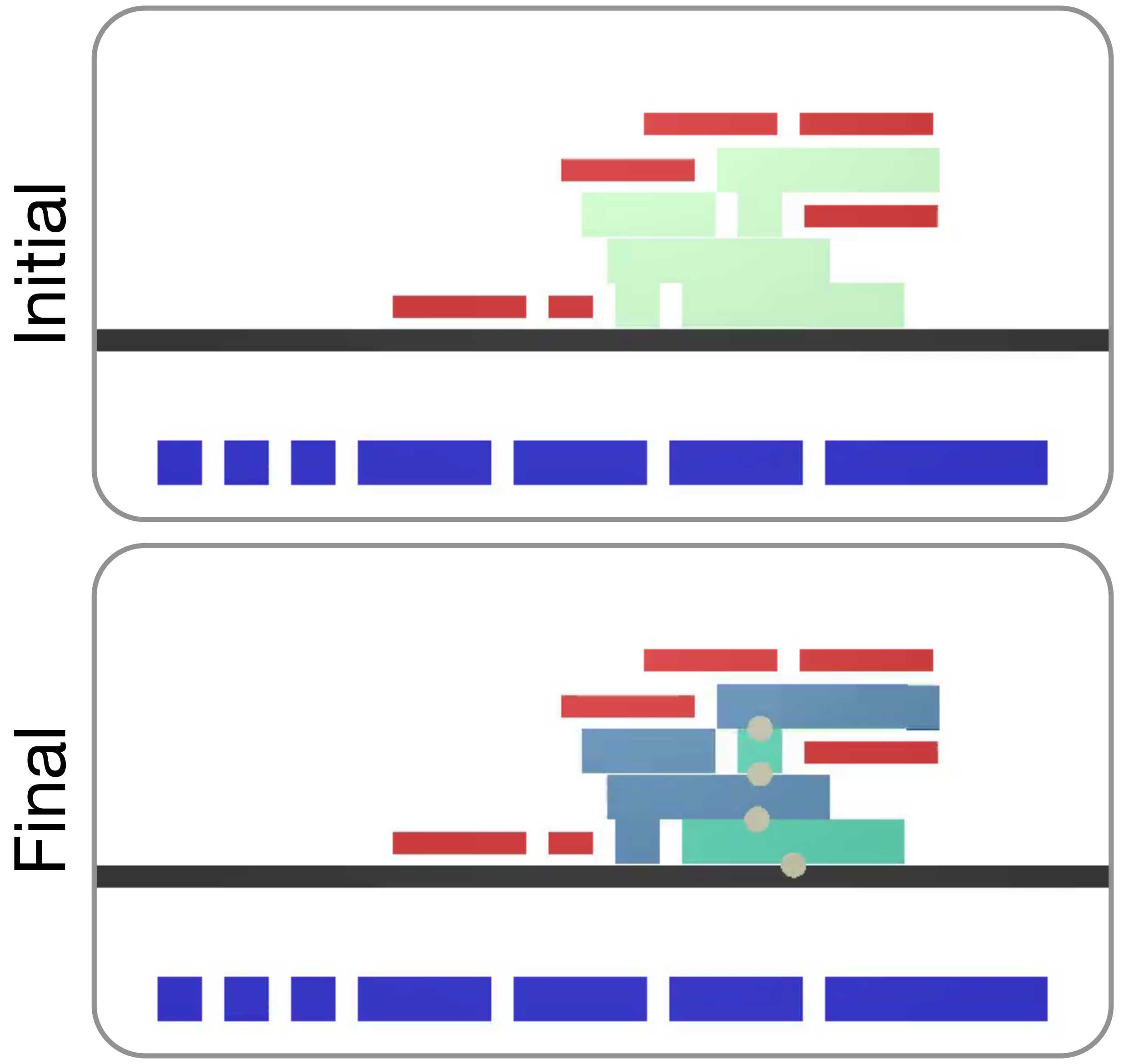


Pick up **blocks** and place them in the scene
(and optionally make them **sticky**)

