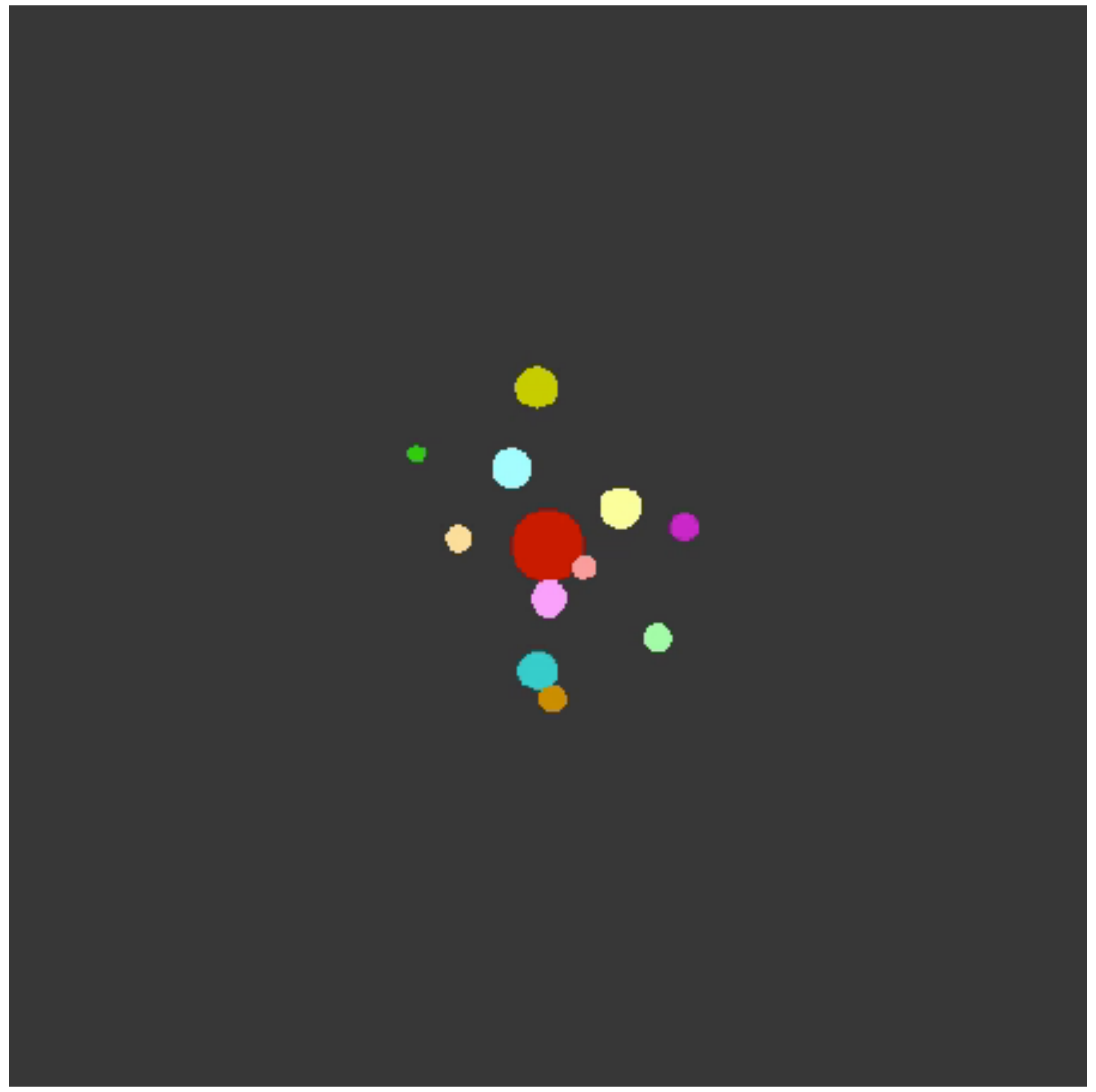
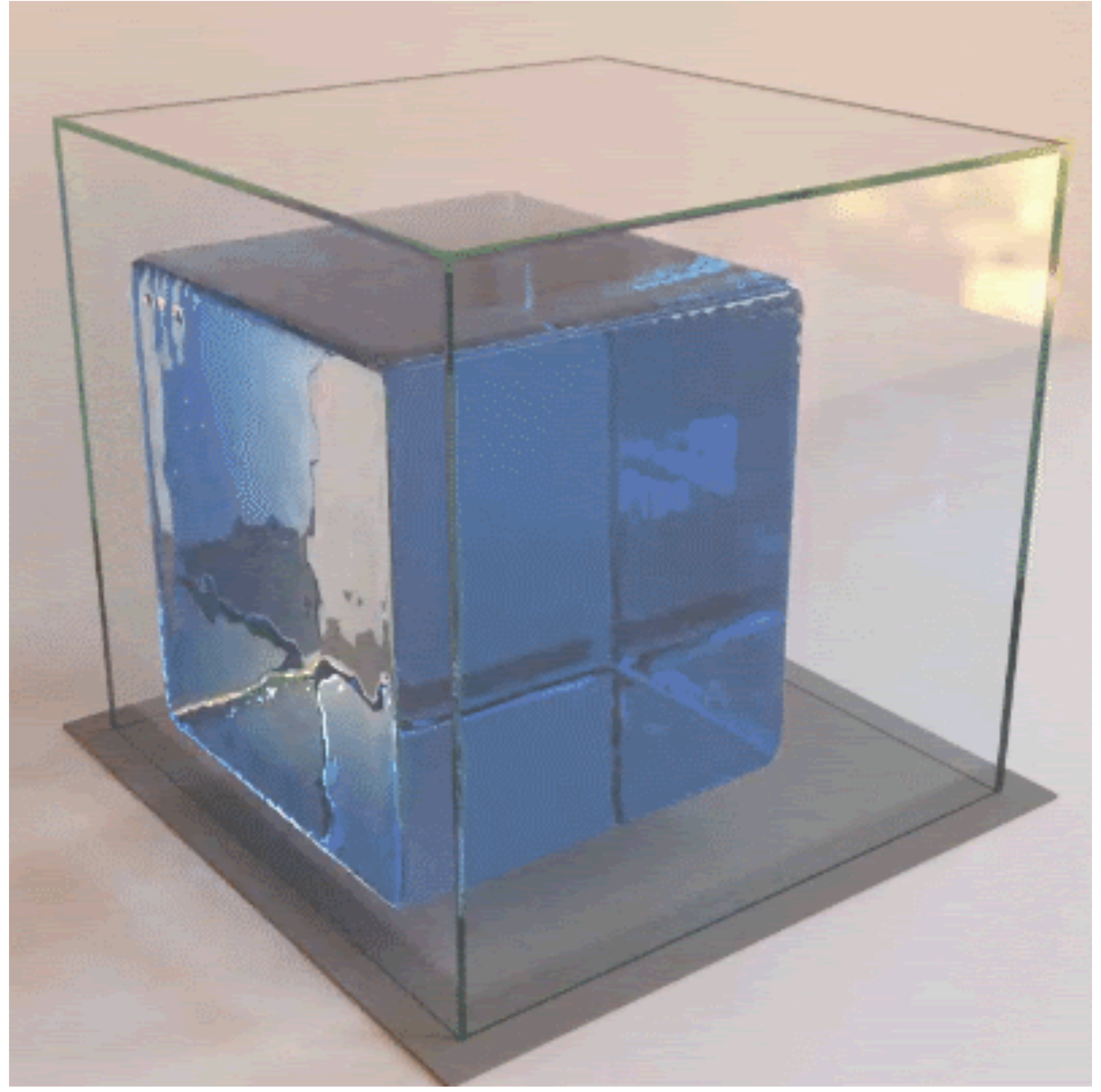
Structured models of objects, relations, and physics


Peter Battaglia

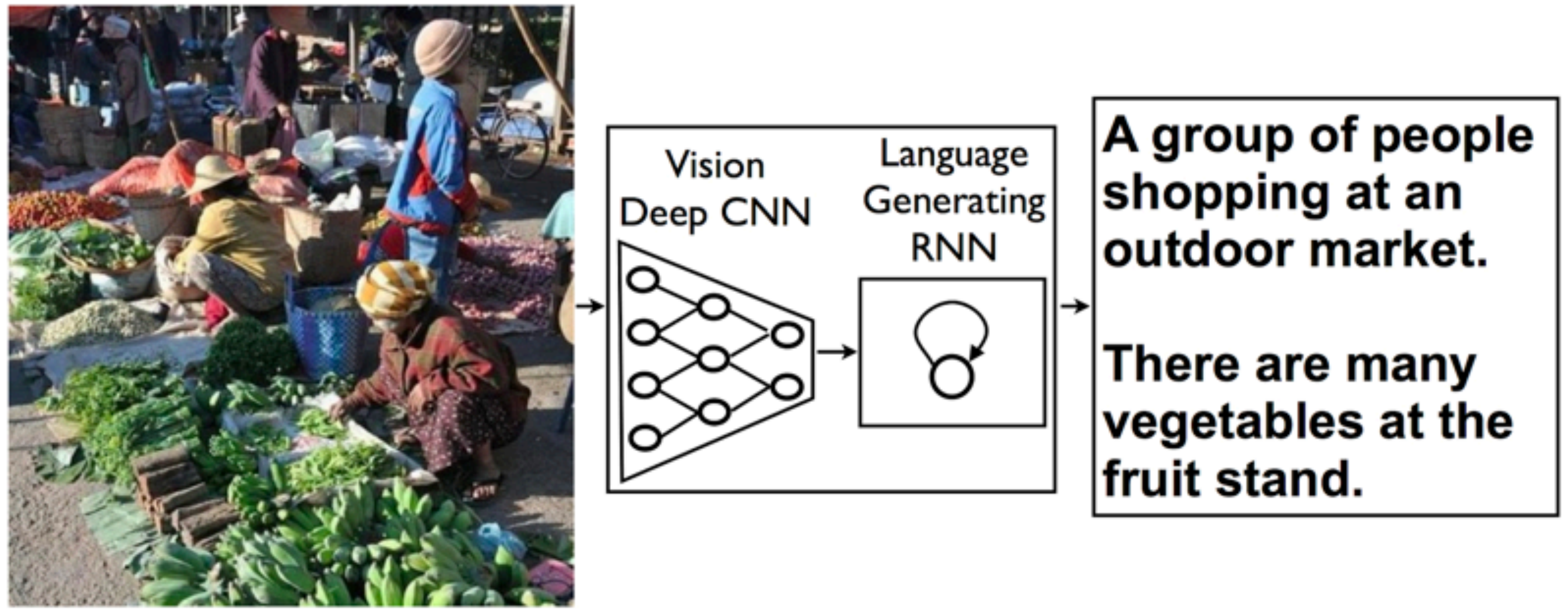
DeepMind

Interexperimental Machine Learning Workshop / Data Science Seminar
CERN (remote) - October 20, 2020

What is deep learning good at?

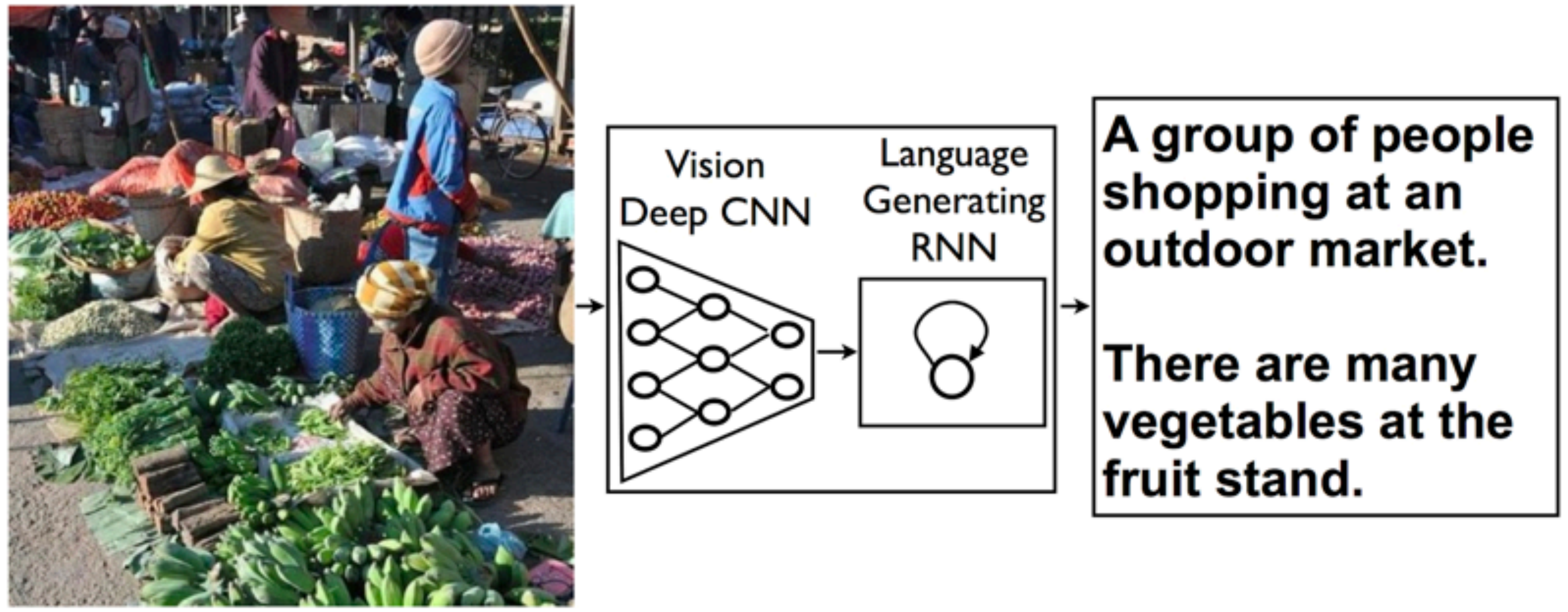
What is deep learning good at?

Image and language processing

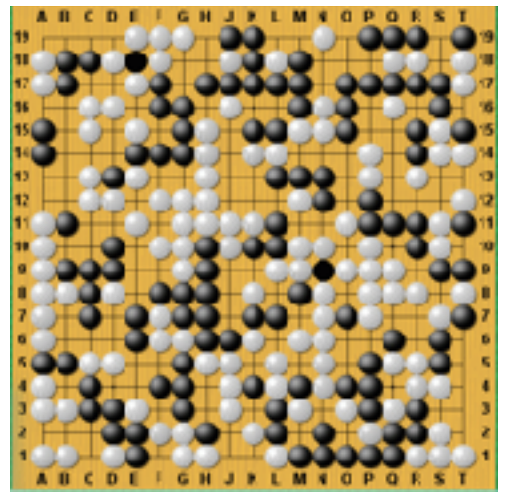


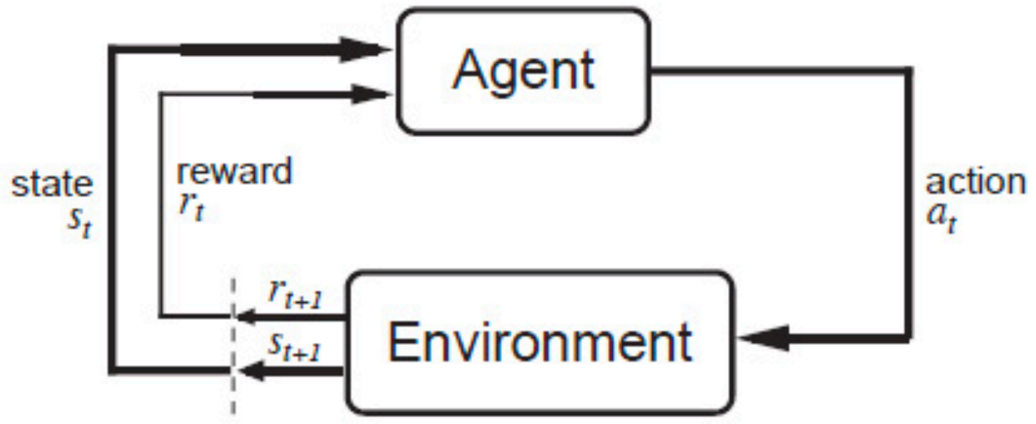
What is deep learning good at?

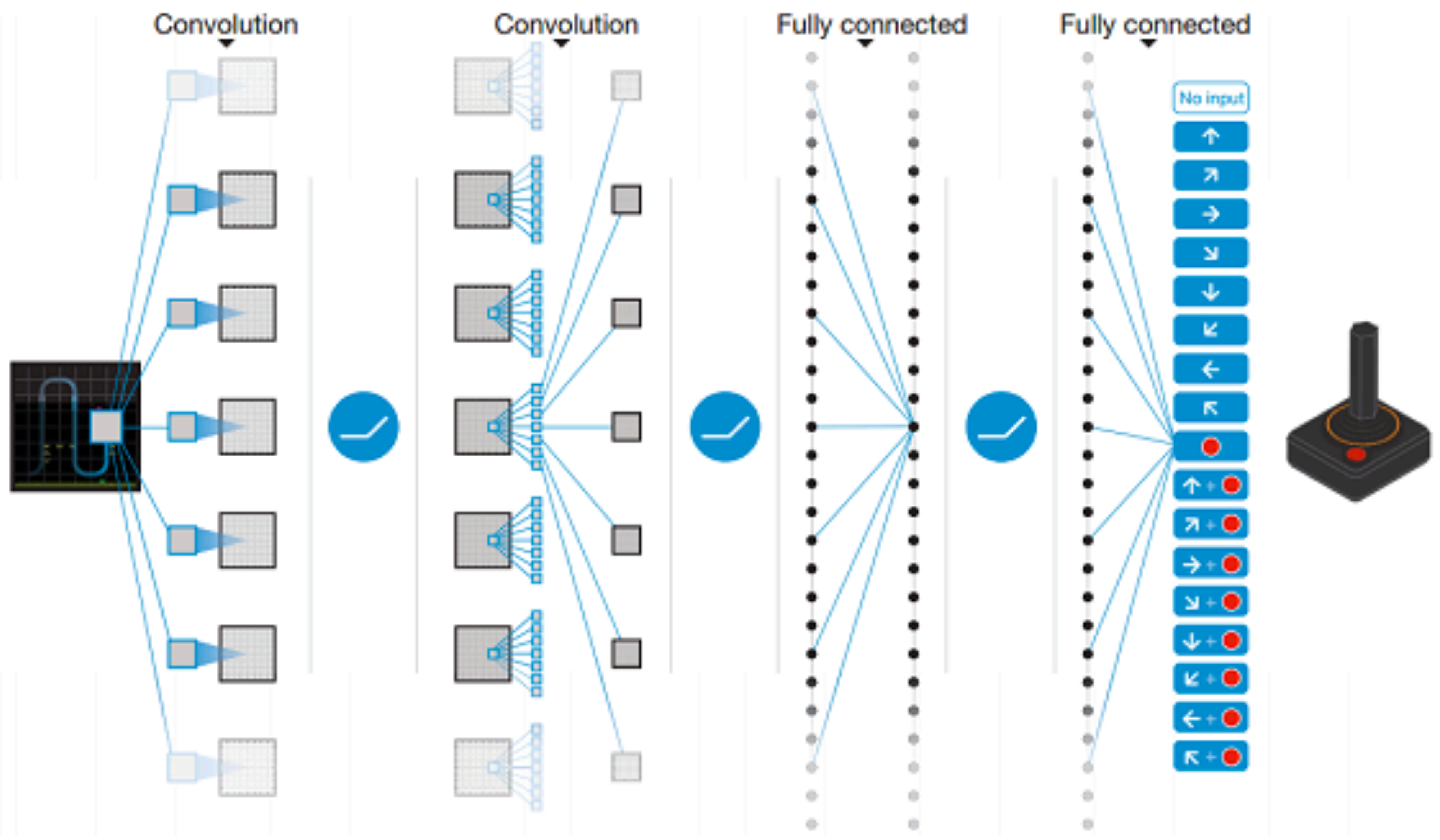
Image and language processing



Games (via deep reinforcement learning)







What is deep learning good at?

Image and language processing

What do many of deep learning's successes have in common?

Games (via deep reinforcement learning)

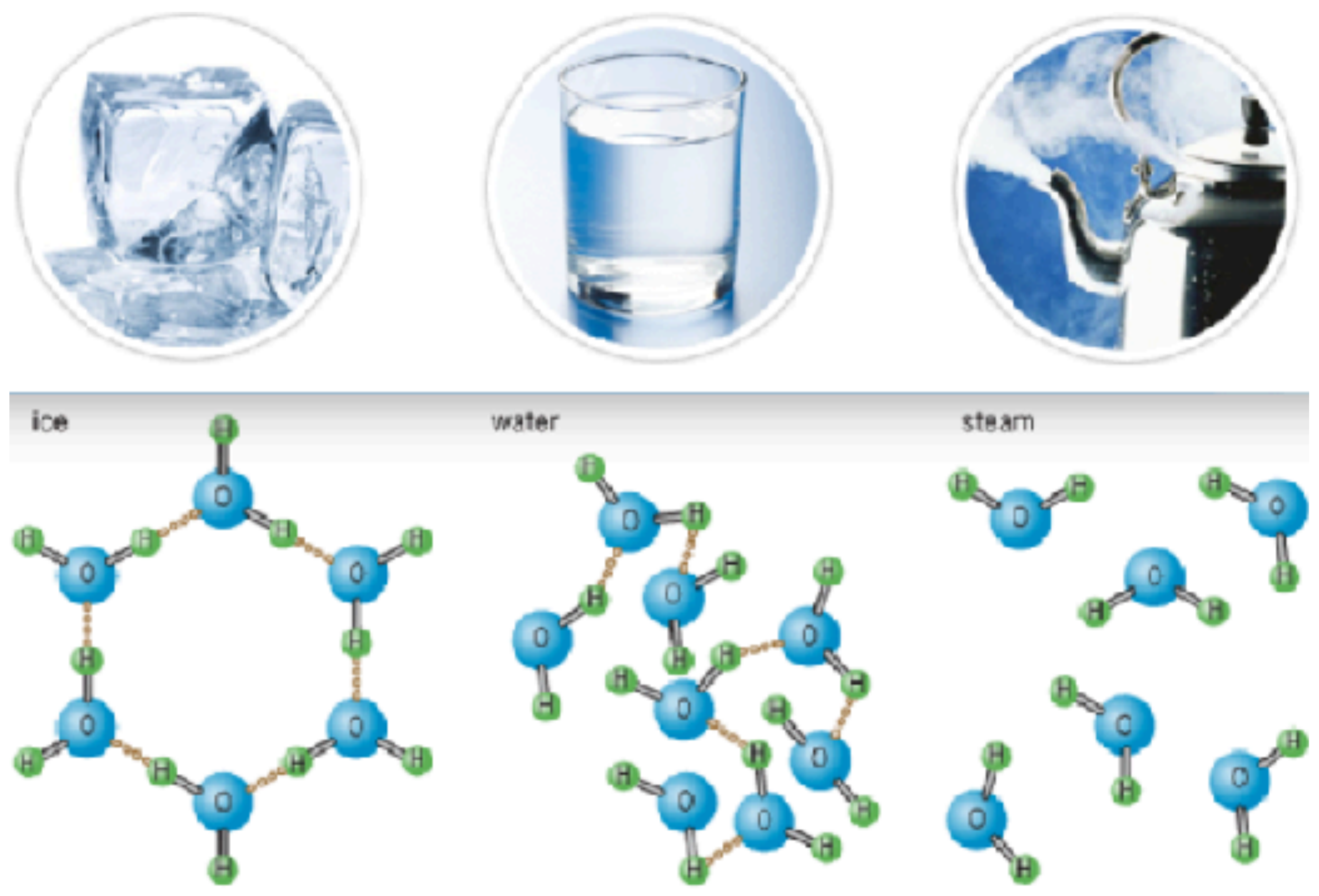
What is deep learning good at?

What do many of deep learning's successes have in common?

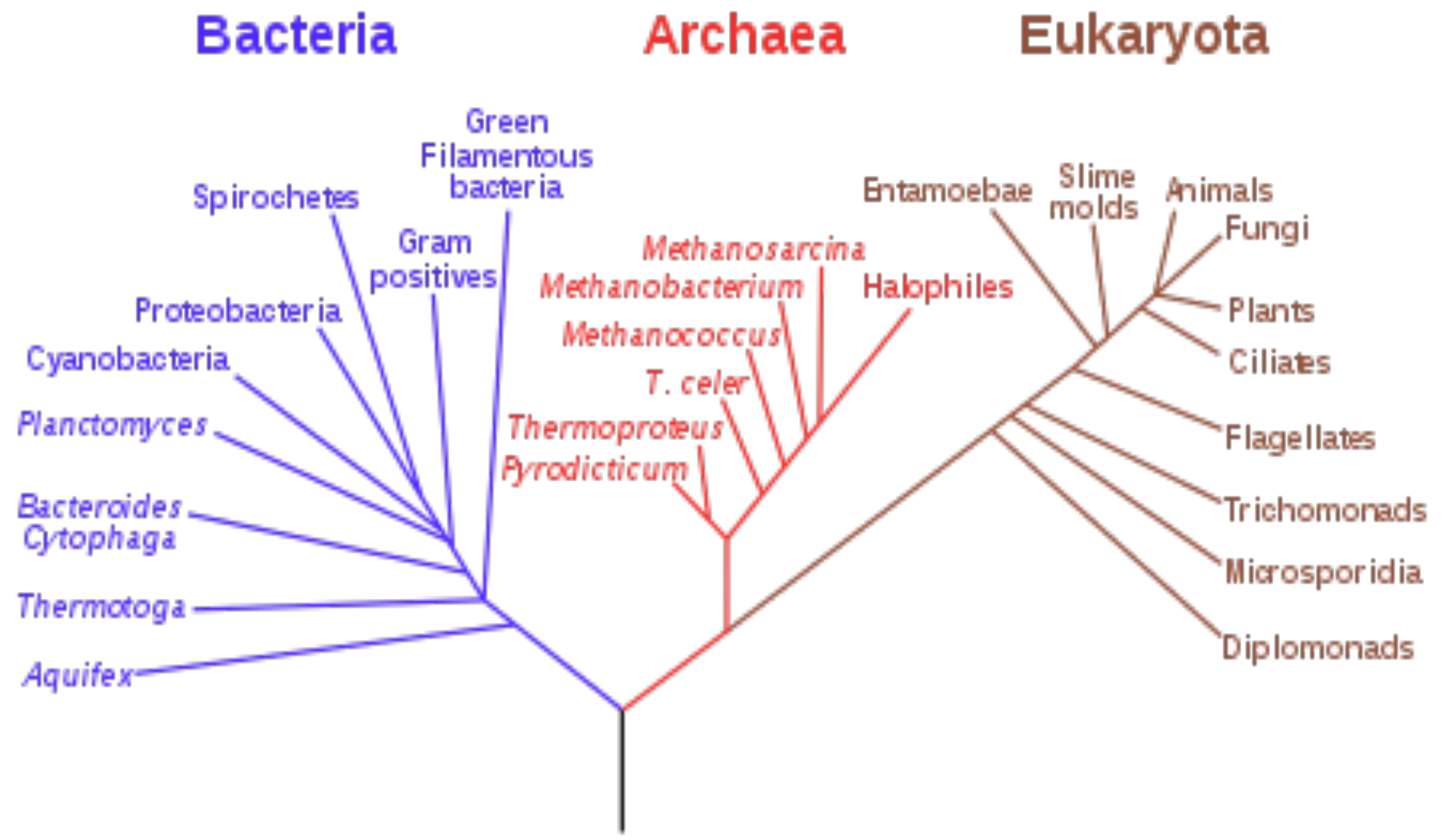
* Vectors
* Grids
* Sequences

But many important domains are richly structured

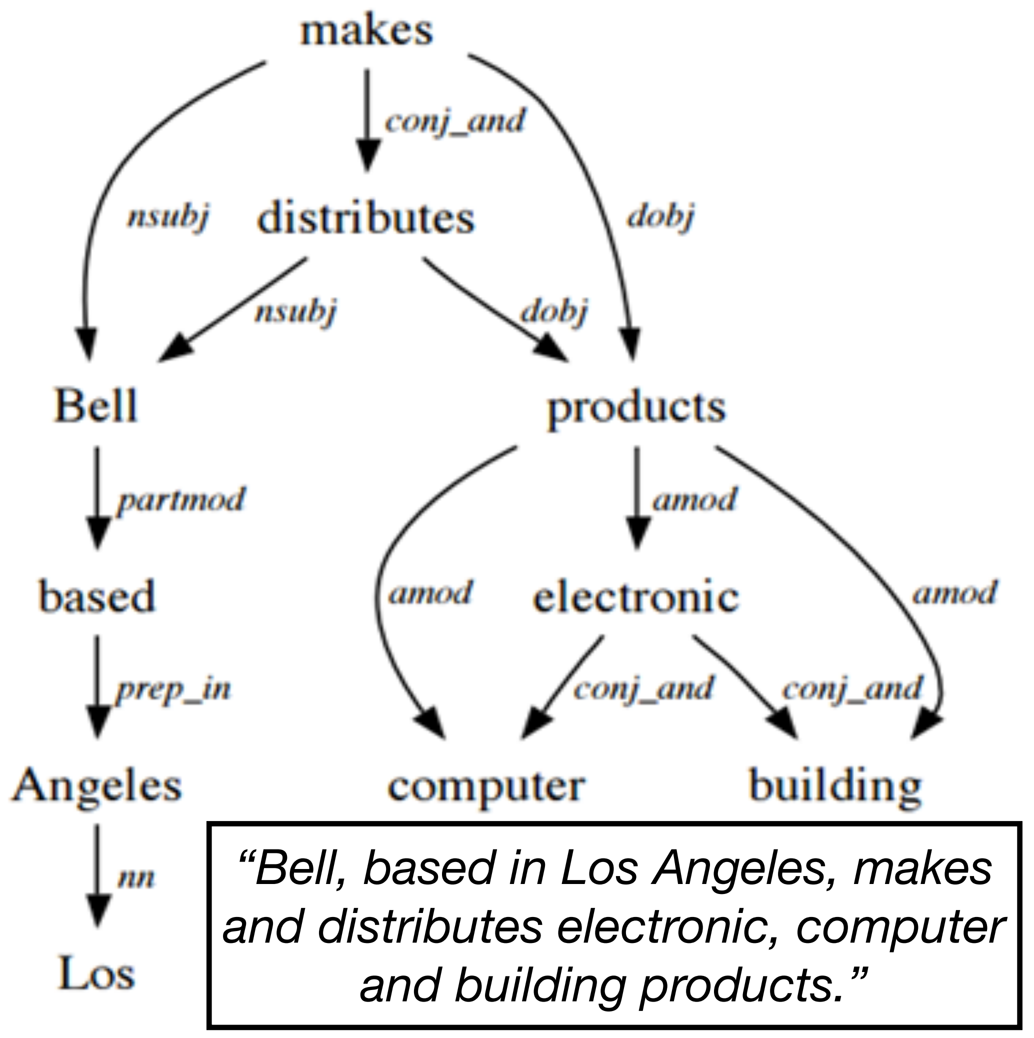
Molecules



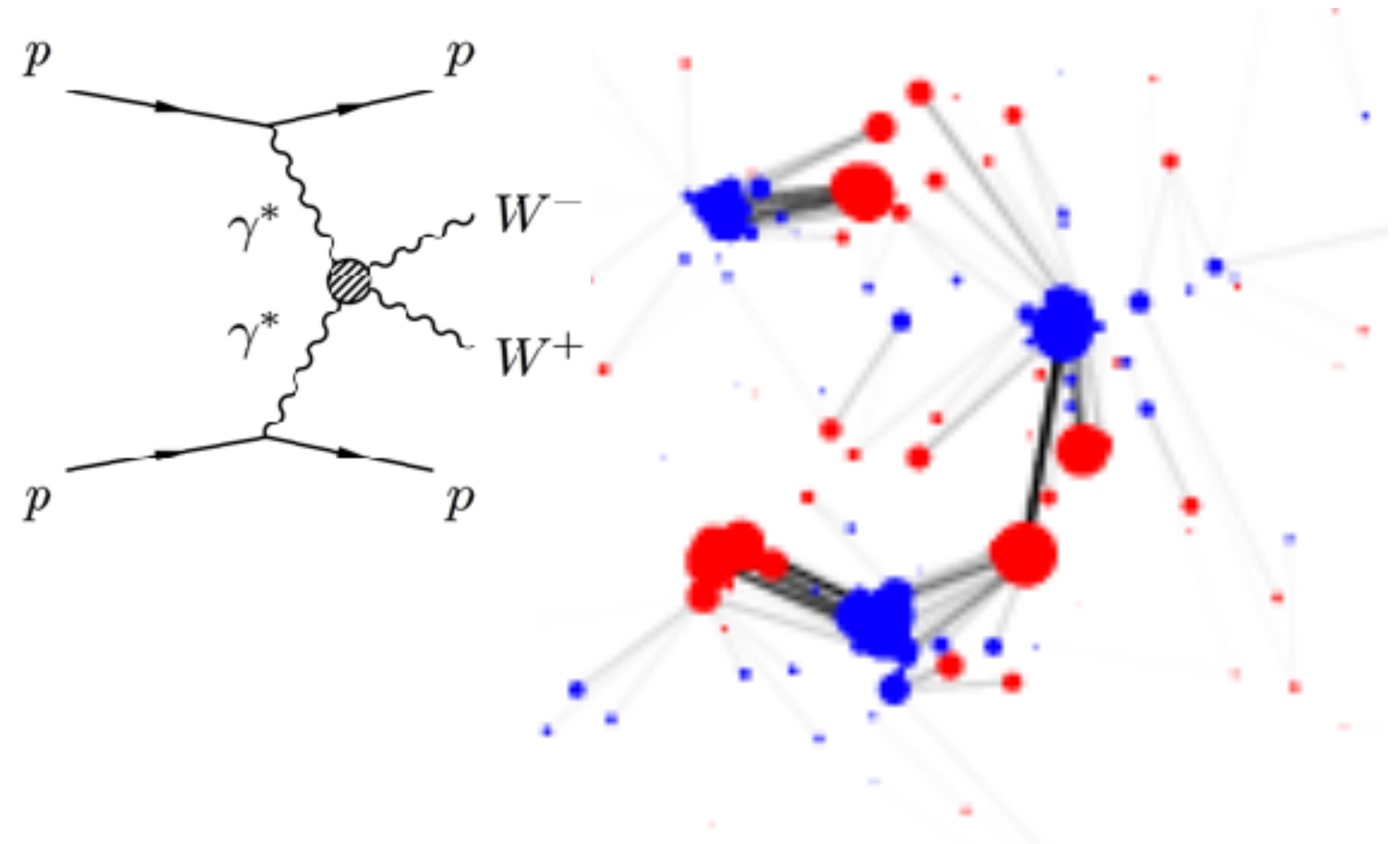
Biological species



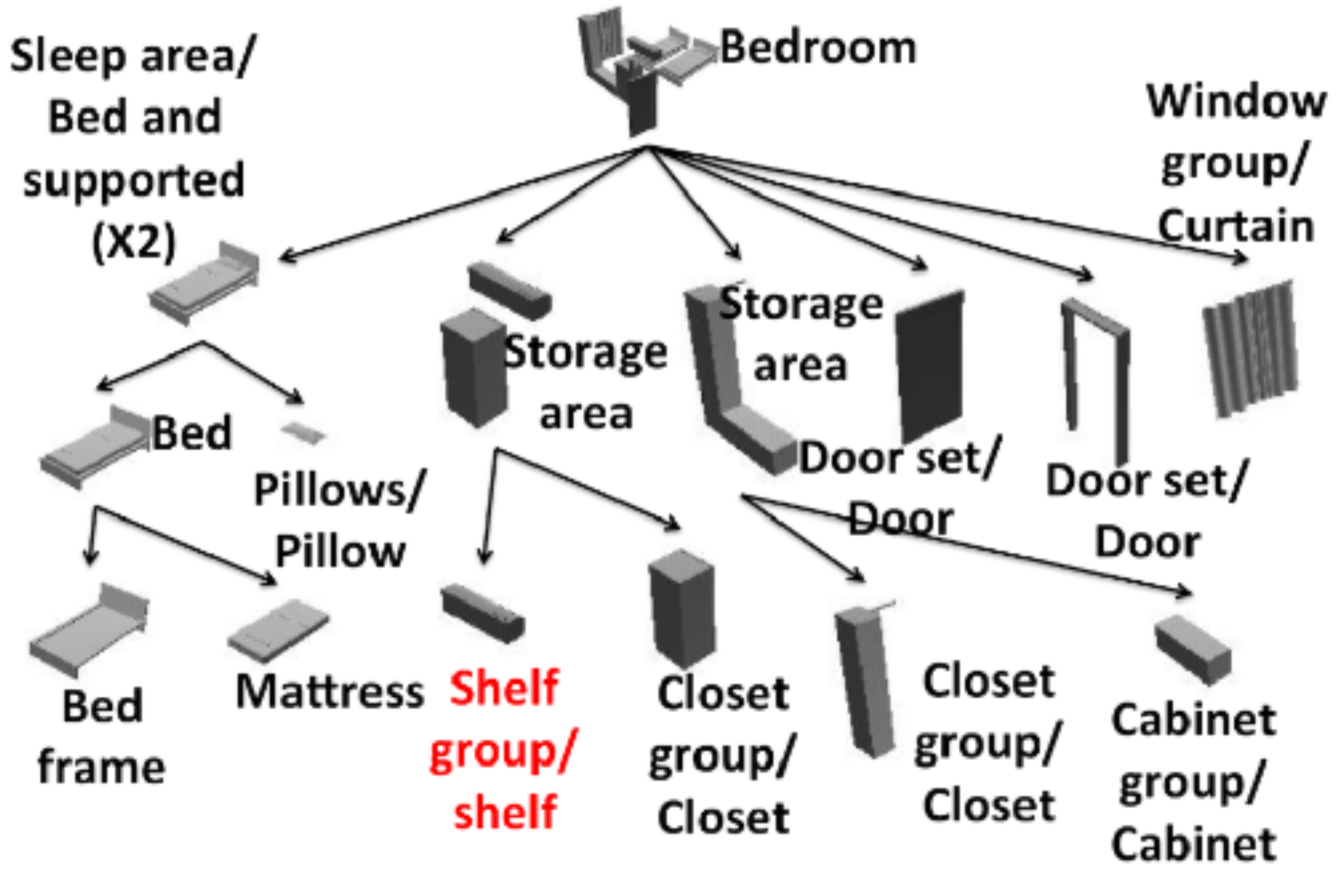
Natural language



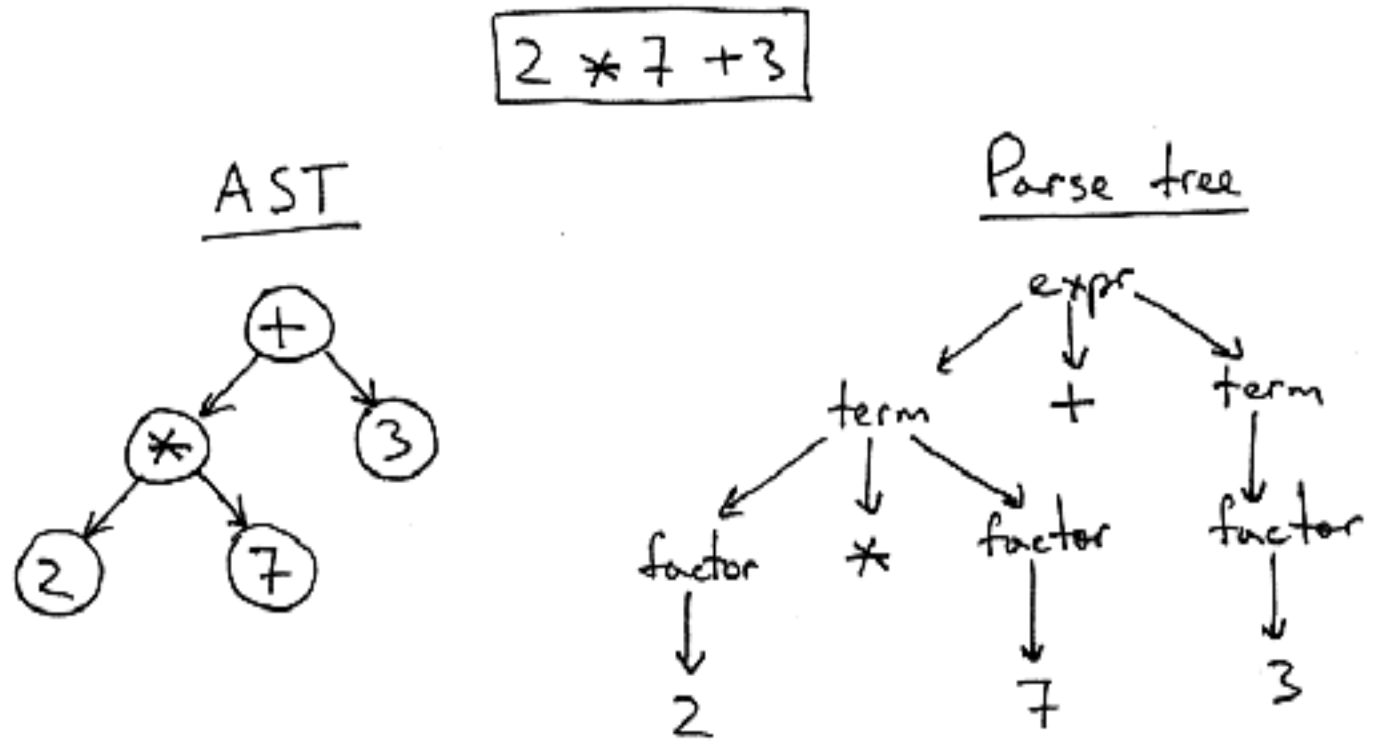
Sub-atomic particles



Everyday scenes



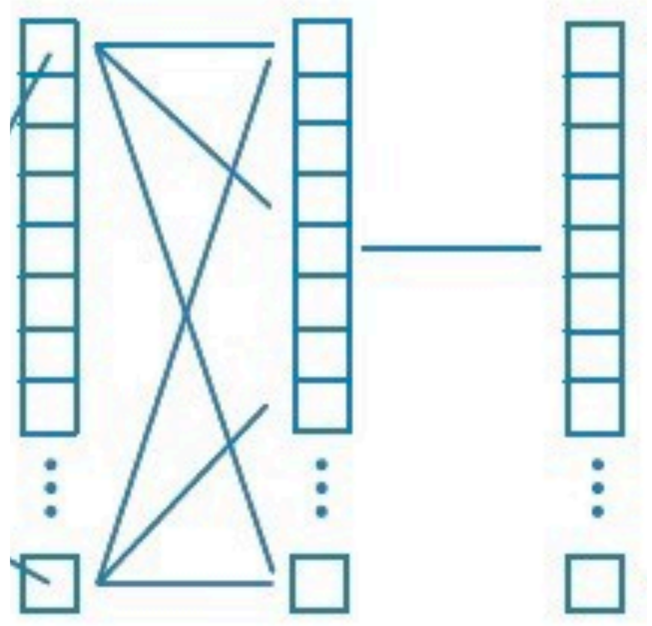
Code



What tool do I need?

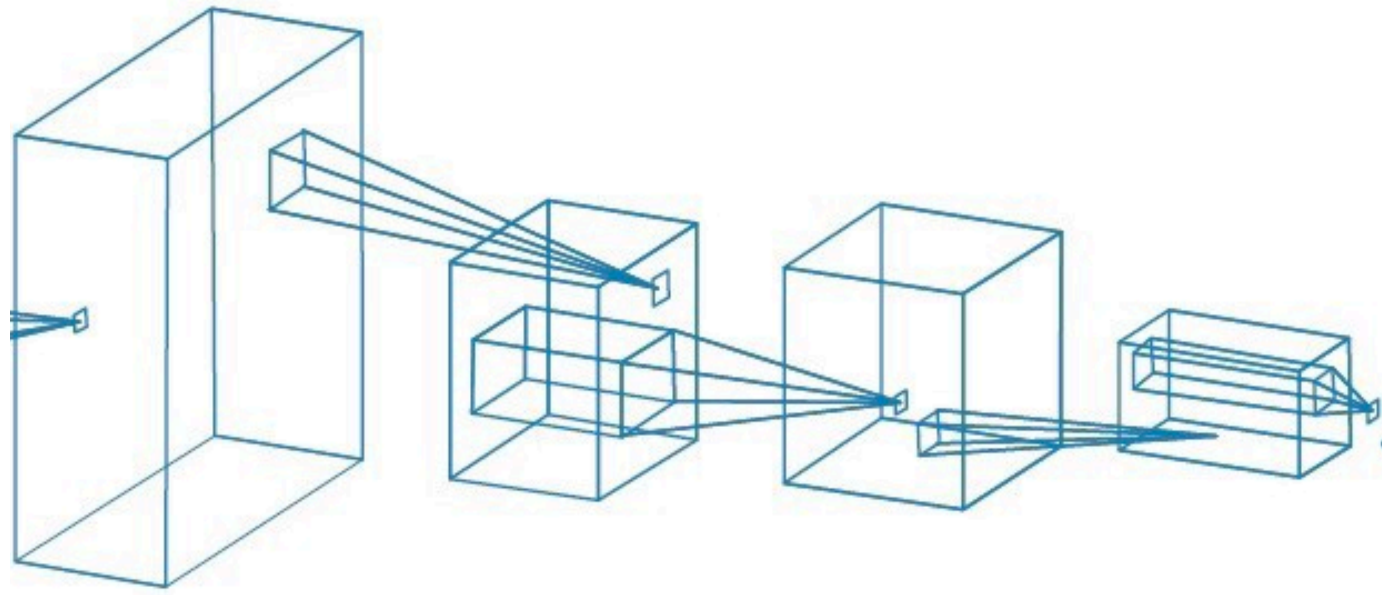
“My data is **vectors**”:

Multi-layer perceptron (MLP)



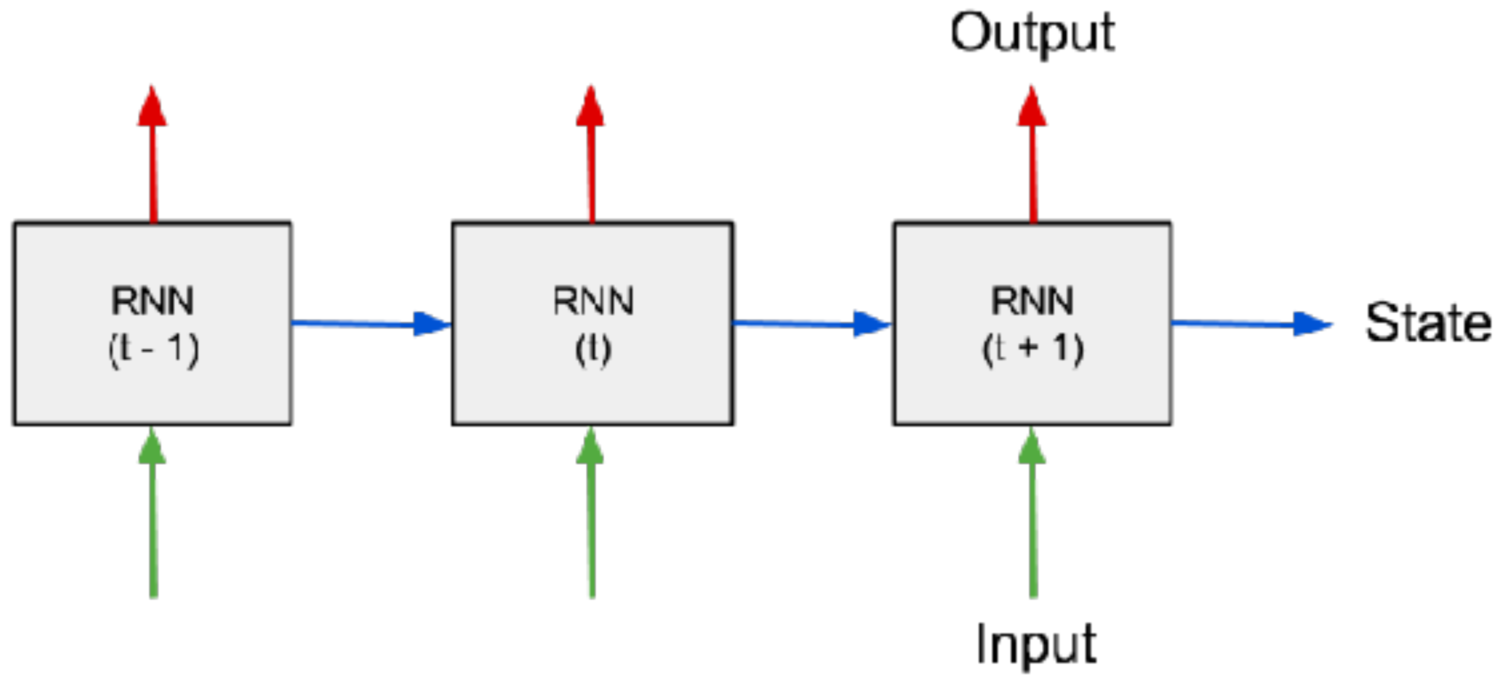
“My data is **grids**”:

Convolutional neural network (CNN)



“My data is **sequences**”:

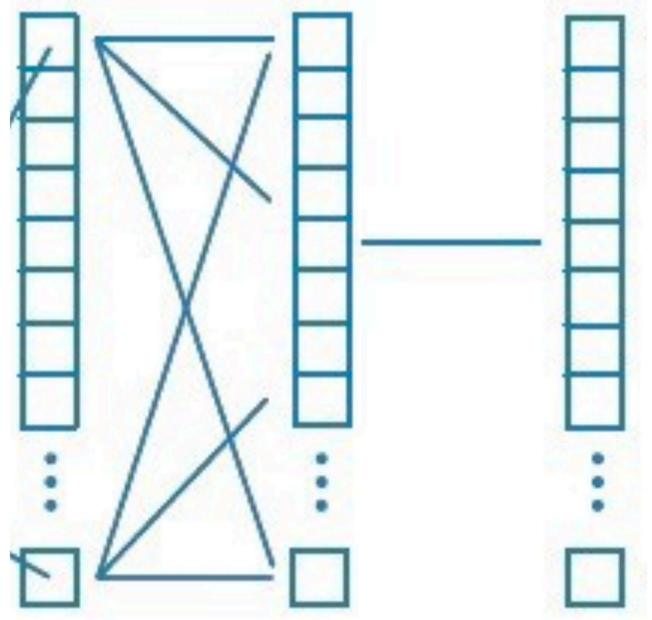
Recurrent neural network (RNN)



What tool do I need?

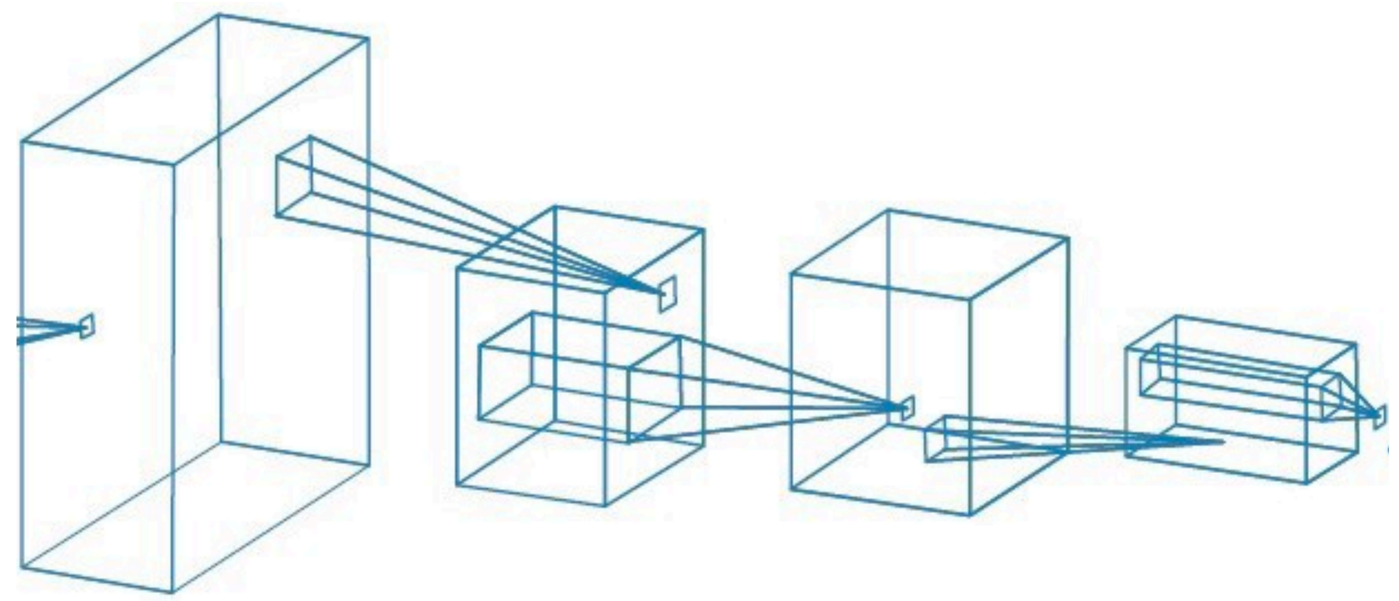
"My data is **vectors**":

Multi-layer perceptron (MLP)



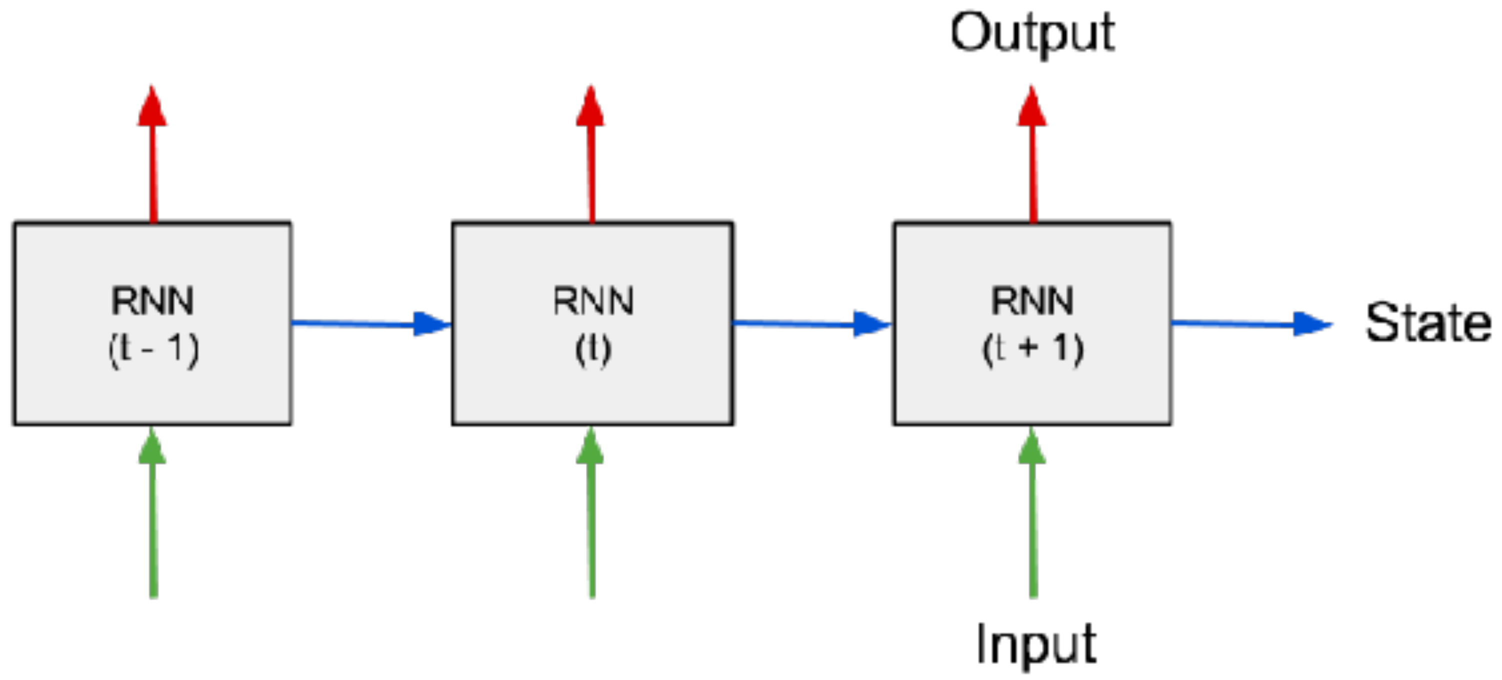
"My data is **grids**":

Convolutional neural network (CNN)

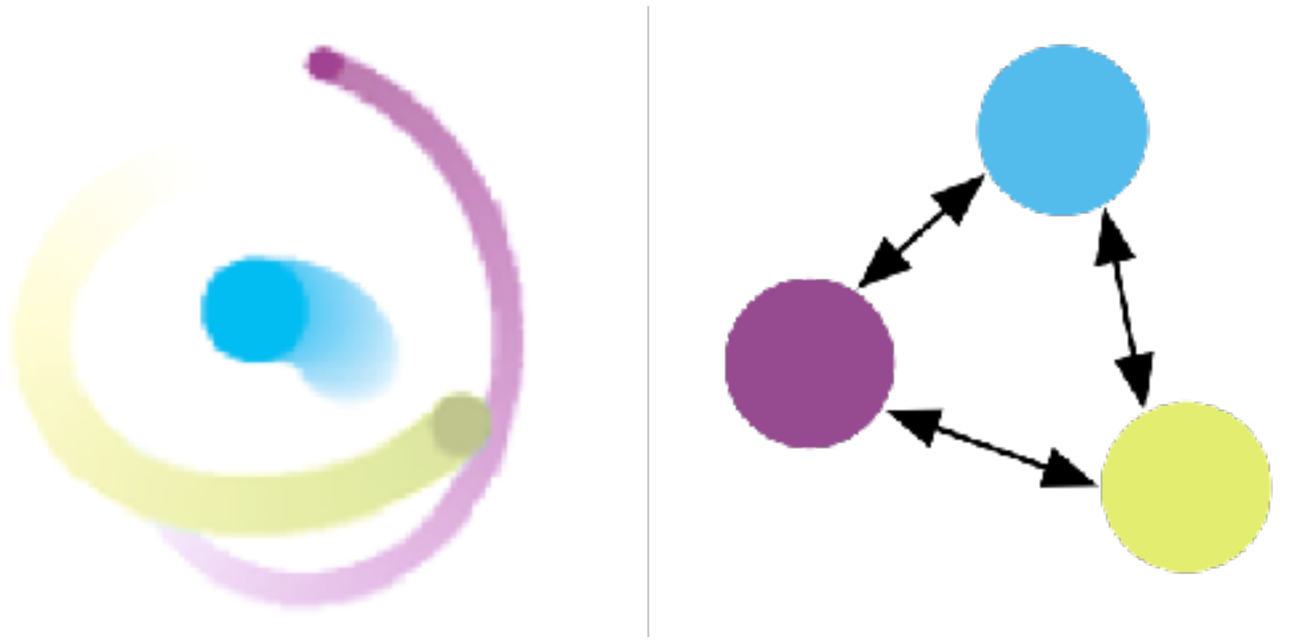


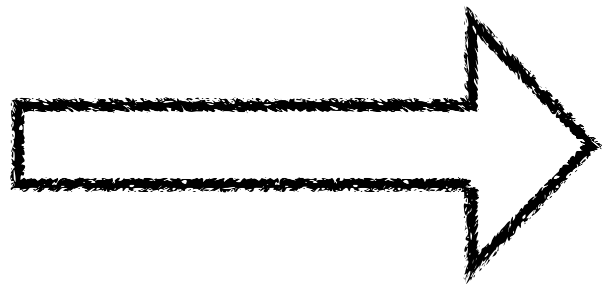
"My data is **sequences**":

Recurrent neural network (RNN)

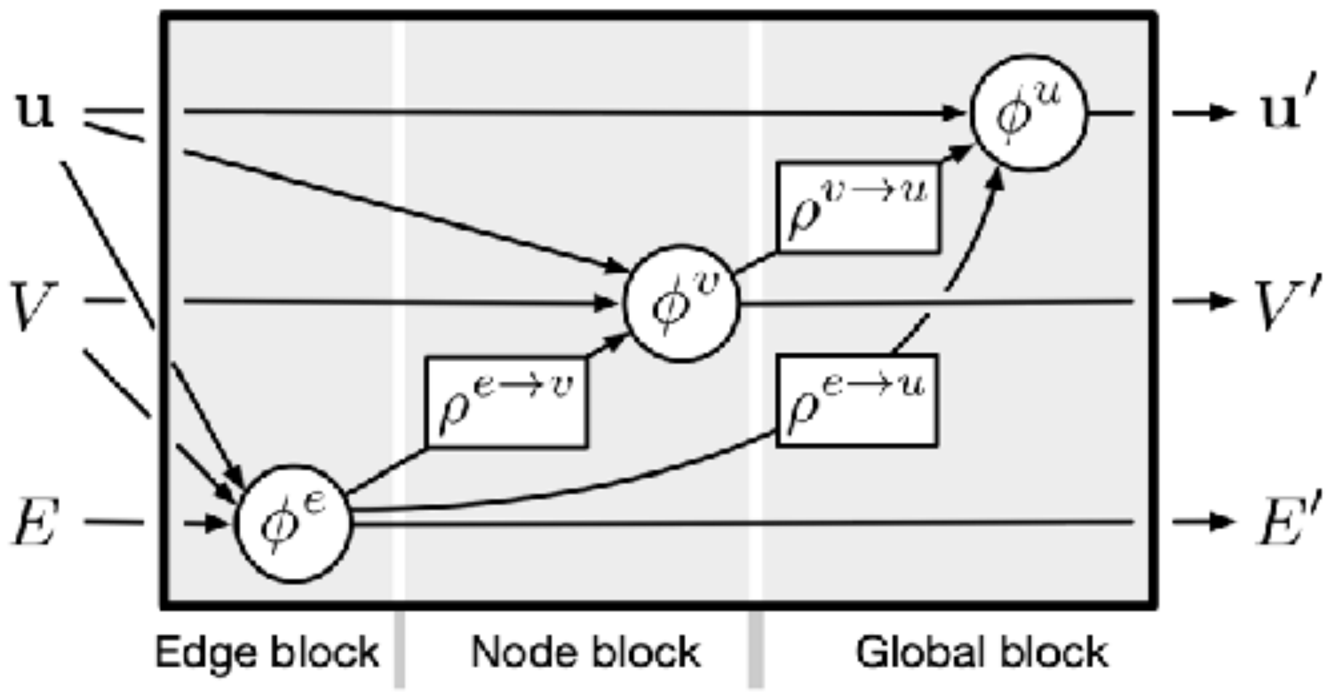


"My data is **structured**":





Graph neural network (GNN)



Background: Graph Neural Networks

General idea

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

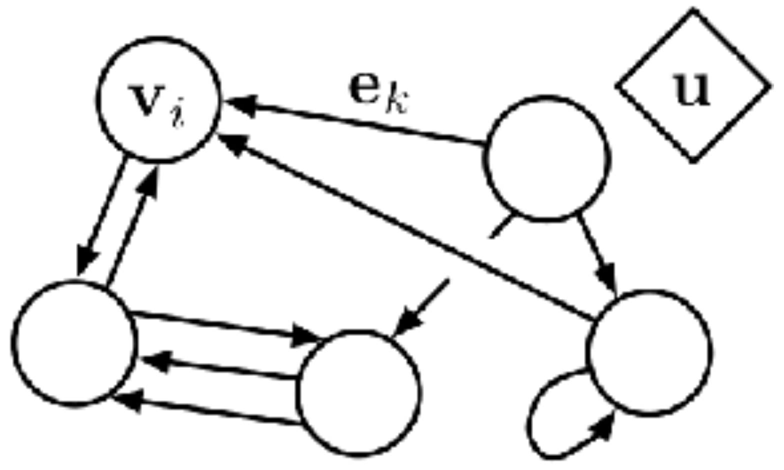
Key literature surveys

- Scarselli et al. (2009) "The Graph Neural Network Model".
  Summarizes the initial papers on the topic from ~2005-2009. Original innovation, 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.

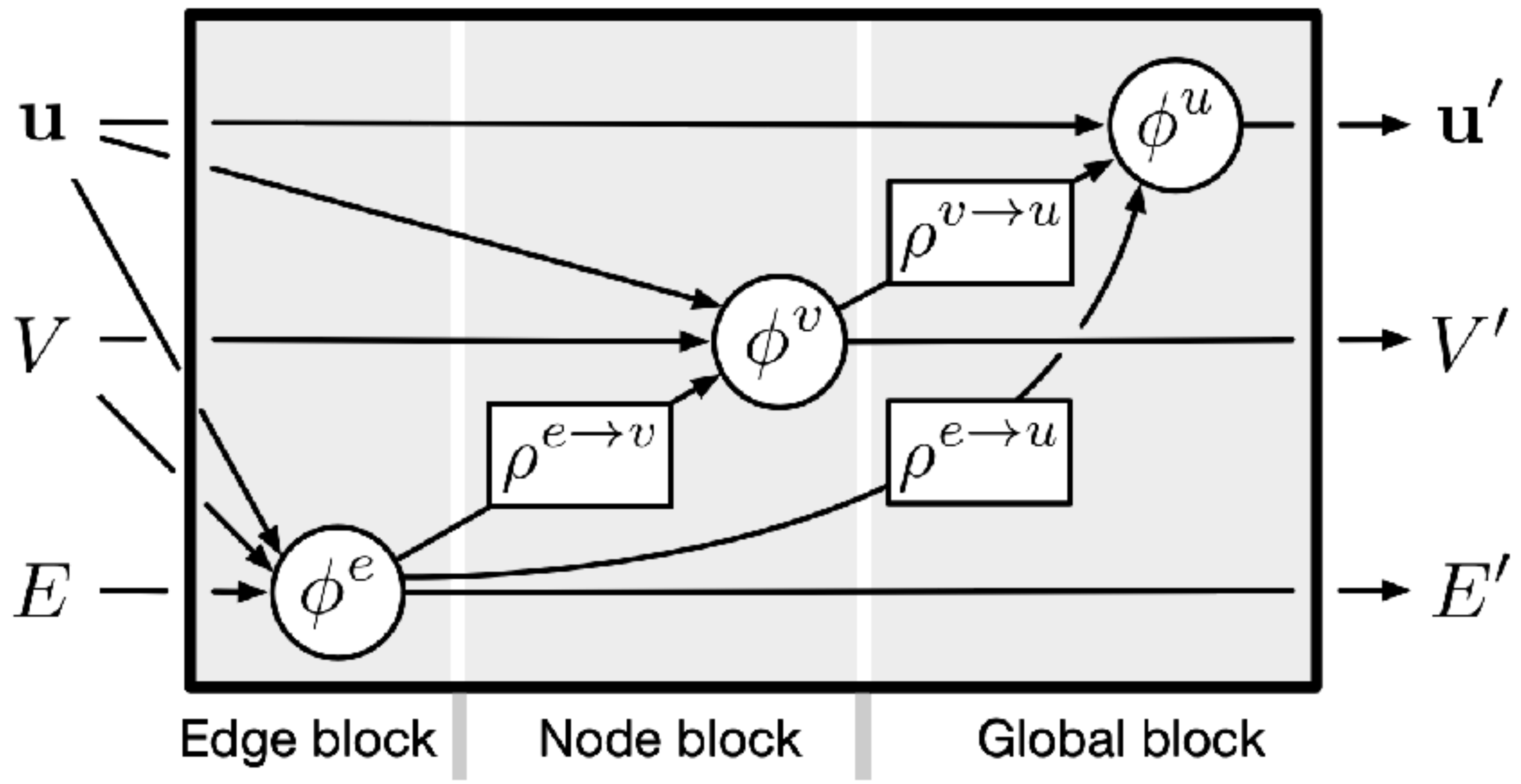
Graph Networks (GNs)

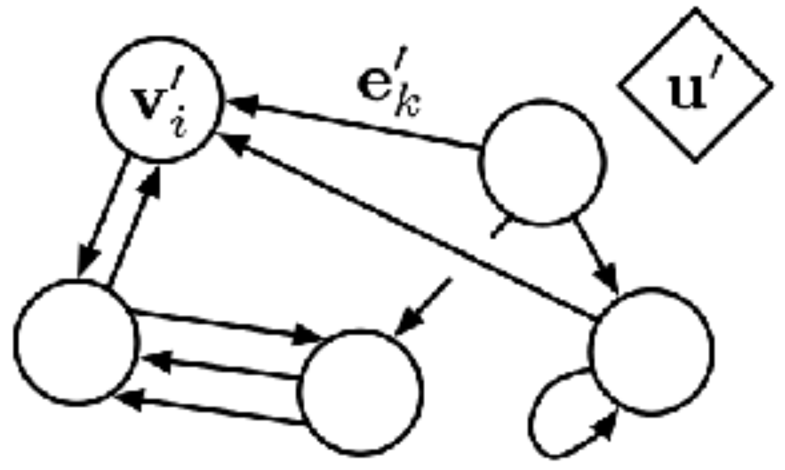
Why do we need another graph neural network variant?