Bapst et al., 2019, ICML

Structured agents for physical construction

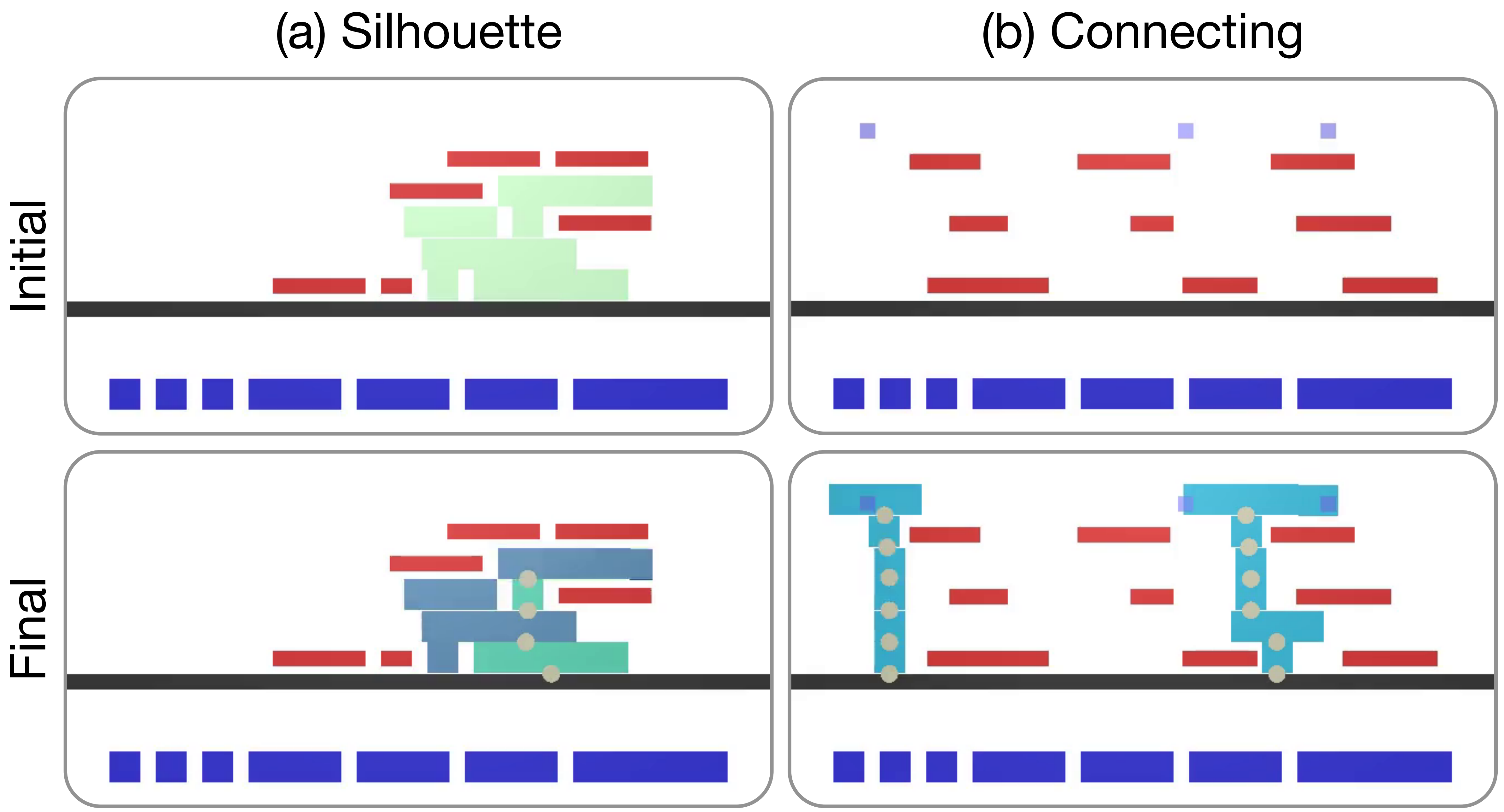
(a) Silhouette



+1 per target
-0.5 per sticky block

Bapst et al., 2019, ICML

Structured agents for physical construction

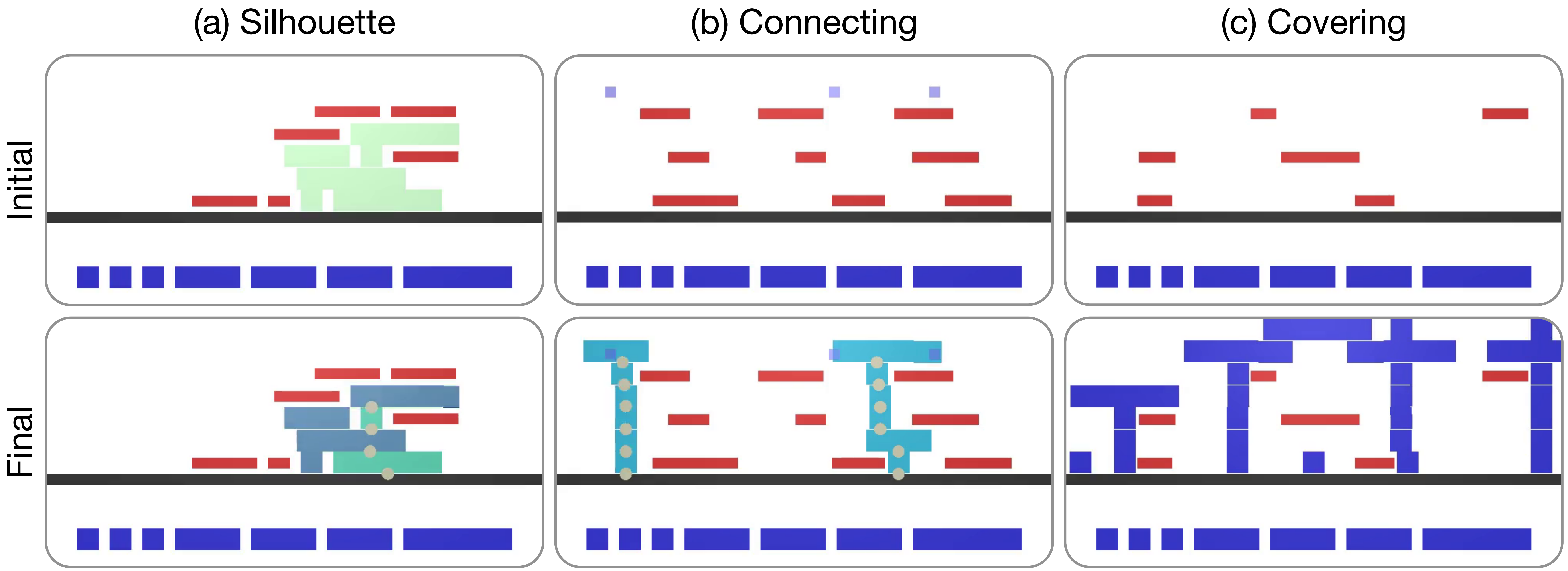


+1 per target
-0.5 per sticky block

+1 per target
Free sticky blocks

Bapst et al., 2019, ICML

Structured agents for physical construction



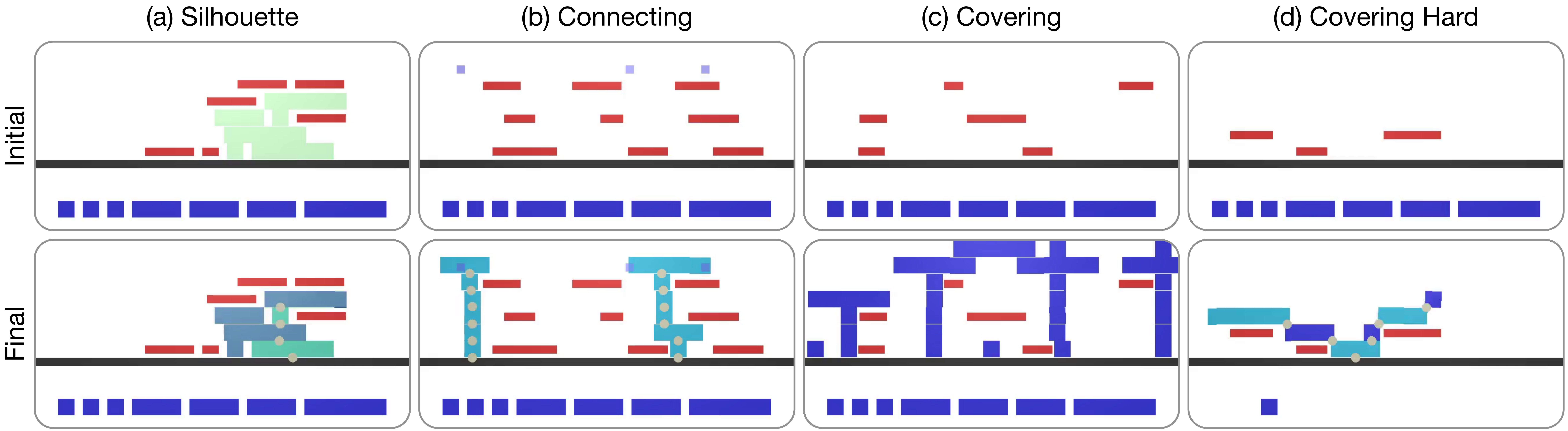
+1 per target
-0.5 per sticky block

+1 per target
Free sticky blocks

Length covered
-2 per sticky block

Bapst et al., 2019, ICML

Structured agents for physical construction

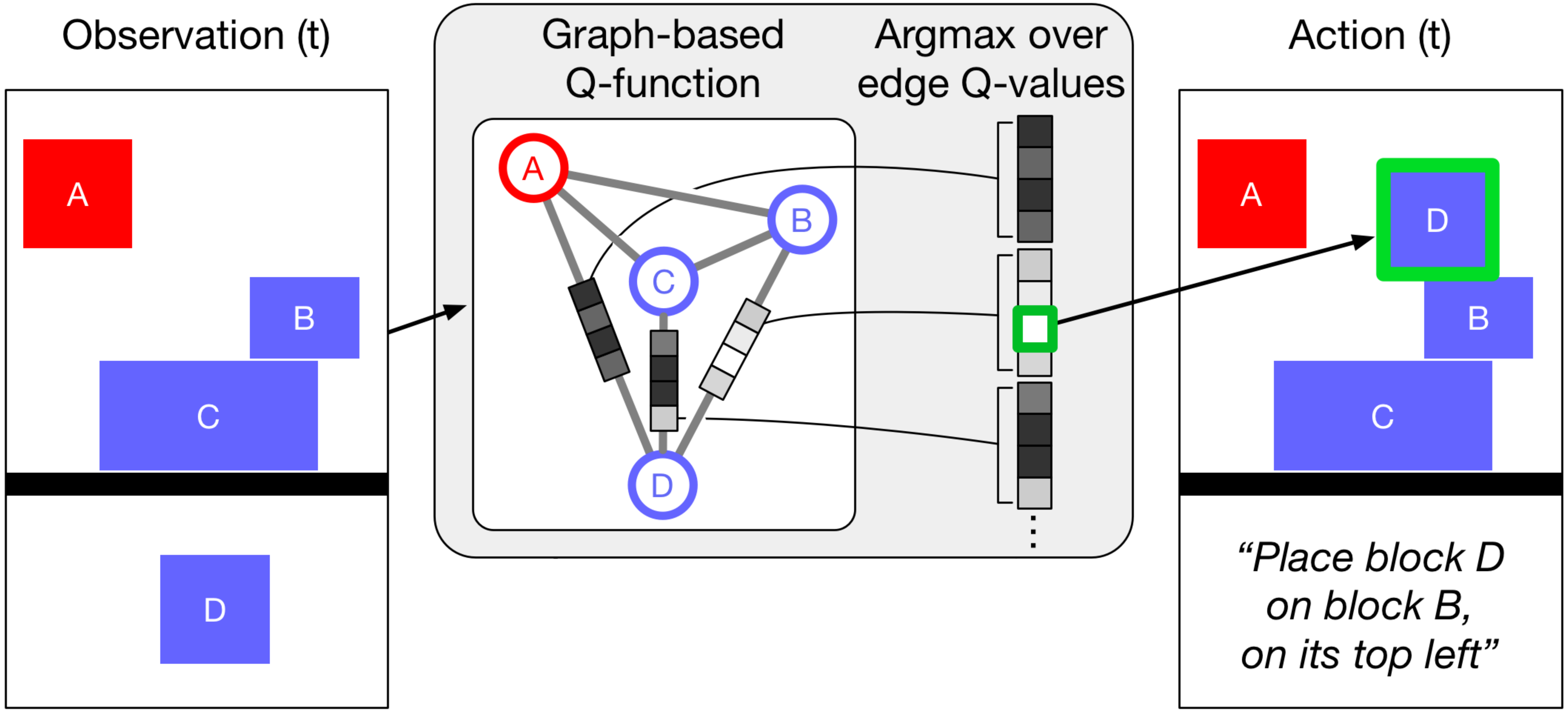


(a) Silhouette	(b) Connecting	(c) Covering	(d) Covering Hard
+1 per target *-0.5 per sticky block*	*+1 per target* *Free sticky blocks*	*Length covered* *-2 per sticky block*	*Length covered* *-0.5 per sticky block*

Bapst et al., 2019, ICML

Graph net-based agent: model-free

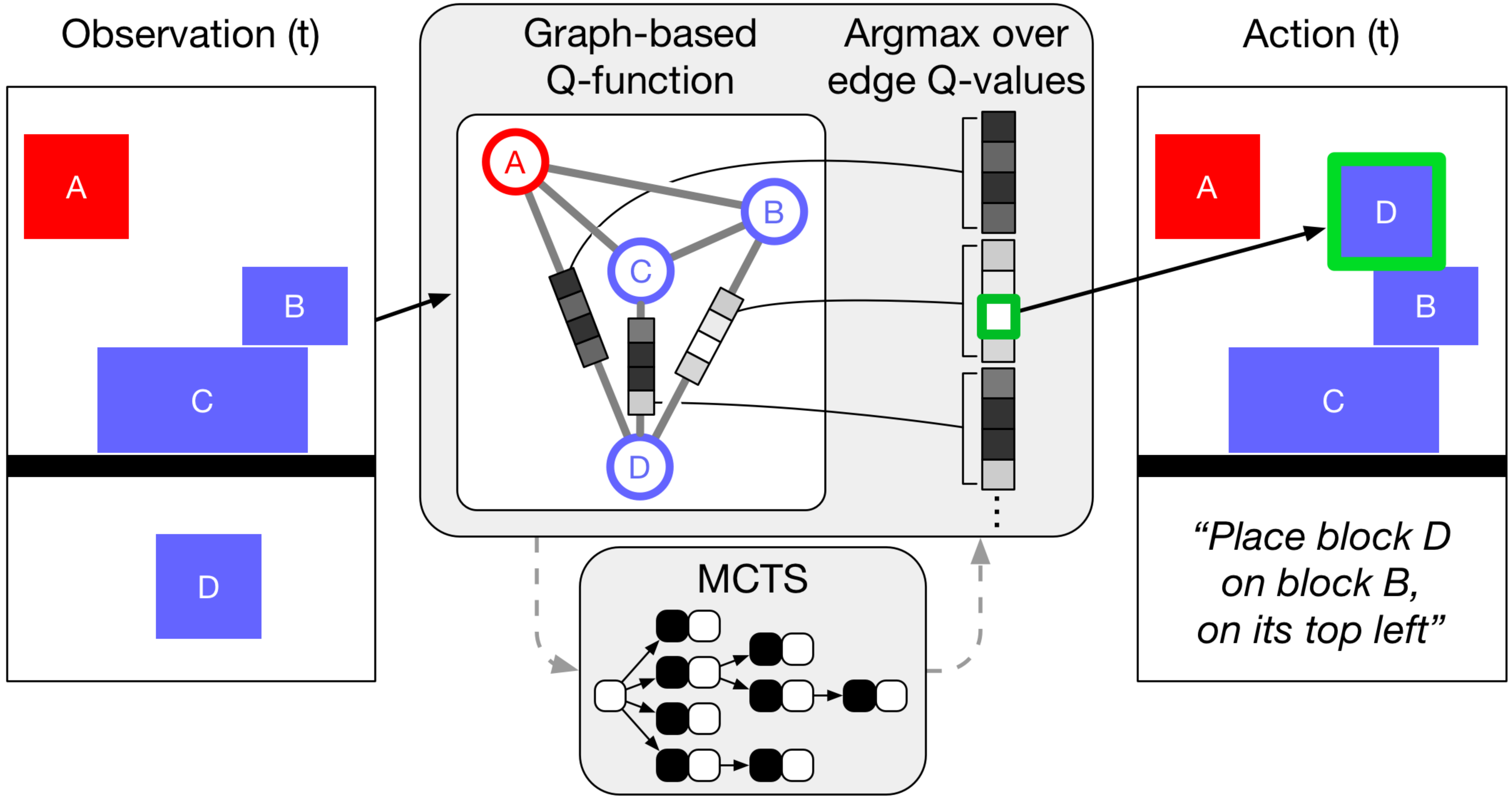
Can be thought of as “graph building” agent