- 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

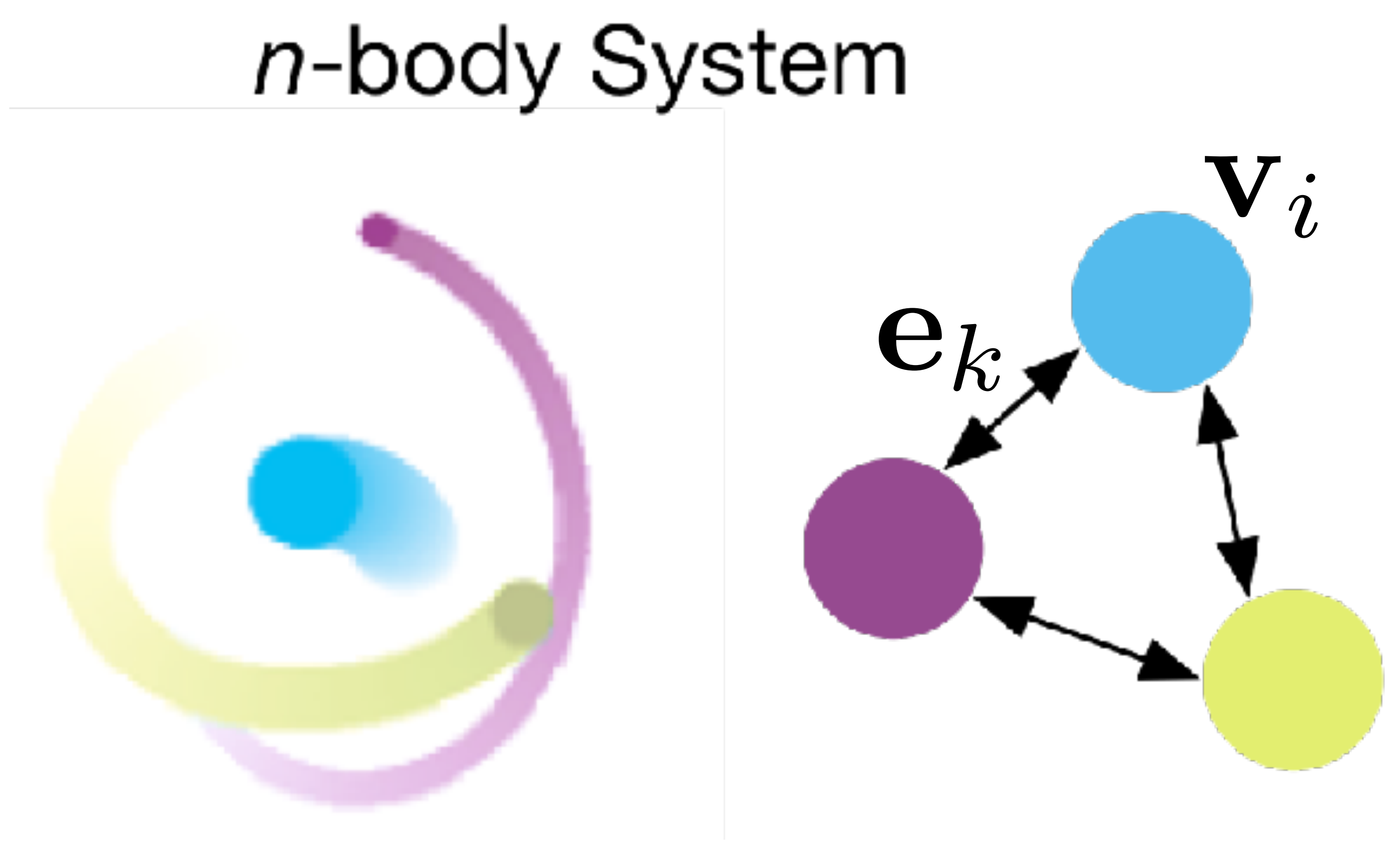


Graph Network
(a type of Graph Neural Network)
Battaglia et al., 2018, arXiv



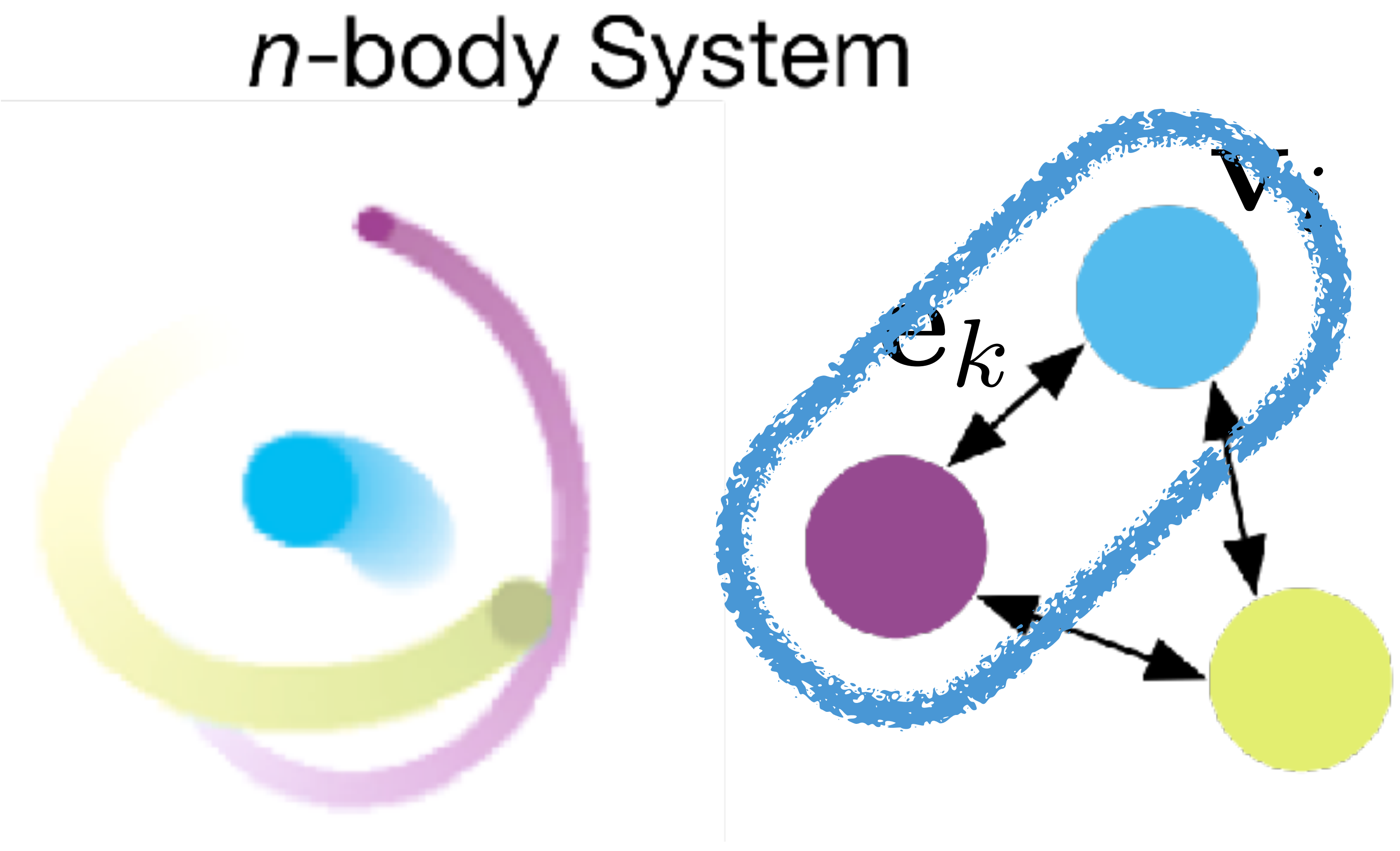


Interaction Network: Learning simulation as message-passing



Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



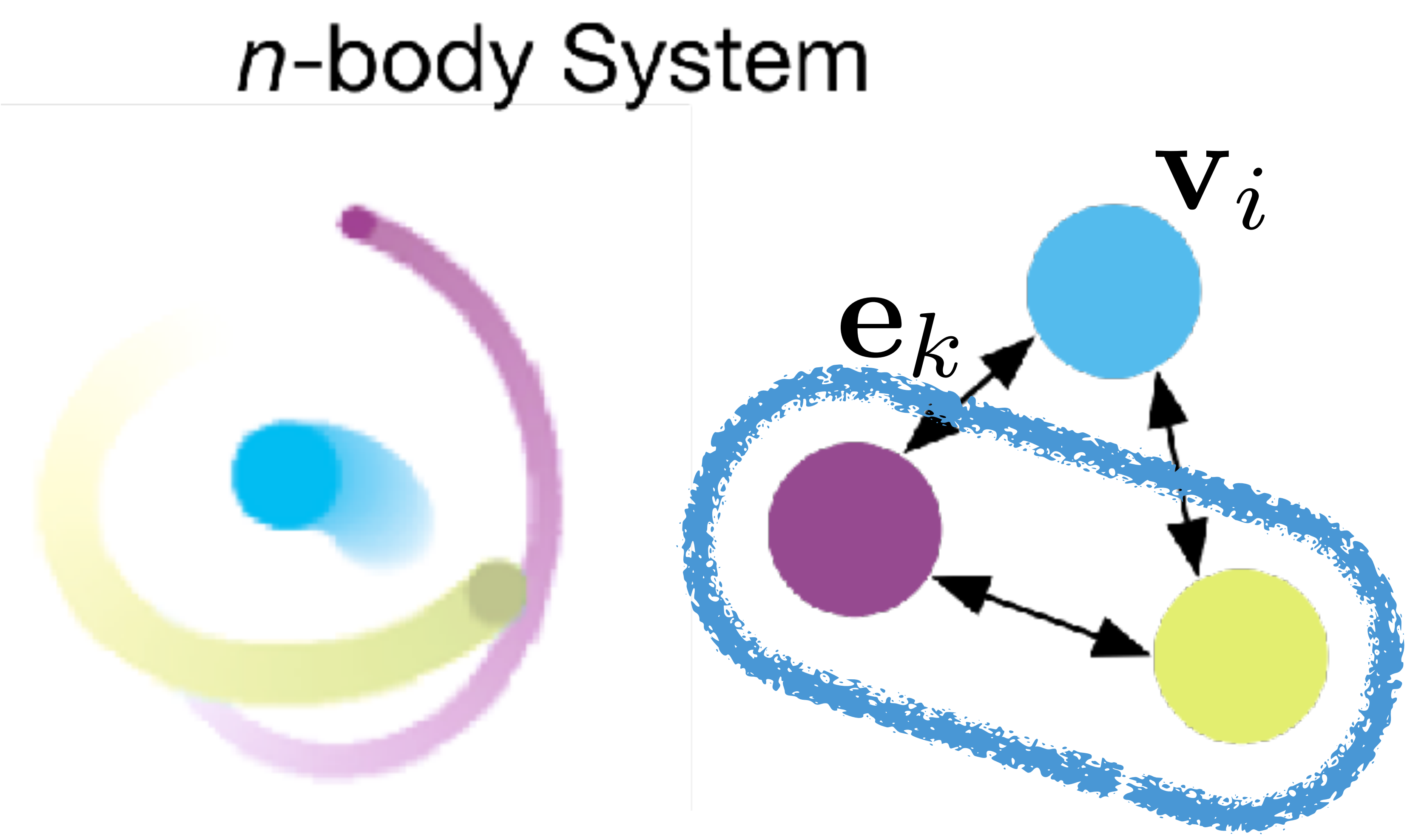
Edge function

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

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

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



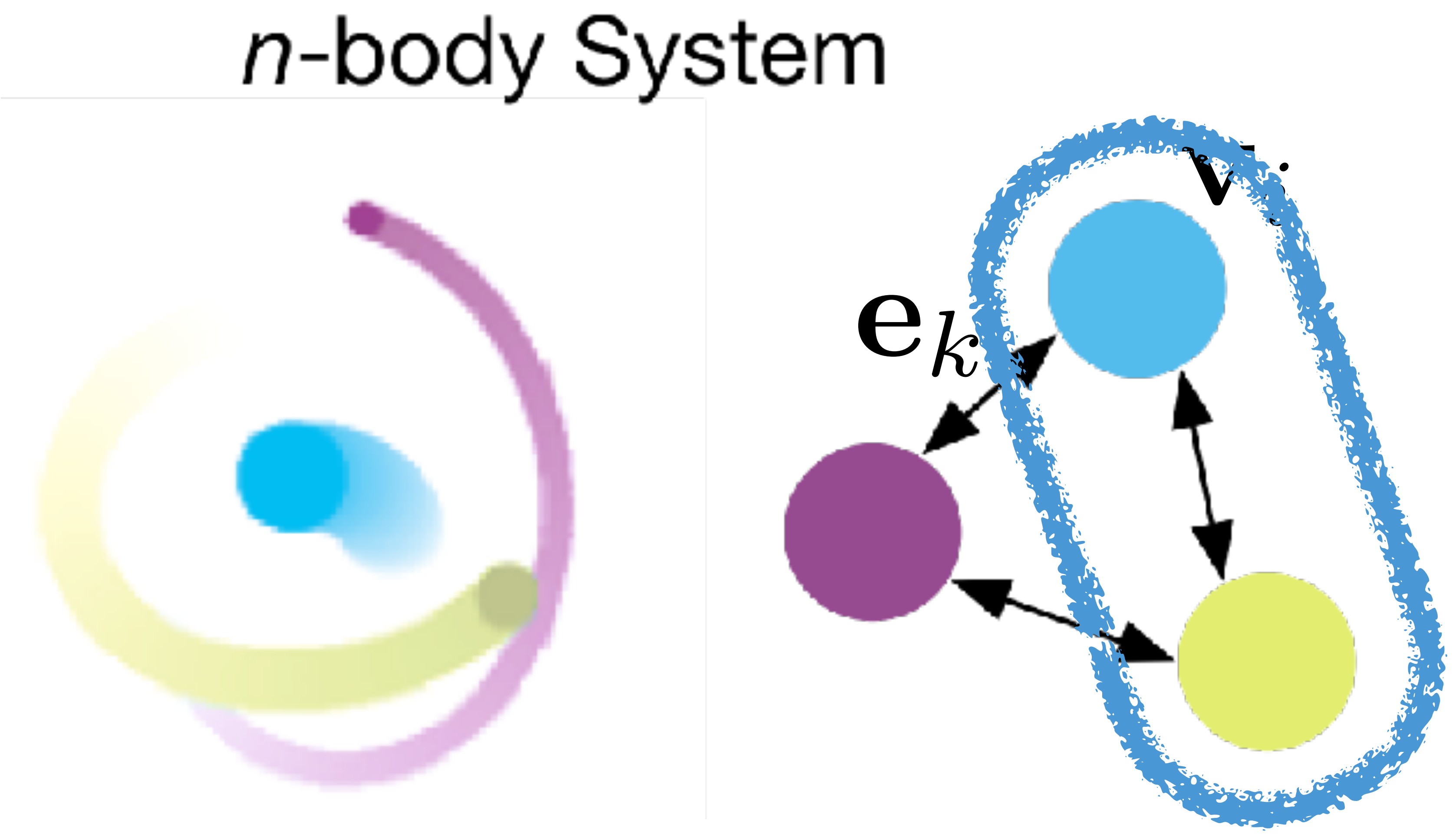
Edge function

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

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

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



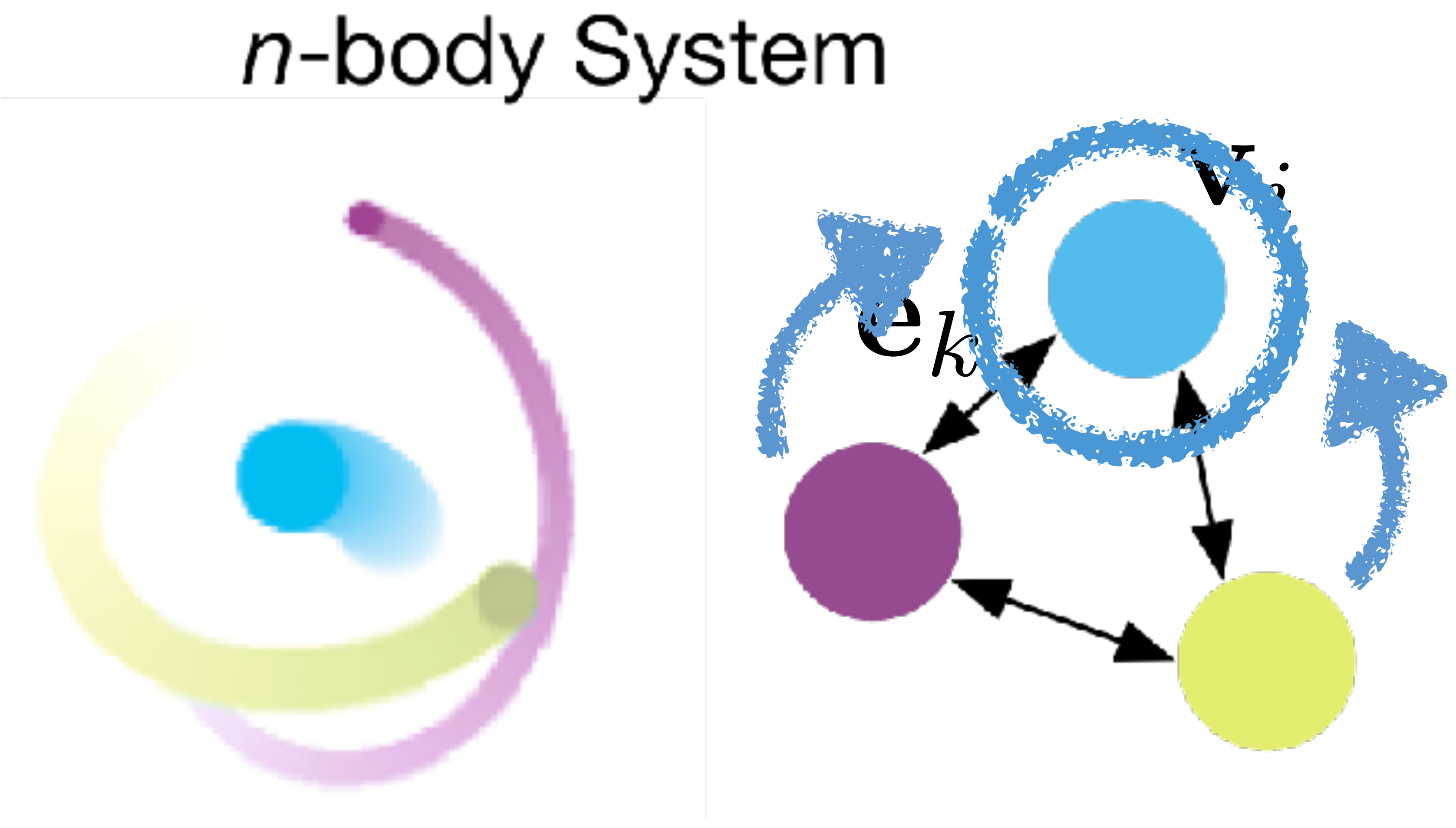
Edge function

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

- 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

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

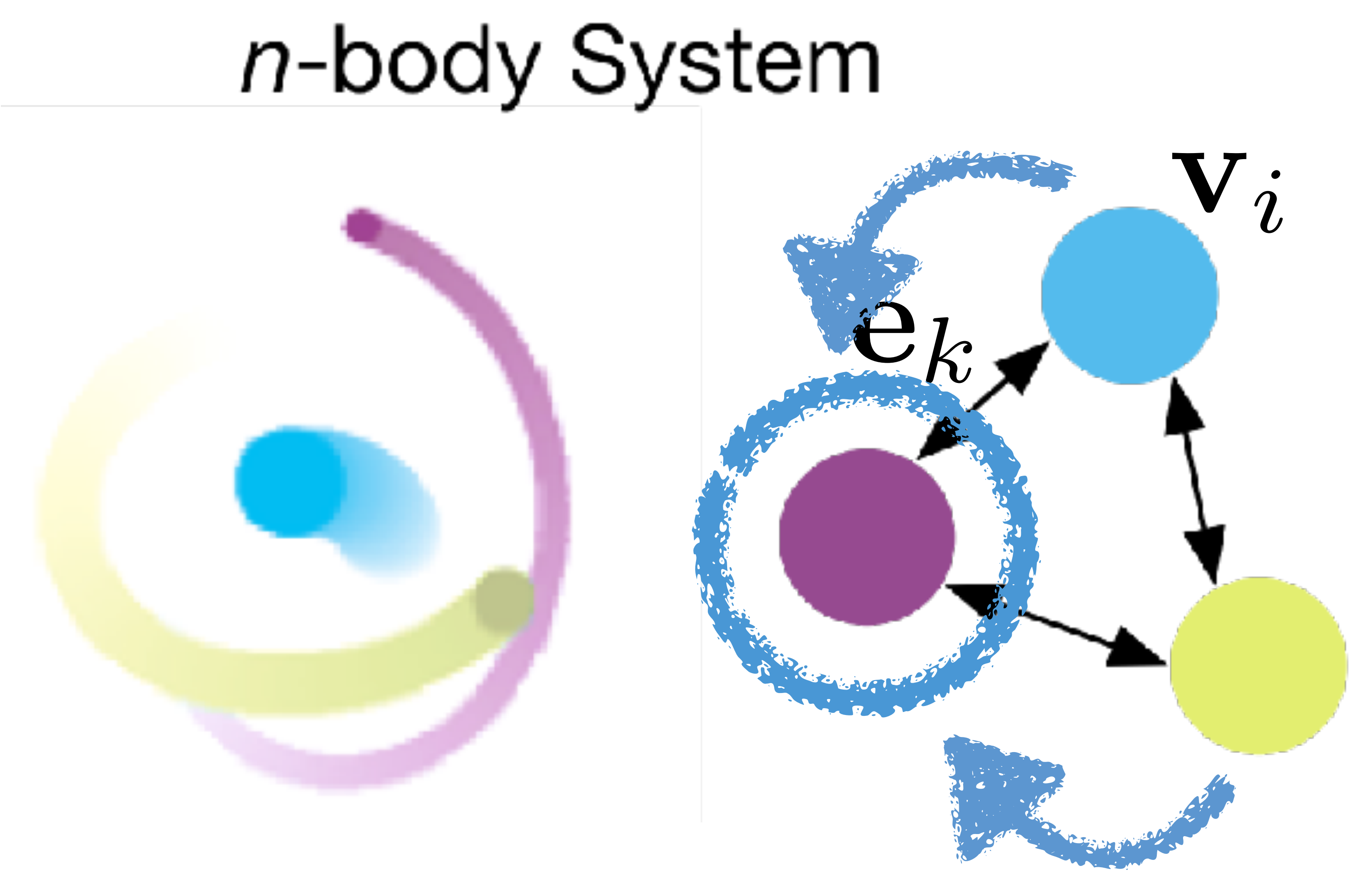
Node function

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

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

Battaglia et al., 2016, NeurIPS

Interaction Network: Learning simulation as message-passing



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

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

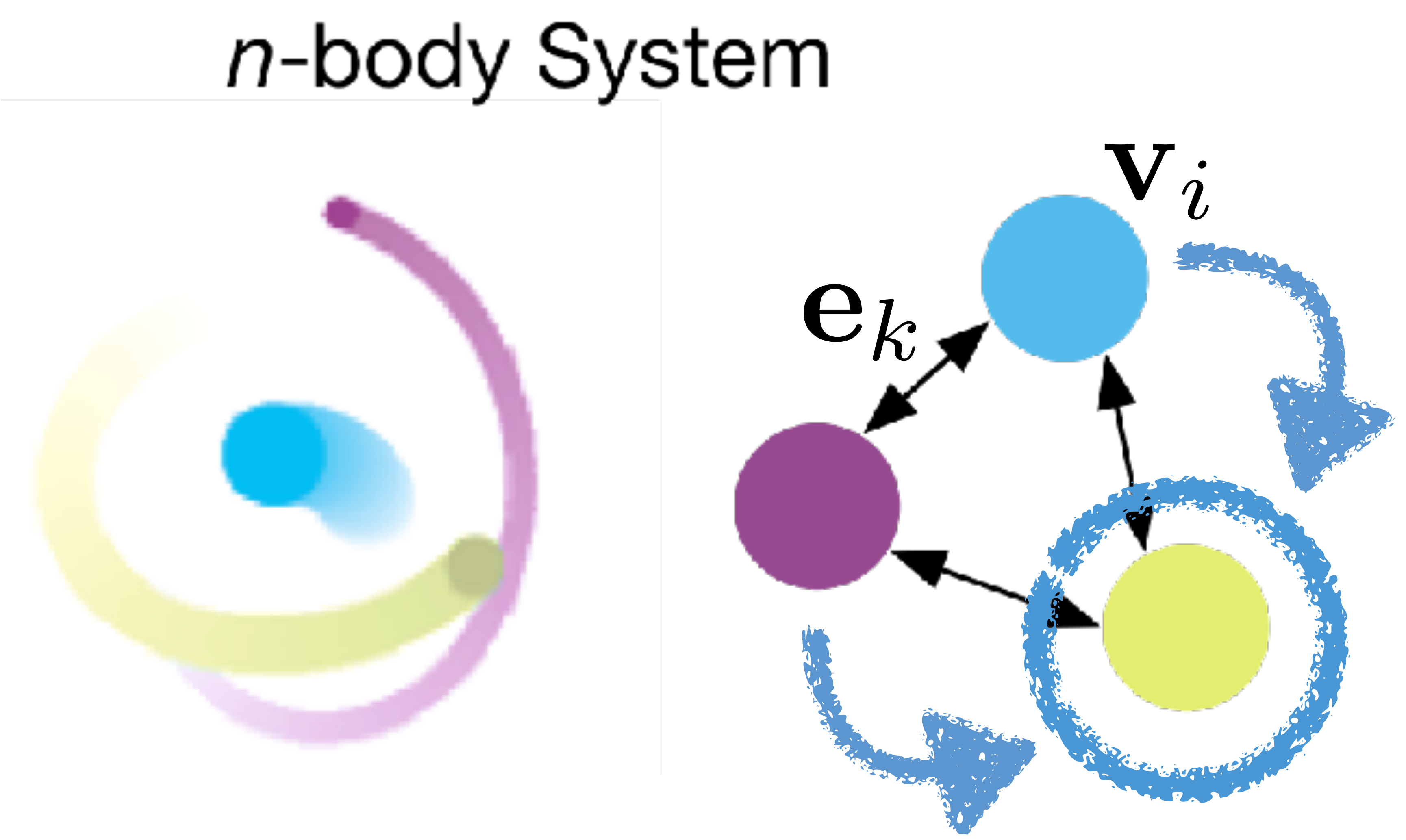
Node function

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

- 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

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

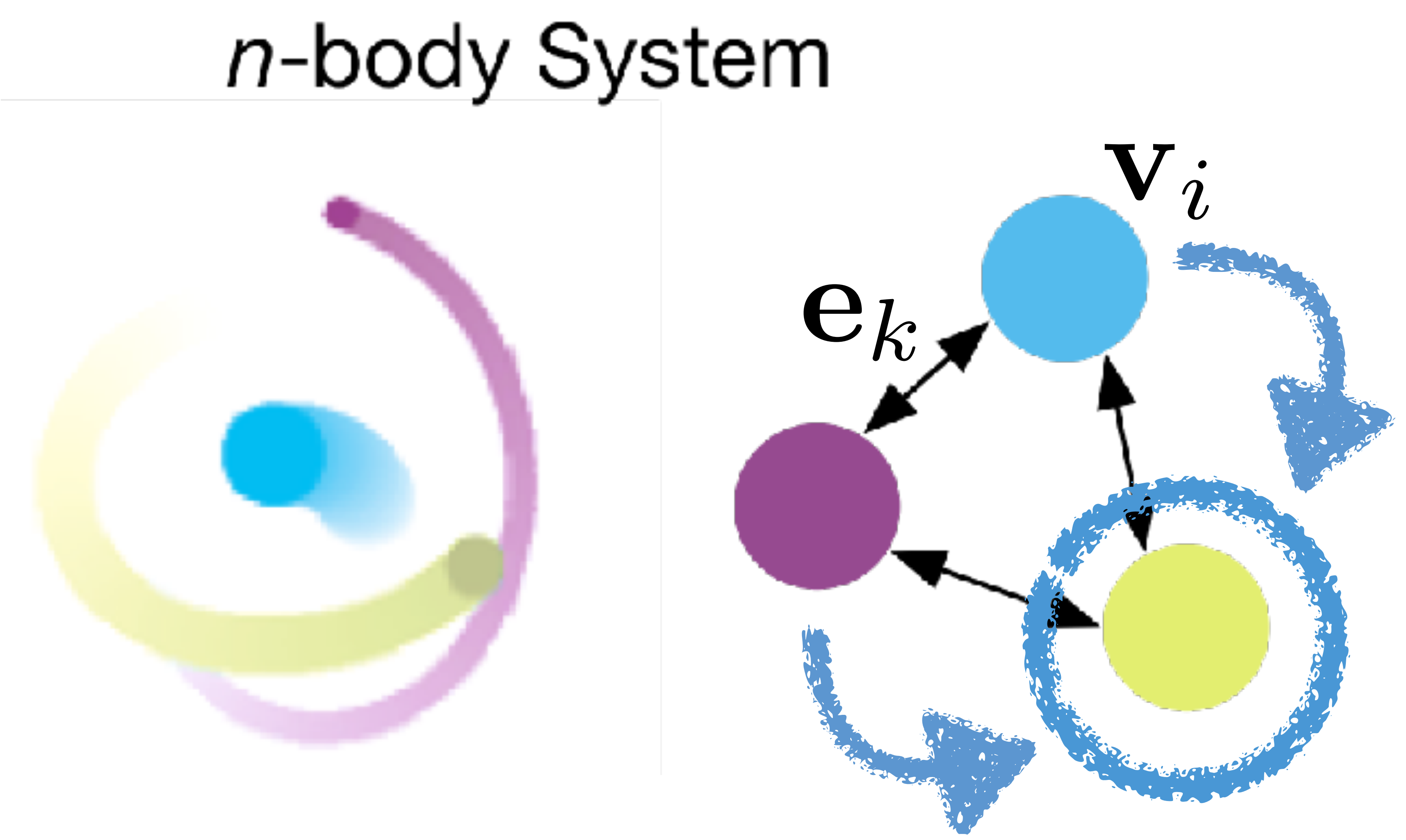
Node function

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

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

Battaglia et al., 2016, NeurIPS

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

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

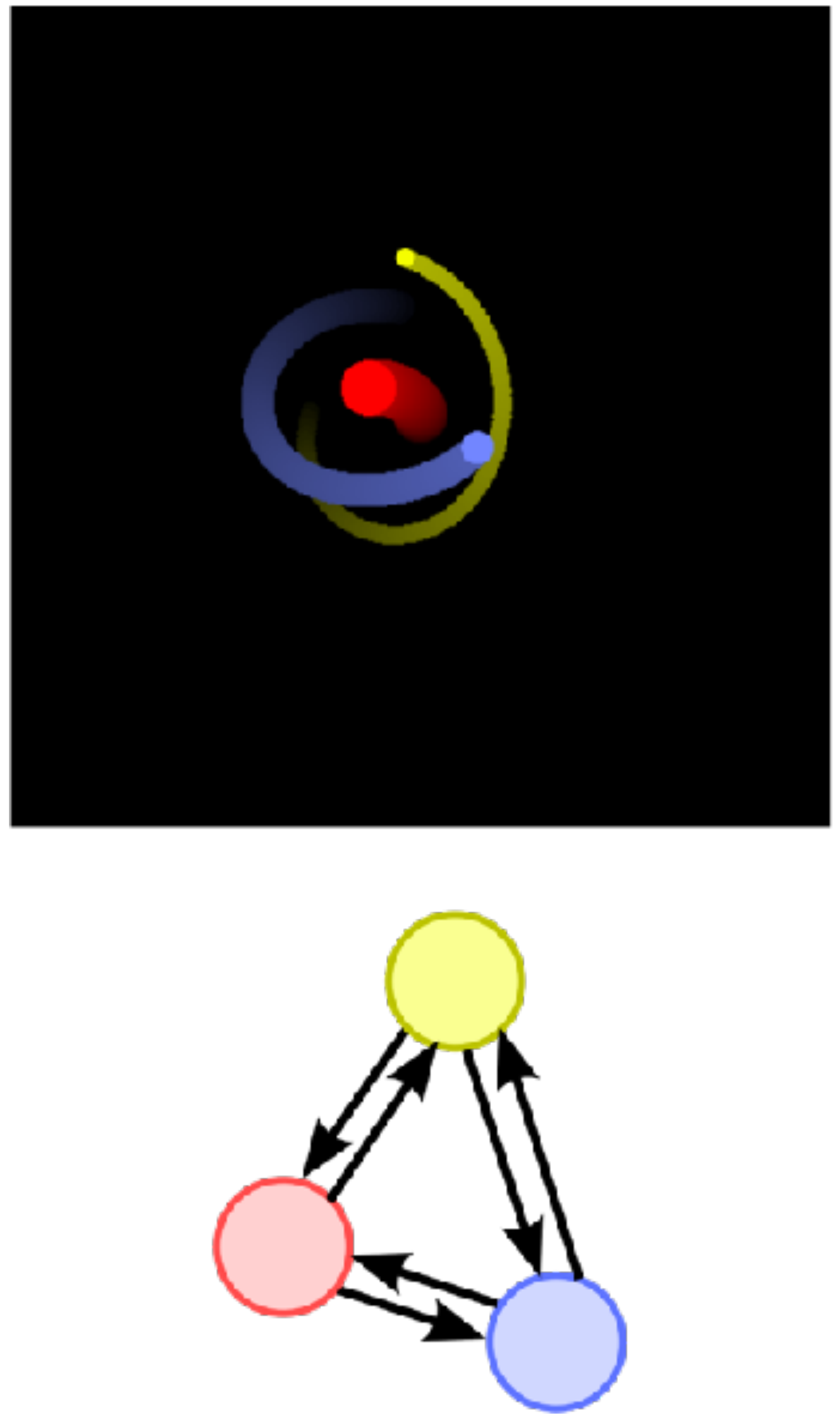
- Update node info from previous node state and aggregated "messages"

Trained to predict body coordinates

Battaglia et al., 2016, NeurIPS

Physical systems as graphs

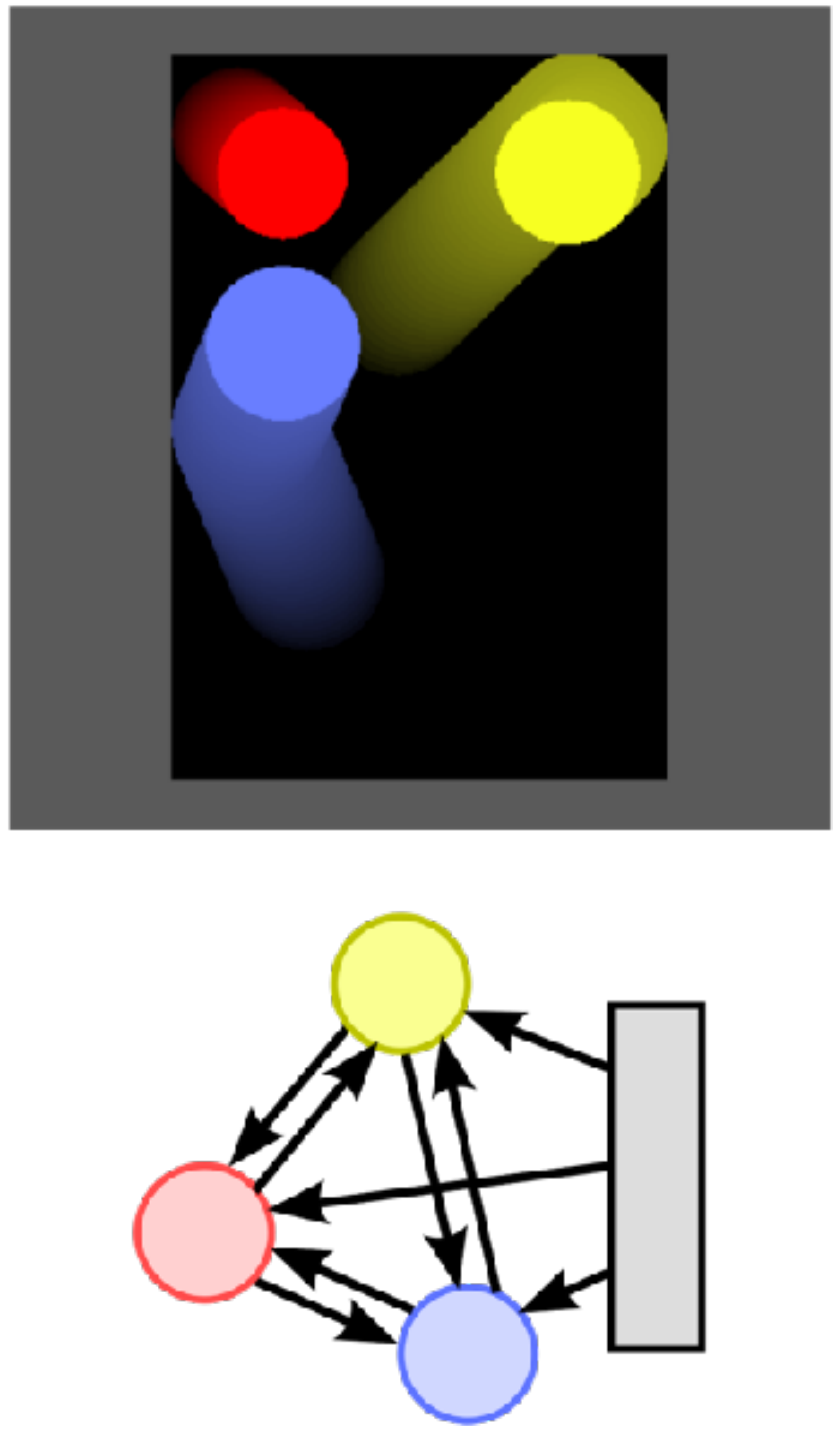
n-body



Nodes: bodies

Edges: gravitational forces

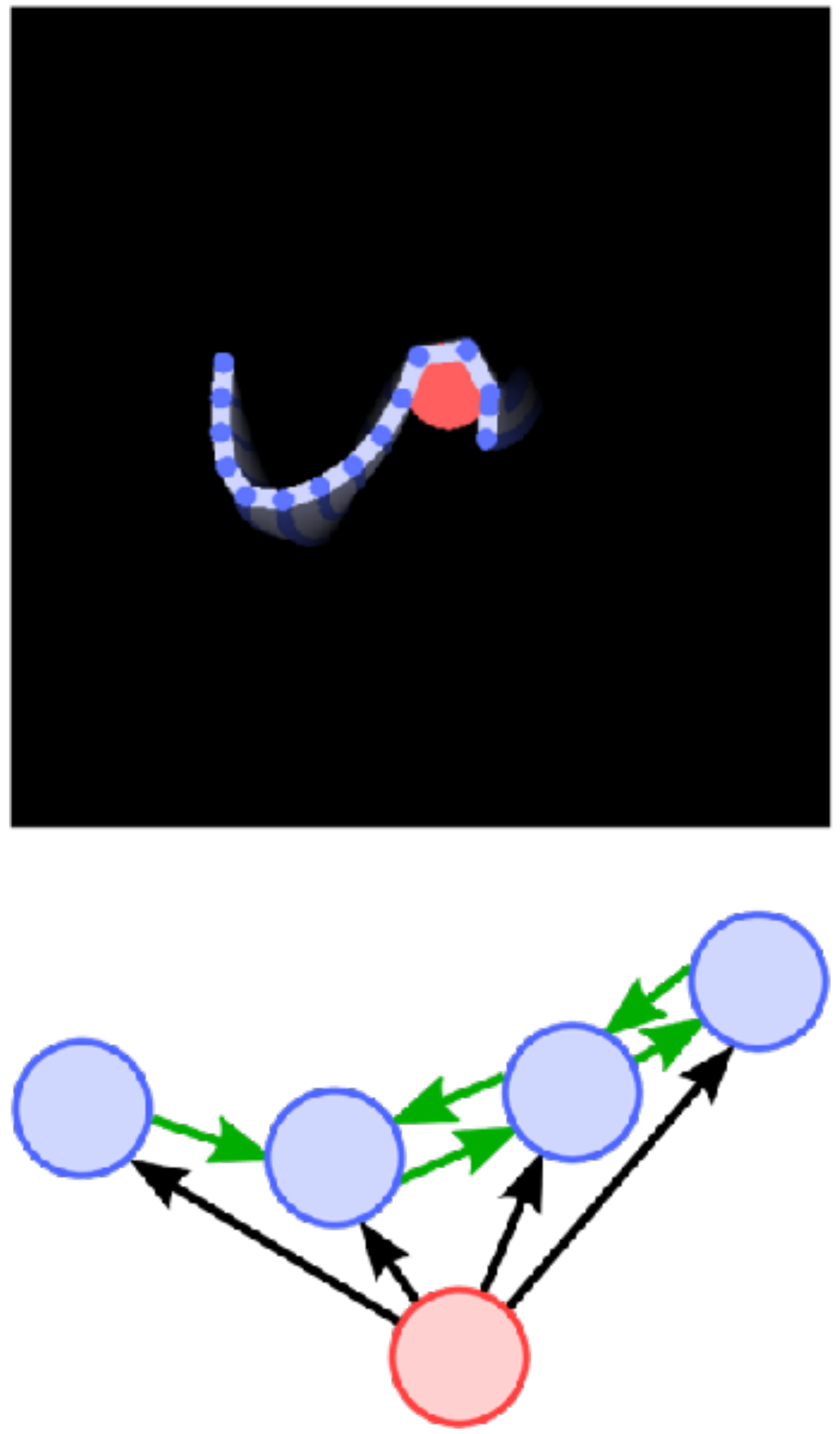
Balls



Nodes: balls

Edges: rigid collisions between balls, and walls

String

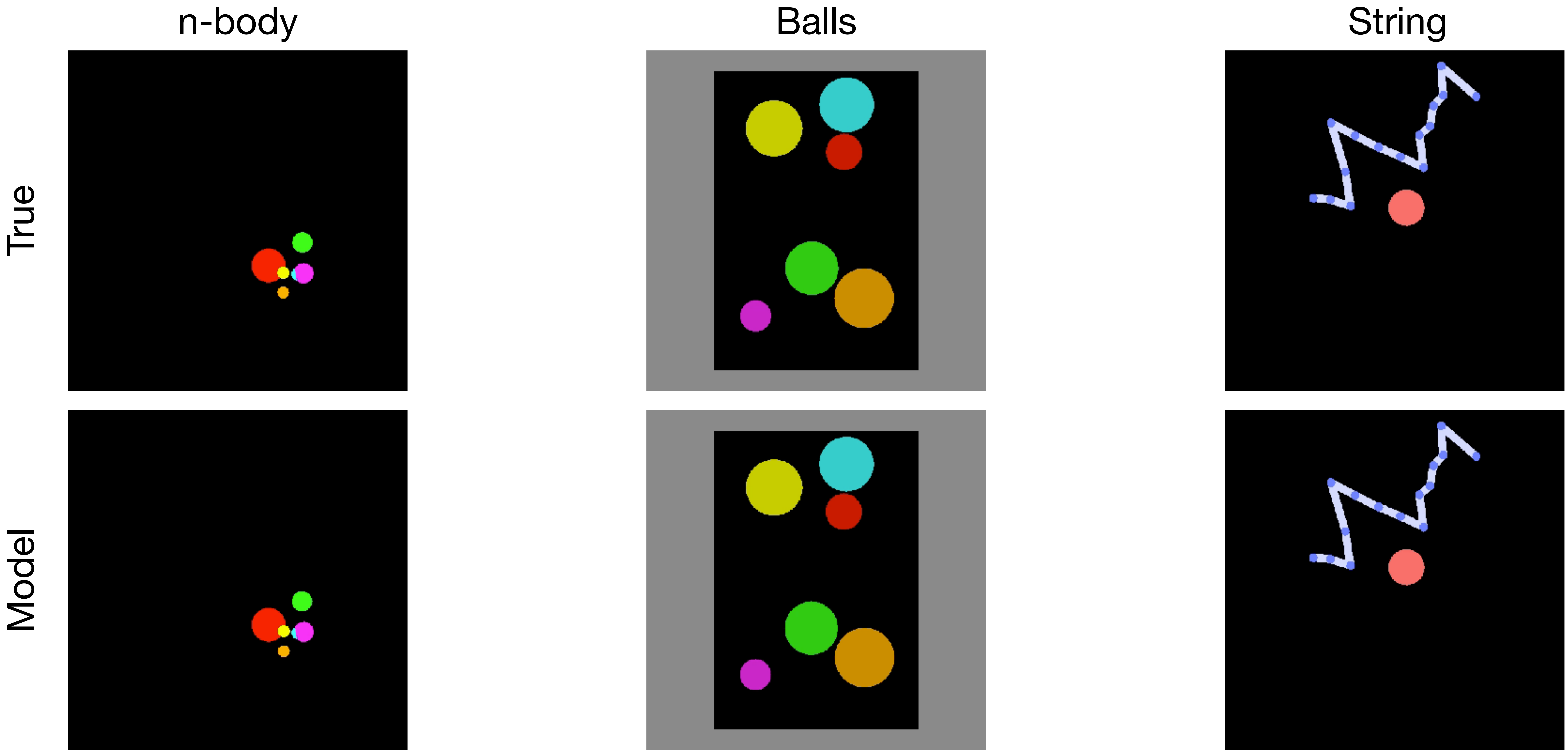


Nodes: masses

Edges: springs and rigid collisions

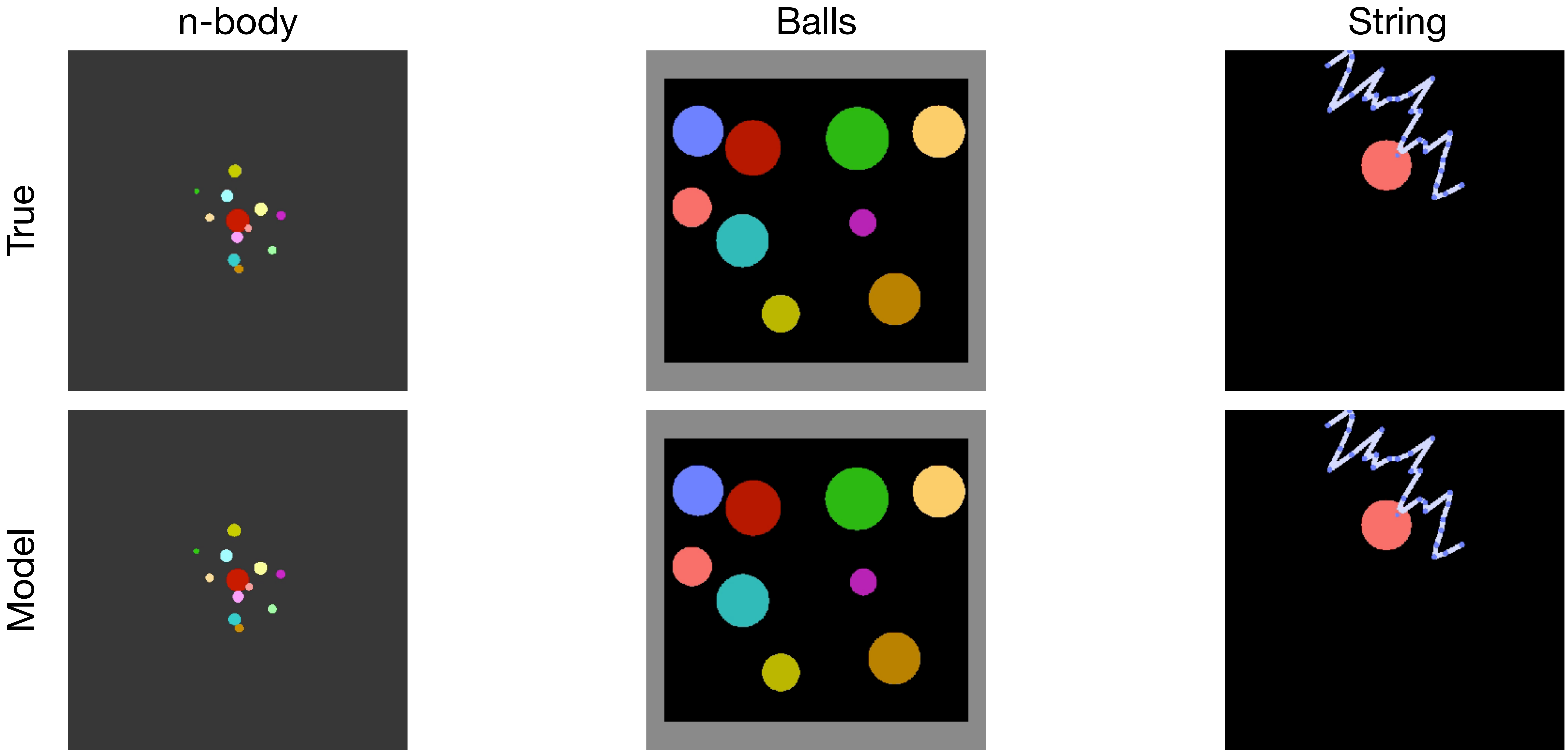
Battaglia et al., 2016, NeurIPS

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



Battaglia et al., 2016, NeurIPS

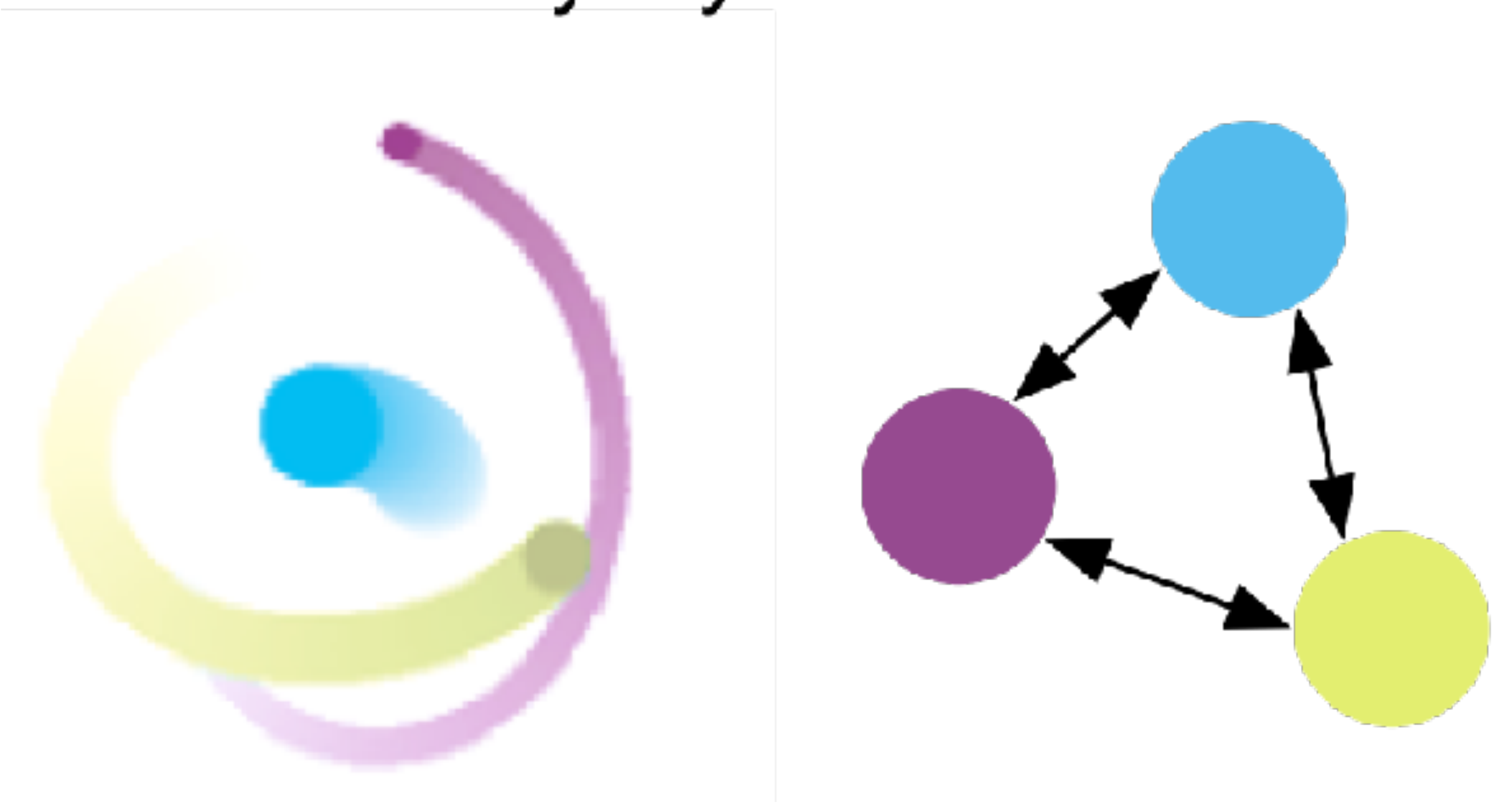
Zero-shot generalisation to larger systems



Battaglia et al., 2016, NeurIPS

Interaction Network: Predicting potential energy

n-body System



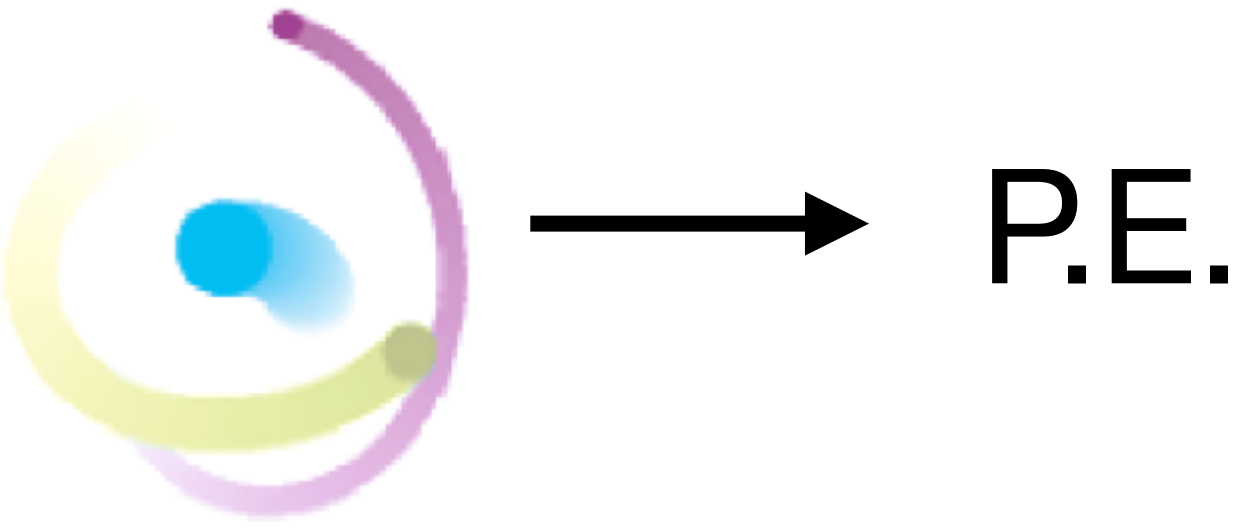
Node aggregation and global function

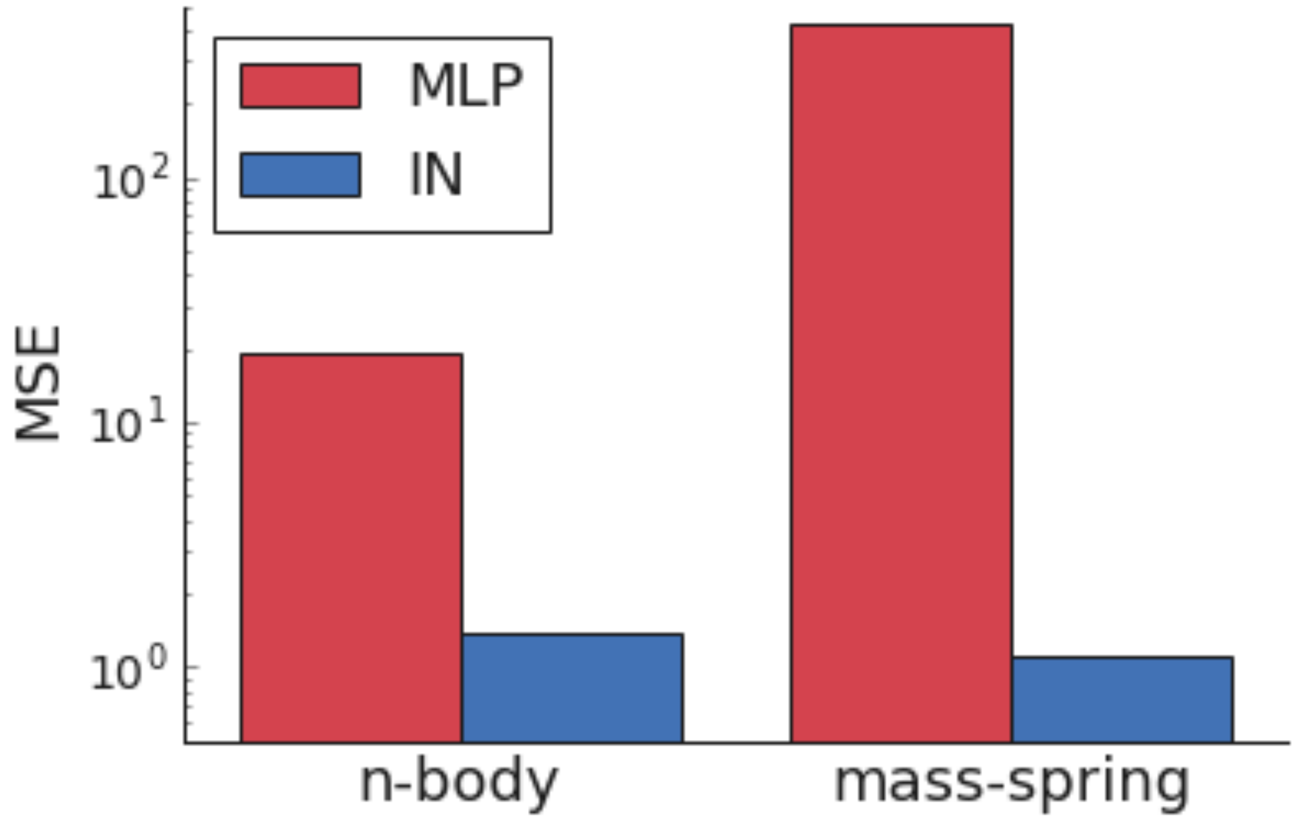
$$\bar{\mathbf{v}}' \leftarrow \sum_i \mathbf{v}'_i$$

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

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

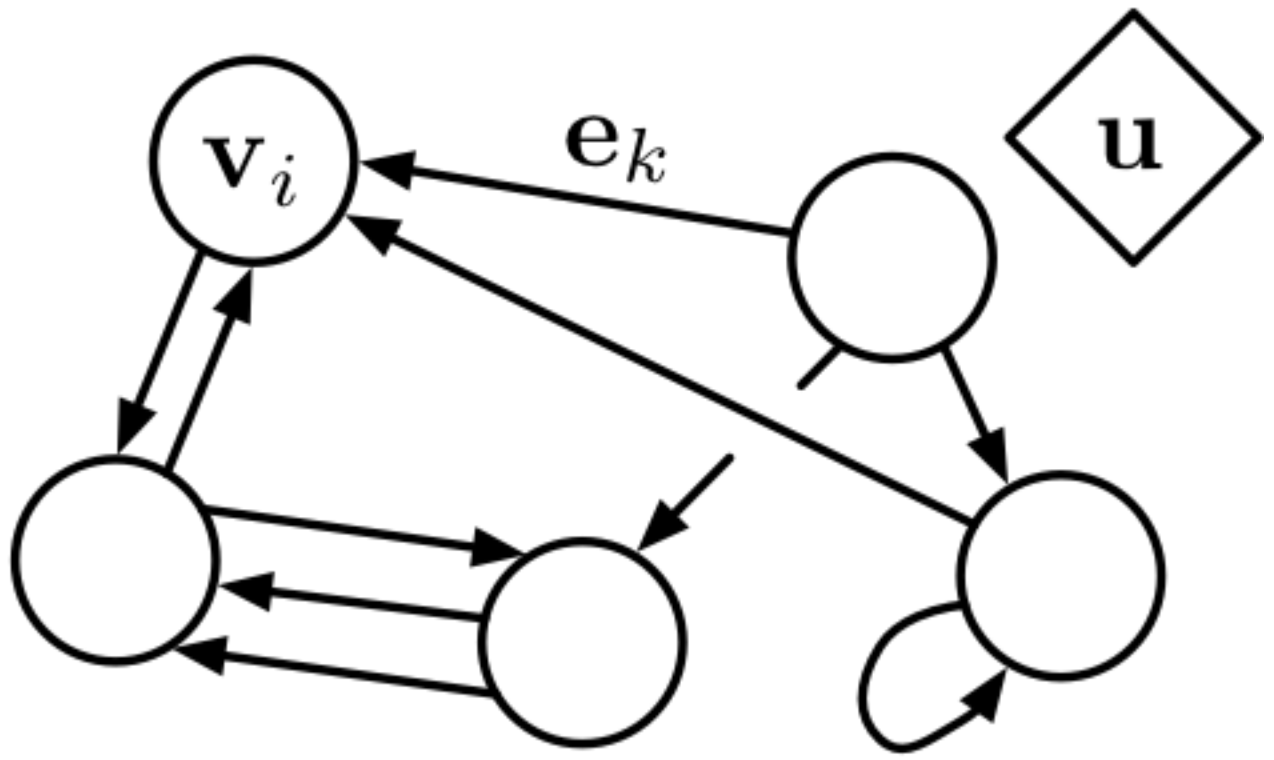
Trained to predict system's potential energy

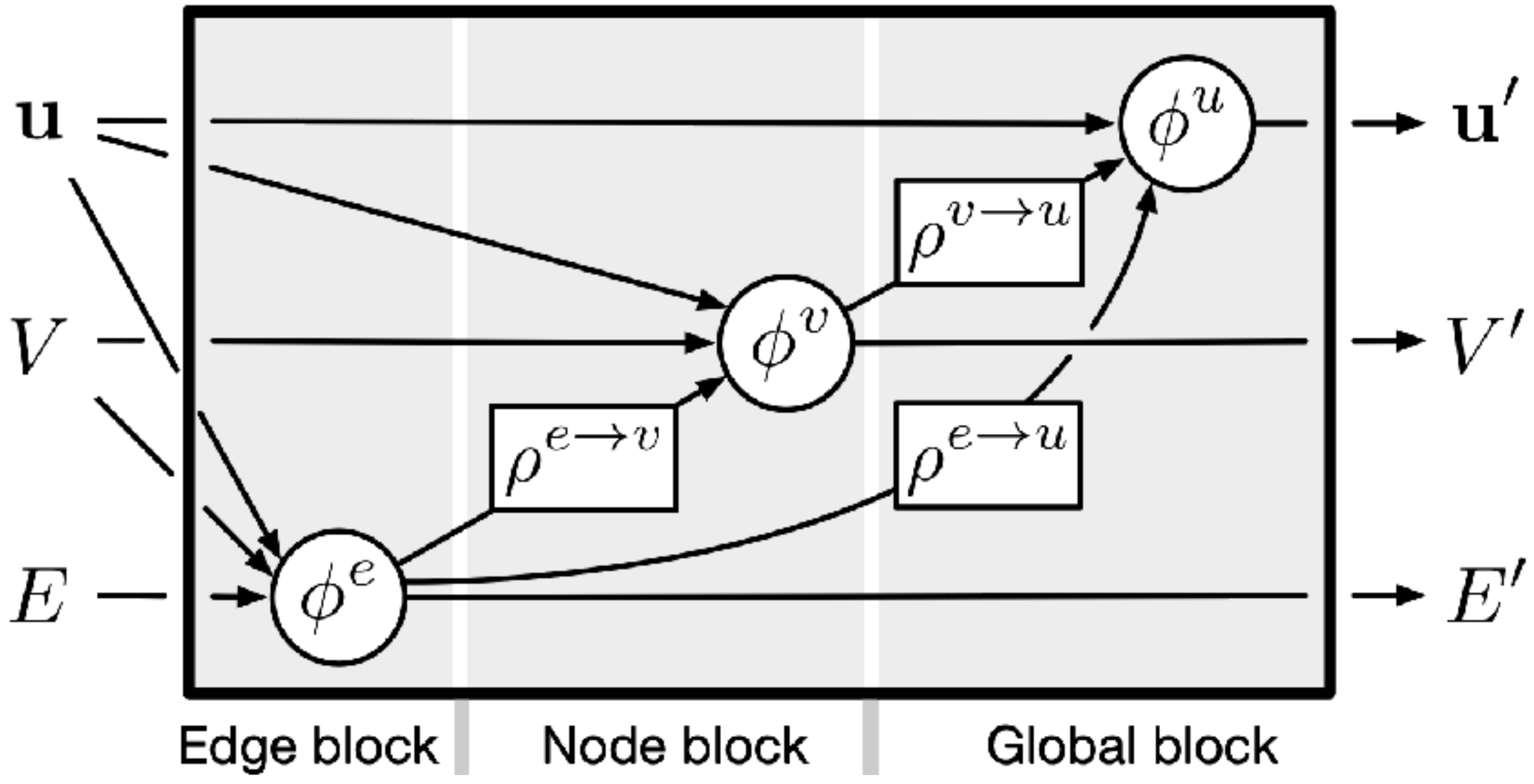




Battaglia et al., 2016, NeurIPS

Our "Graph Network" generalizes/extends "Interaction Network"

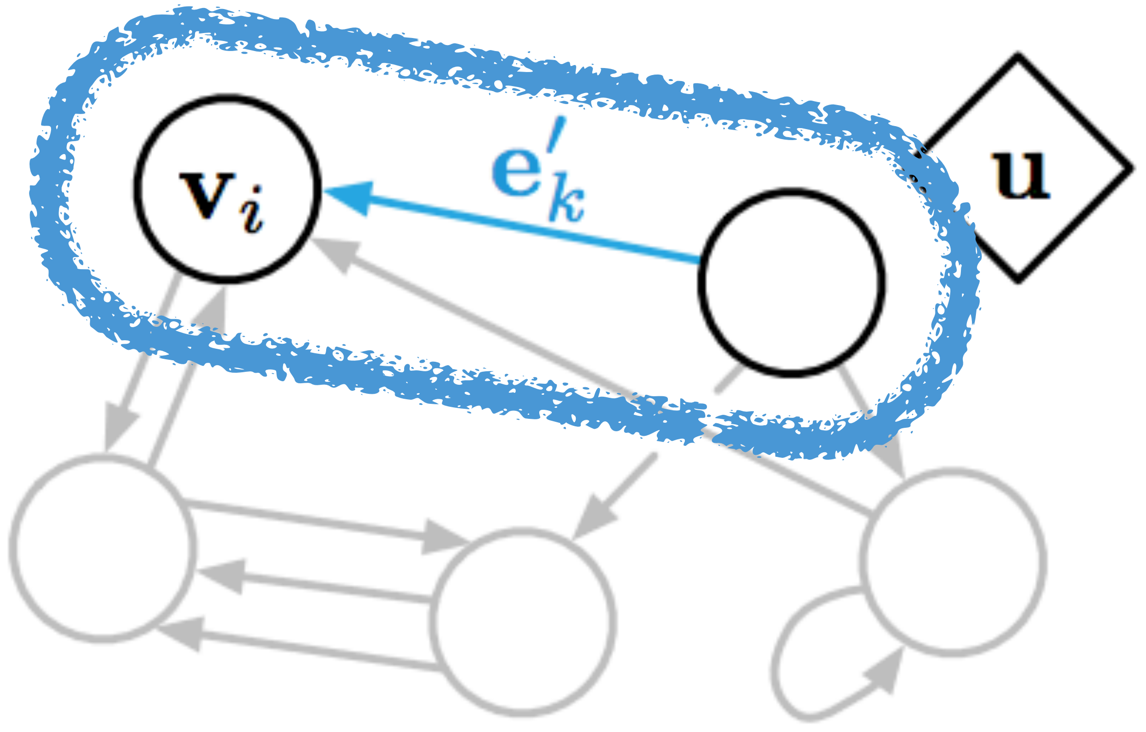




Edge block

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

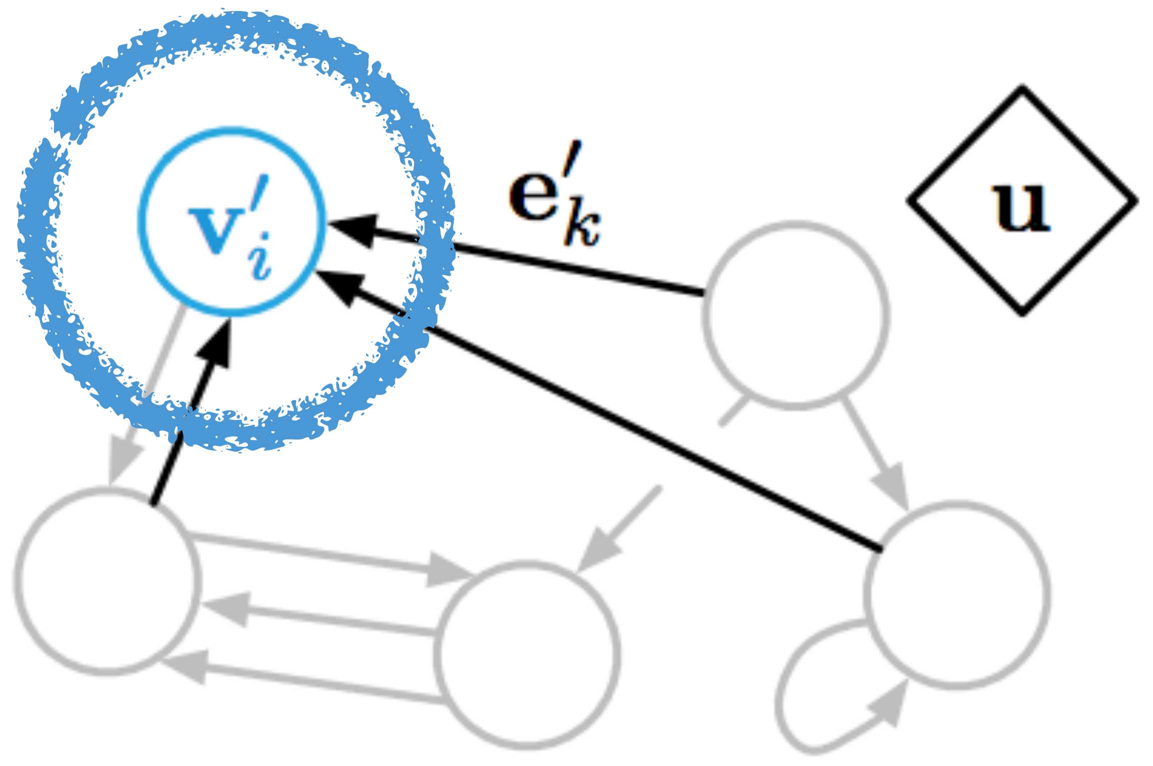
$$\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$$



Node block

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

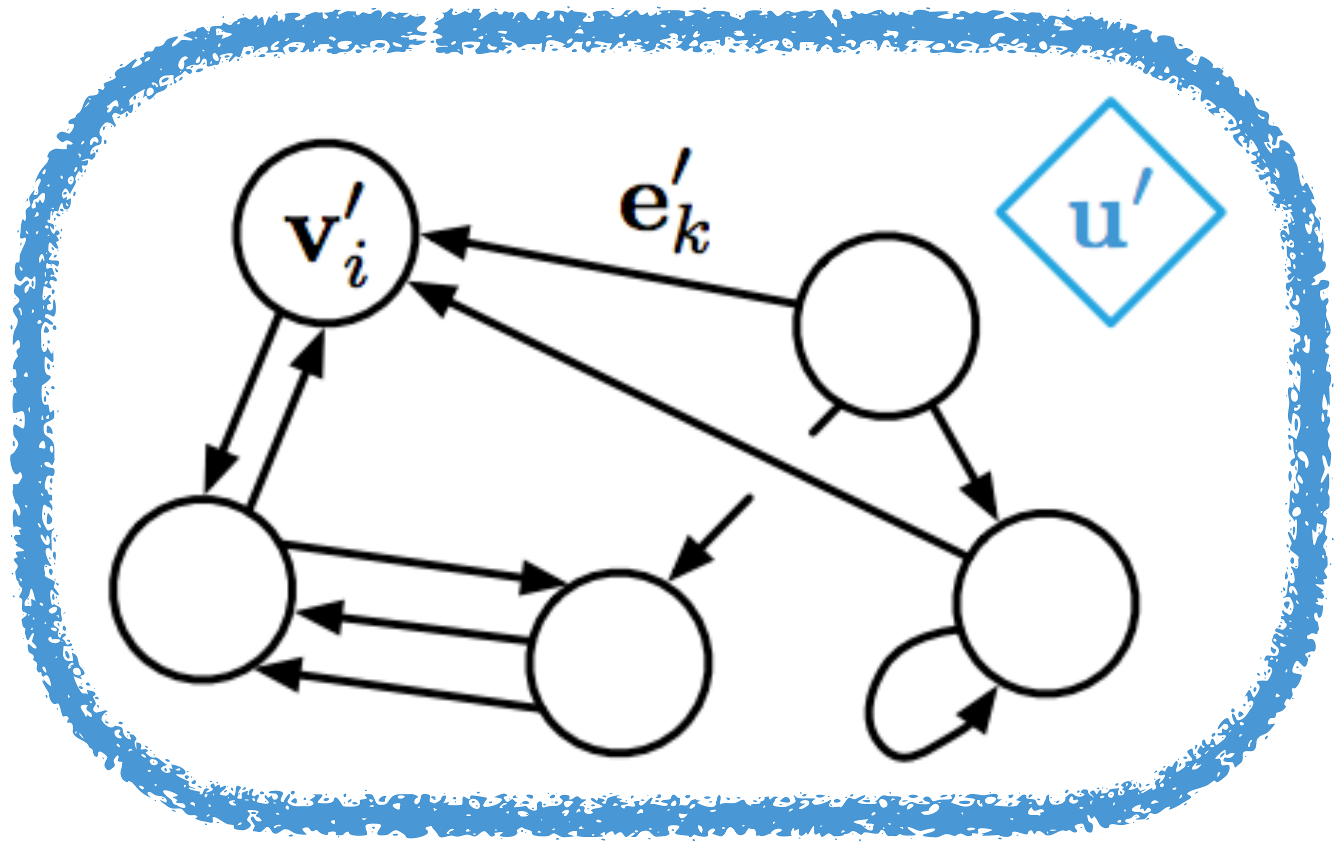
$$\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$$



Global block

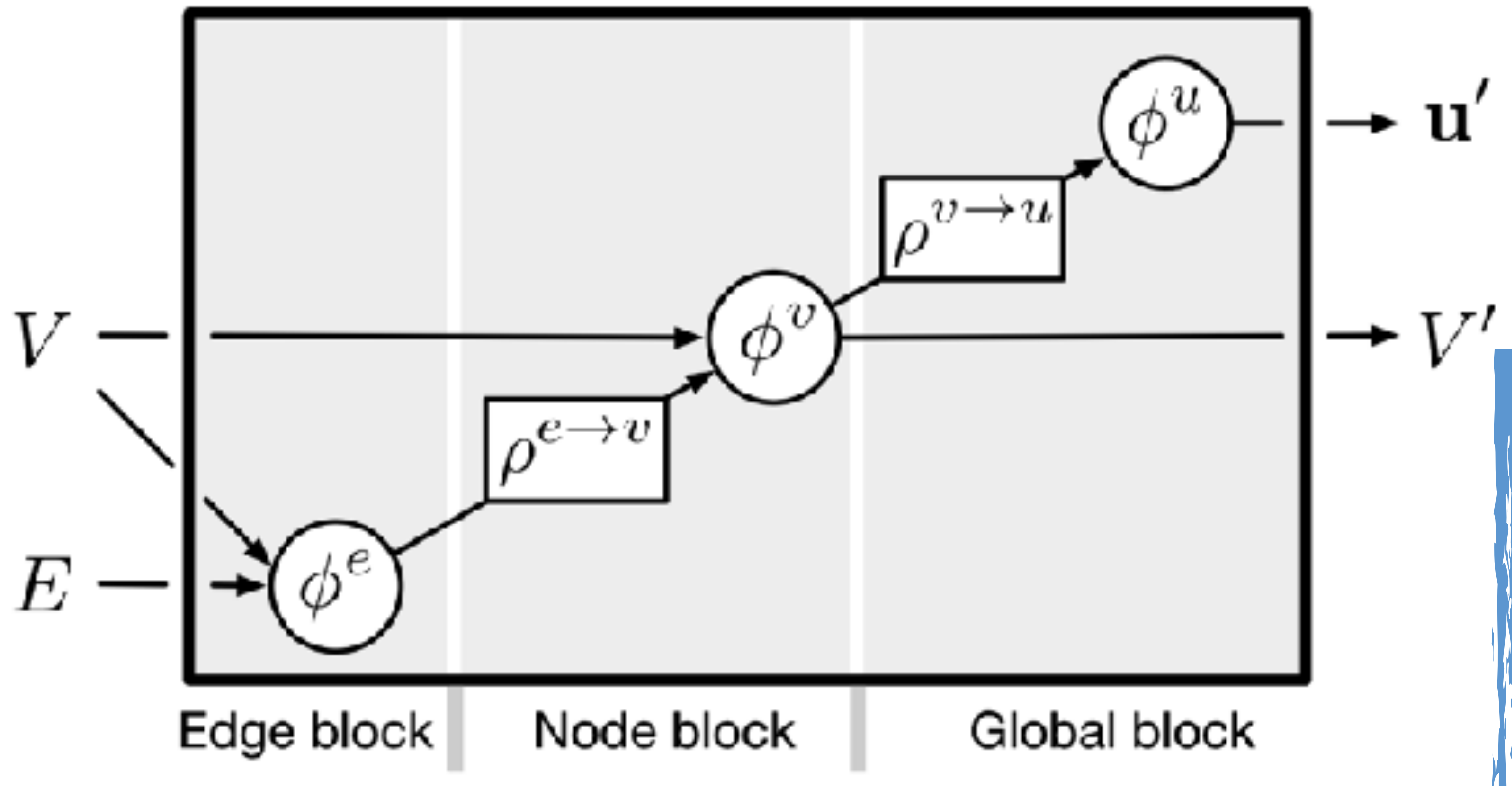
Across the graph, $\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}$, are passed to a "global function":

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



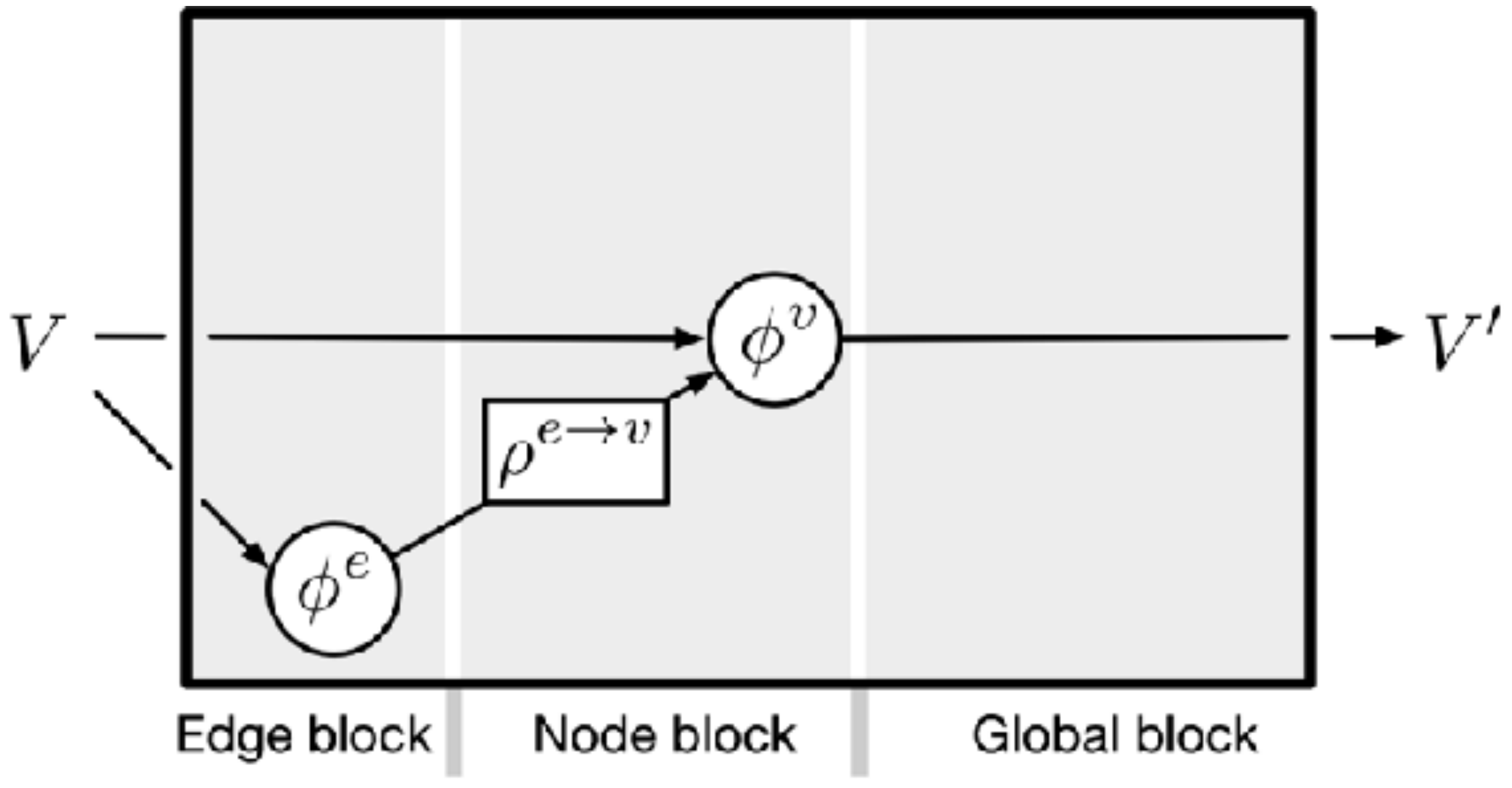
Message-Passing NN (eg. Interaction Net, GCN)

Gilmer et al. 2017



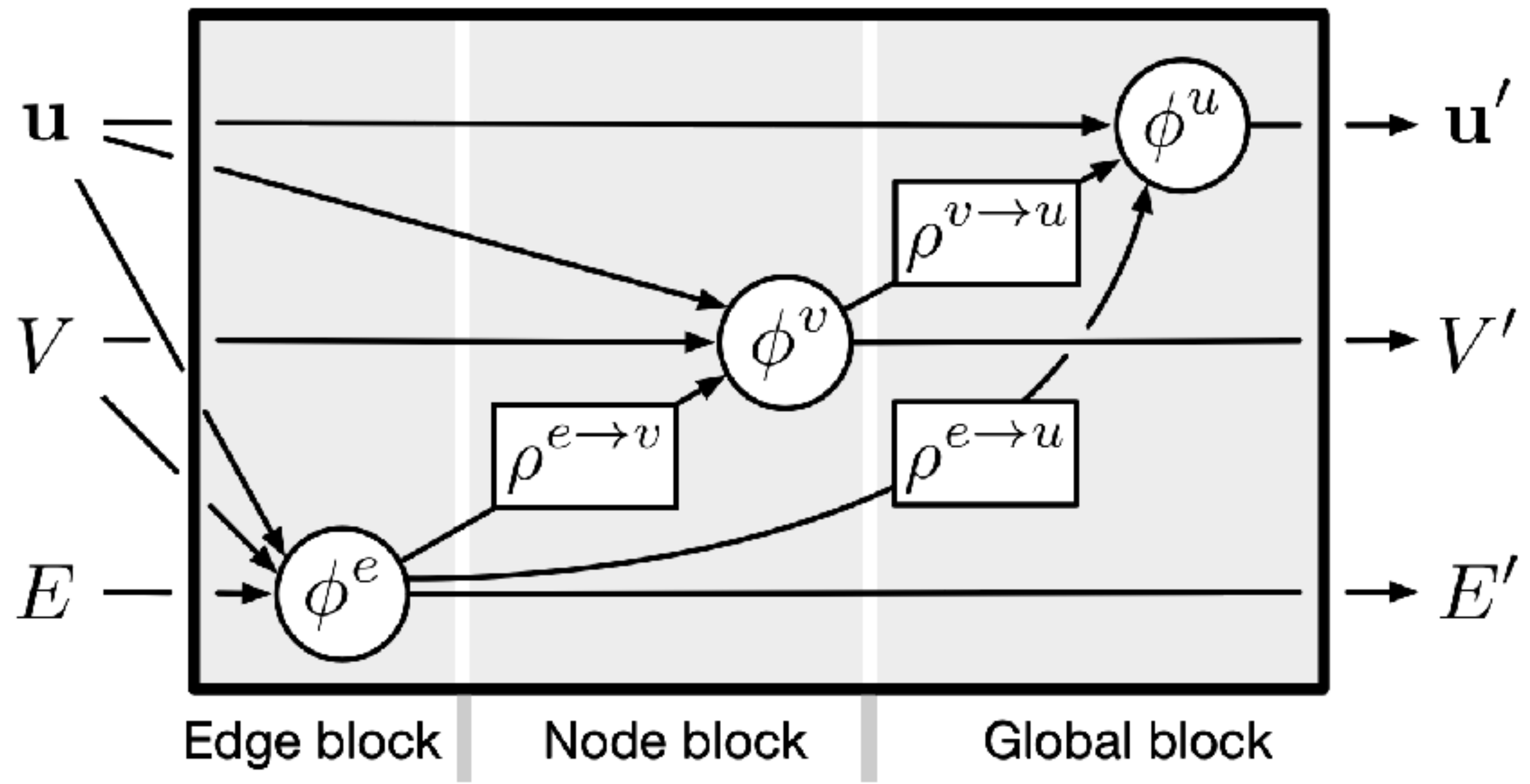
Non-Local NN (eg. Transformer)

Vaswani et al. 2017; Wang et al. 2017



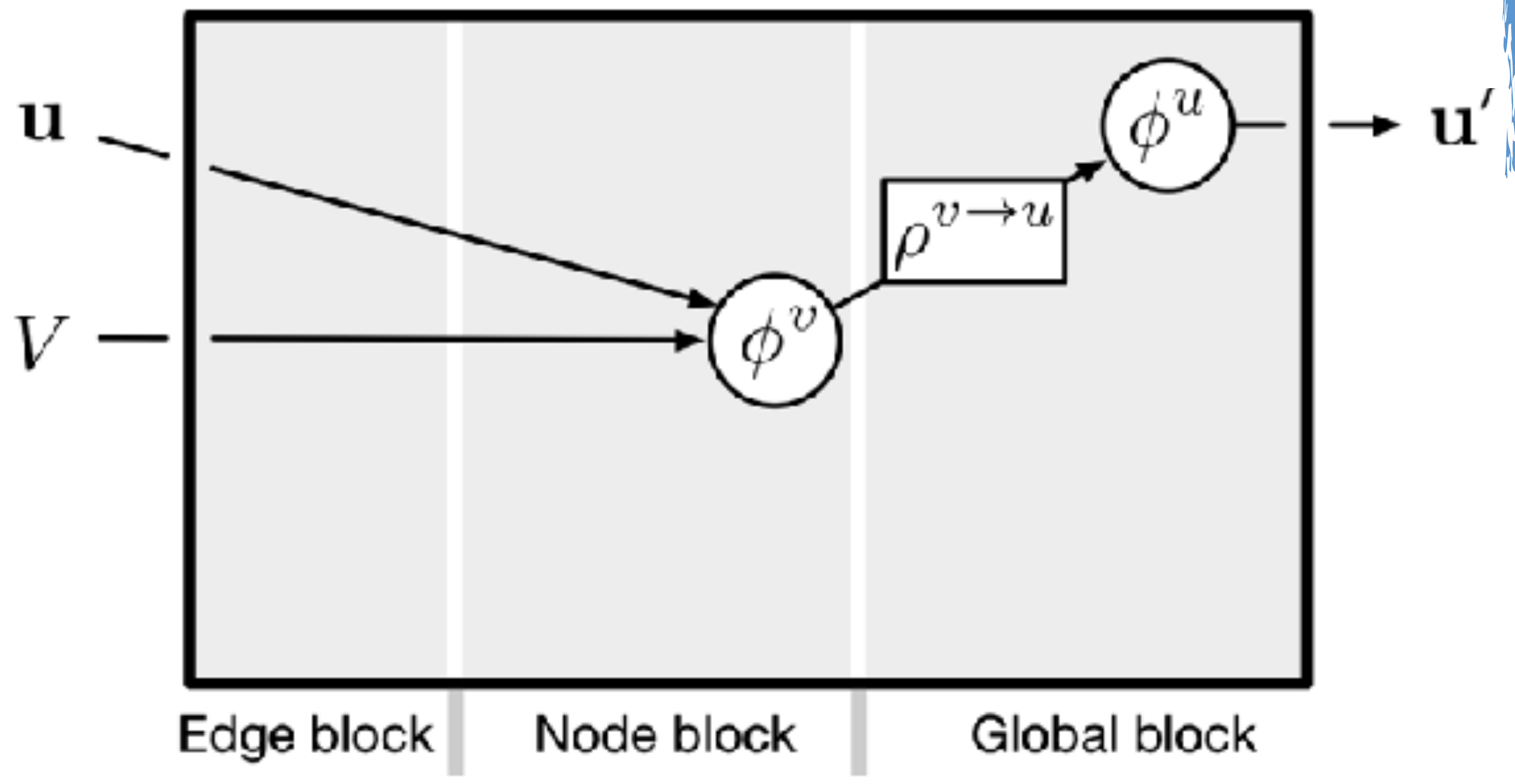
Graph Network
(a type of Graph Neural Network)