Bapst et al., 2019, ICML

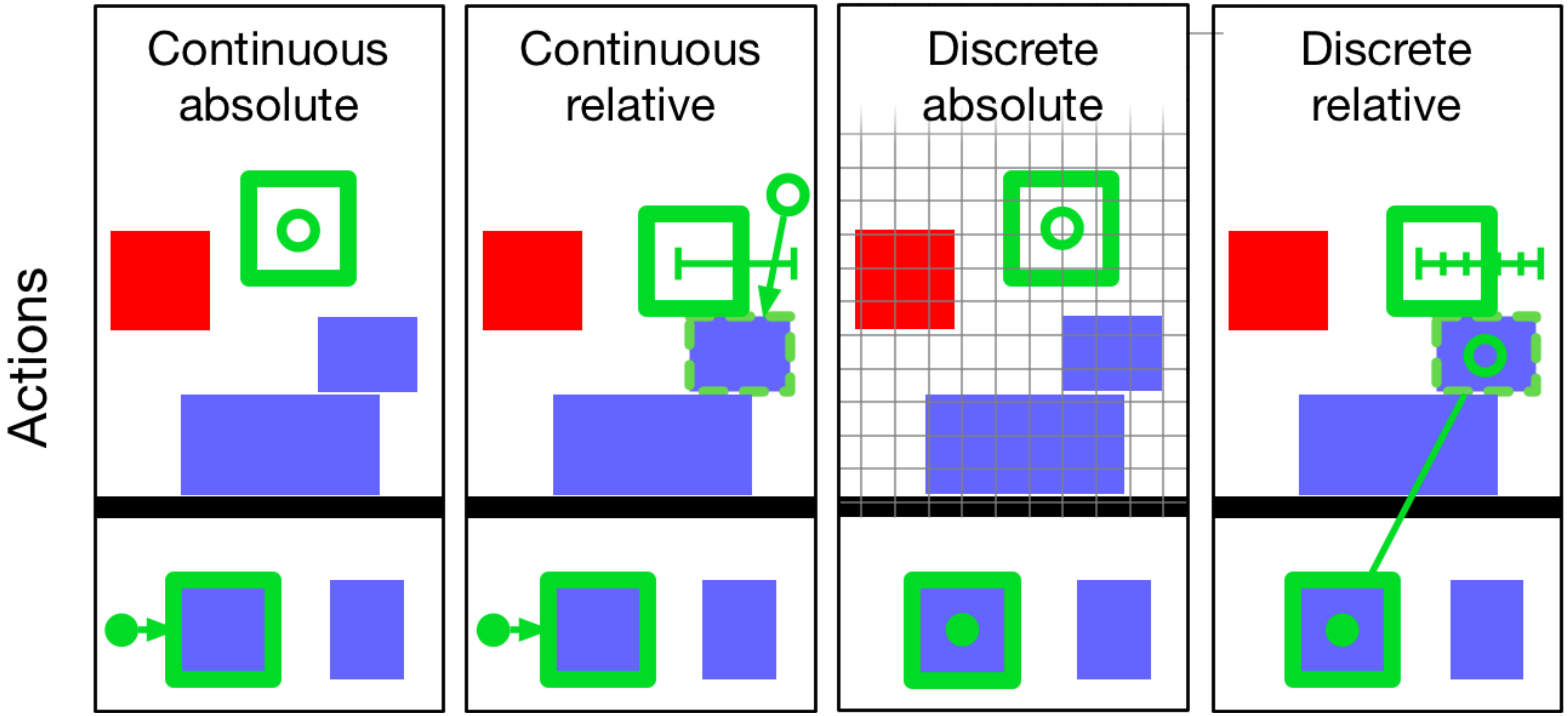
Graph net-based agent: model-based

Can be thought of as “graph building” agent



Bapst et al., 2019, ICML

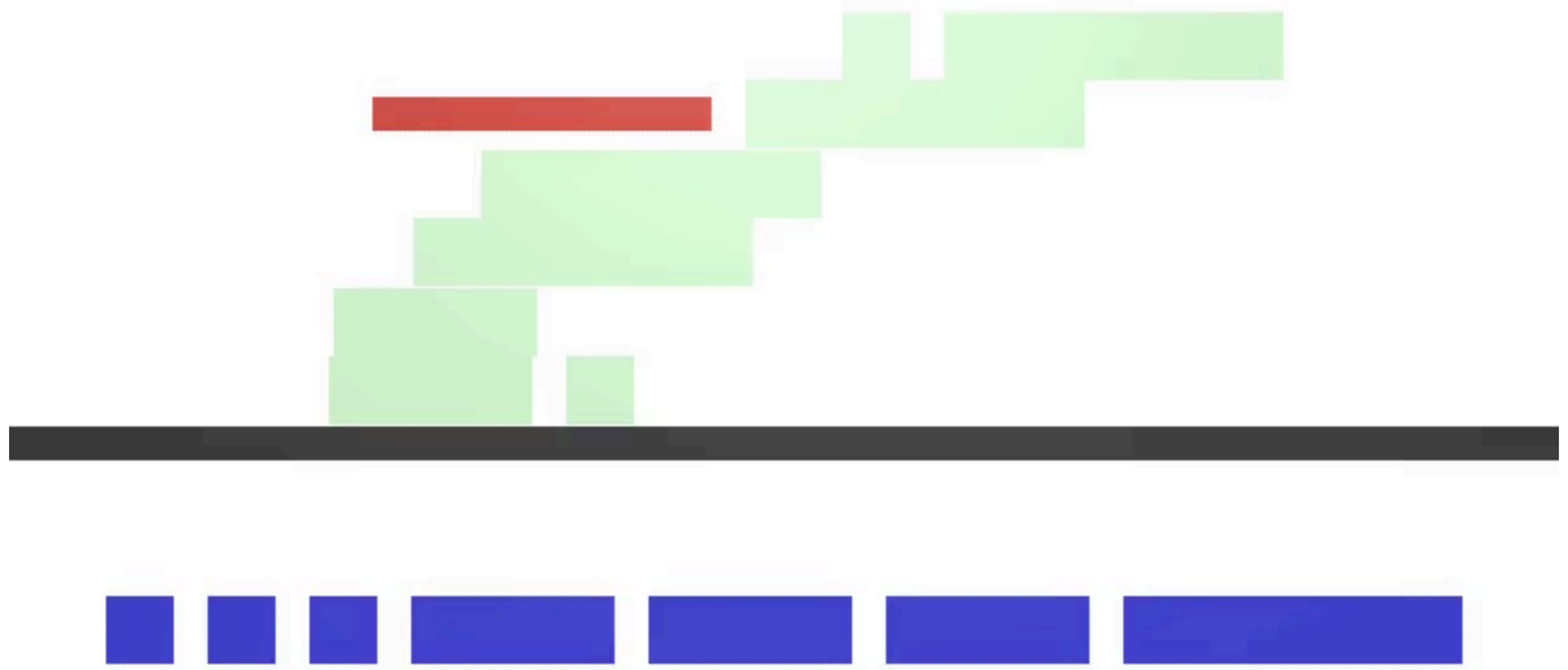
Absolute vs relative actions



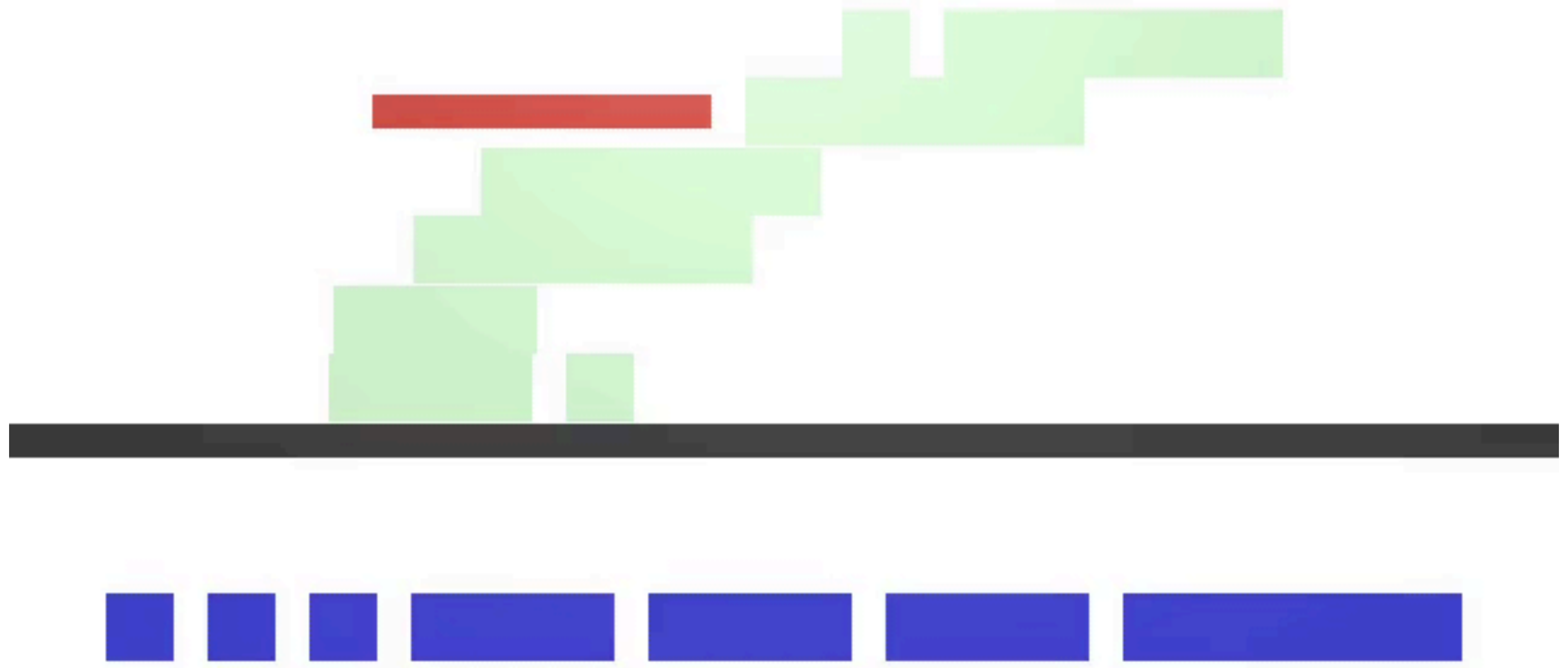
Bapst et al., 2019, ICML

Results: “Silhouette” task

Absolute Actions



Object-Centric Actions

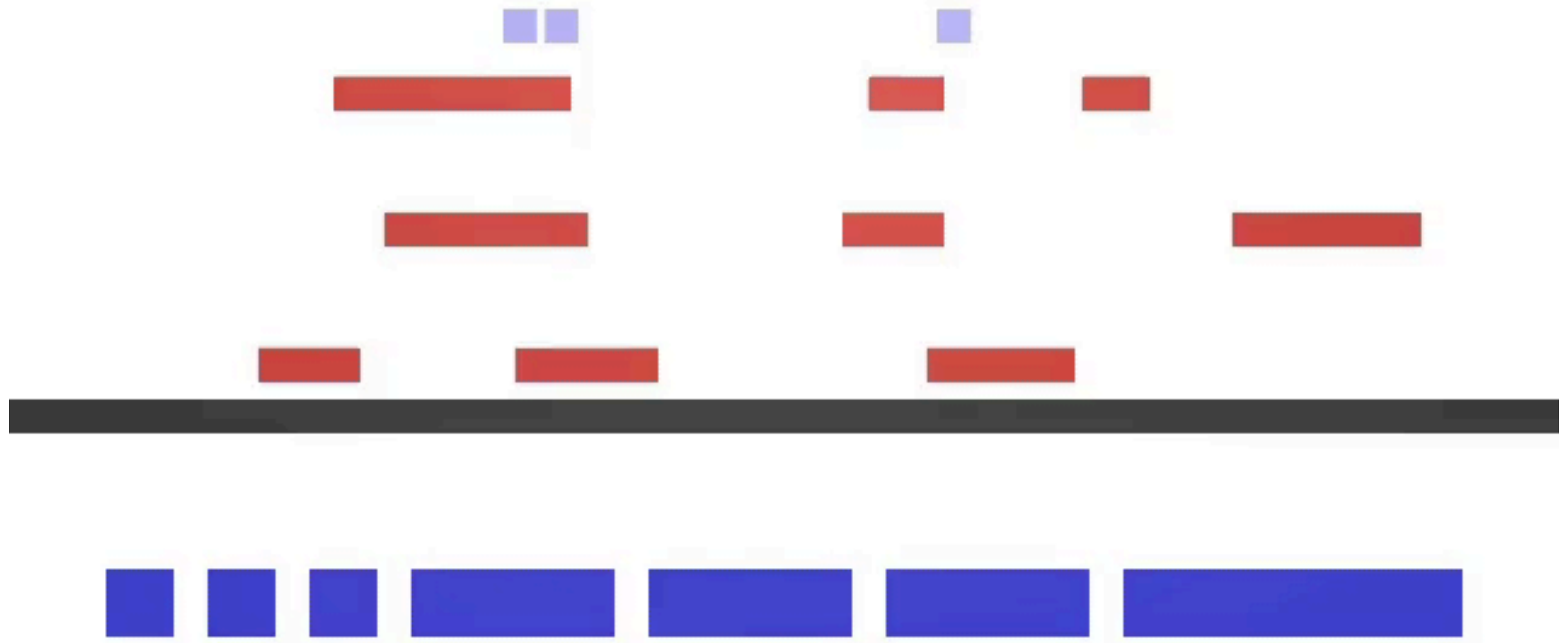


Reward: +1 per target, -0.5 per sticky block

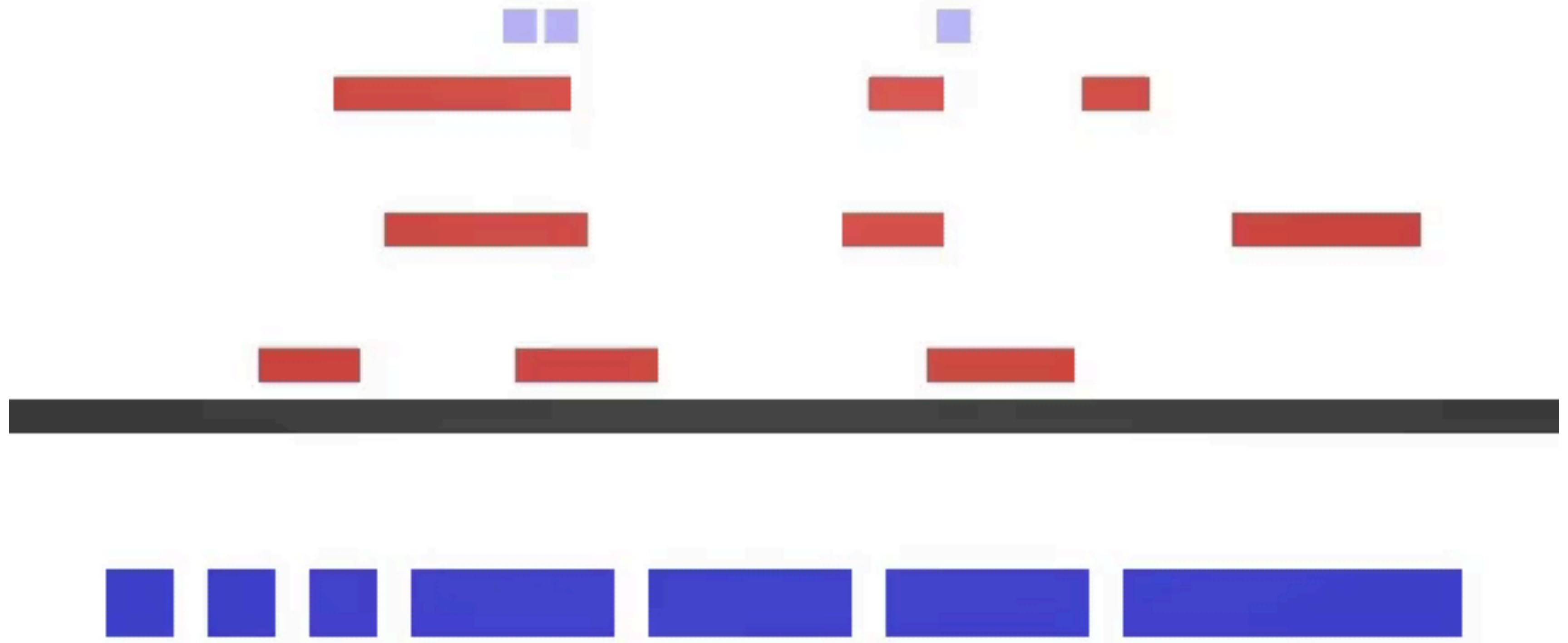
Bapst et al., 2019, ICML

Results: “Connecting” task

Absolute Actions



Object-Centric Actions

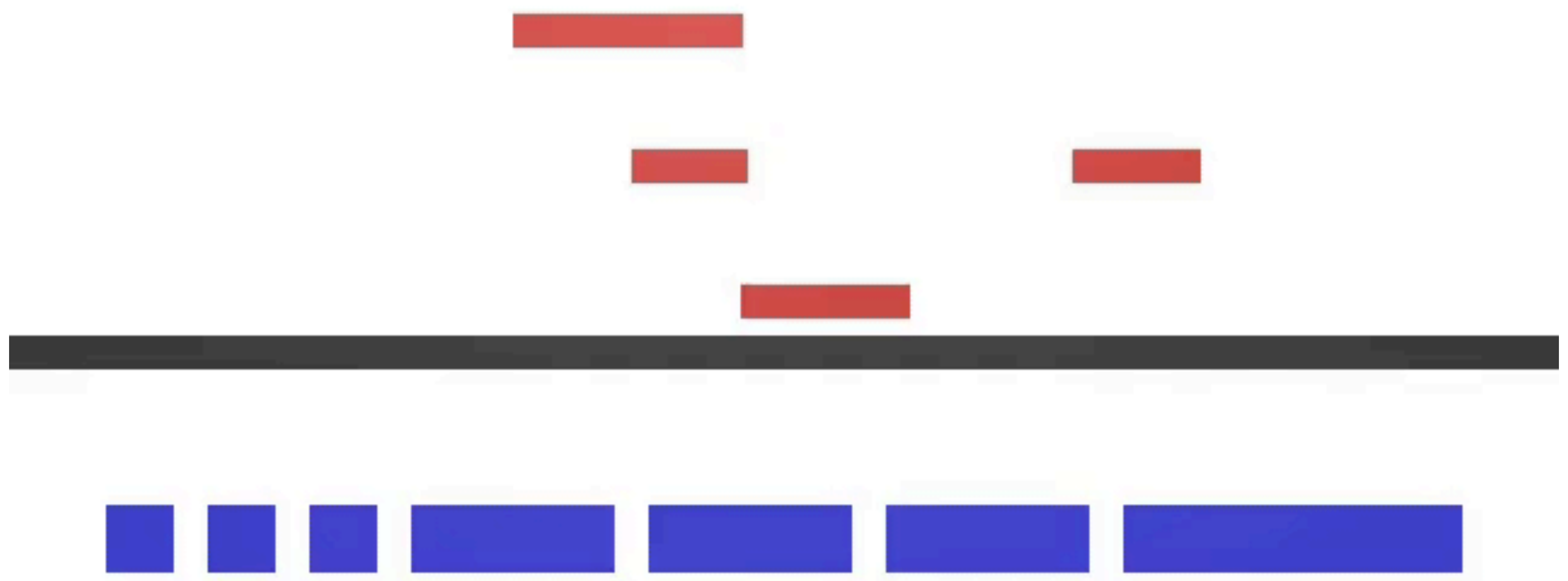


Reward: +1 per target, free sticky blocks

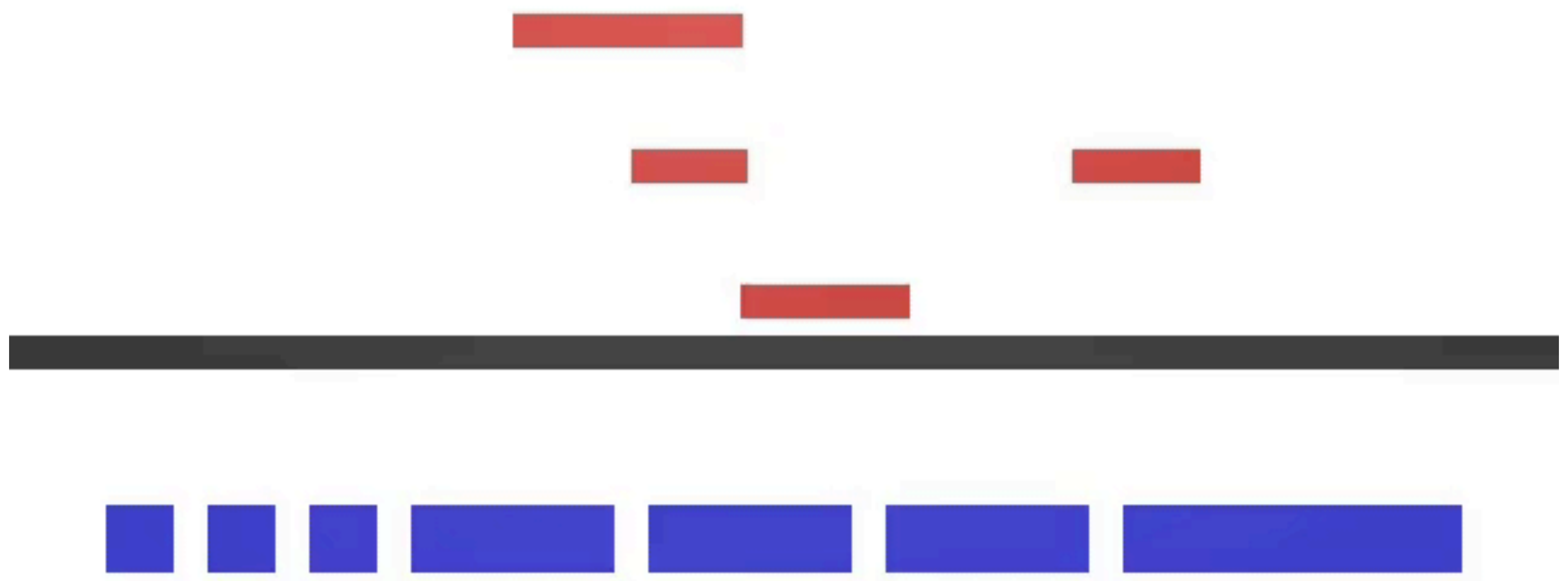
Bapst et al., 2019, ICML

Results: “Covering” task

Absolute Actions



Object-Centric Actions

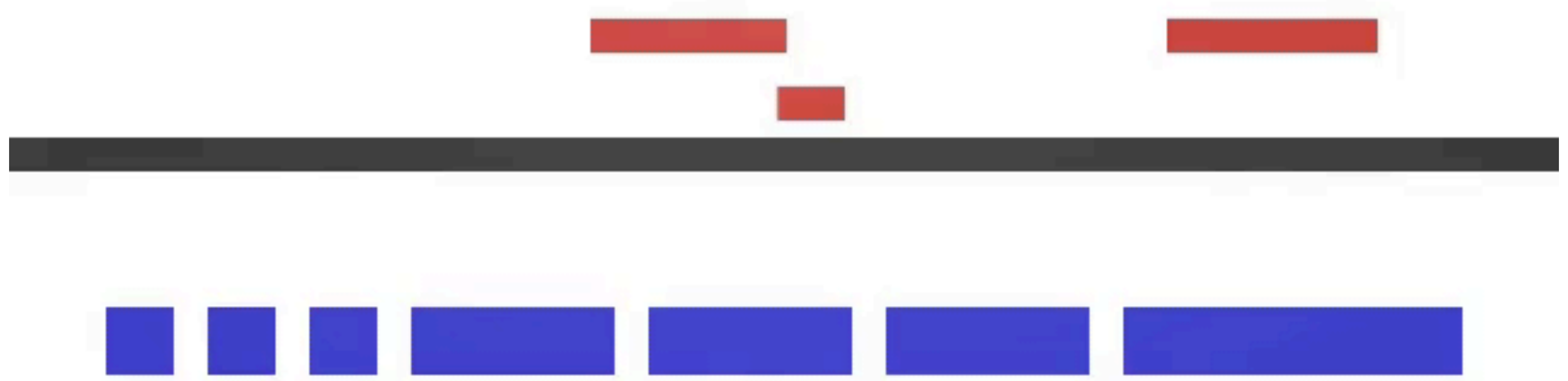


Reward: proportional to length covered, -2 per sticky block

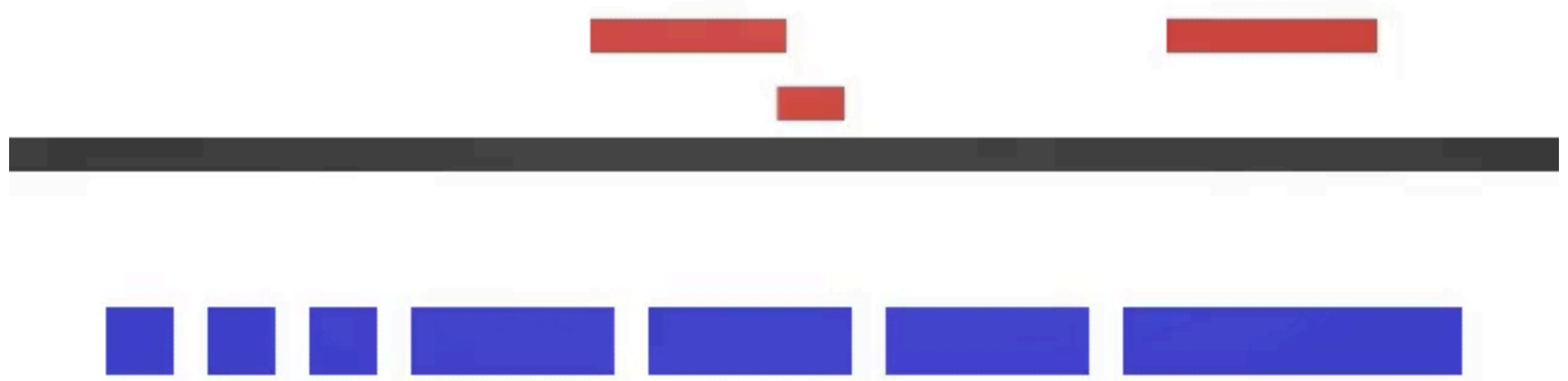
Bapst et al., 2019, ICML

Results: “Covering hard” task

Absolute Actions



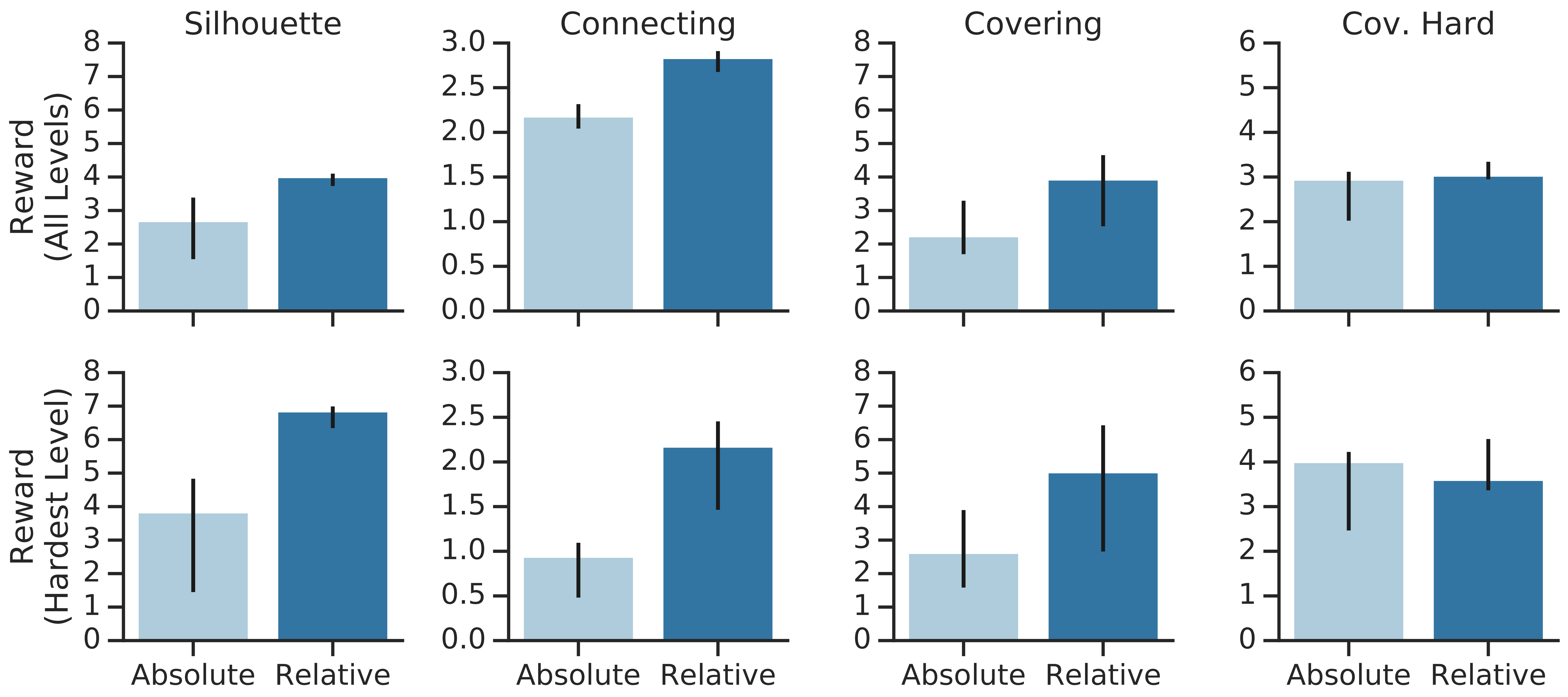
Object-Centric Actions



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