Battaglia et al. 2018



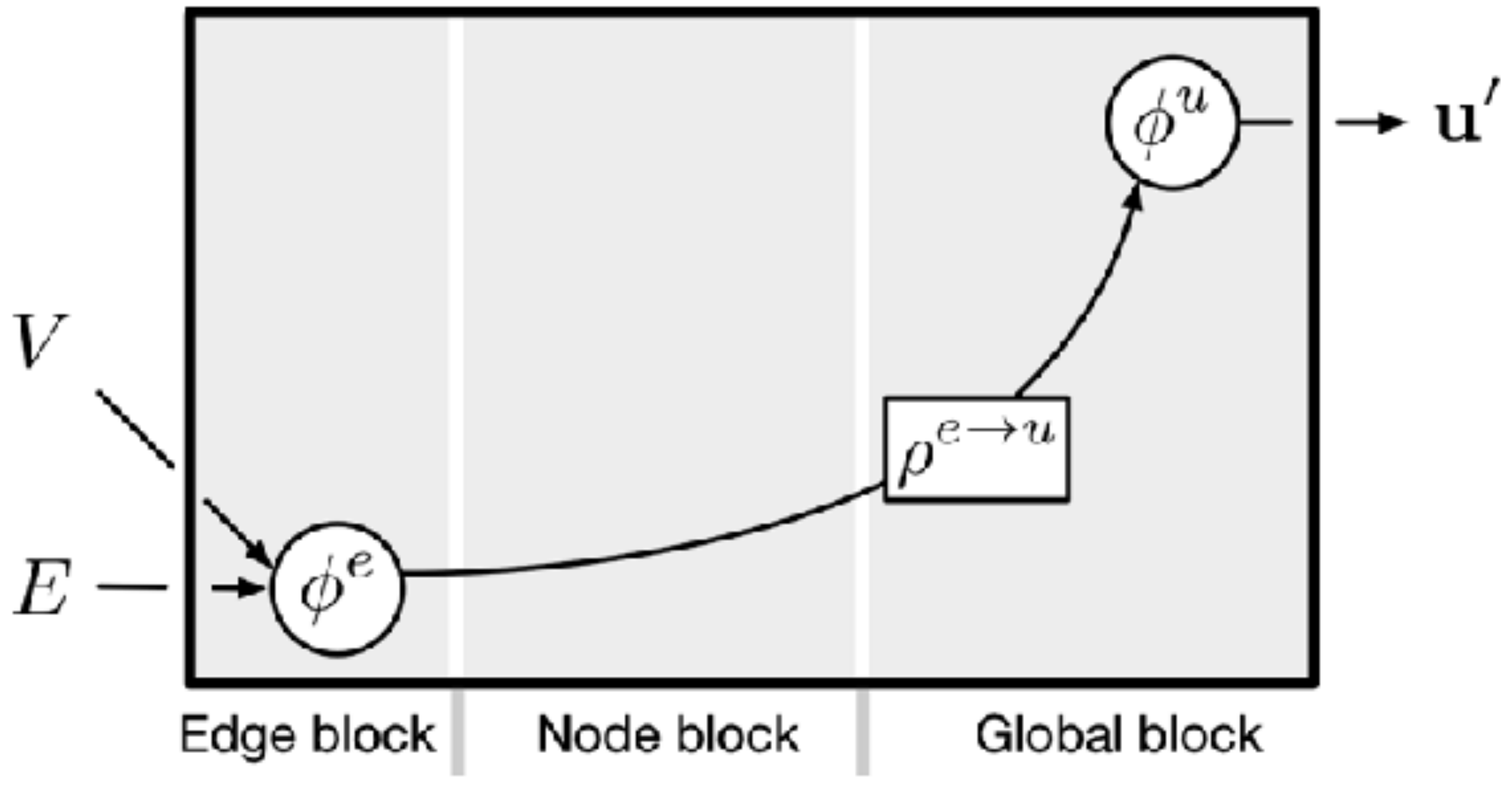
Deep Sets

Zhang et al. 2017



Relation Network

Raposo et al. 2017; Santoro et al. 2017

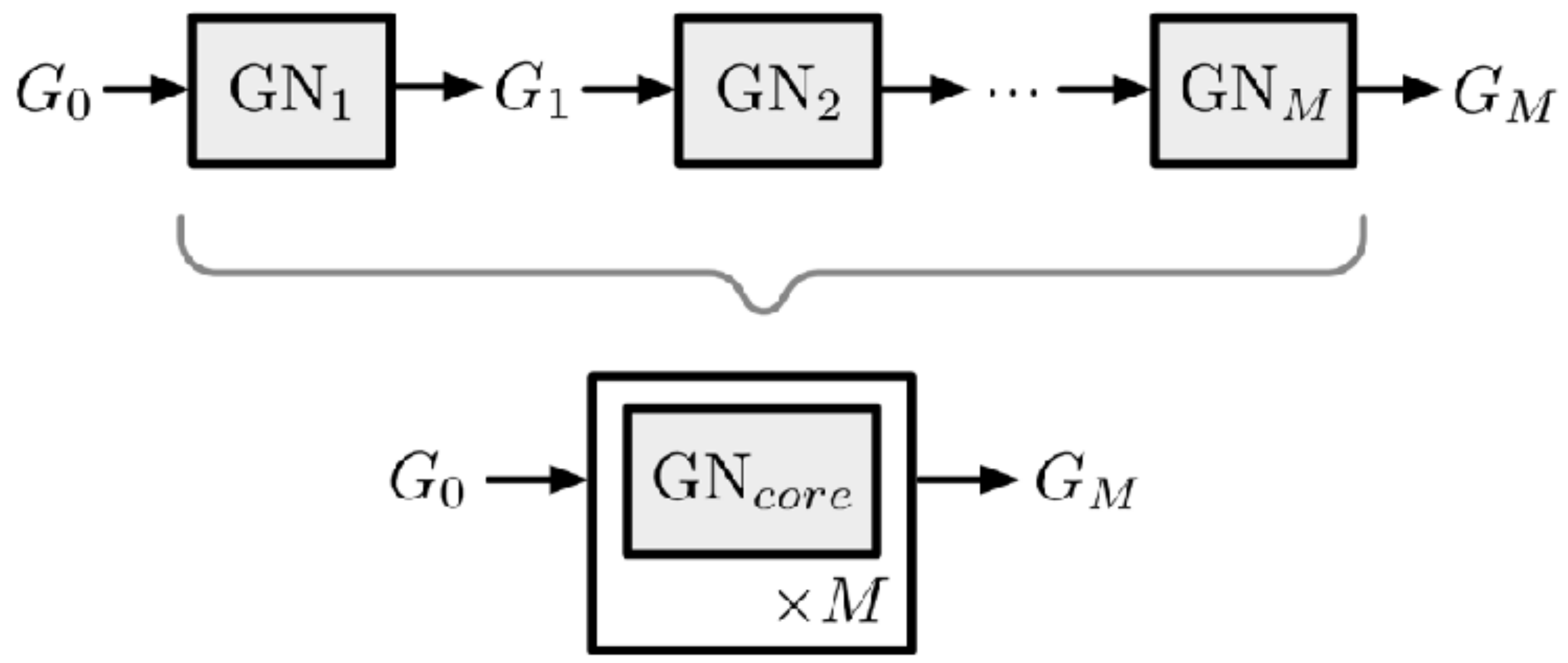


Battaglia et al., 2018, arXiv

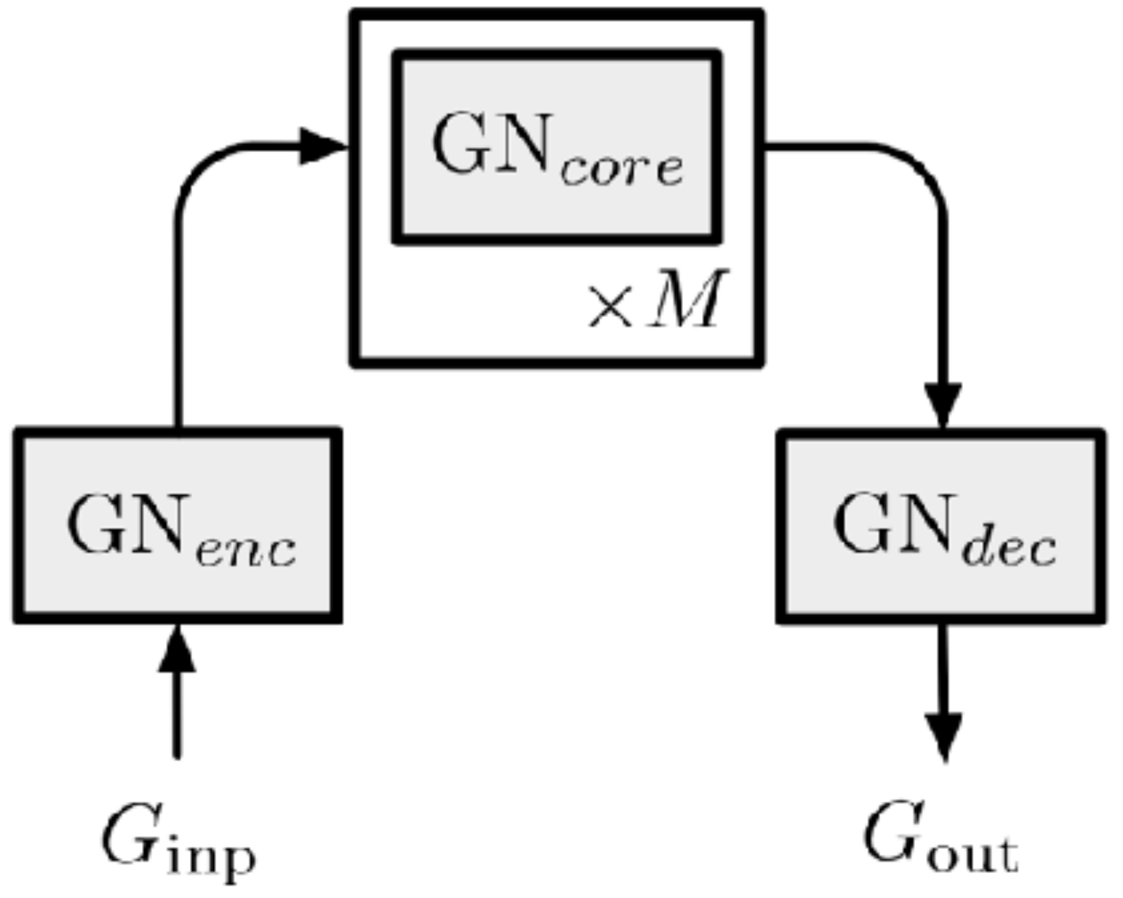
Composing GN blocks

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

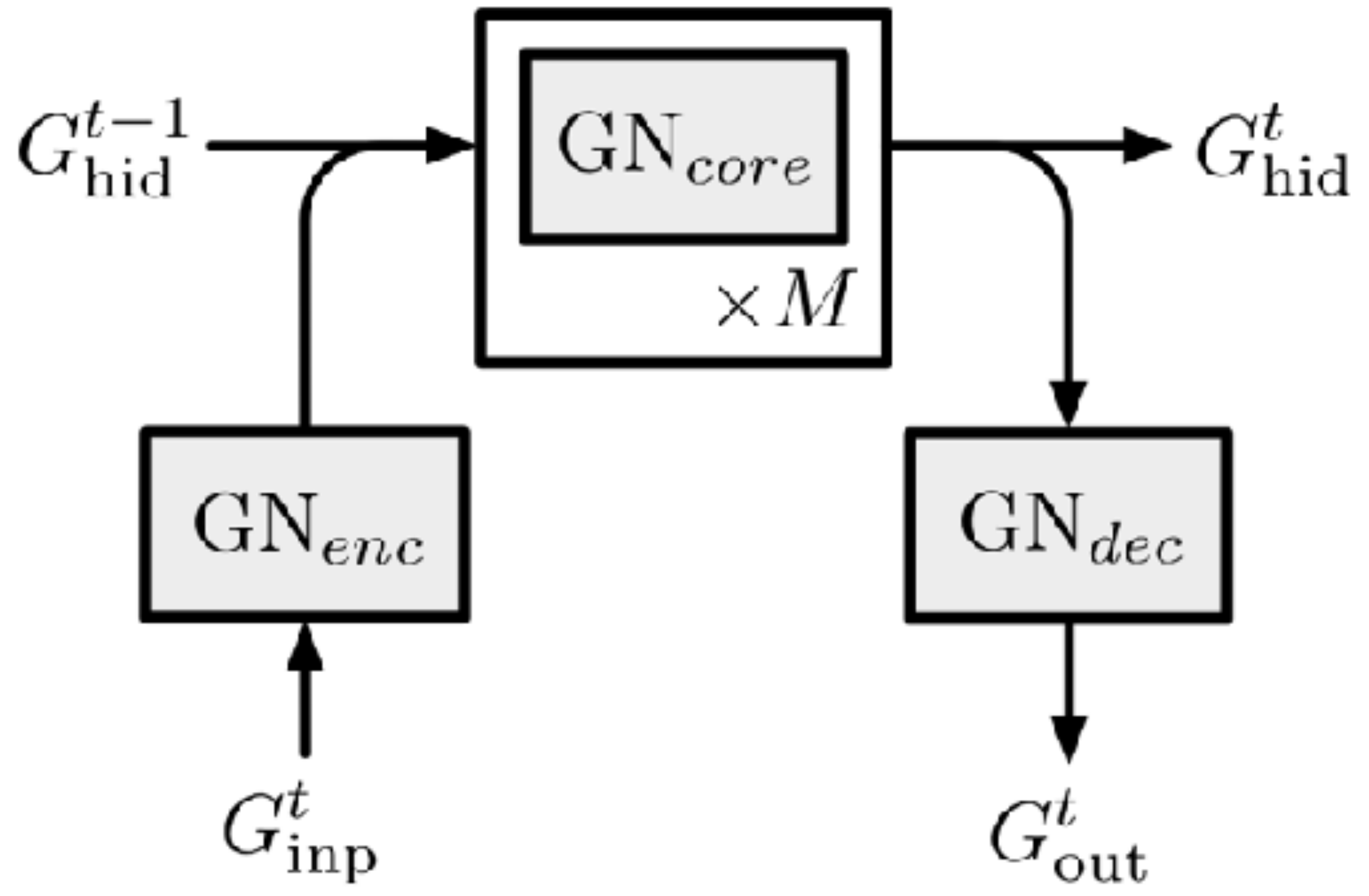
Shared GN core



Encode-process-decode

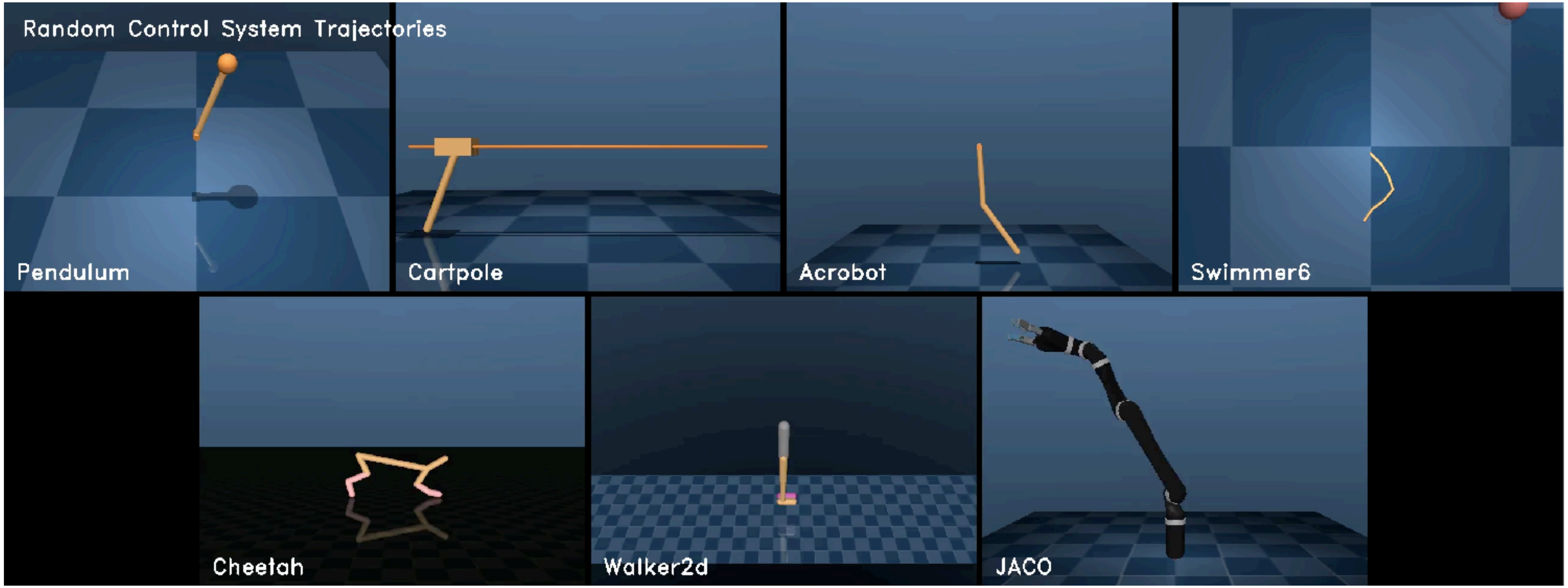


Recurrent GN architecture



Battaglia et al., 2018, arXiv

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



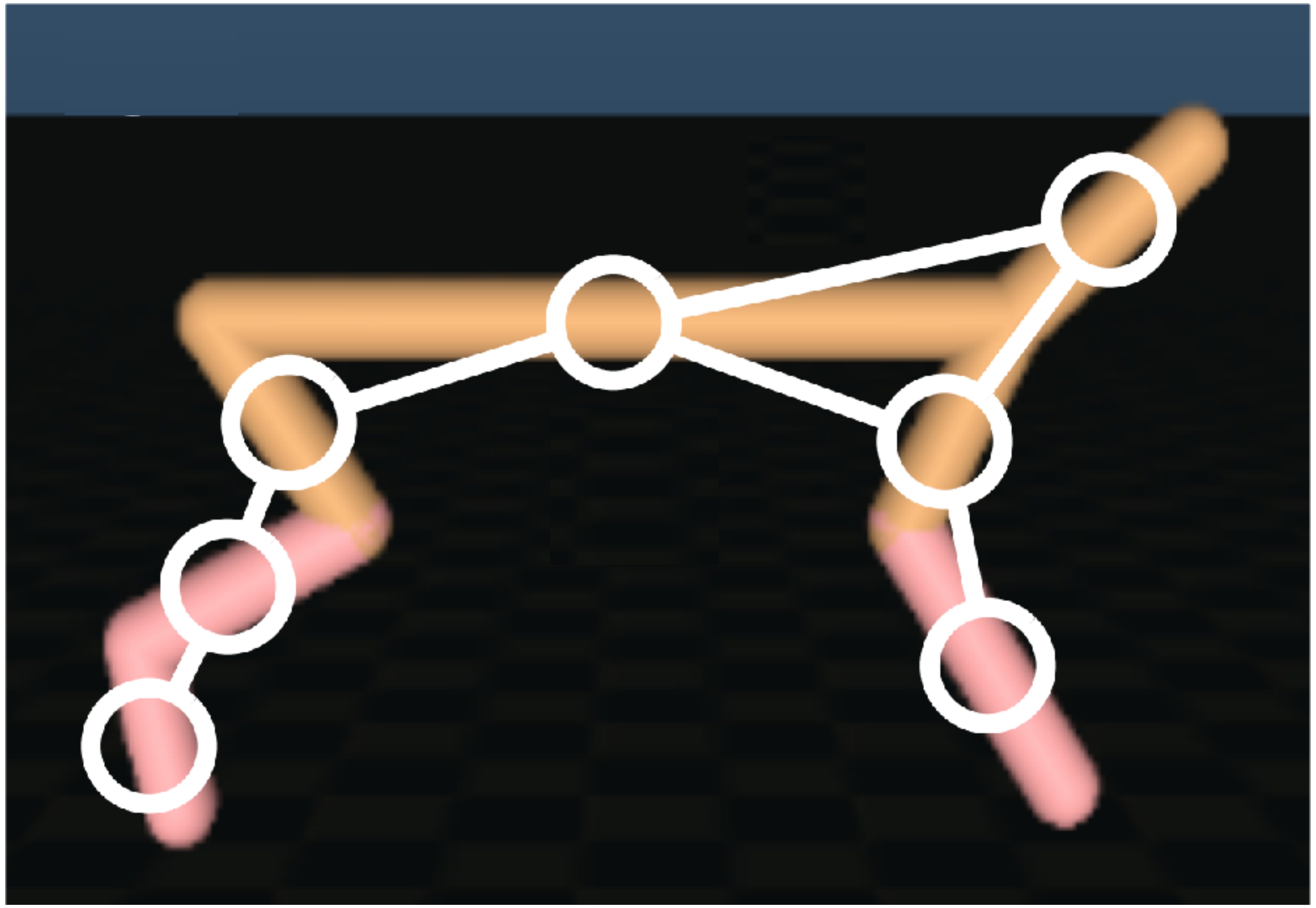
DeepMind Control Suite (Tassa et al., 2018)

JACO Arm

Kinematic tree of the actuated system as a graph

Controllable physical system as a graph:

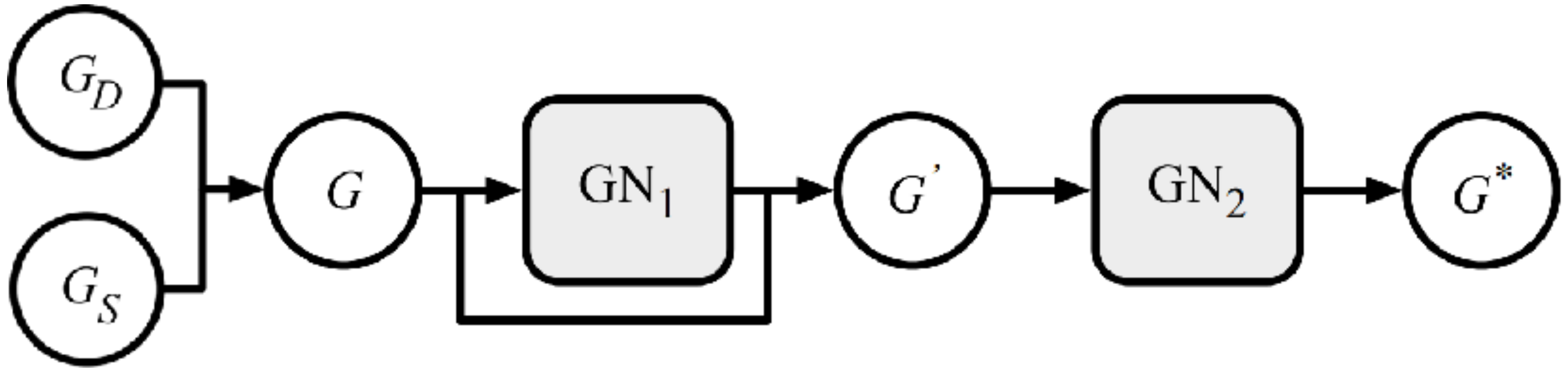
- Bodies → Nodes
- Joints → Edges
- Global properties



Sanchez-Gonzalez et al., 2018, ICML

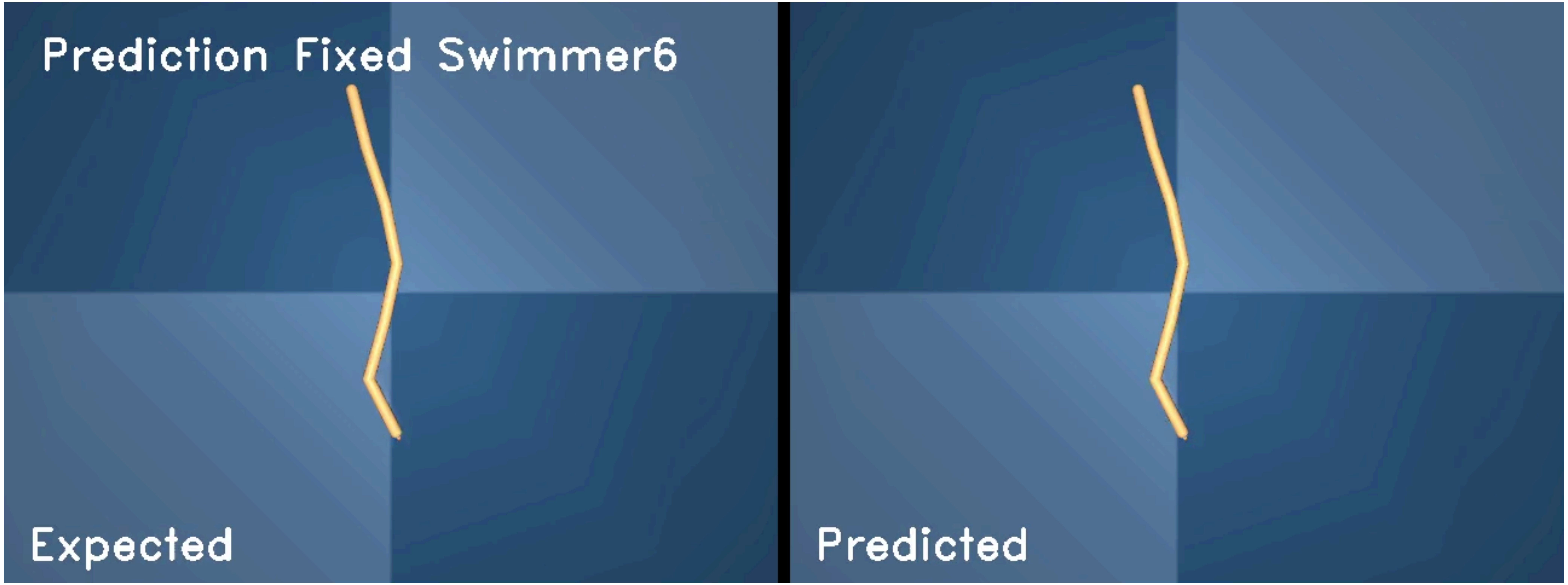
Forward model: supervised, 1-step training w/ random control inputs

Input graph (t)



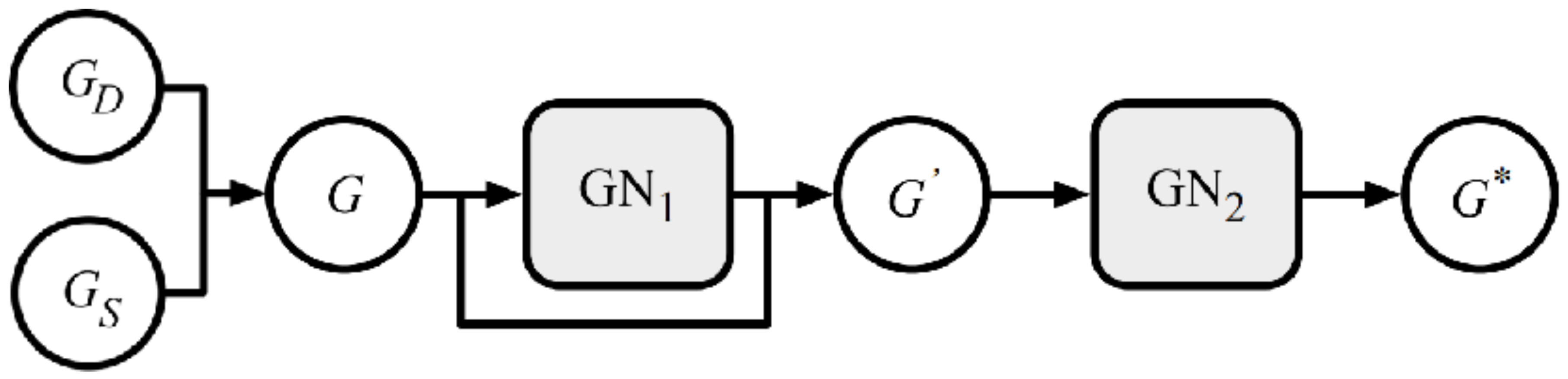
Next graph (t+1)

Chained 100-step predictions



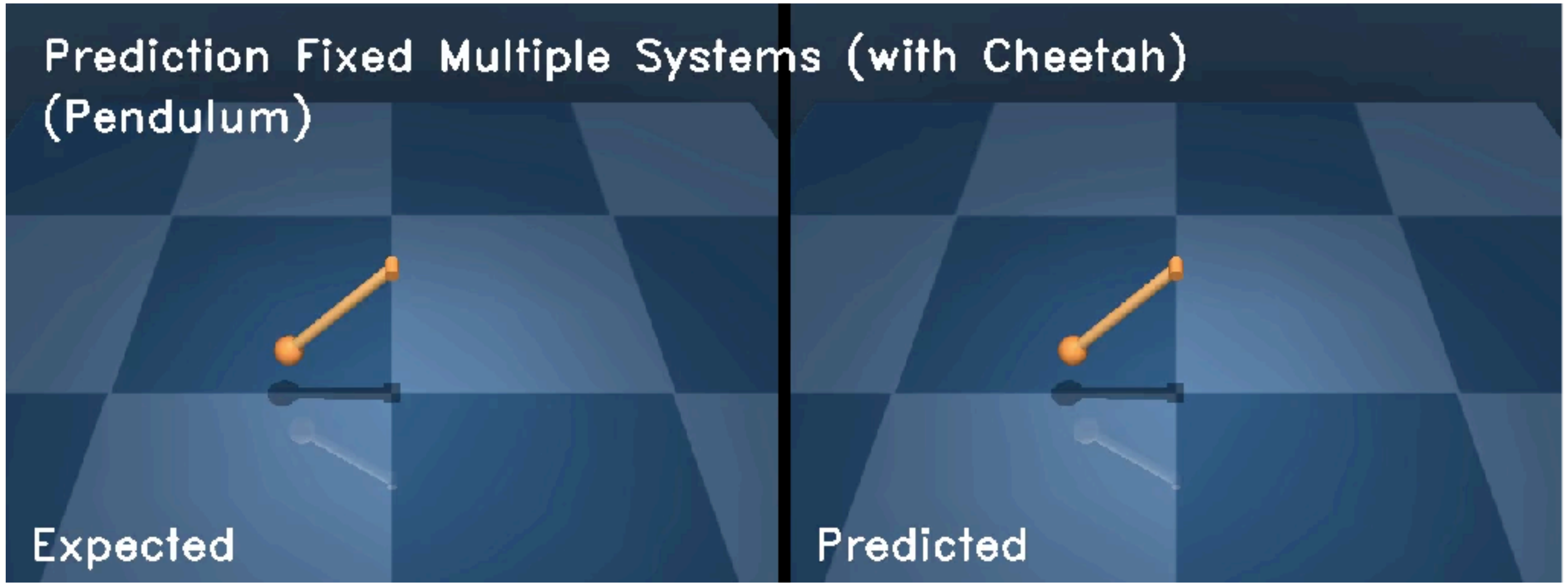
Sanchez-Gonzalez et al., 2018, ICML

Forward model: Multiple systems & zero-shot generalization



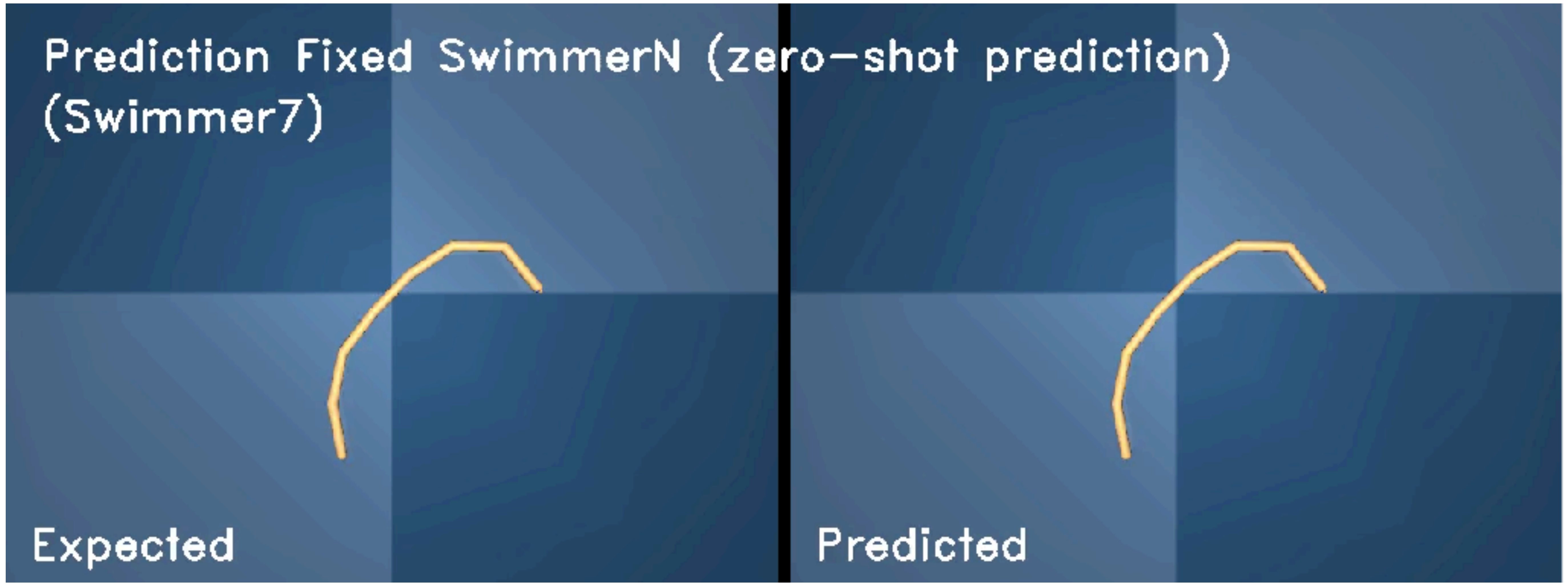
Single model trained:

- Pendulum, Cartpole, Acrobot, Swimmer6 & Cheetah



Zero-shot generalization: Swimmer

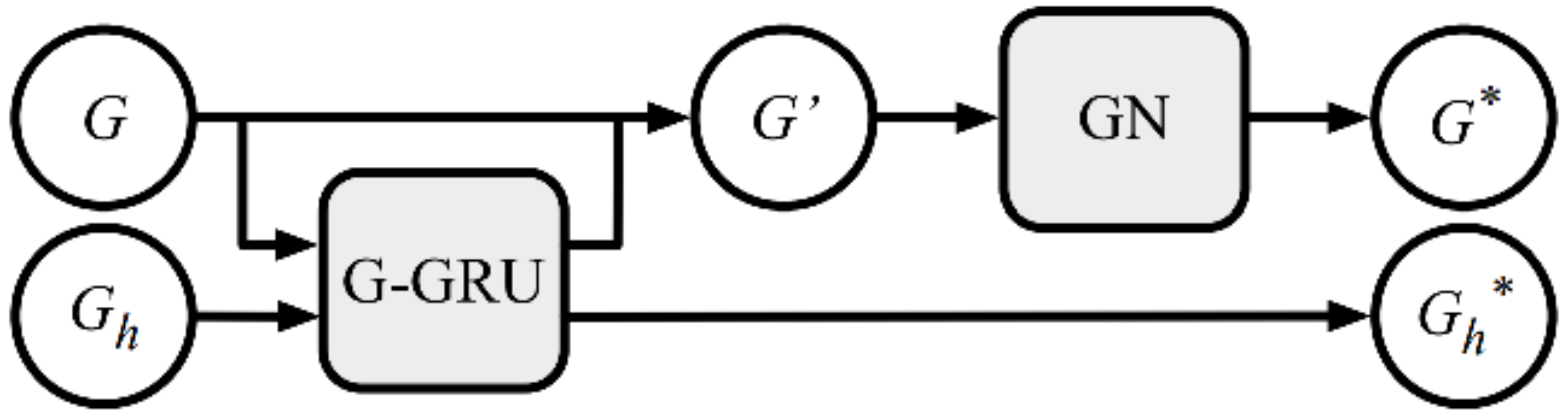
- # training links: {**3**, **4**, **5**, **6**, -, **8**, **9**, -, -, …}
- # testing links: {-, -, -, -, **7**, -, -, **10-14**}

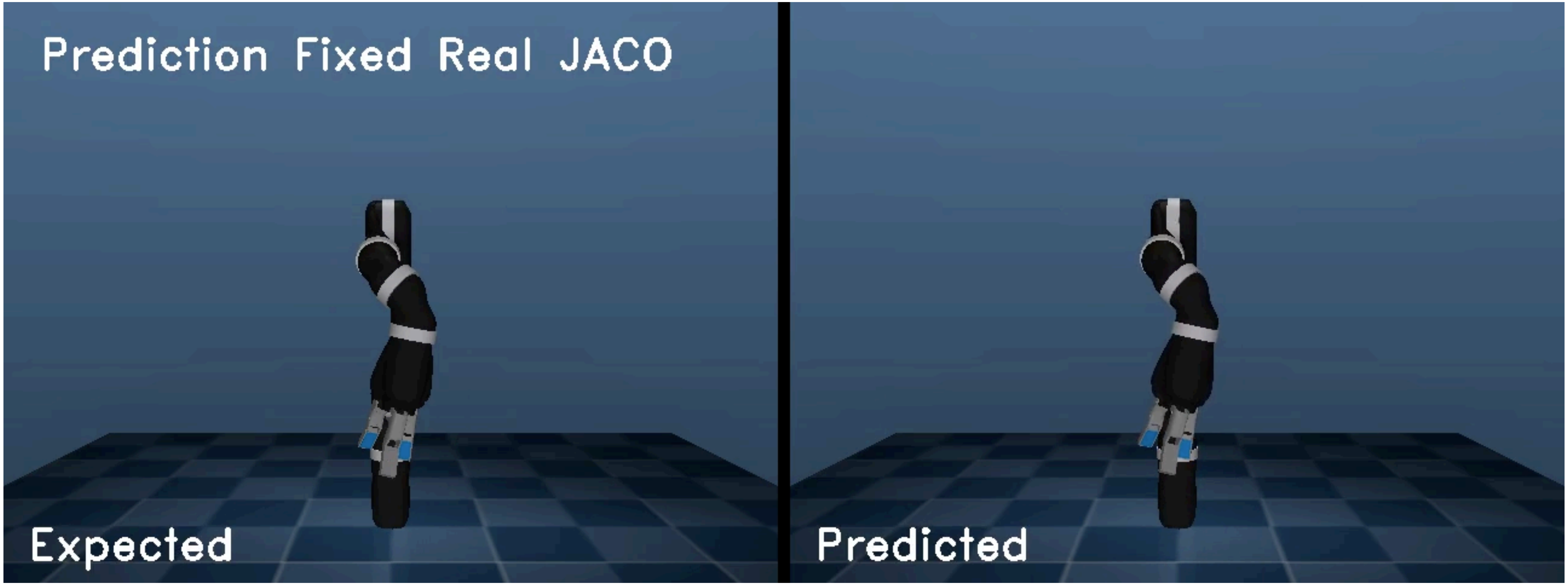


Sanchez-Gonzalez et al., 2018, ICML

Forward model: Real JACO data

Recurrent GN

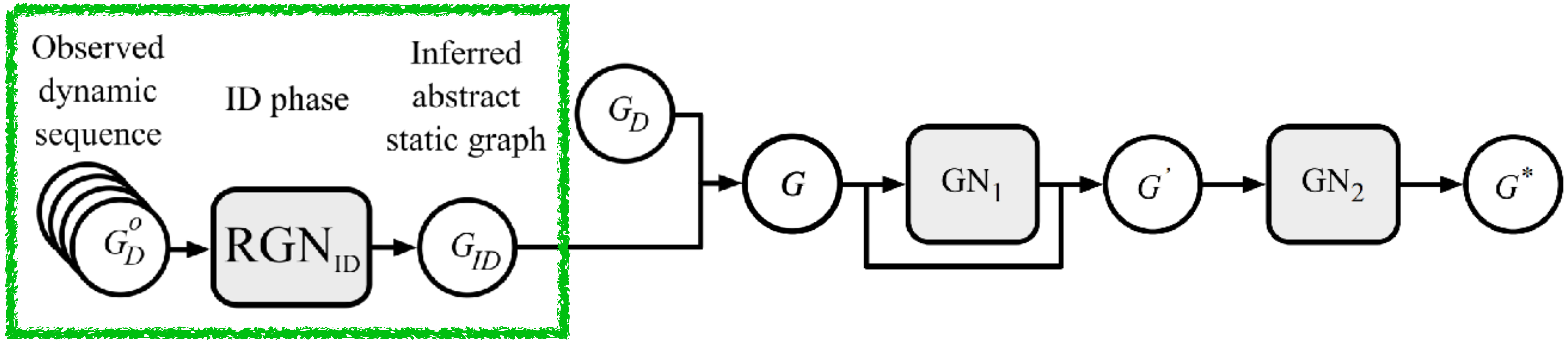




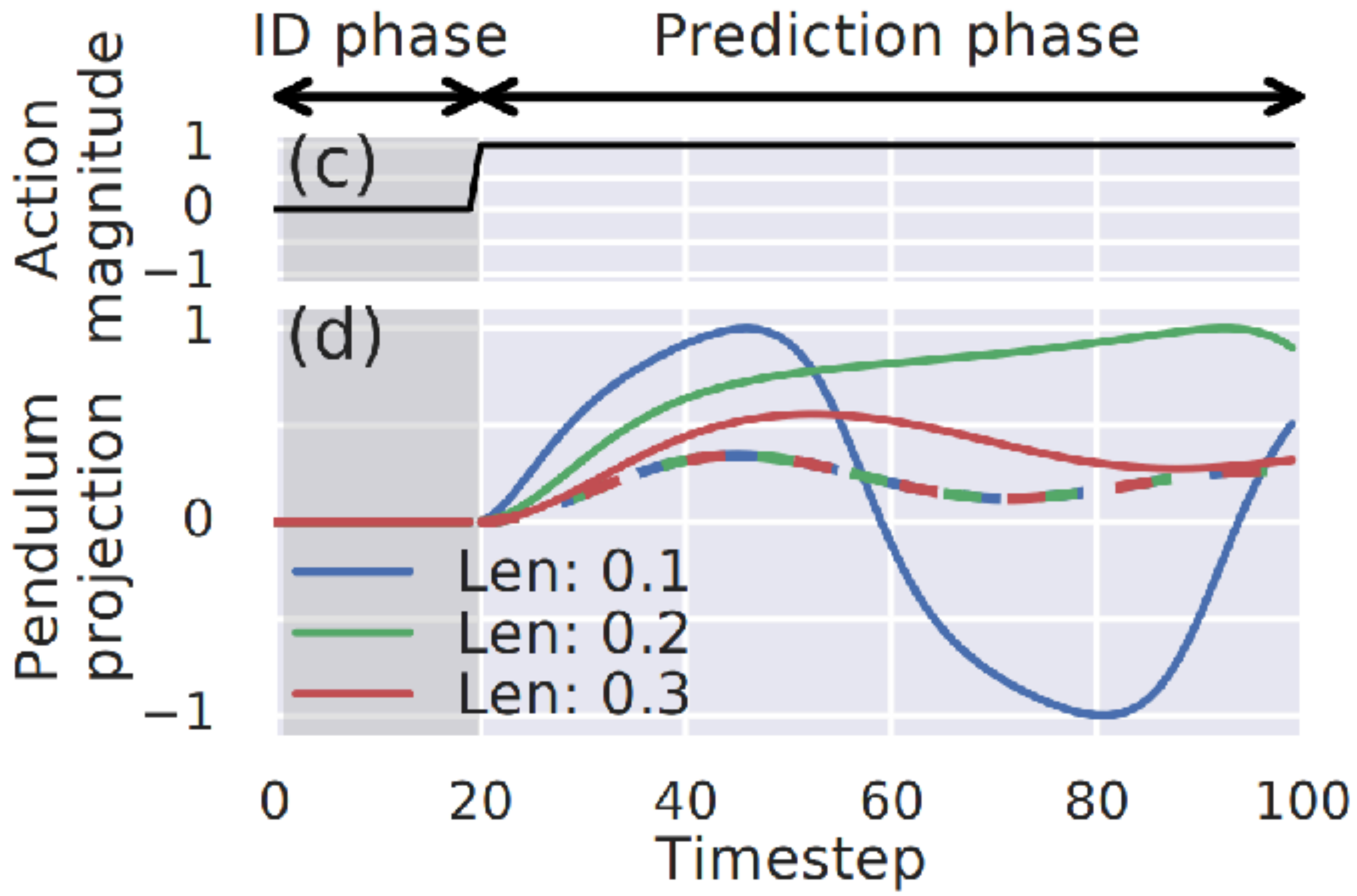
Sanchez-Gonzalez et al., 2018, ICML

Inference: GN-based system identification

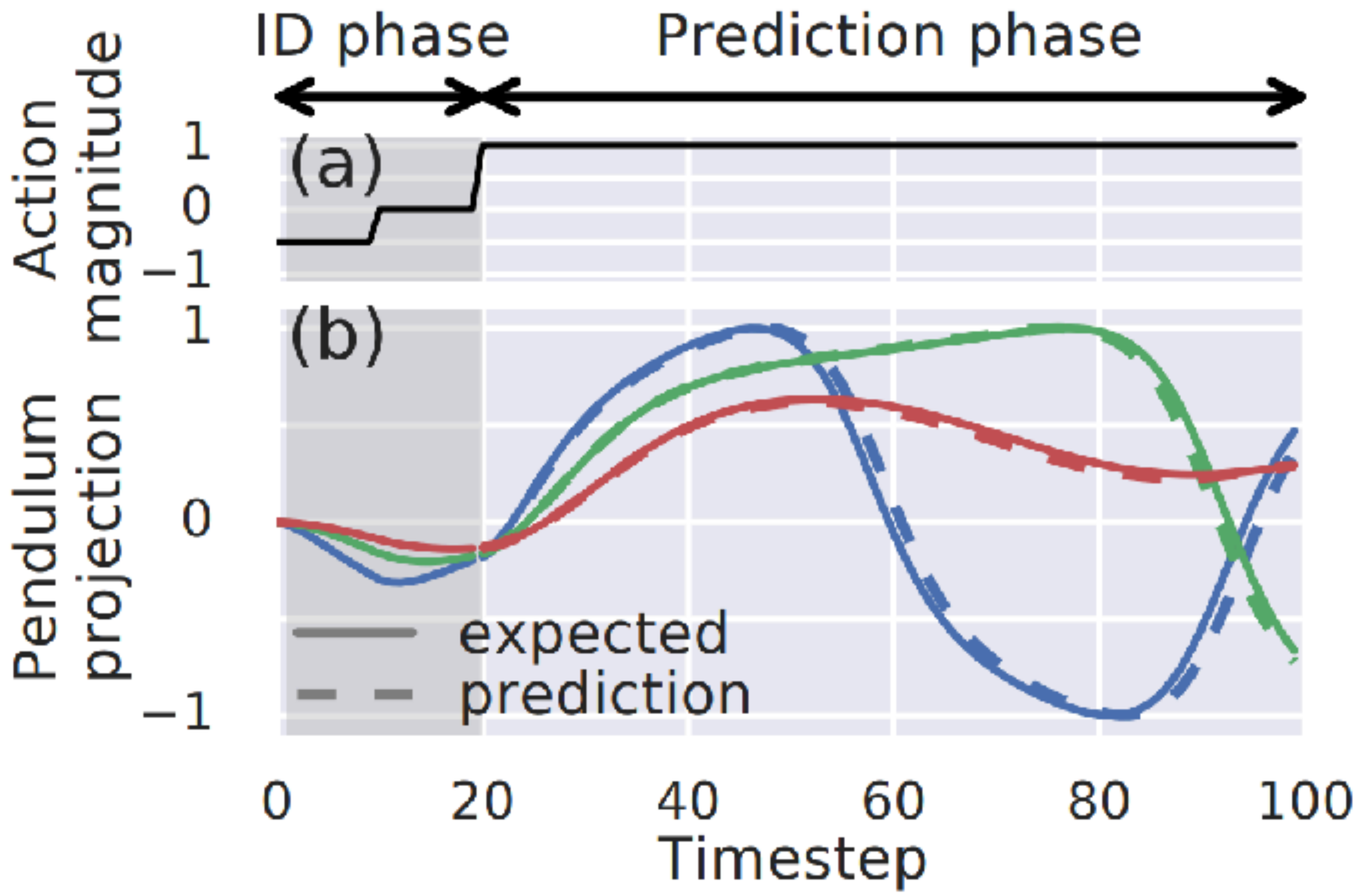
Unobserved system parameters (e.g. mass, length) are implicitly inferred