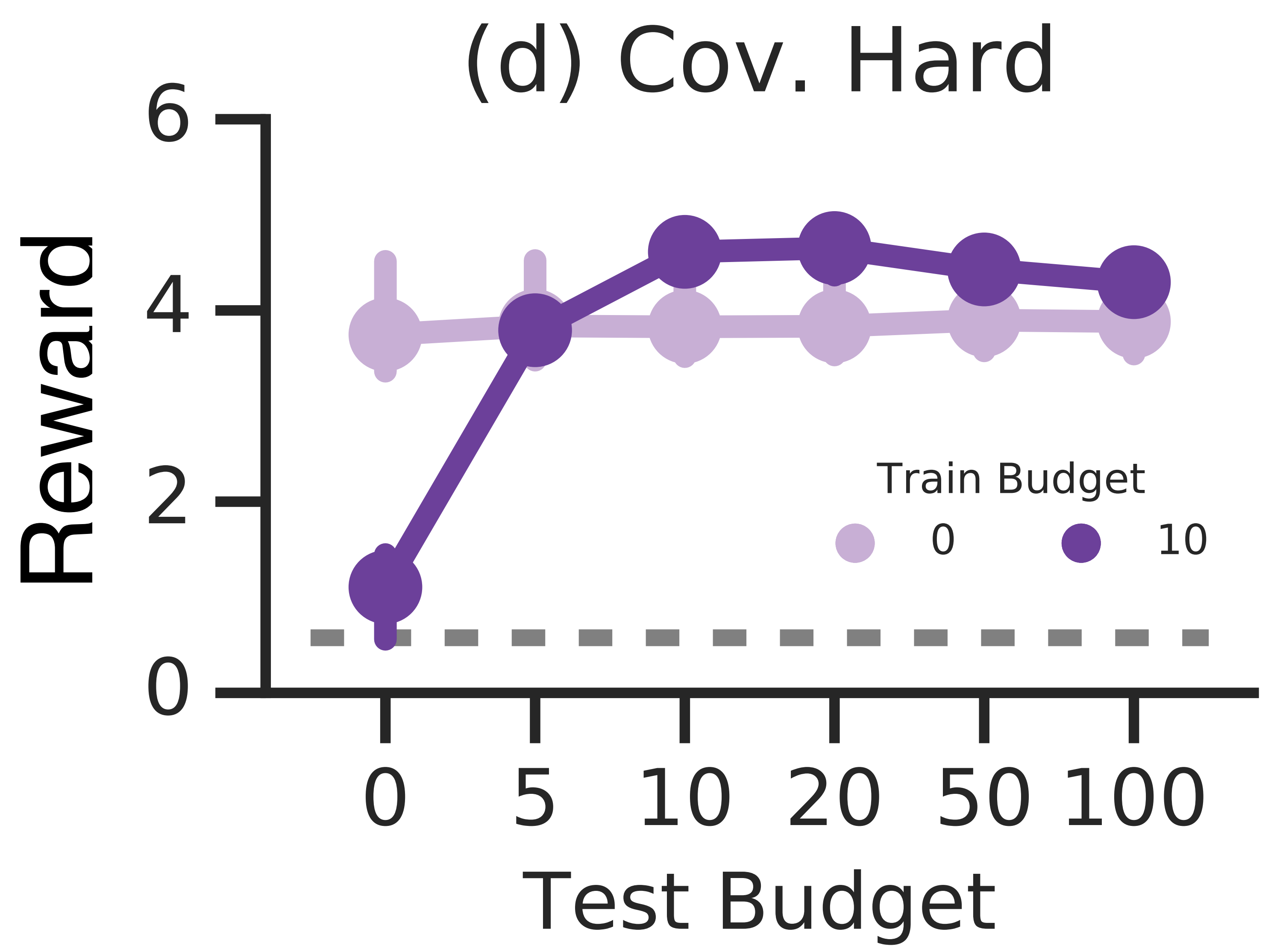
Bapst et al., 2019, ICML

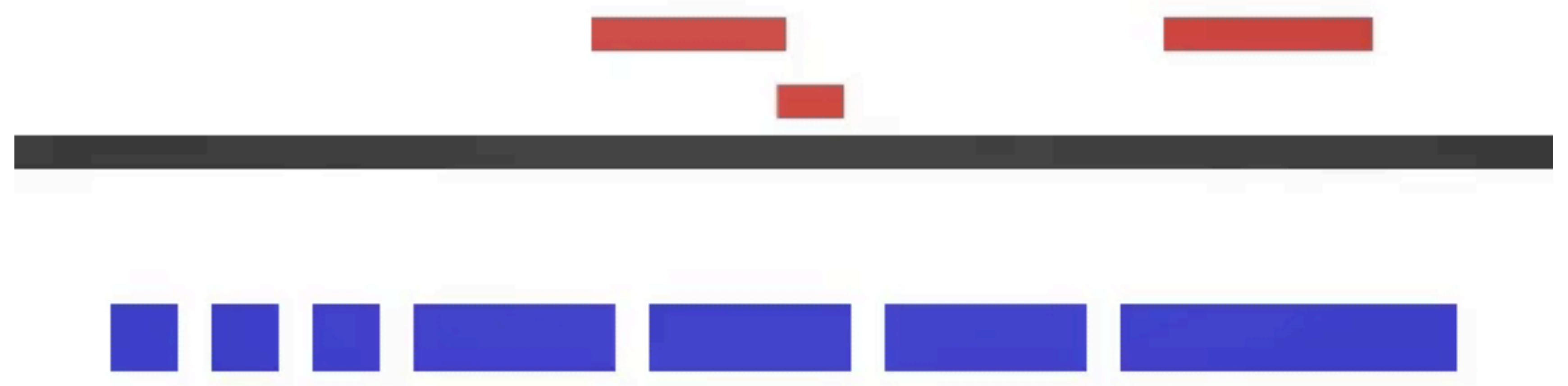
Results: Absolute vs relative actions



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

```
# 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:

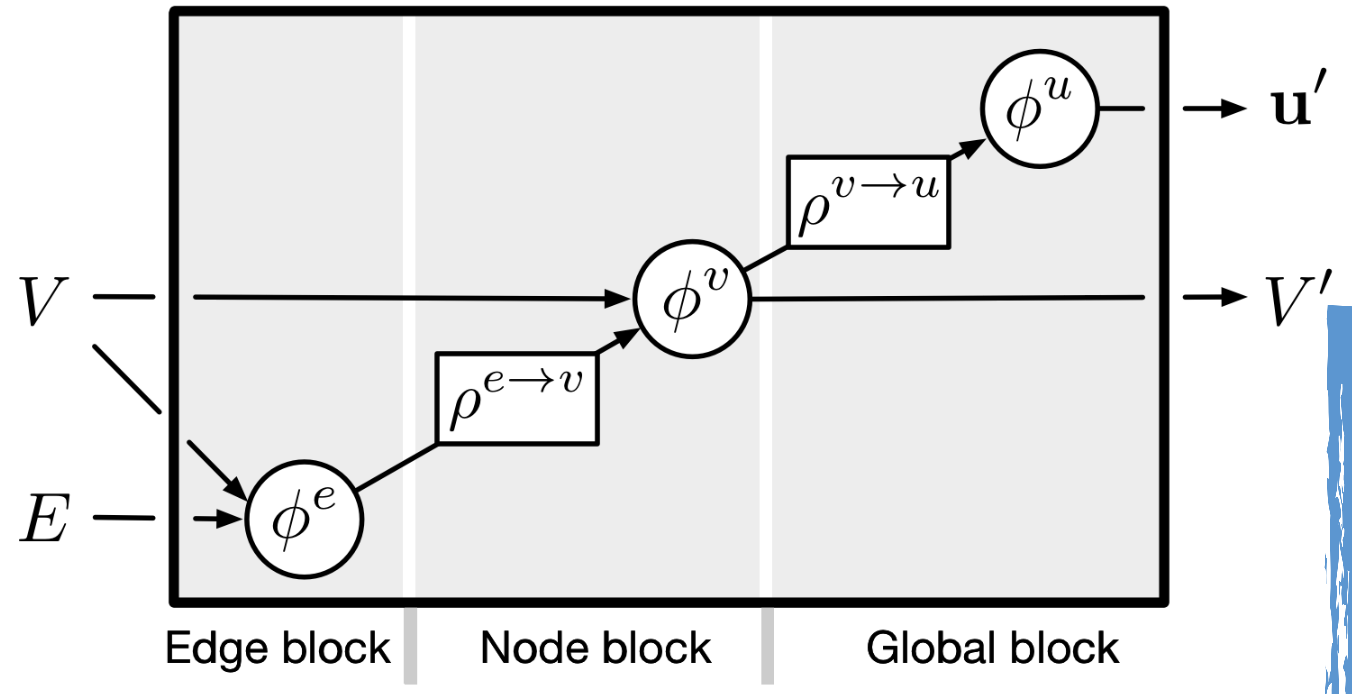
- pytorch_geometric: github.com/rusty1s/pytorch_geometric (for a GN analog, see `MetaLayer`)
- Deep Graph Library: github.com/dmlc/dgl

Build Graph Nets in Tensorflow

github.com/deepmind/graph_nets

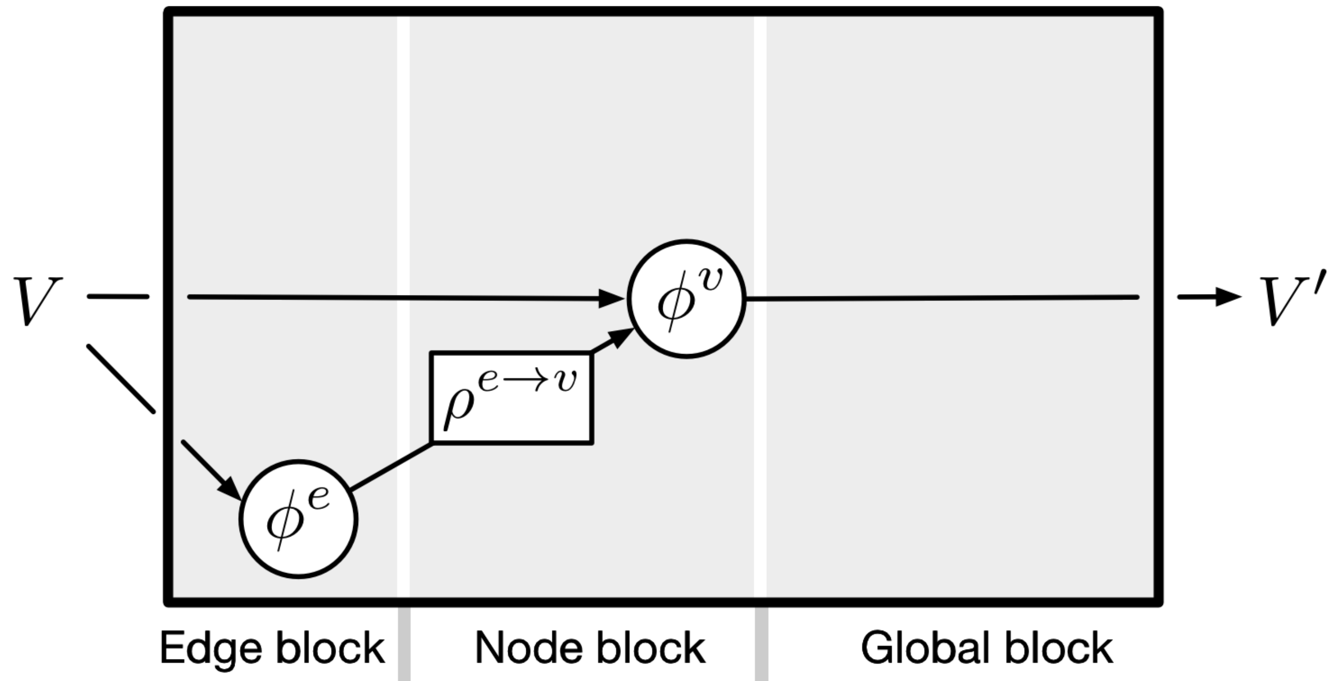
Message-Passing NN (eg. Interaction Net)

Gilmer et al. 2017