Unidentifiable condition



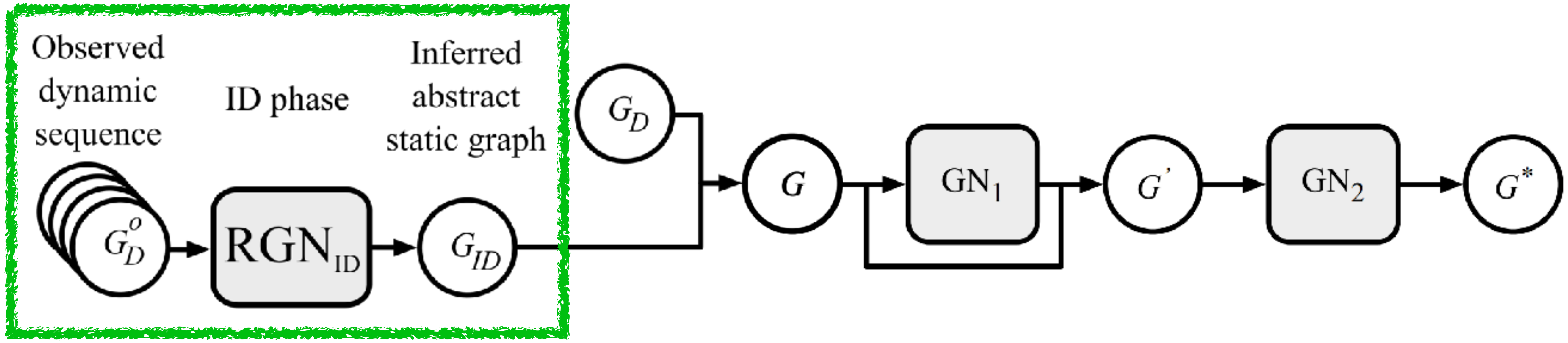
Identifiable condition

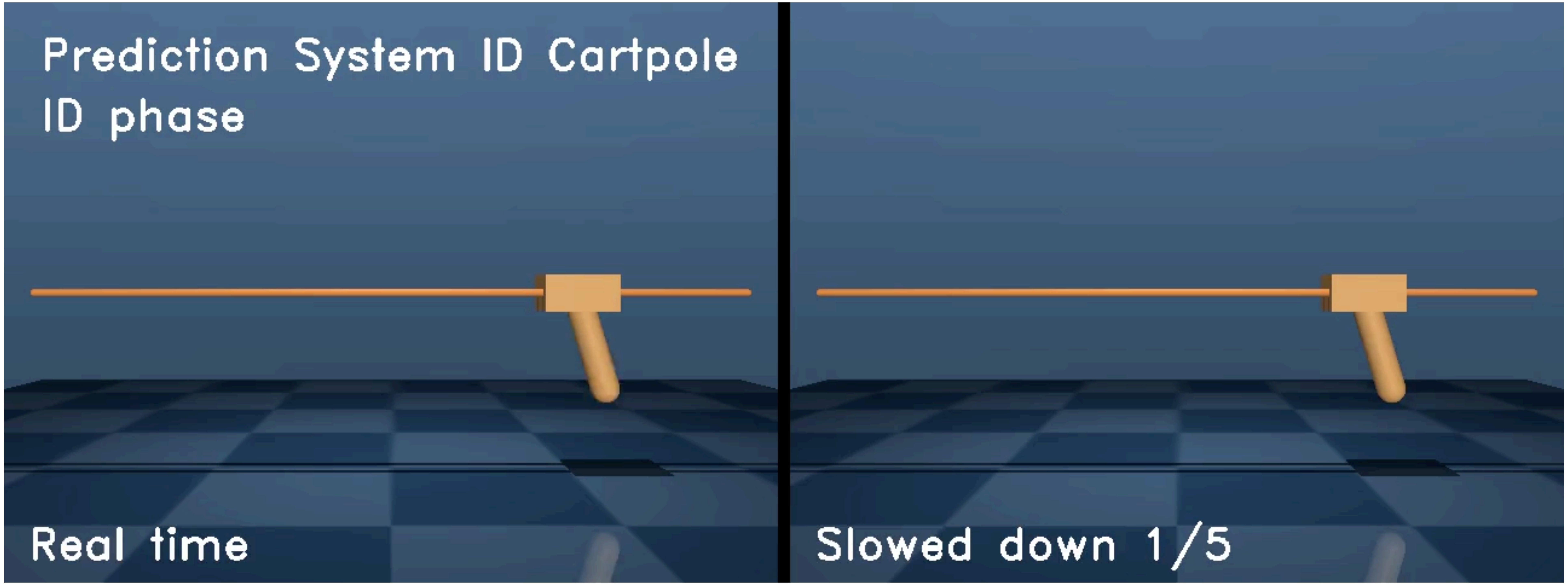


Sanchez-Gonzalez et al., 2018, ICML

Inference: GN-based system identification

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

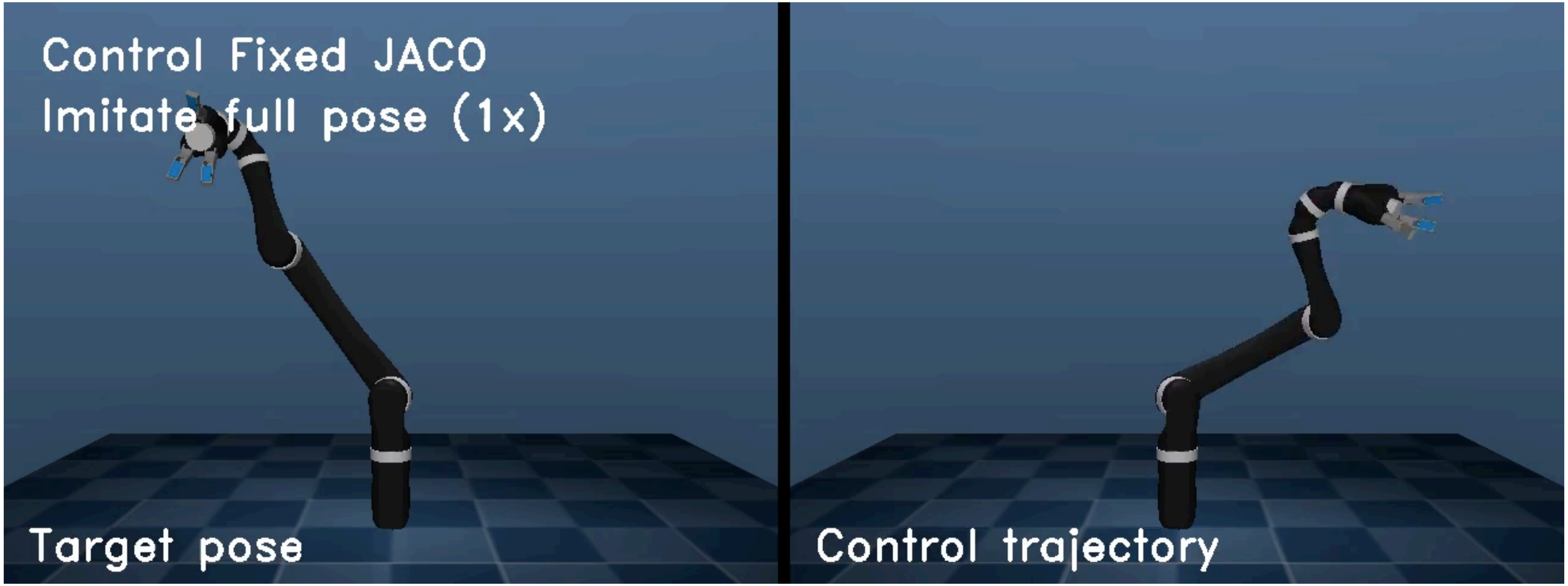




Sanchez-Gonzalez et al., 2018, ICML

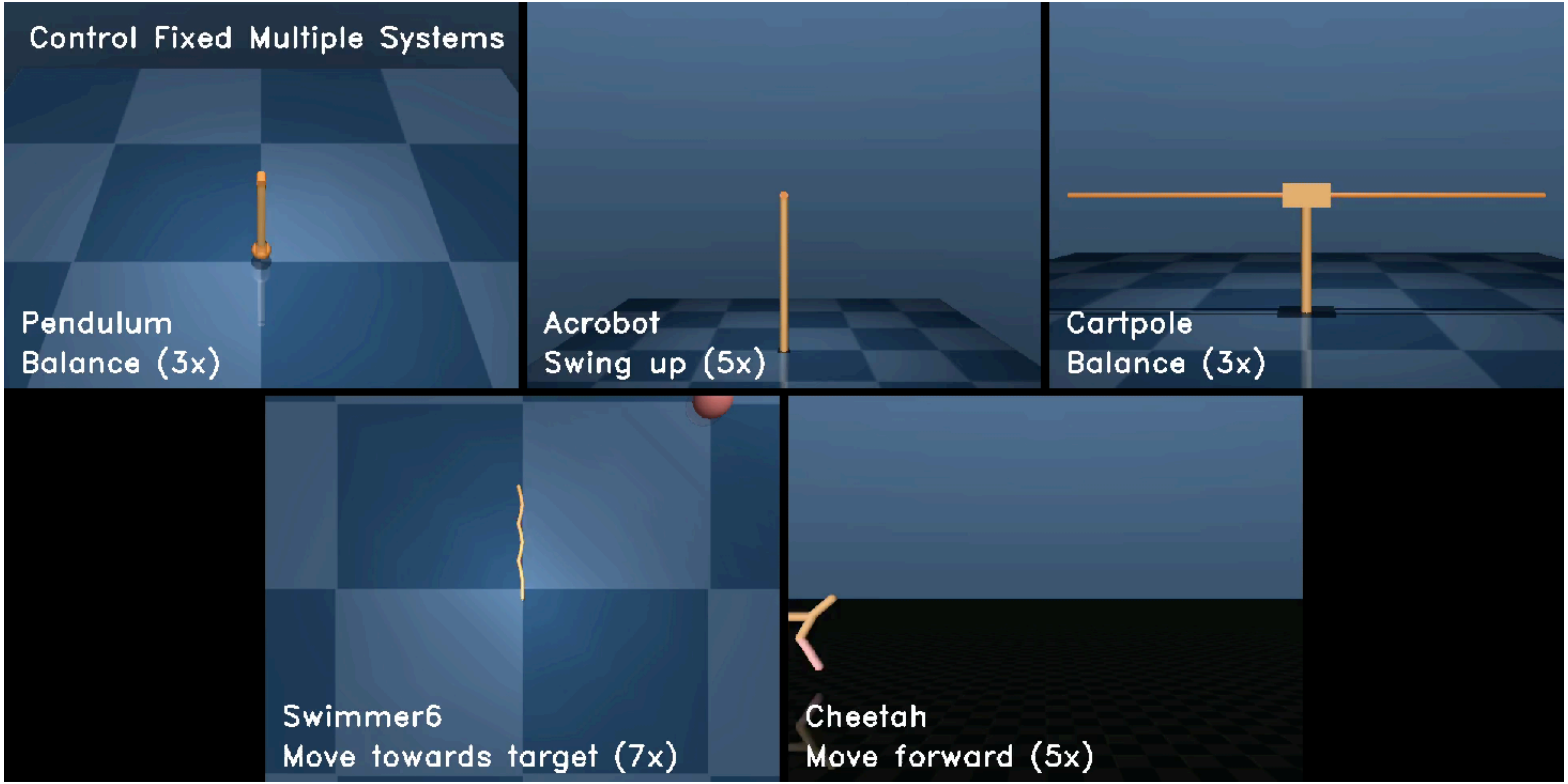
Control: Model-based planning

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



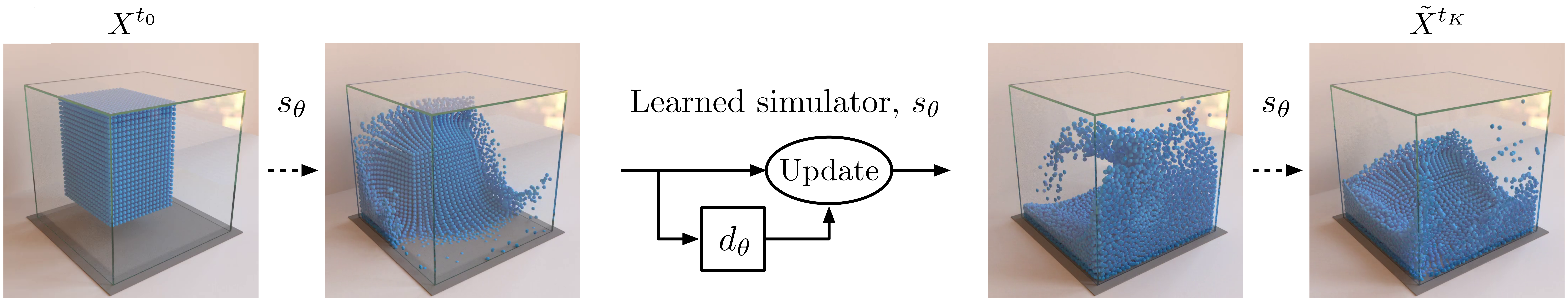
Sanchez-Gonzalez et al., 2018, ICML

Control: Multiple systems via a single model



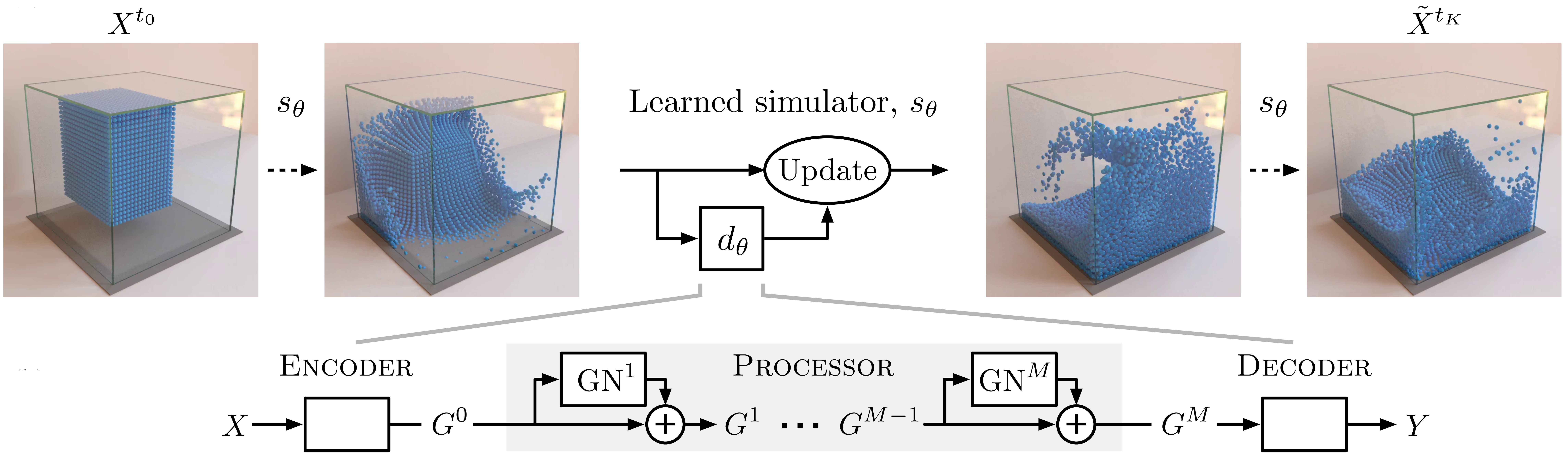
Sanchez-Gonzalez et al., 2018, ICML

Learning to simulate fluids and complex materials



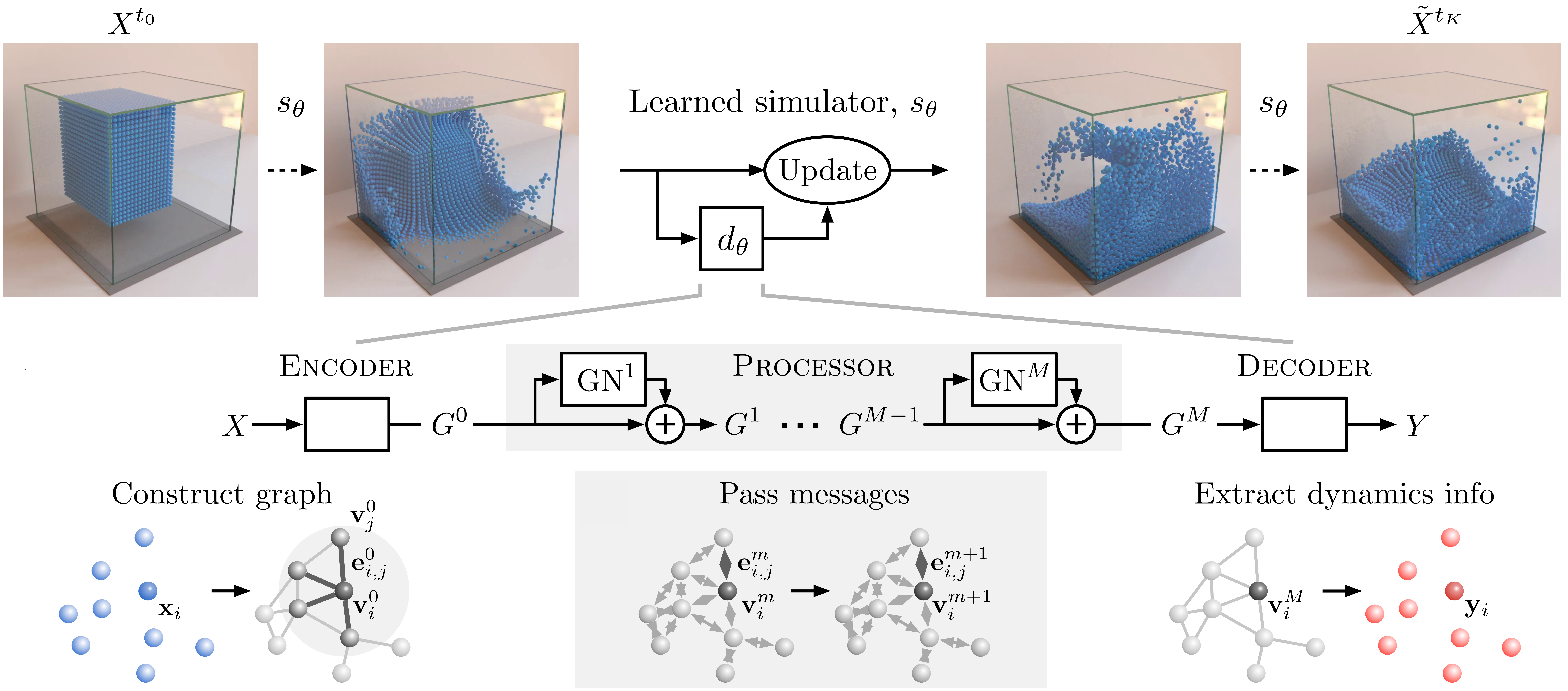
Sanchez-Gonzalez et al., 2020, ICML

Learning to simulate fluids and complex materials



Sanchez-Gonzalez et al., 2020, ICML

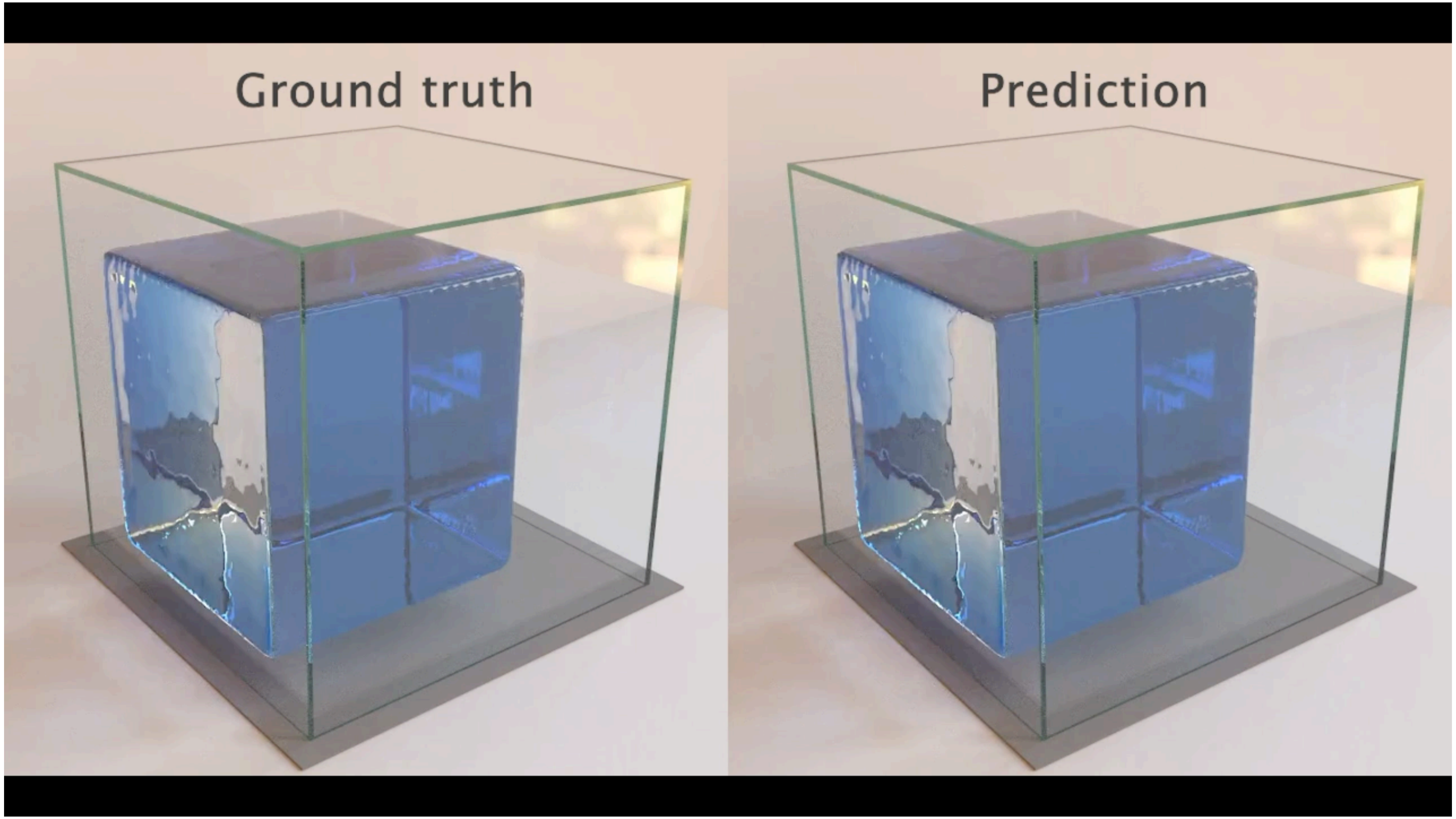
Learning to simulate fluids and complex materials



Sanchez-Gonzalez et al., 2020, ICML

Learning to simulate fluids and complex materials

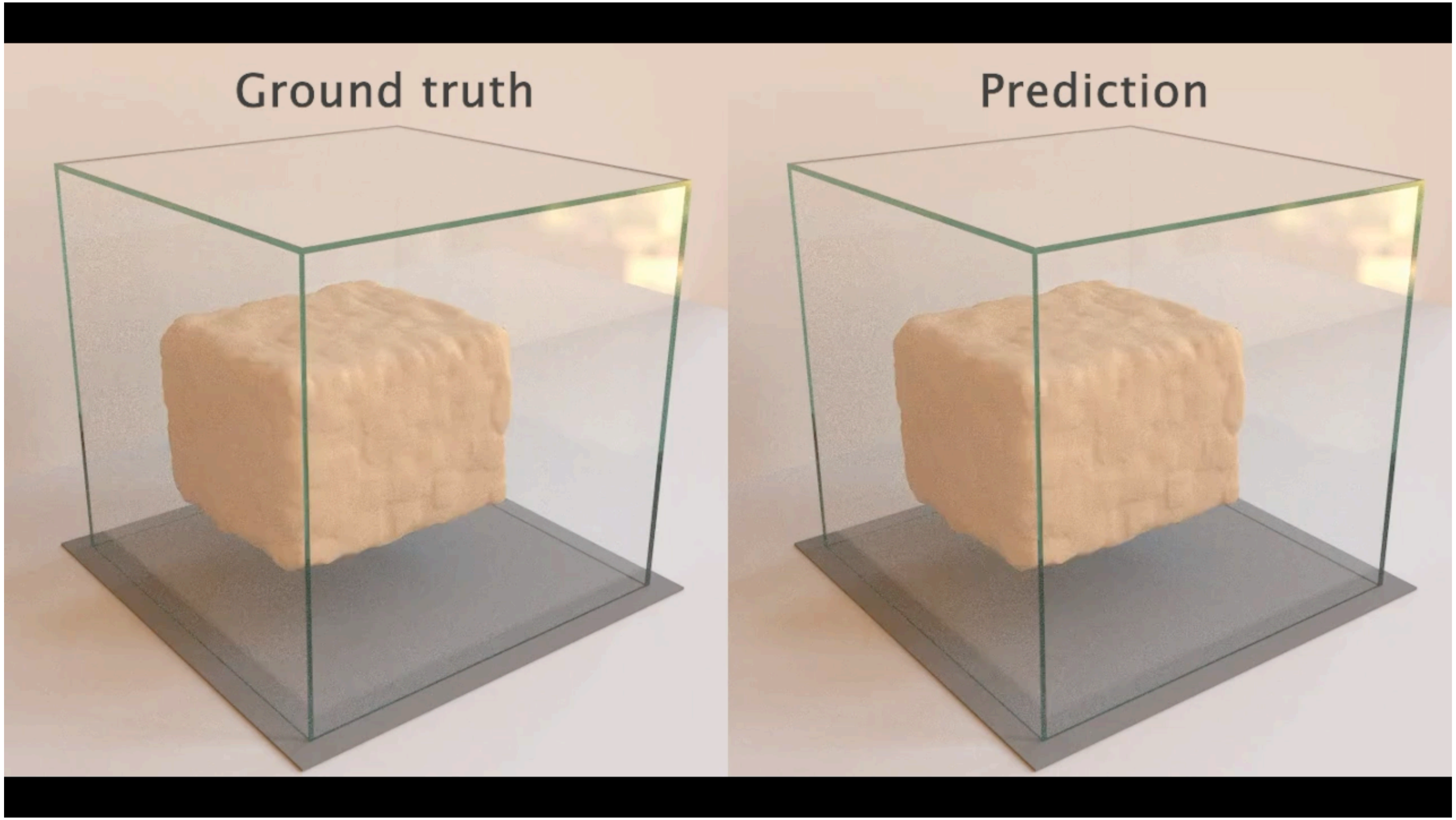
Water-3D (14k particles, SPH)



Sanchez-Gonzalez et al., 2020, ICML

Learning to simulate fluids and complex materials

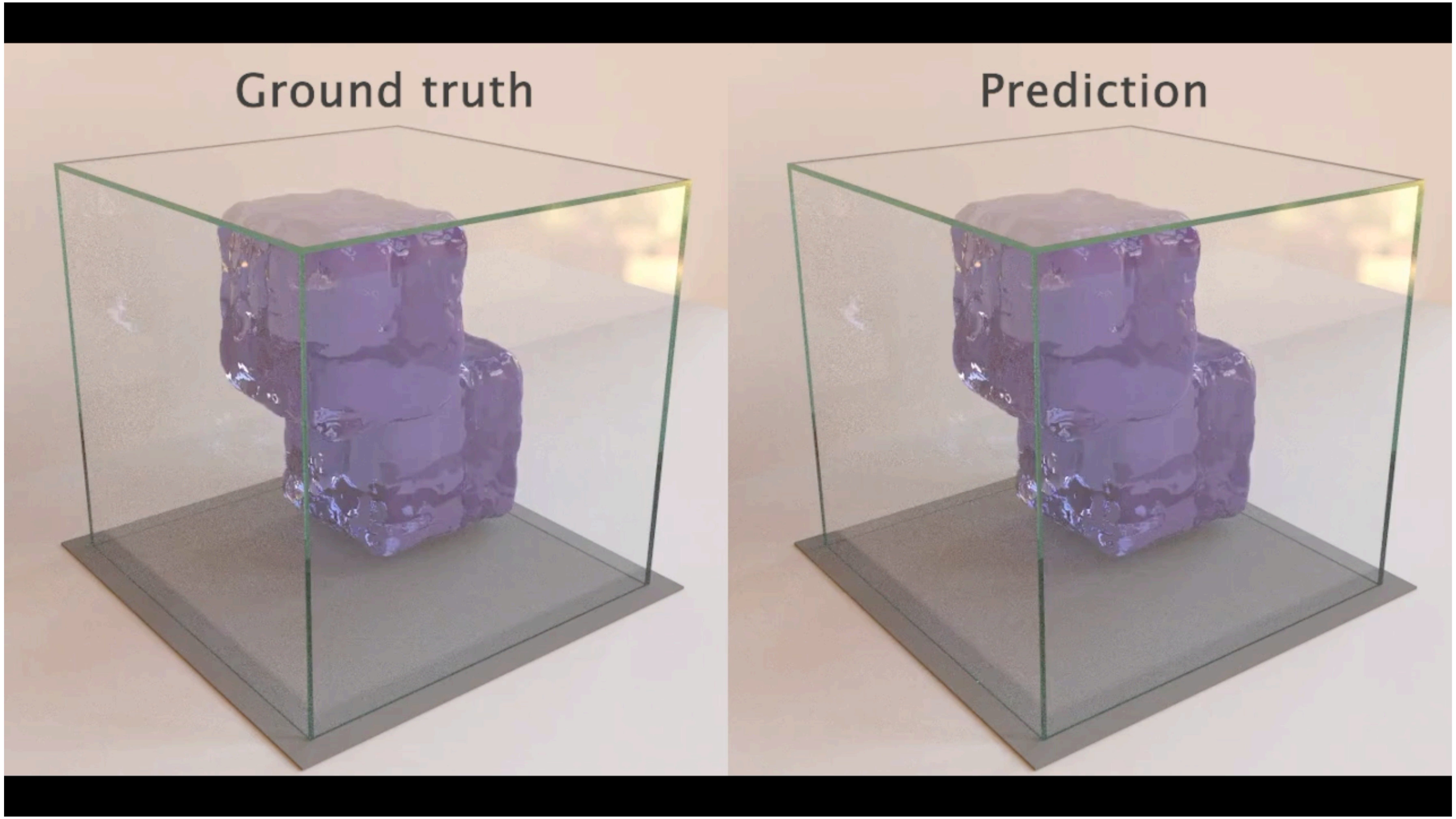
Sand-3D (19k particles, MPM)



Sanchez-Gonzalez et al., 2020, ICML

Learning to simulate fluids and complex materials

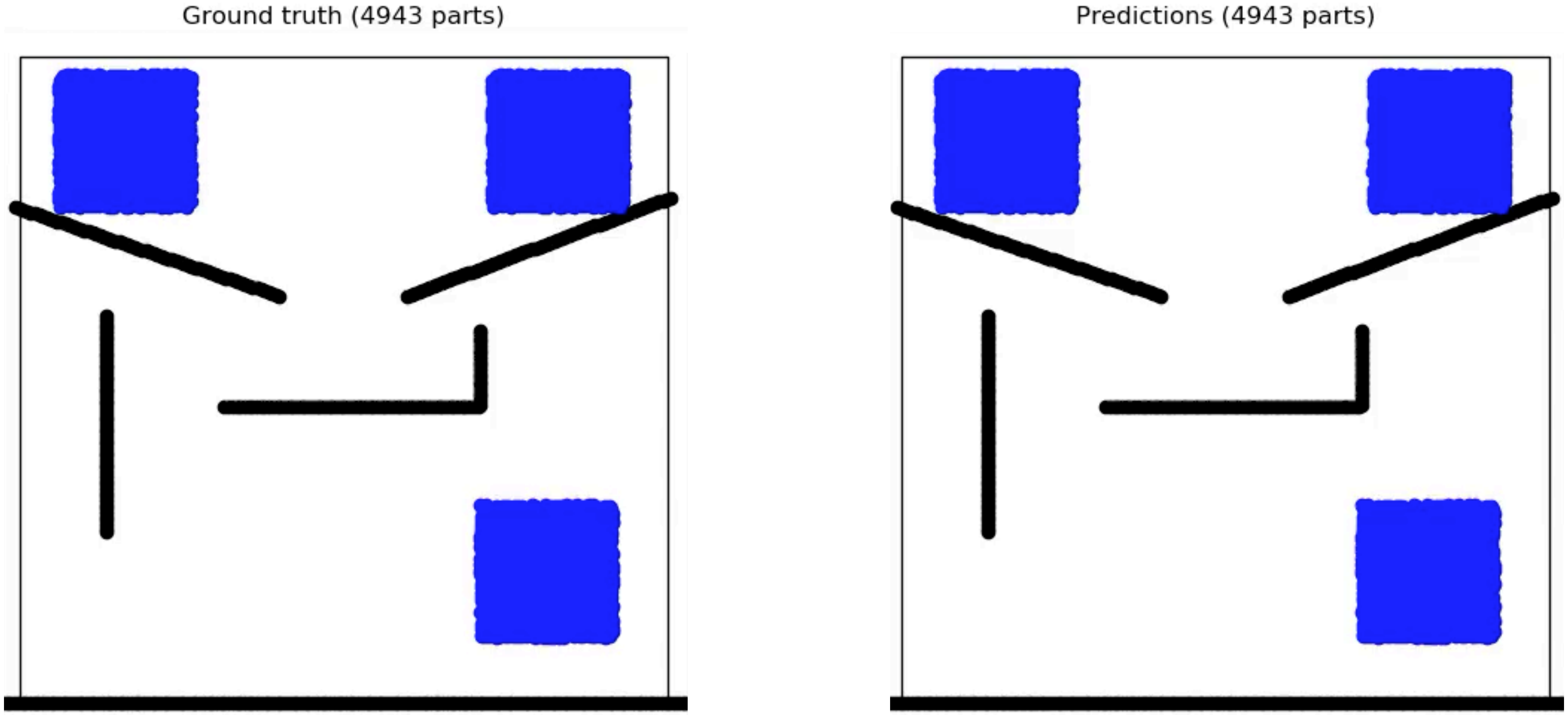
Goop-3D (19k particles, MPM)