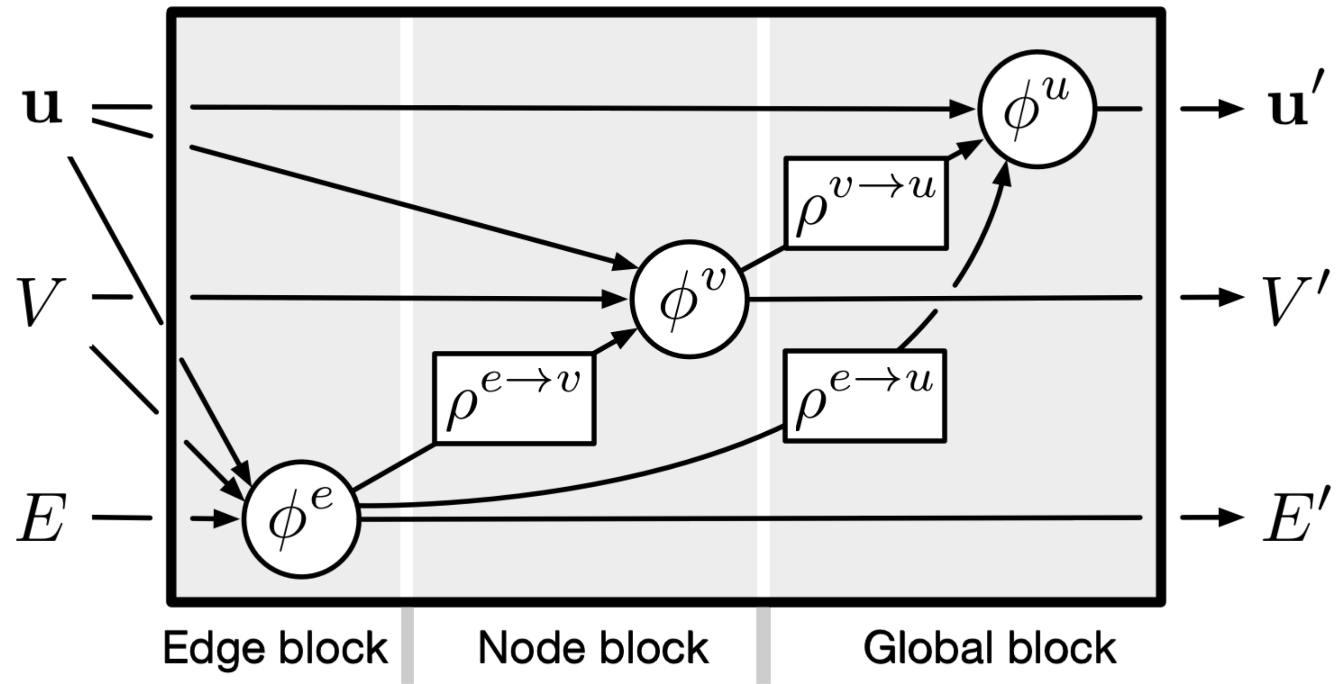


Non-Local NN (eg. Transformer)

Vaswani et al. 2017; Wang et al. 2017

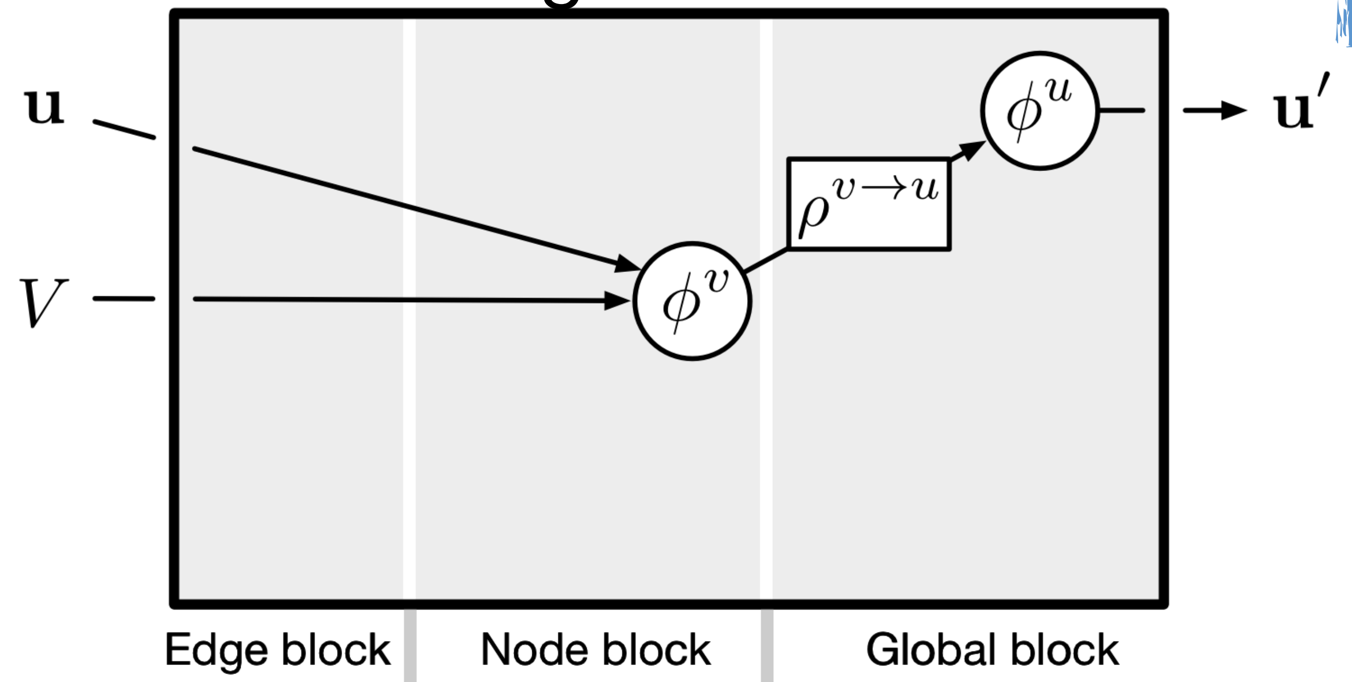


Graph Network
(a type of Graph Neural Network)



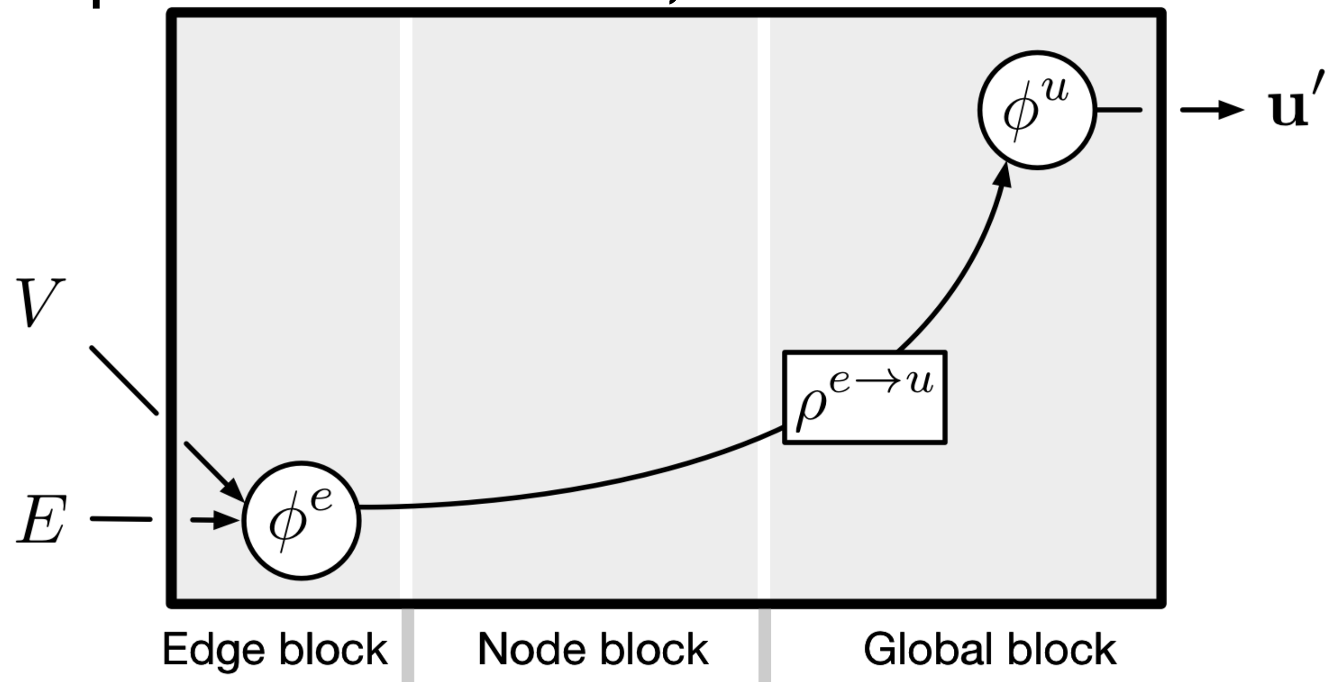
Deep Sets

Zhang et al. 2017



Relation Network

Raposo et al. 2017; Santoro et al. 2017



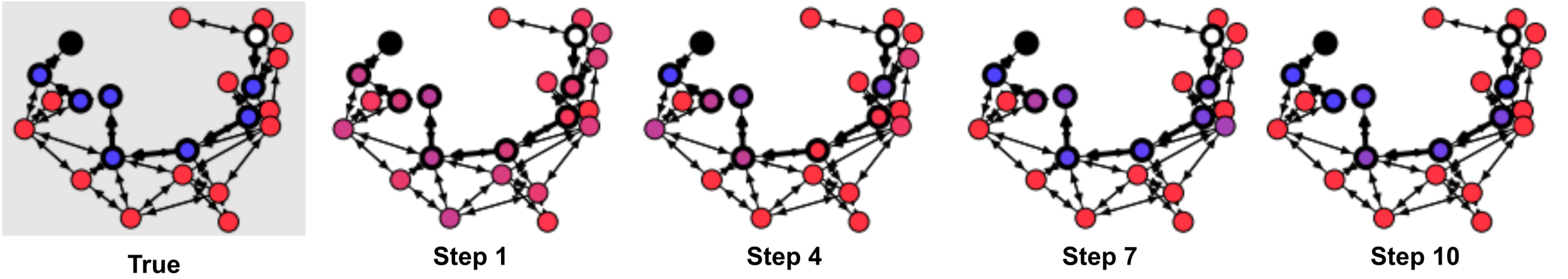
Build Graph Nets in Tensorflow

github.com/deepmind/graph_nets

IPython Notebook demos
(All use same architecture)

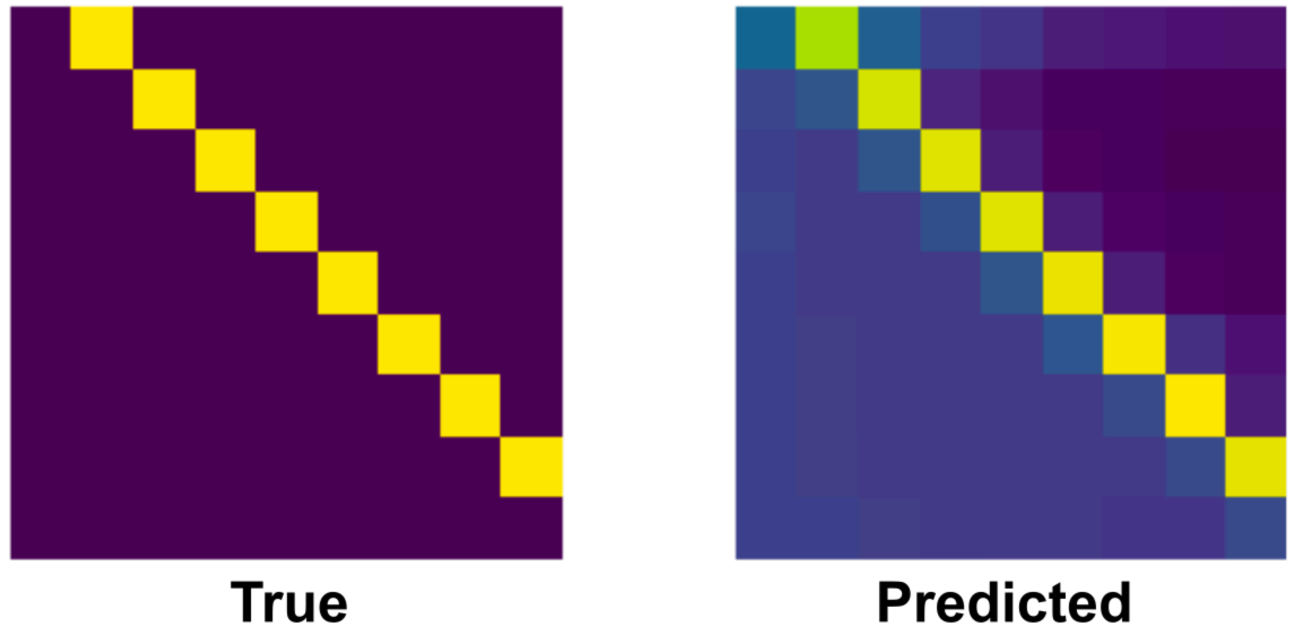
Shortest path:

Shortest path: predictions at each message-passing step



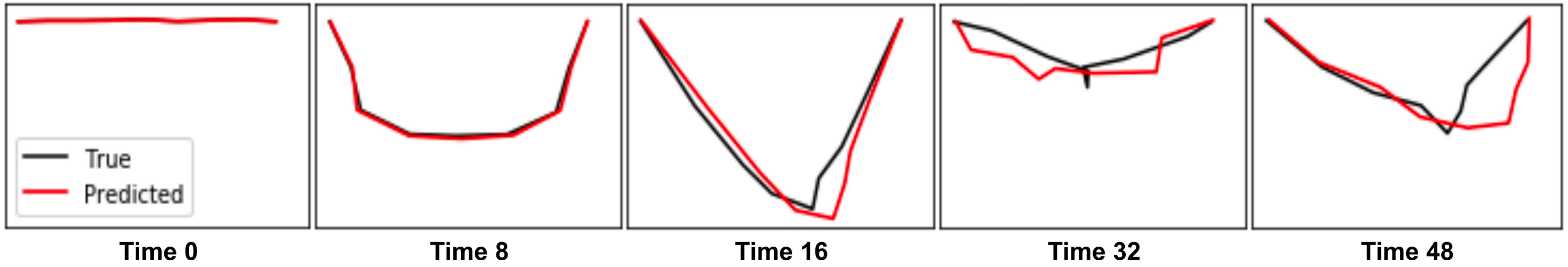
Sorting:

Sort: item-to-item connections



Predicting physics:

Physics: rollout of mass-spring system pinned at ends



Conclusions

- Graph neural networks: a first-class member of the deep learning toolkit.
- Learned message-passing on graphs supports simulation, as well as other forms of structured control and decision-making.
- Models with rich internal structure offer unique opportunities for interpretability.
- Build Graph Nets in Tensorflow: github.com/deepmind/graph_nets.

Key collaborators

Alvaro Sanchez-Gonzalez
Victor Bapst
Jess Hamrick
Razvan Pascanu
Nick Watters
Andrea Tacchetti
Theophane Weber
Daniel Zoran
Kelsey Allen
Yujia Li
Kim Stachenfeld
Carl Doersch
Nicholas Heess
Koray Kavukcuoglu
Oriol Vinyals
Shirley Ho
Miles Cranmer
Rui Xu
Kyle Cranmer

References

Battaglia et al. 2018 arXiv
Battaglia et al., 2016, NeurIPS
Watters et al., 2017, NeurIPS
Cranmer et al., 2019, arXiv/NeurIPS workshop
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS workshop
Sanchez-Gonzalez et al., 2018, ICML
Tacchetti et al., 2019, ICLR
Li et al., 2018, arXiv
Hamrick et al., 2018, Proc Cog Sci
Bapst et al., 2019, ICML