Sanchez-Gonzalez et al., 2020, ICML

Learning to simulate fluids and complex materials

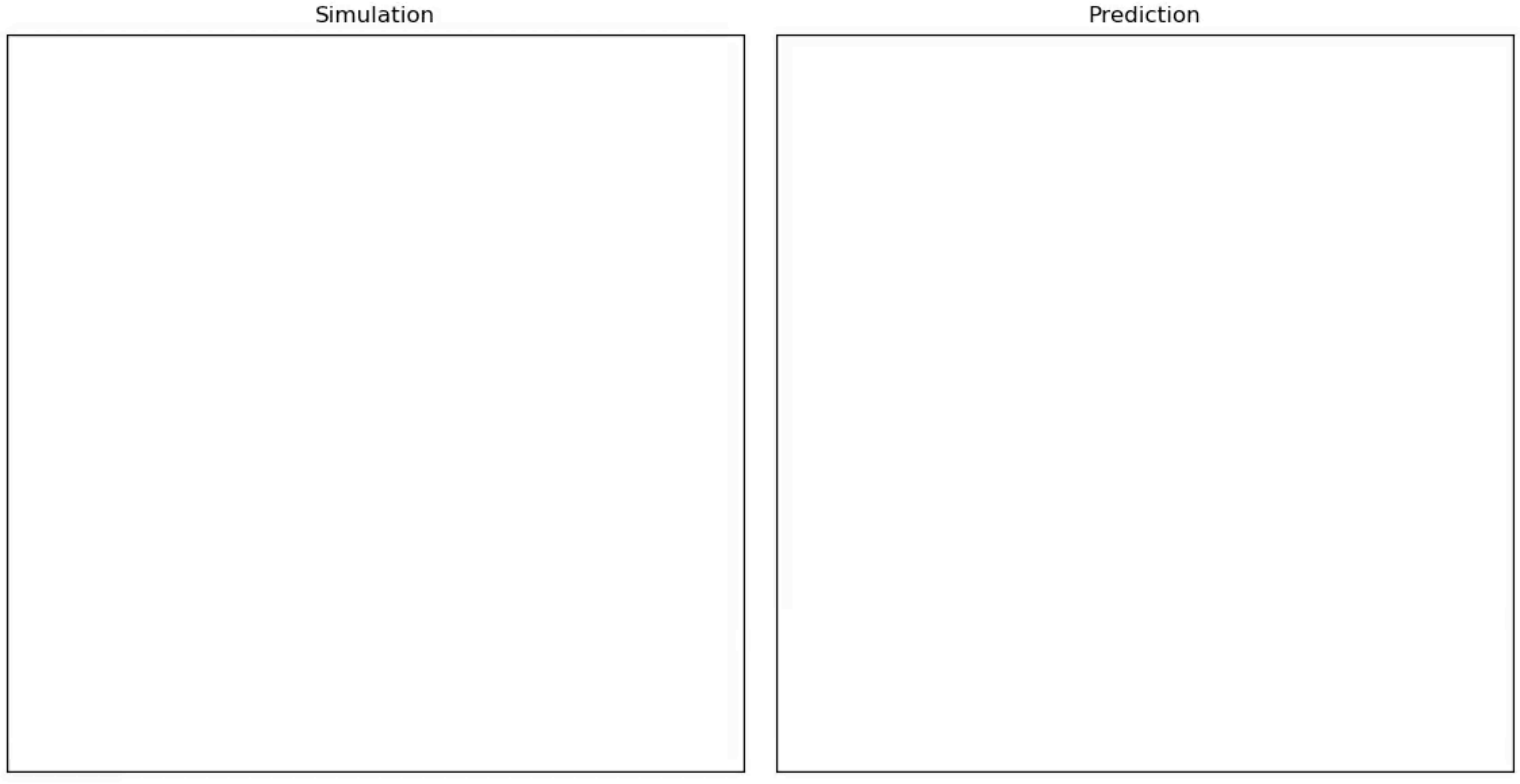
Water ramps



Sanchez-Gonzalez et al., 2020, ICML

Learning to simulate fluids and complex materials

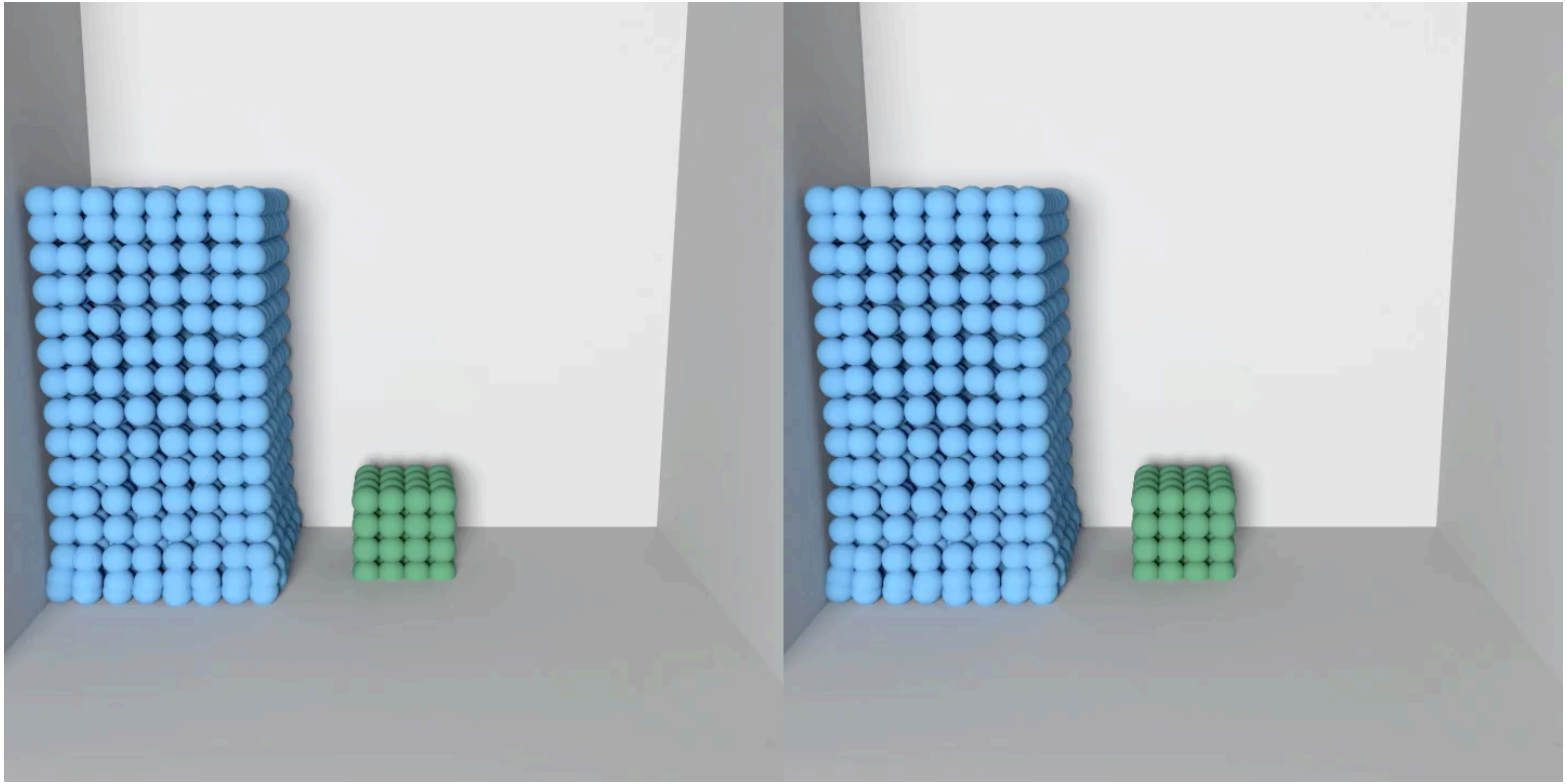
Multiple materials, generalization



Sanchez-Gonzalez et al., 2020, ICML

Learning to simulate fluids and complex materials

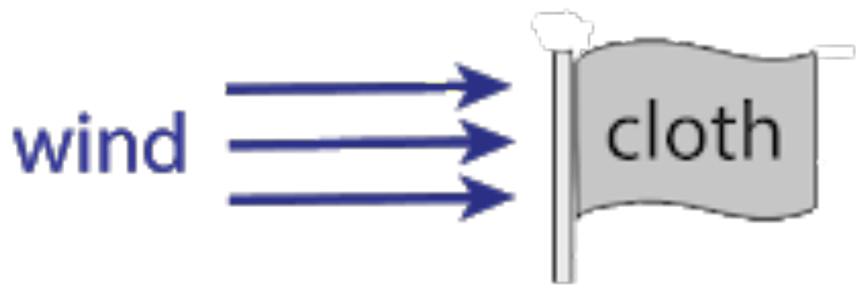
BoxBath



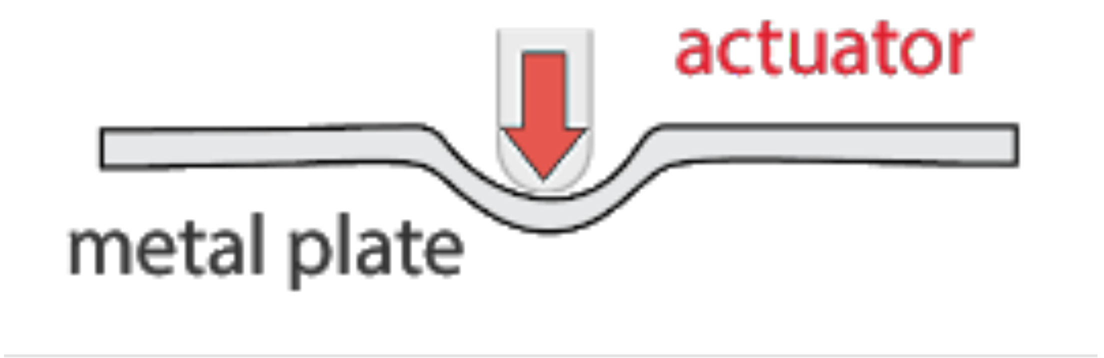
Sanchez-Gonzalez et al., 2020, ICML

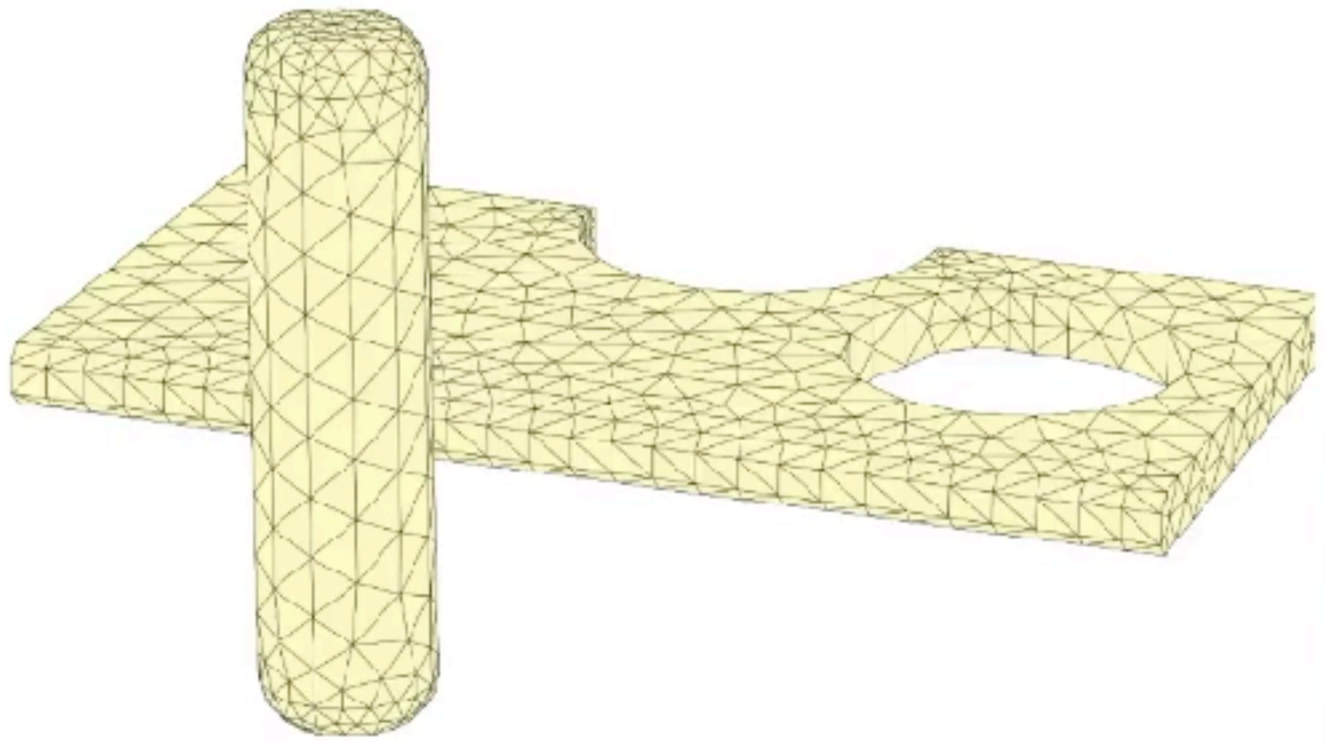
Learning to simulate meshes

Cloth

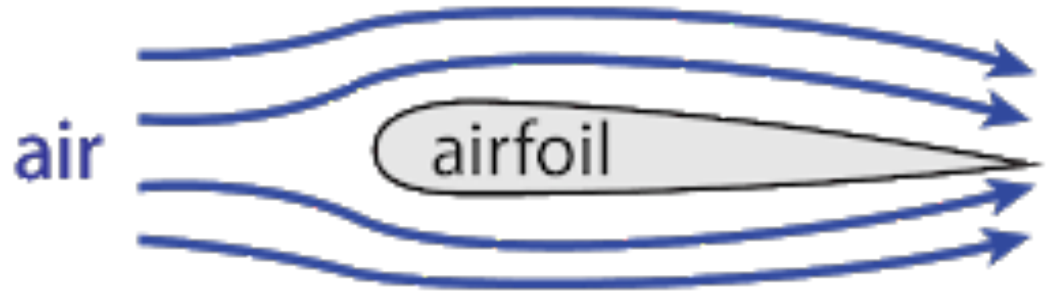


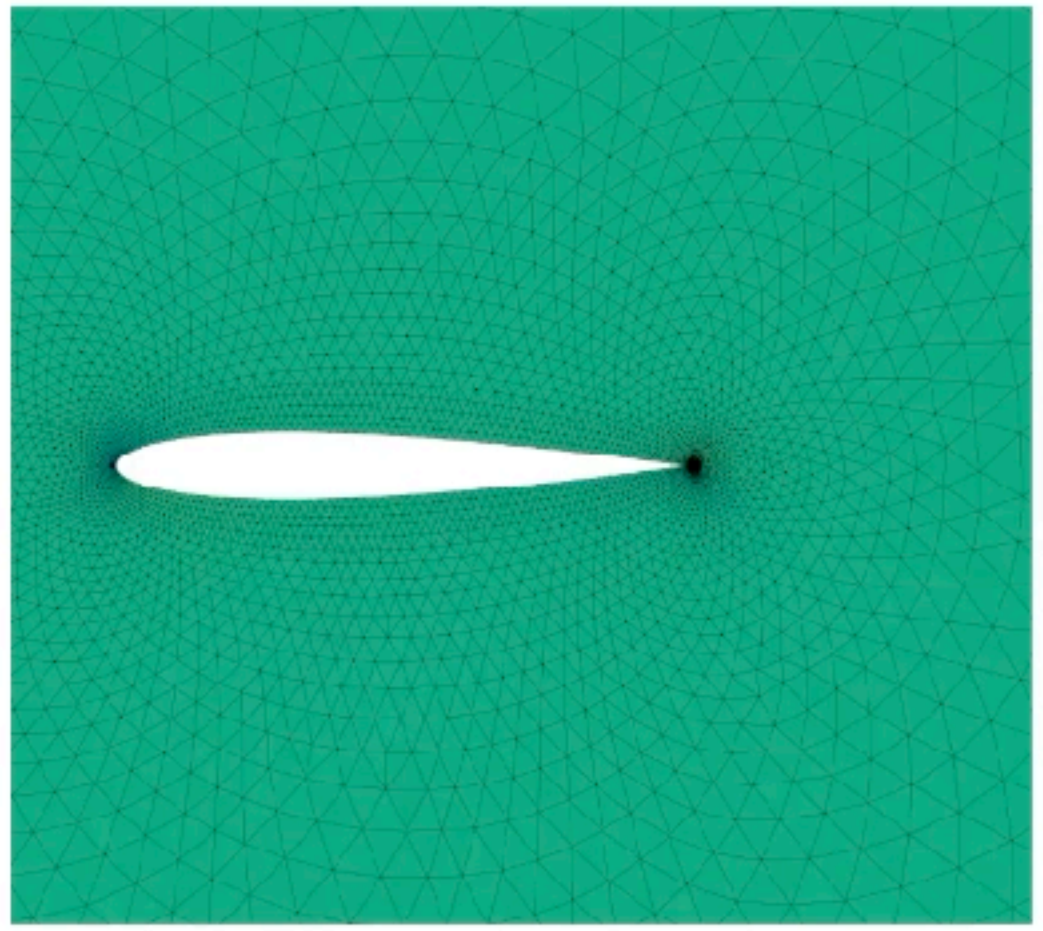
Structural mechanics



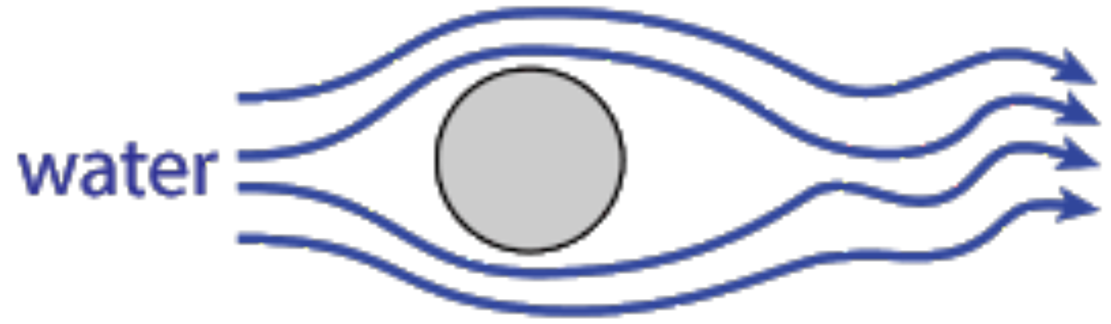


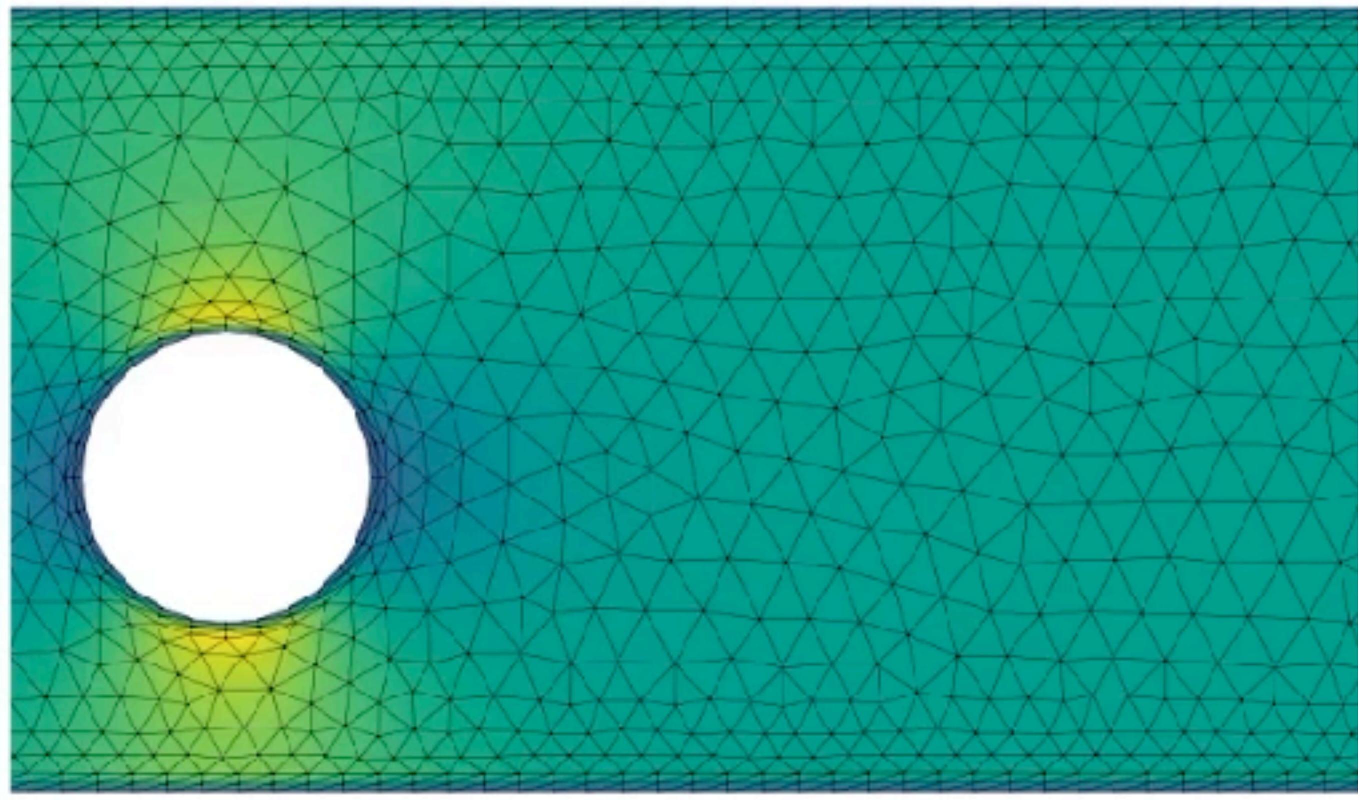
Aerodynamics (compressible)





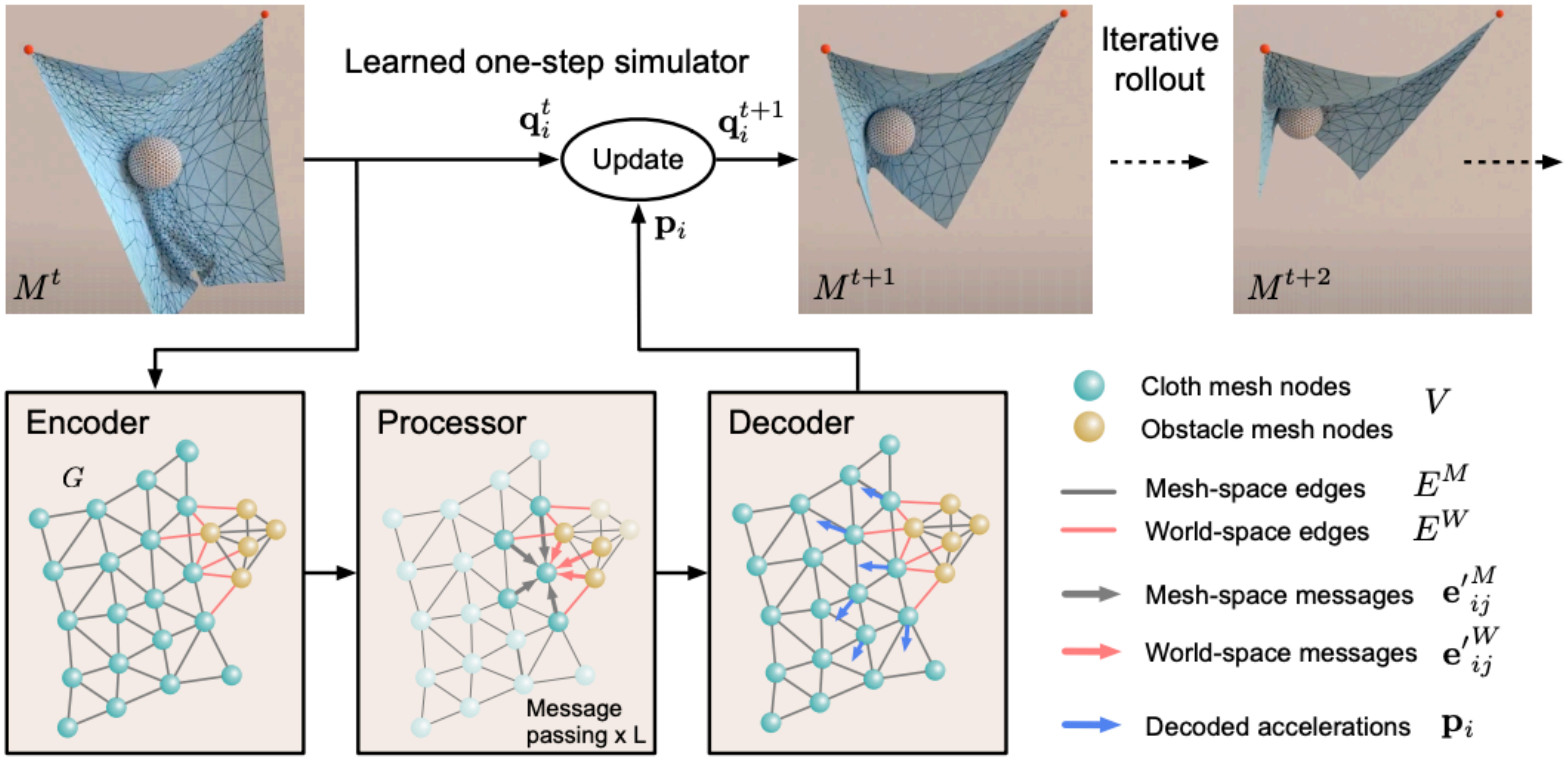
Fluids (incompressible)





Pfaff et al., 2020, arXiv/under review

Learning to simulate meshes

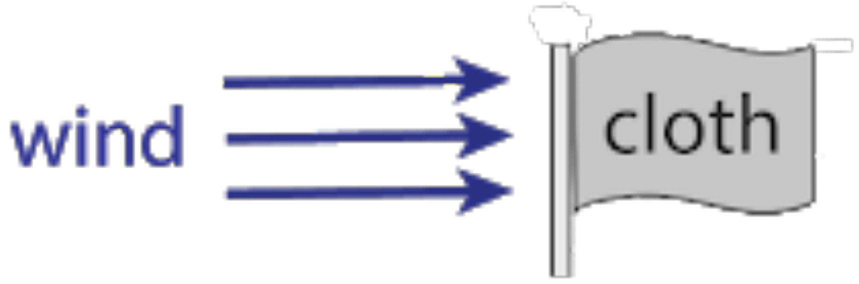


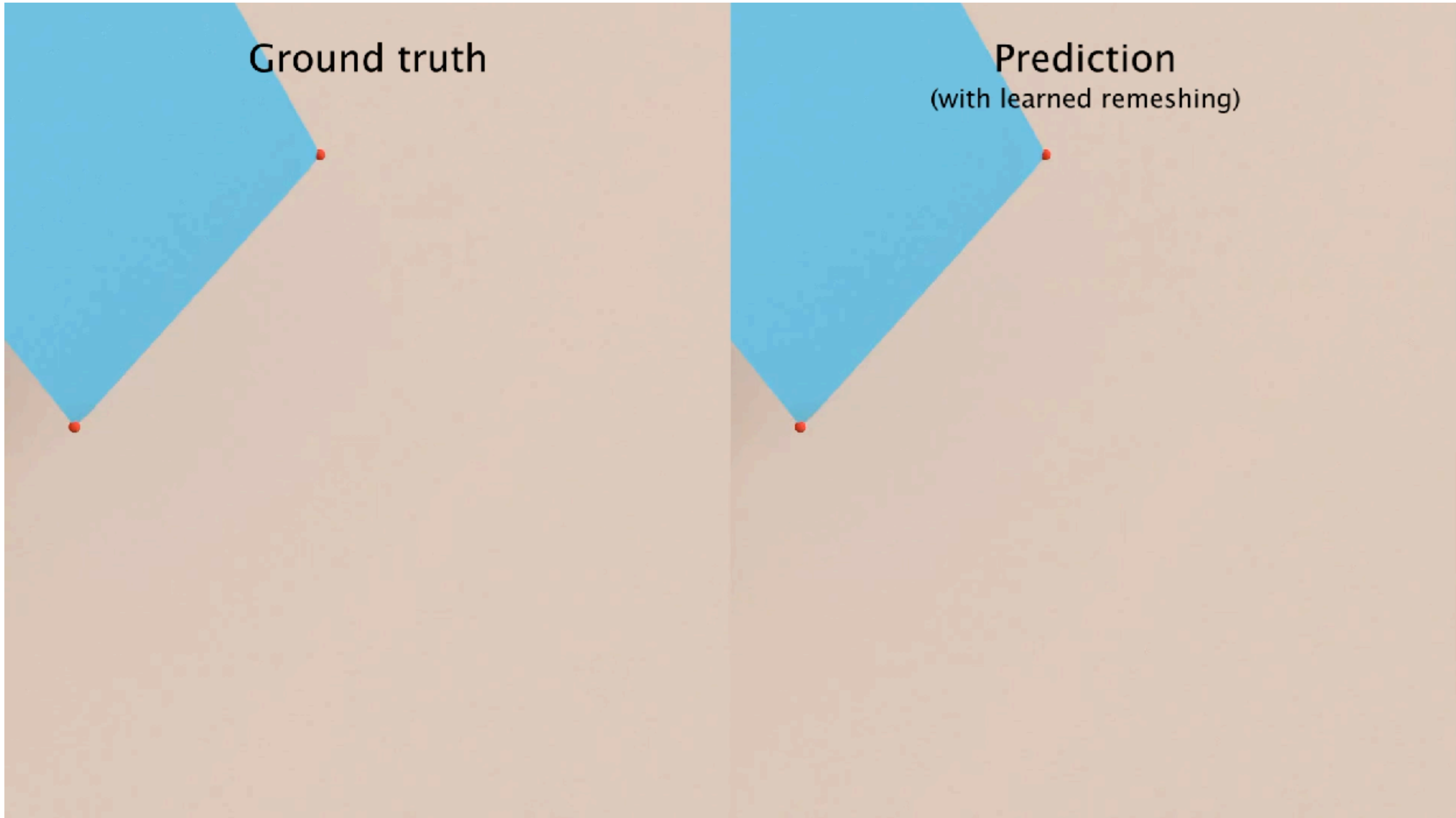
10-100x faster than the simulator on which it was trained

Pfaff et al., 2020, arXiv/under review

Learning to simulate meshes

Cloth

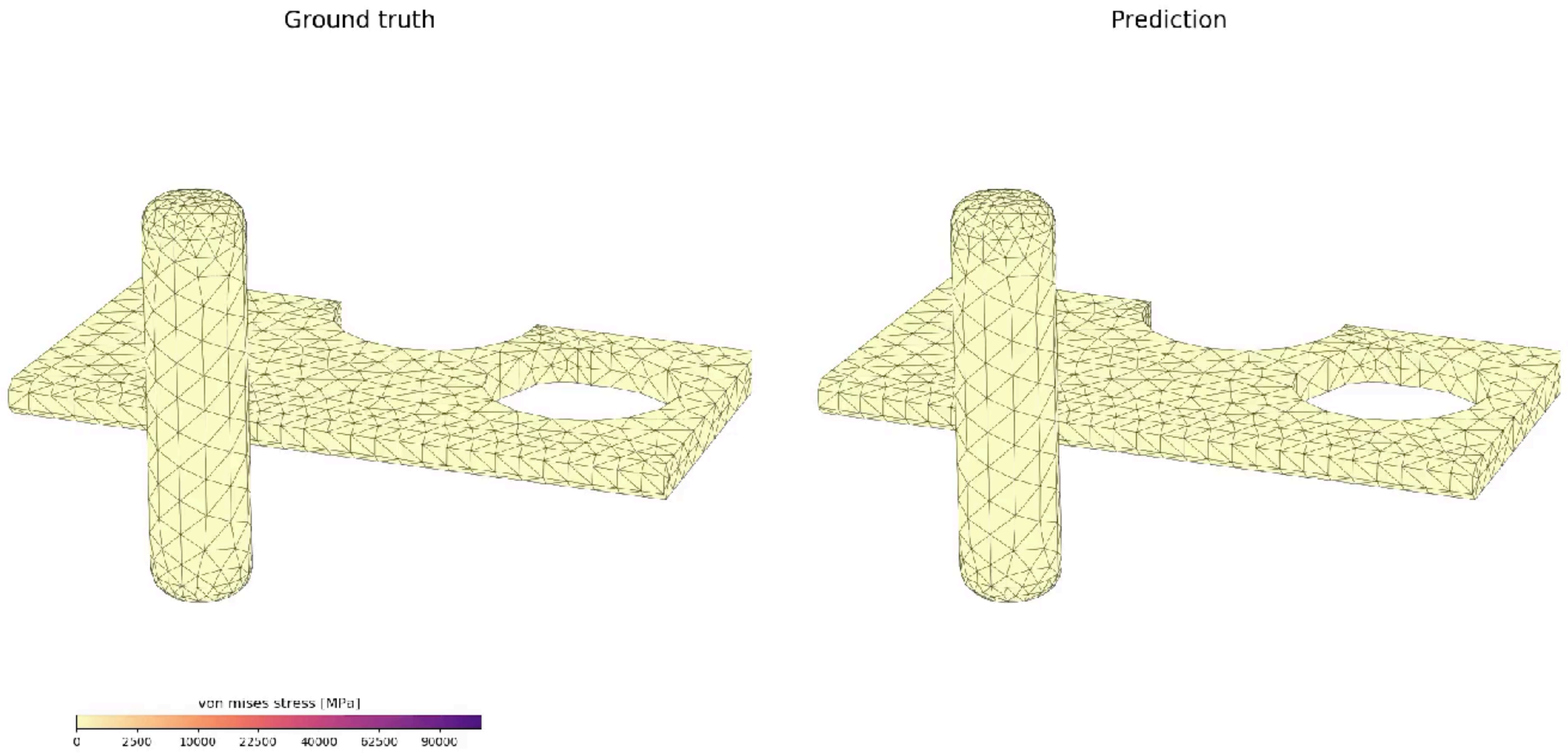




Pfaff et al., 2020, arXiv/under review

Learning to simulate meshes

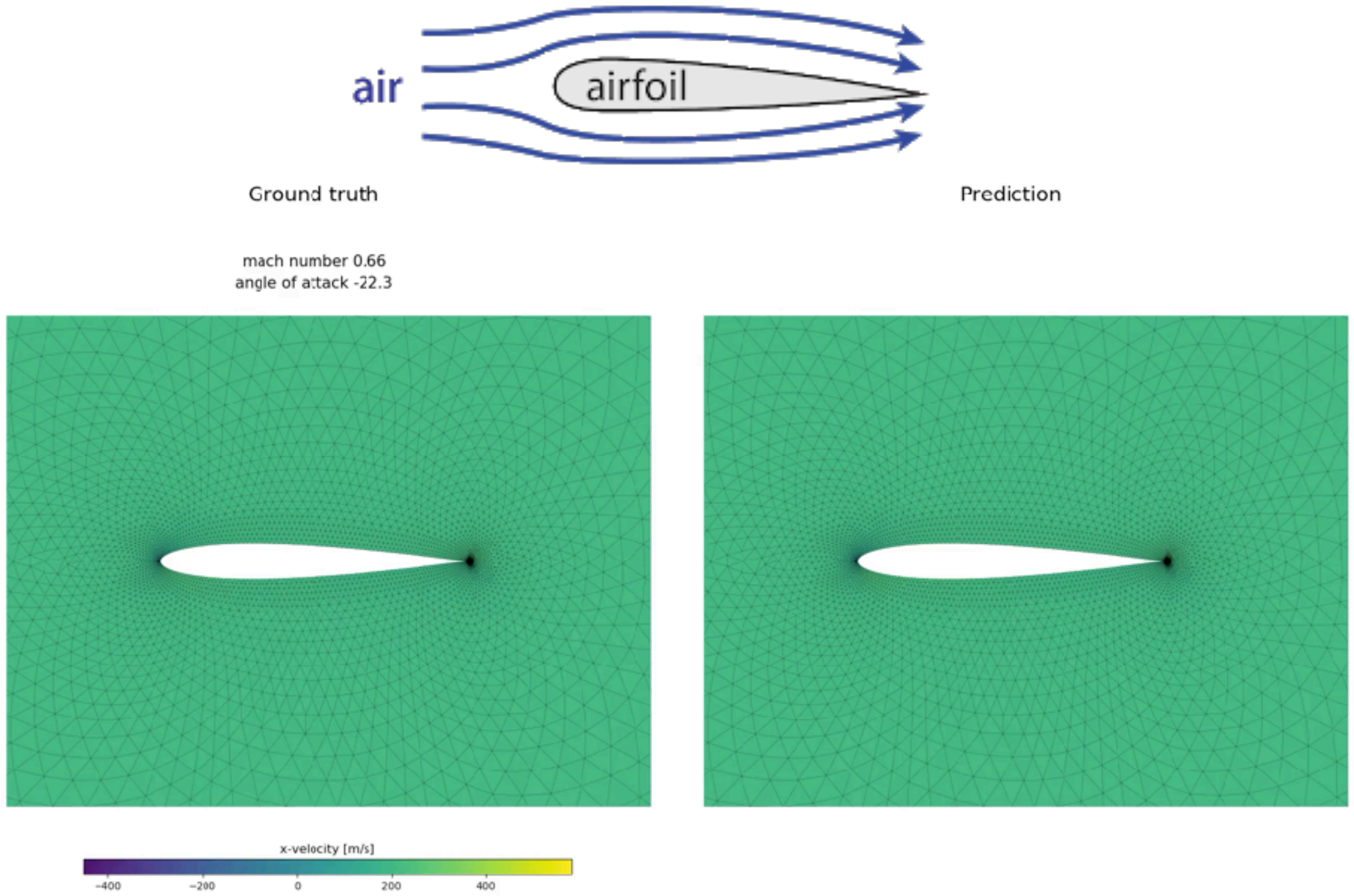
Structural mechanics



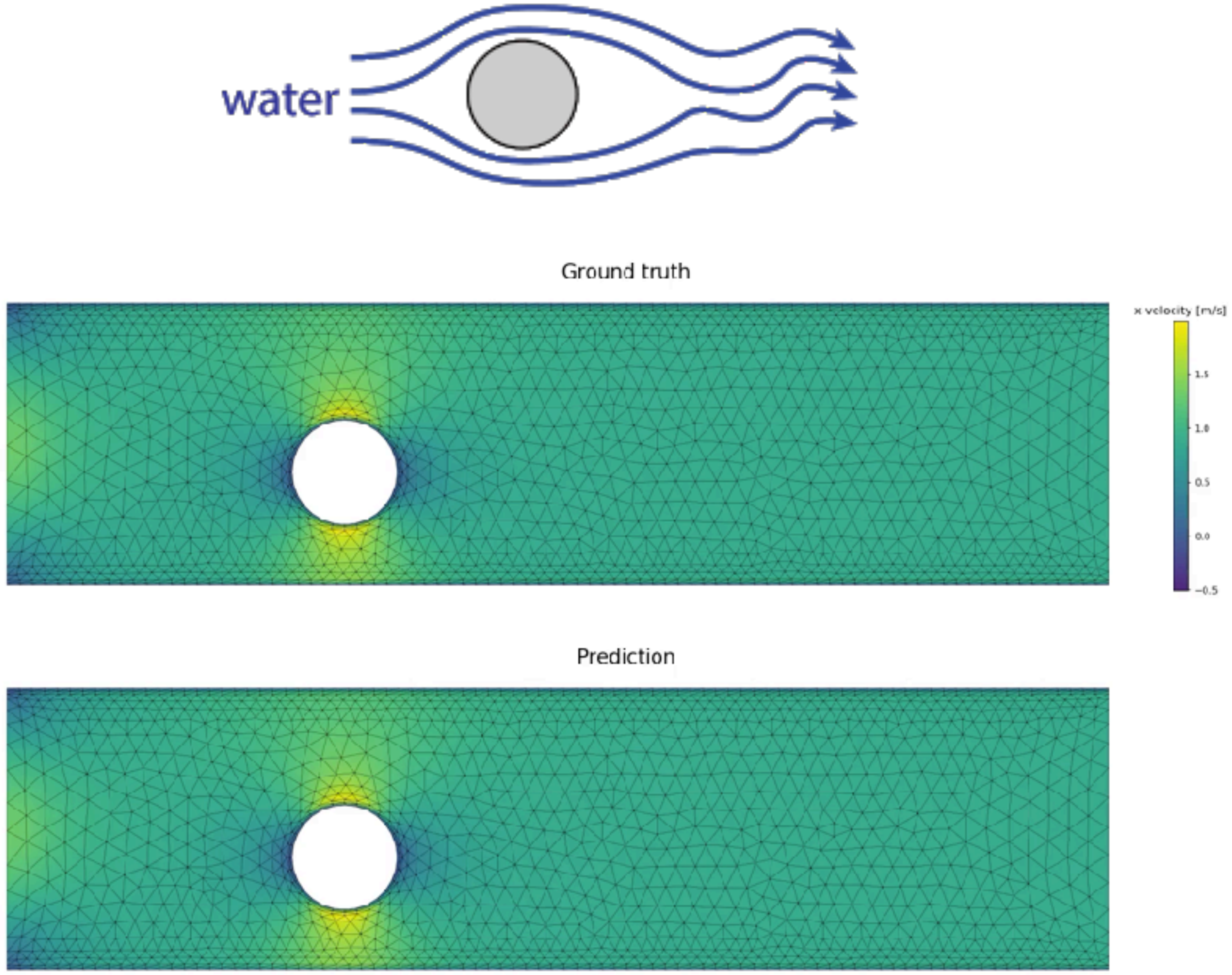
Pfaff et al., 2020, arXiv/under review

Learning to simulate meshes

Aerodynamics (compressible)



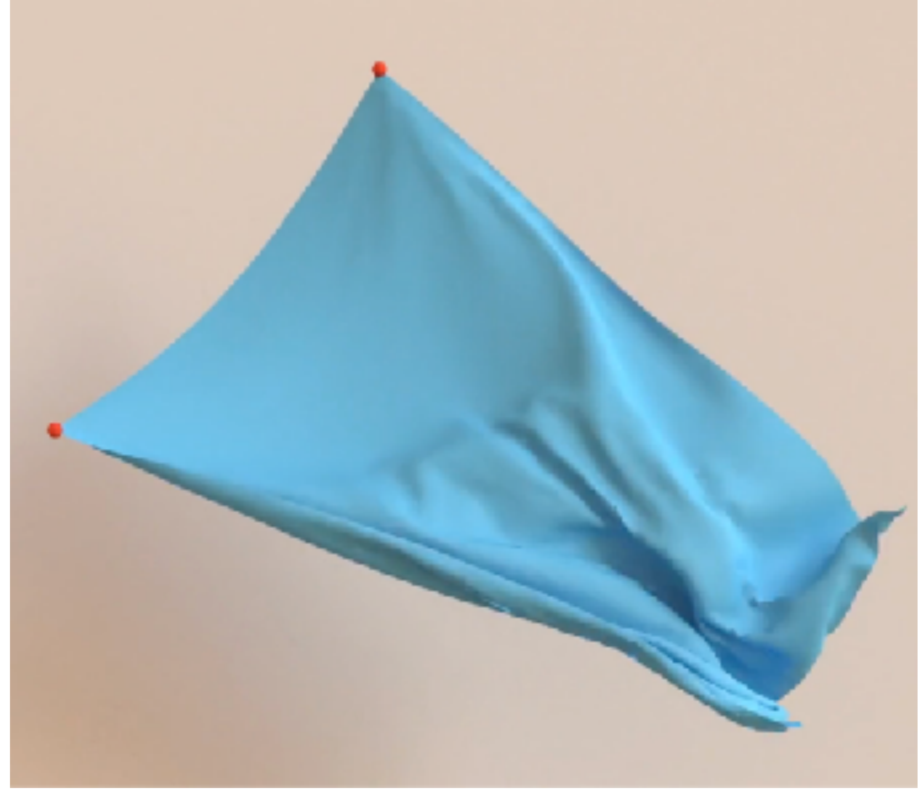
Fluids (incompressible)



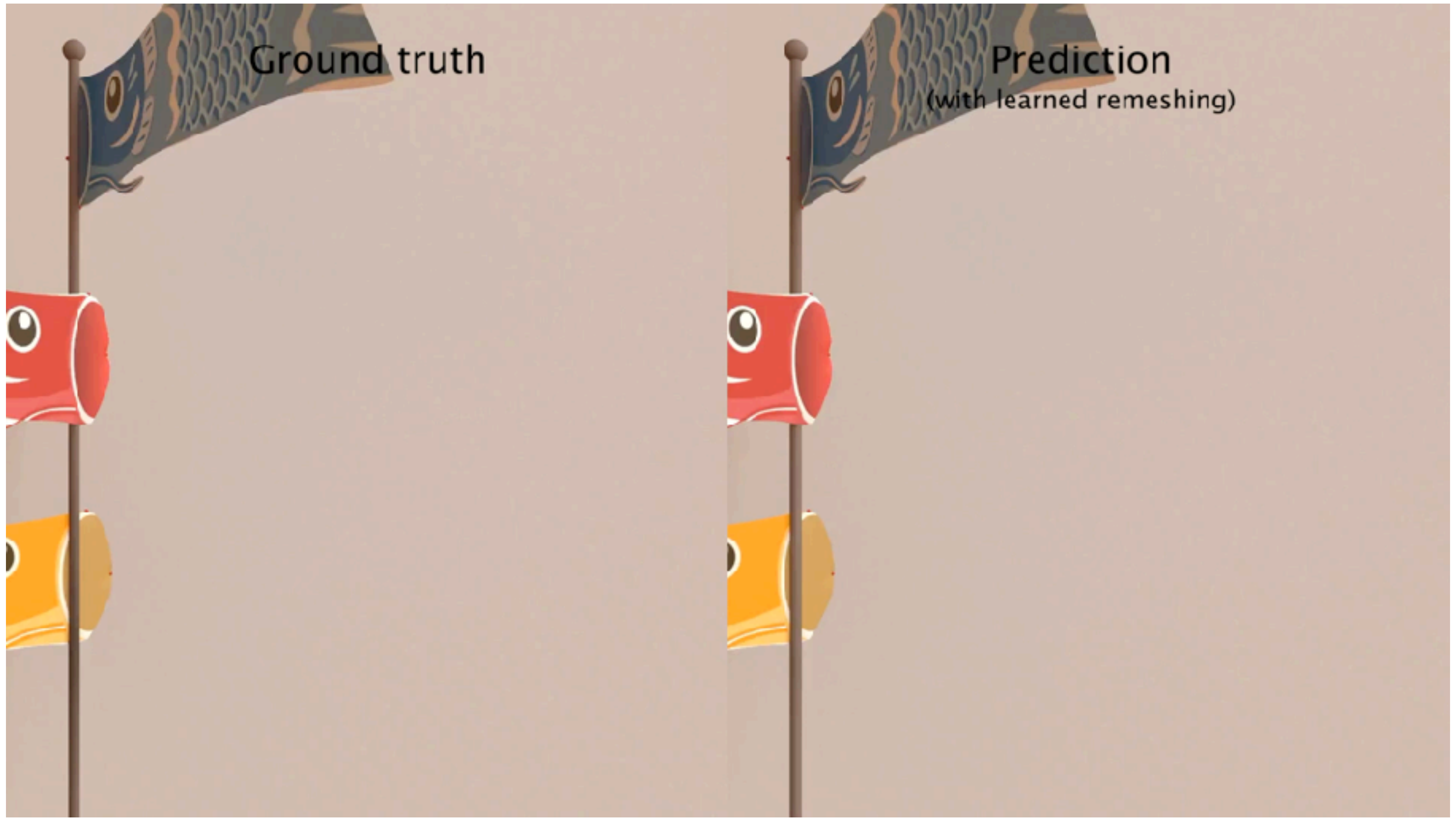
Pfaff et al., 2020, arXiv/under review

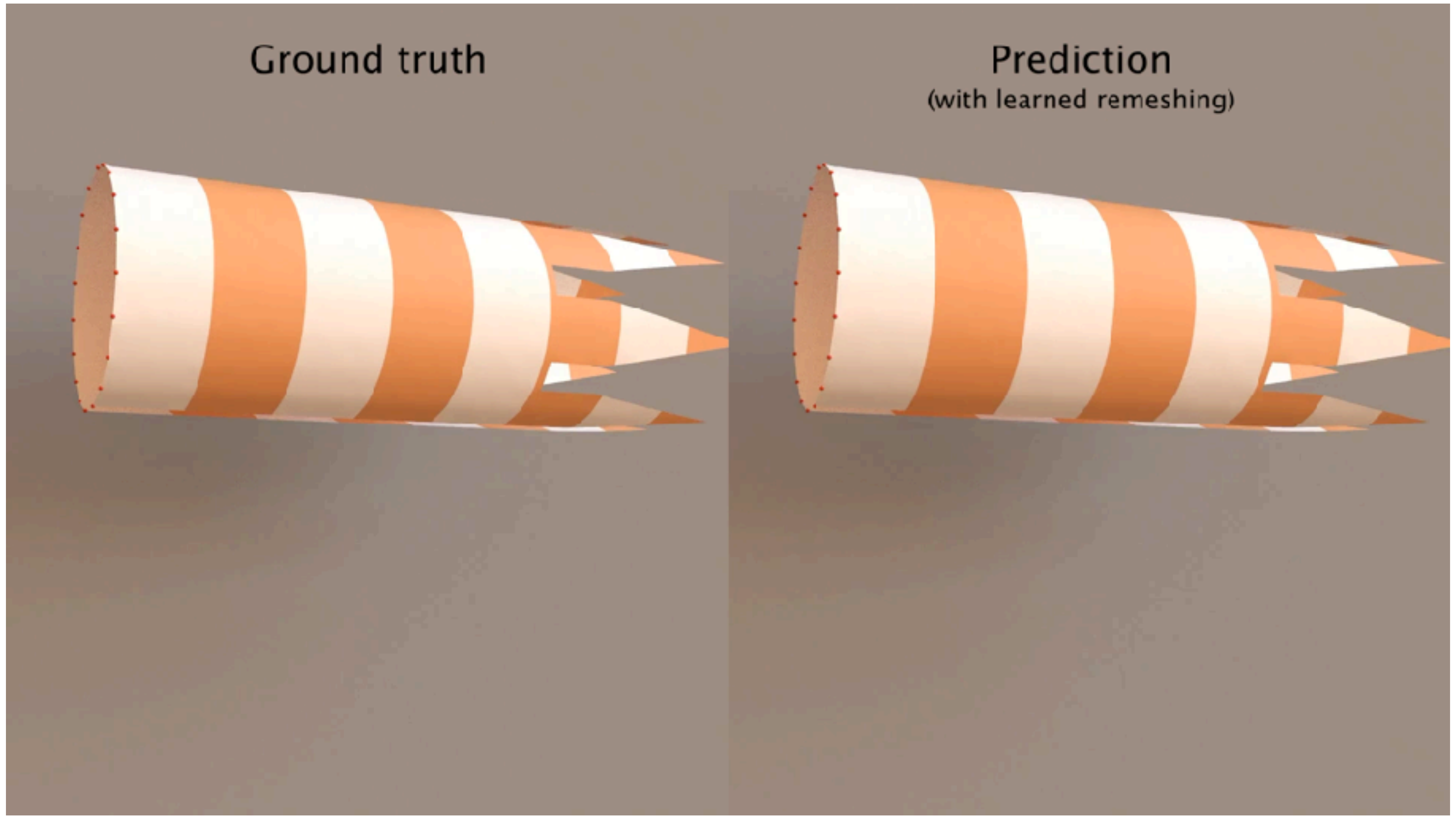
Learning to simulate meshes

Training



Generalization to different shapes



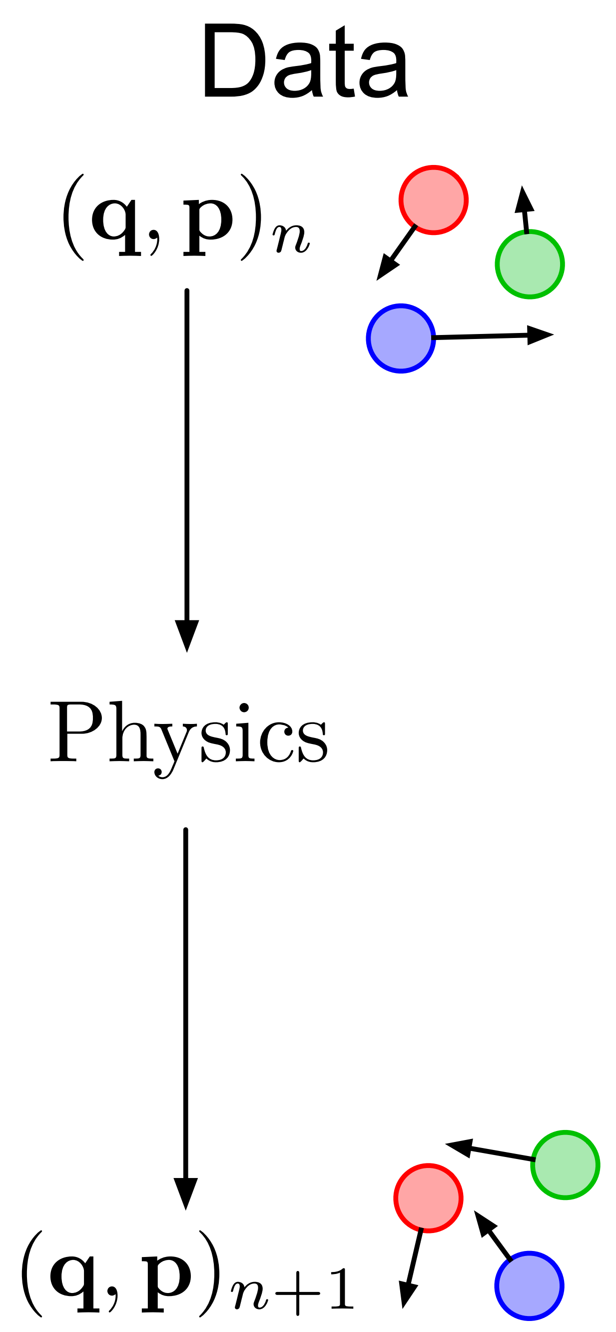


Pfaff et al., 2020, arXiv/under review

For a deep dive on using graph networks for learning complex simulations, see Alvaro Sanchez-Gonzalez’s presentation:

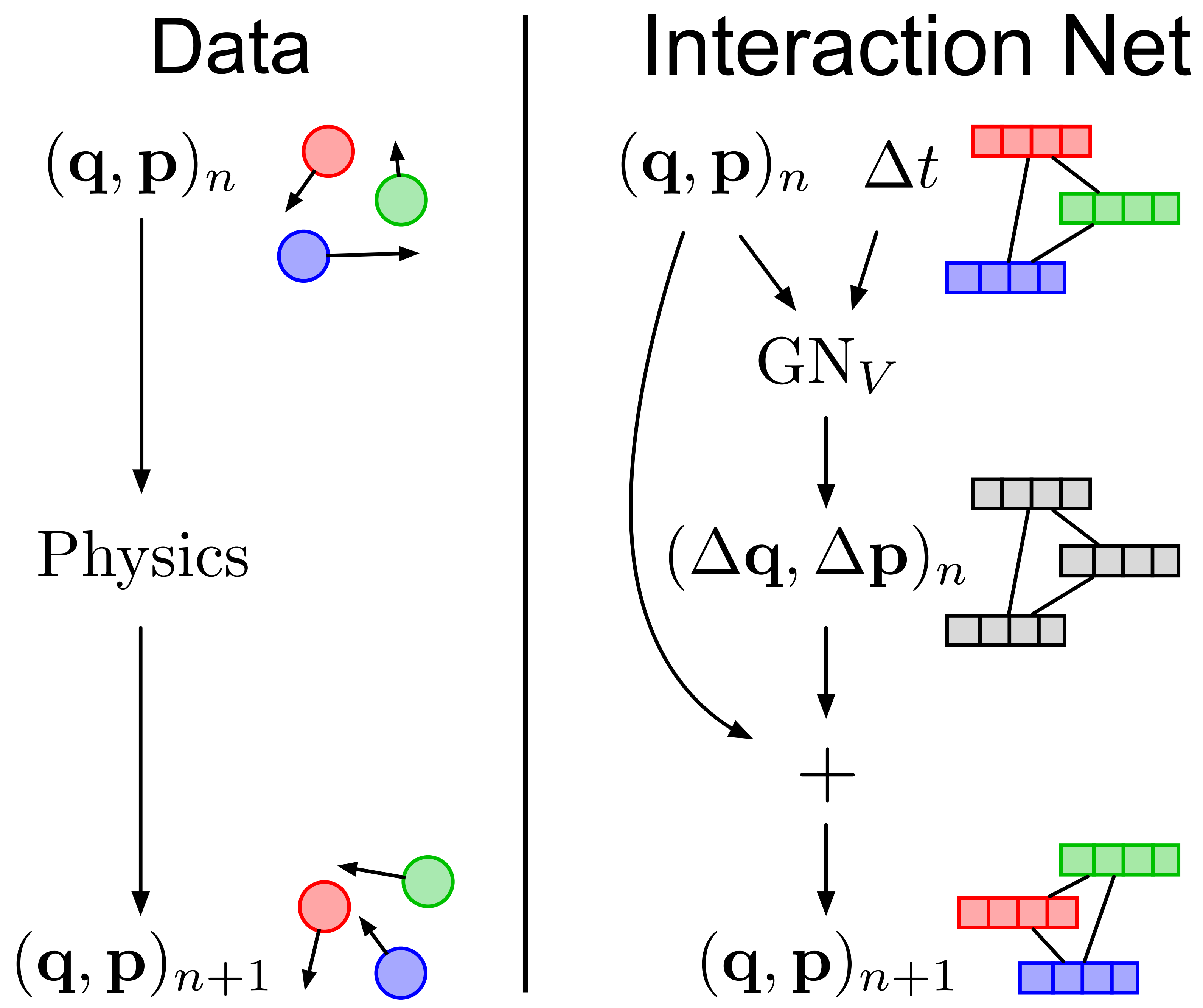
Thursday @ 4-5pm

Hamiltonian ODE Graph Network



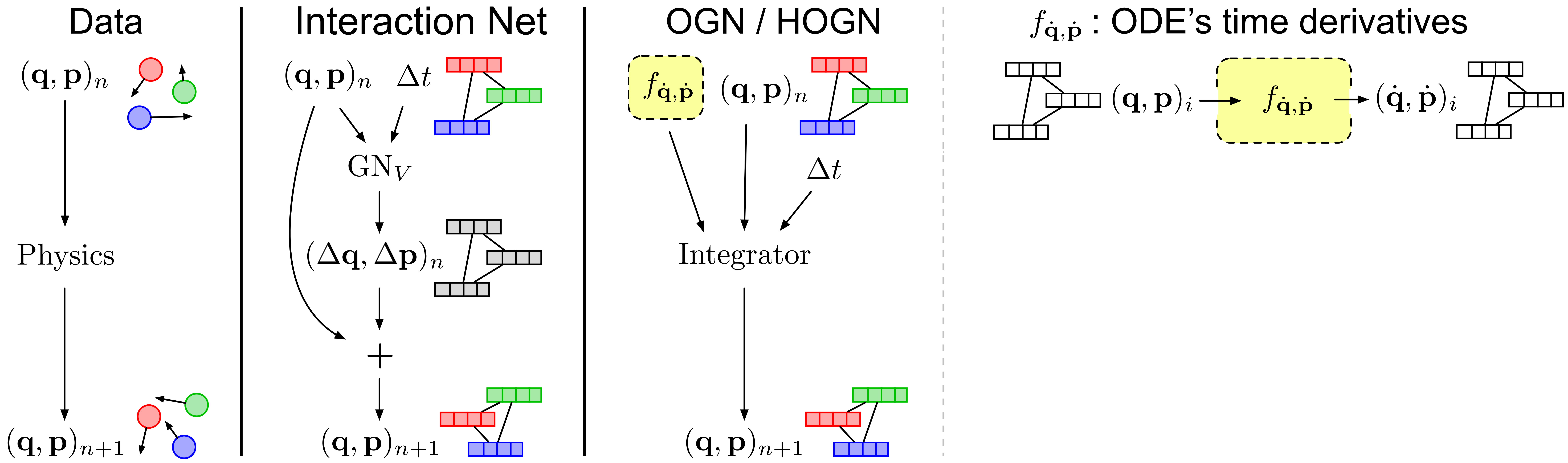
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



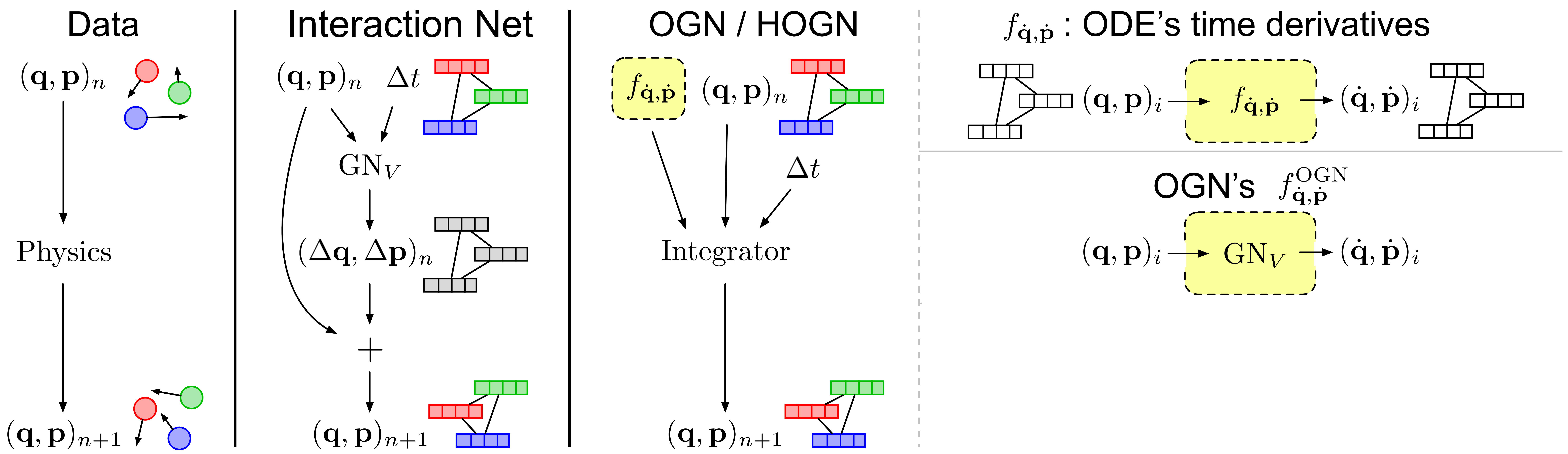
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



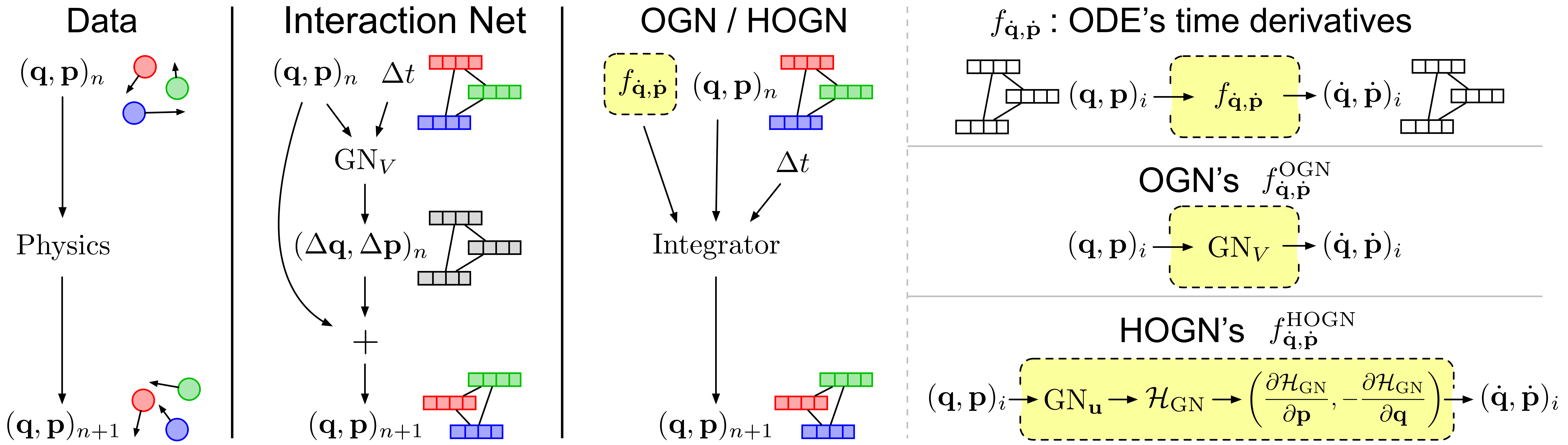
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



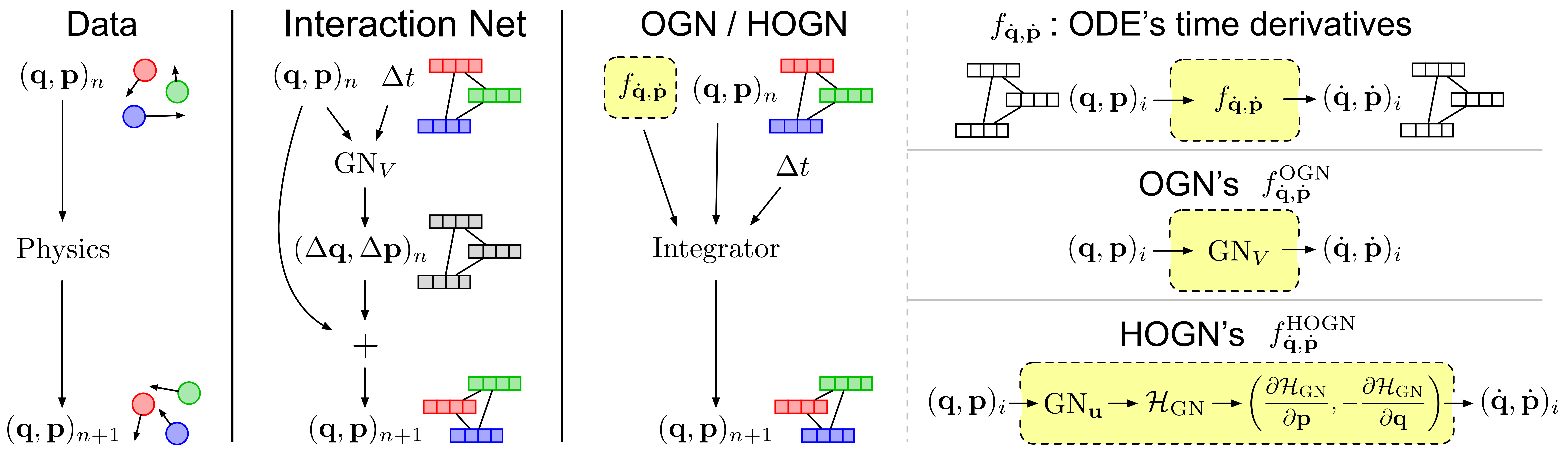
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network



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

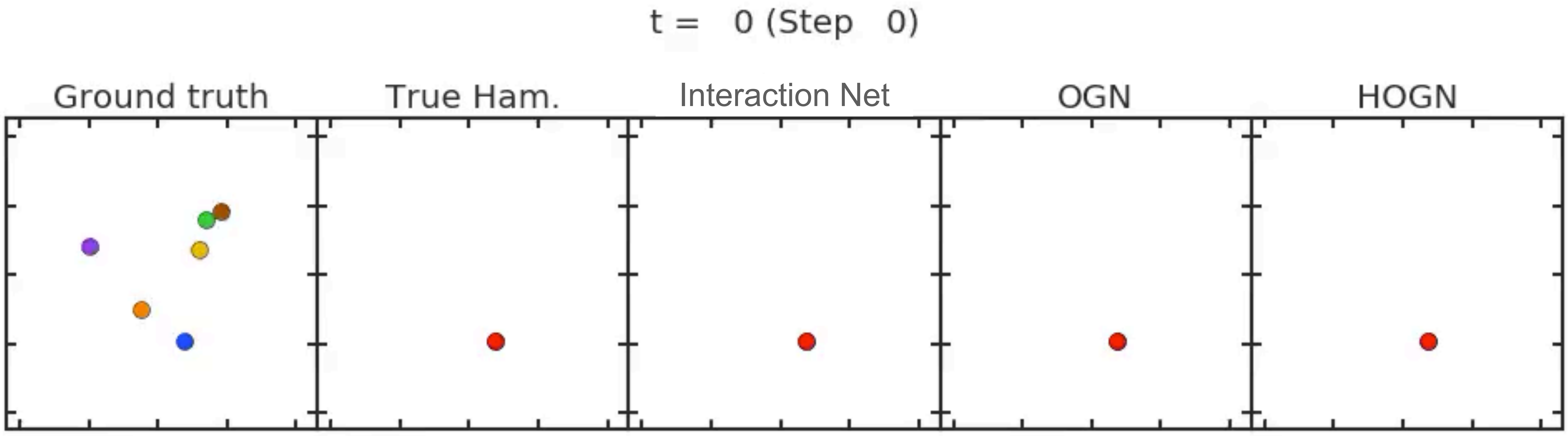
Hamiltonian ODE Graph Network



- * The general idea came from Kyle Cranmer, who was a co-author on this.
- * Also related to Greydanus et al. 2020 “Hamiltonian Neural Networks”.

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

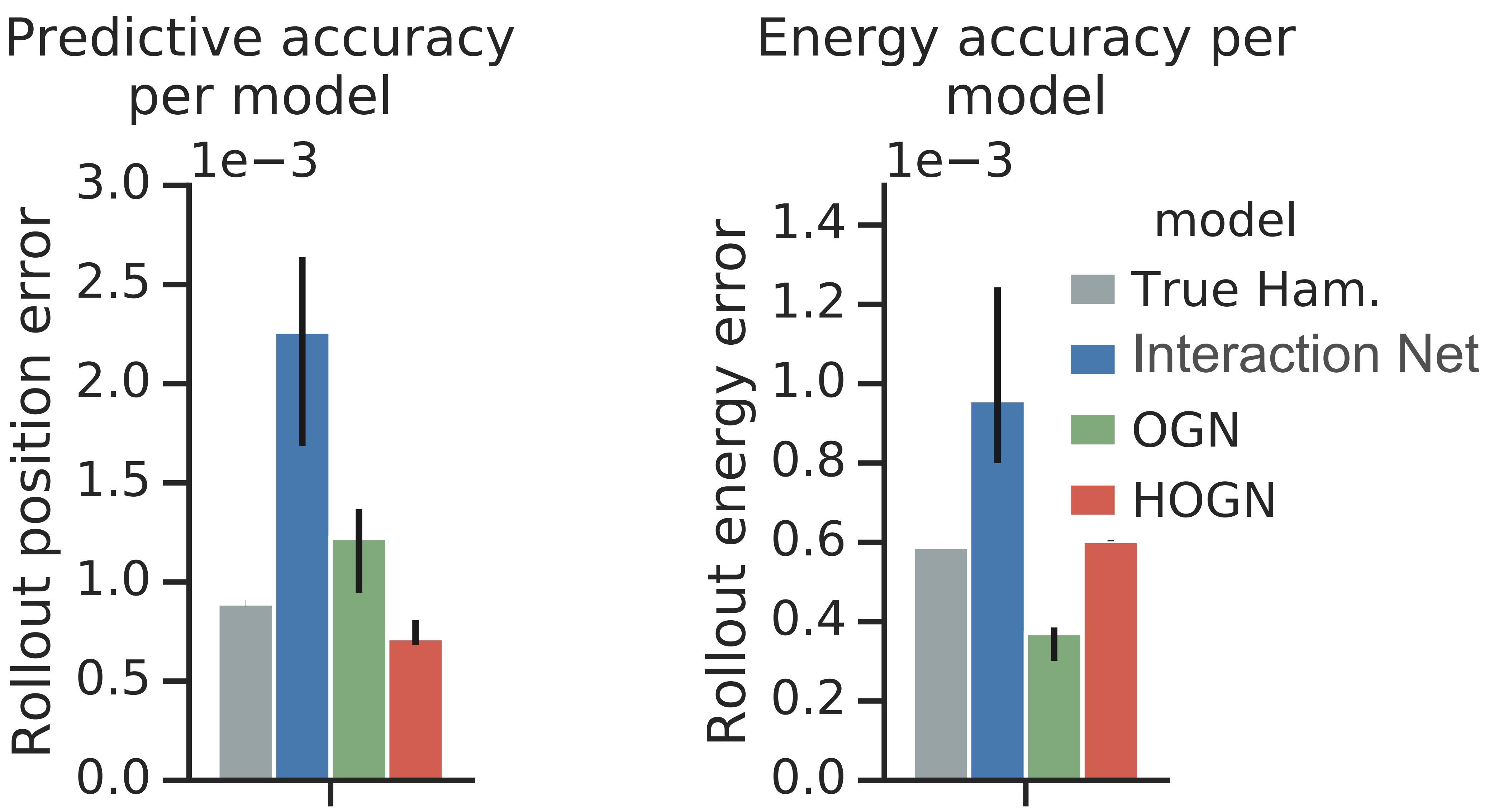
Hamiltonian ODE Graph Network



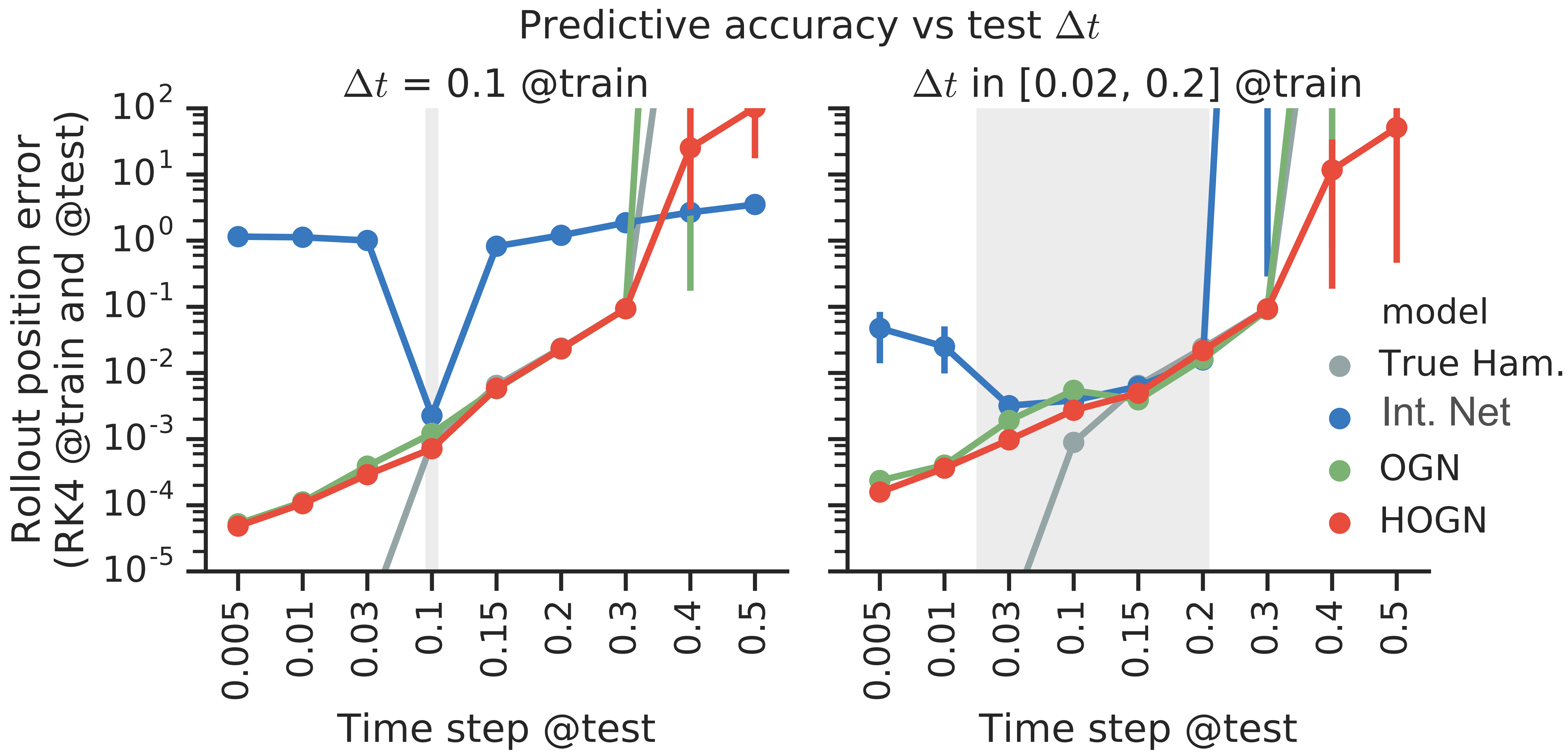
Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS 2019 workshop

Hamiltonian ODE Graph Network

Performance



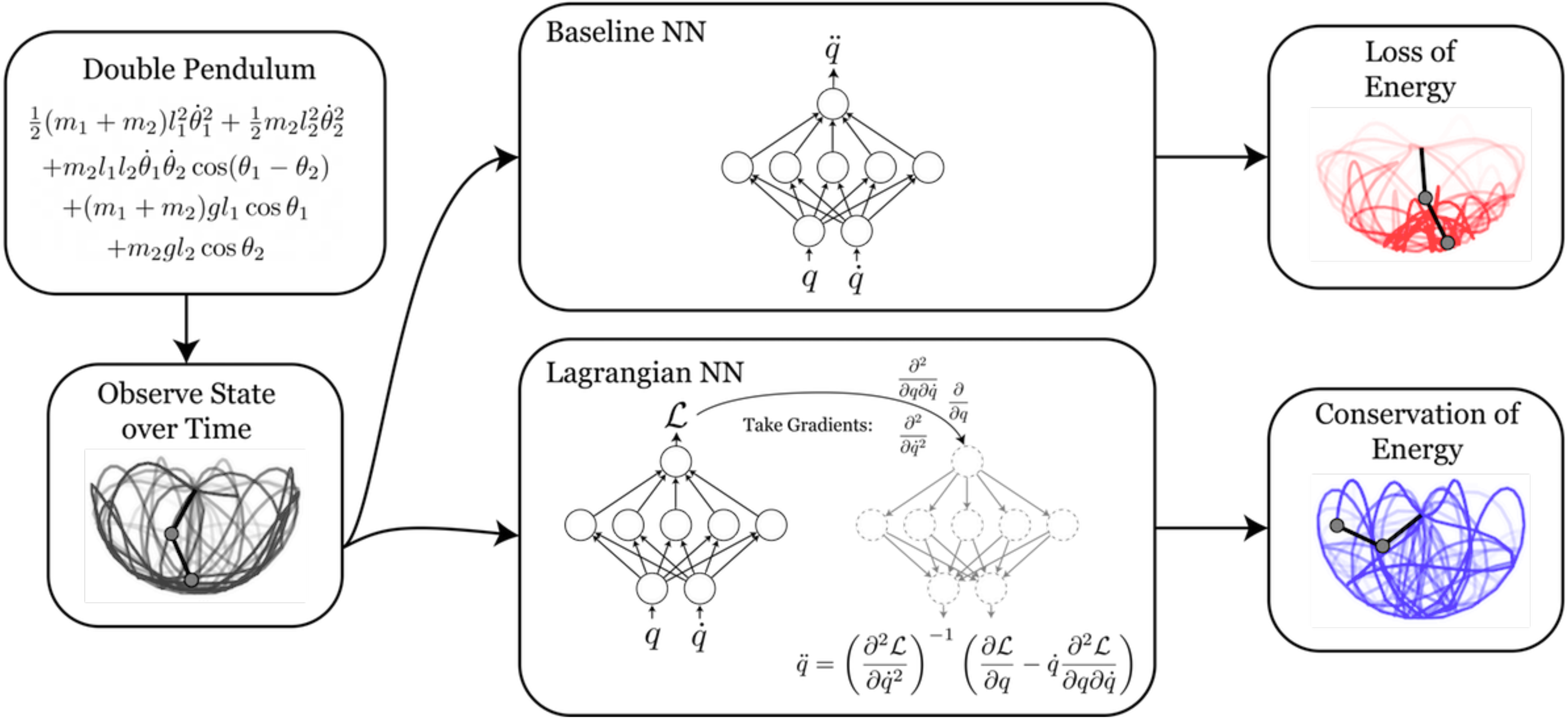
Generalization to untrained time steps



- 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

Lagrangian Neural Network



Cranmer et al., 2020, arXiv/ICLR 2020 workshop

Lagrangian Neural Network

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} = \frac{\partial \mathcal{L}}{\partial q_j}$$

Euler-Lagrange equation

$$\frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} = \nabla_q \mathcal{L}$$

Euler-Lagrange equation (vector form)

$$(\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L})\ddot{q} + (\nabla_q\nabla_{\dot{q}}^{\top}\mathcal{L})\dot{q} = \nabla_q \mathcal{L}$$

Apply chain rule to get $\dot{q}$ and $\ddot{q}$

$$\ddot{q} = (\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L})^{-1}[\nabla_q \mathcal{L} - (\nabla_q\nabla_{\dot{q}}^{\top}\mathcal{L})\dot{q}]$$

Solve for $\ddot{q}$

$\mathcal{L}$ is represented by a trainable neural network, and JAX's auto-differentiation features let us compute $\ddot{q}$ in a few lines of code, which is then passed to an integrator

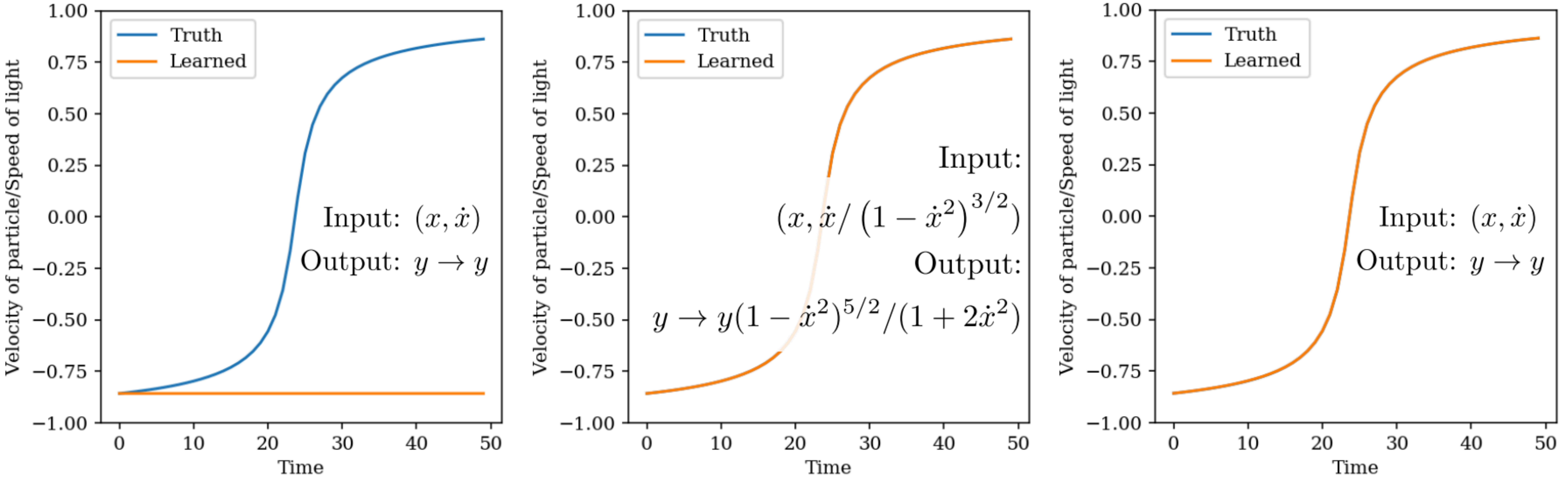
Cranmer et al., 2020, arXiv/ICLR 2020 workshop

Lagrangian Neural Network

Advantage: You don't need canonical position and momentum coordinates
—remember the Hamiltonian approach required $\mathbf{p}$ to compute $\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}$

Relativistic particle task, where canonical momentum is: $\dot{q}(1-\dot{q}^2)^{-3/2}$

Hamiltonian NN (arbitrary coordinates) | Hamiltonian NN (canonical coordinates) | Lagrangian NN (arbitrary coordinates)

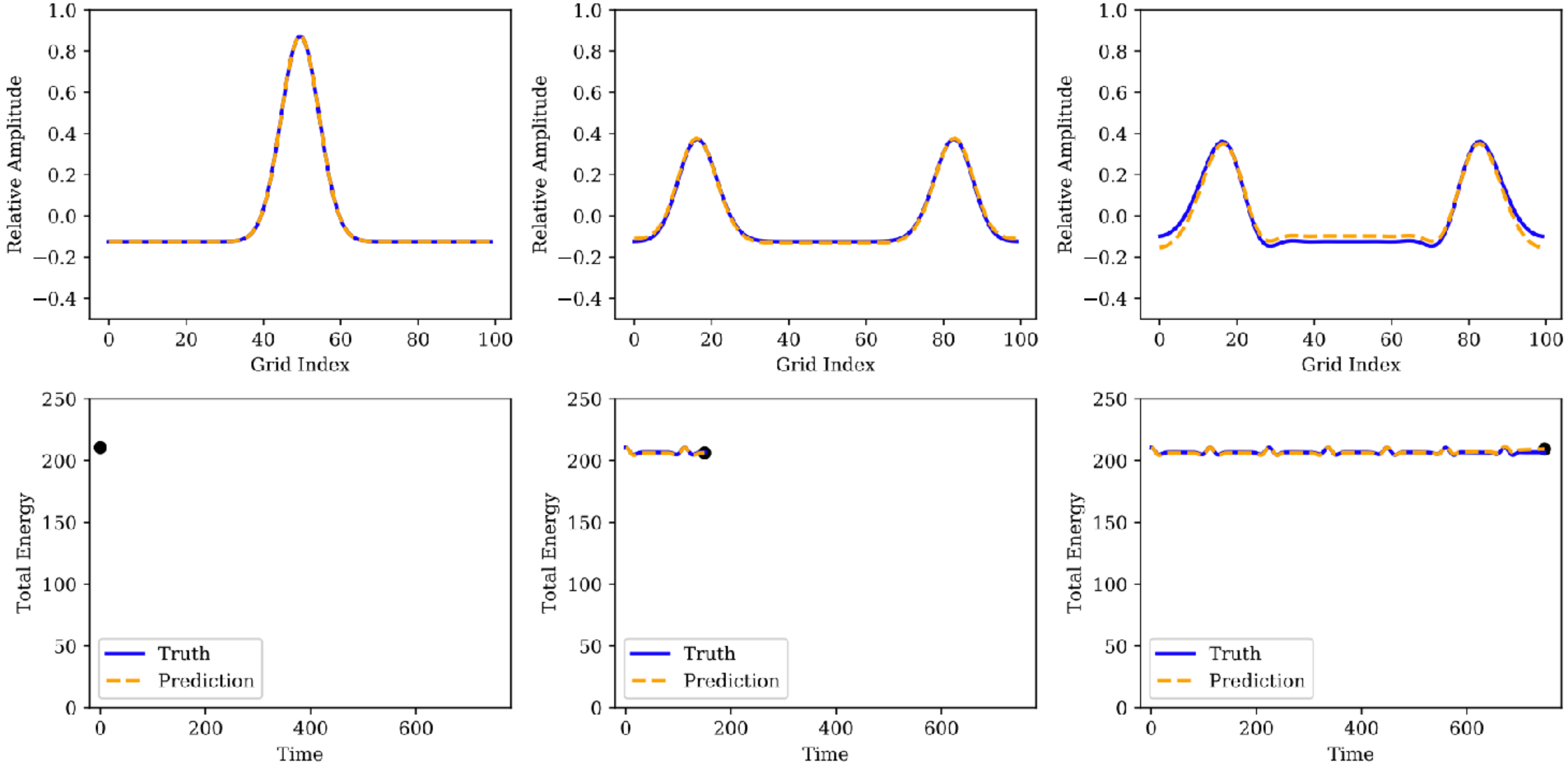


Cranmer et al., 2020, arXiv/ICLR 2020 workshop

Lagrangian Graph Network

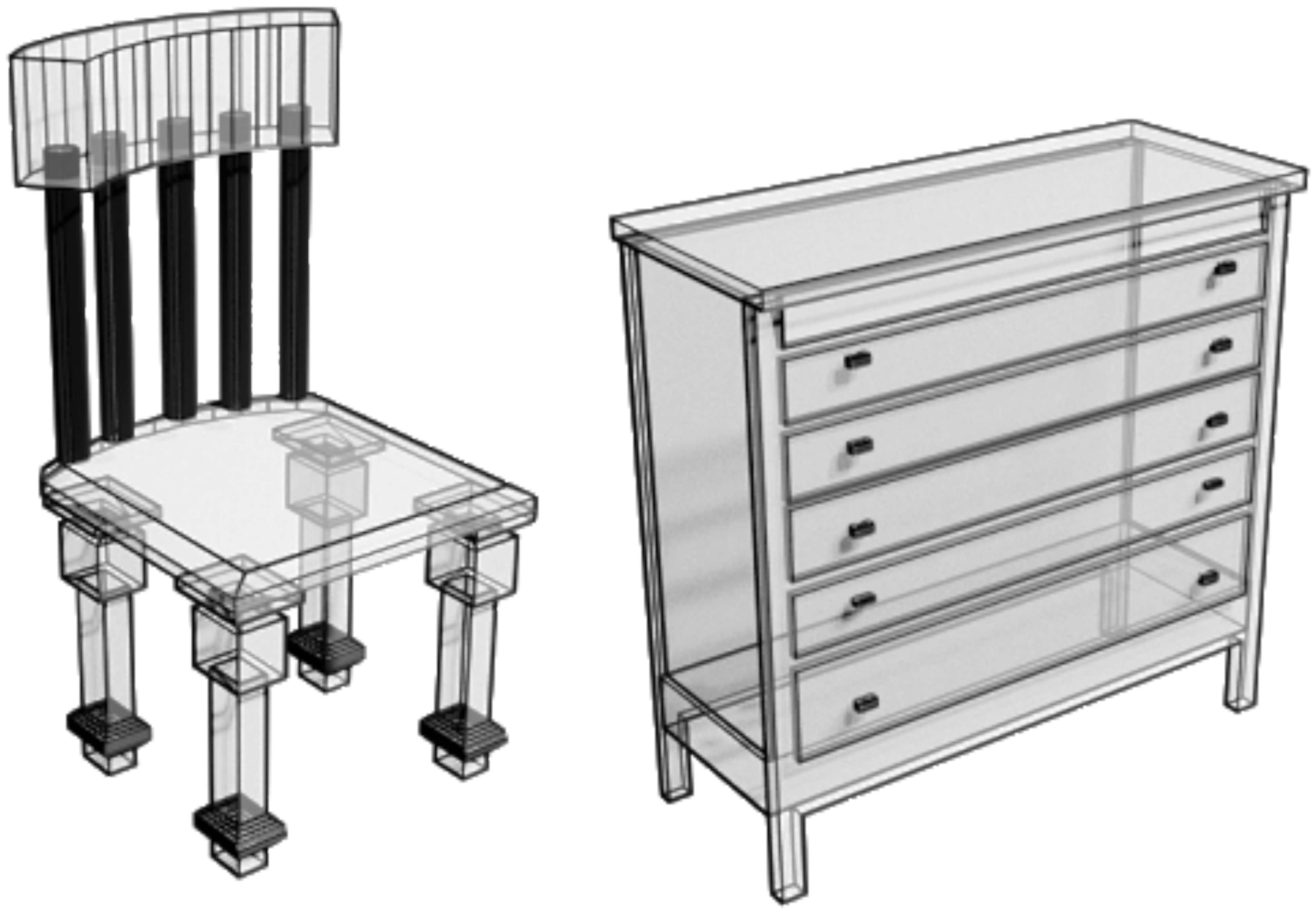
Can also use extend the LNN to apply to graph structures

1D wave equation (periodic boundary conditions) - nodes are vertices on the mesh



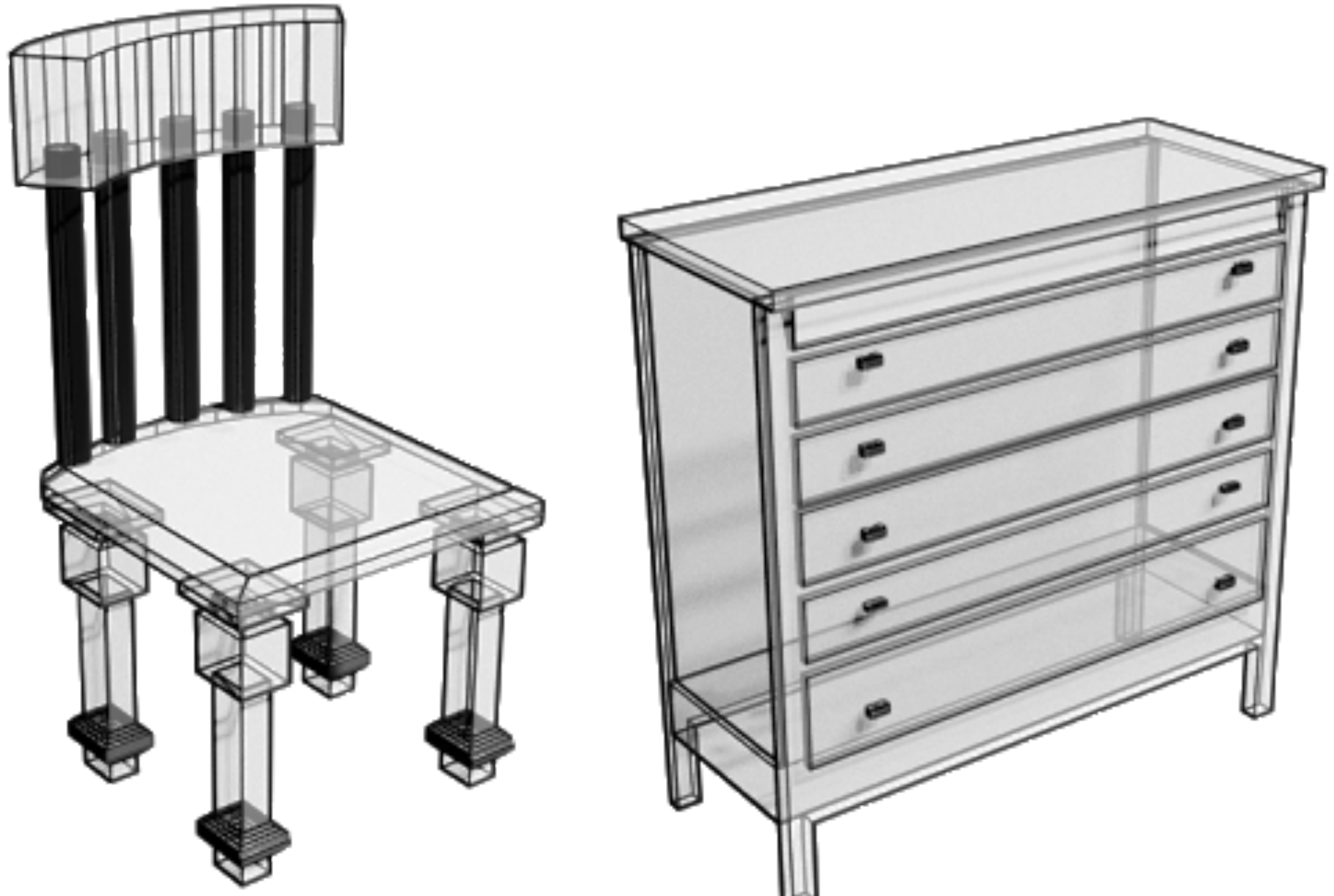
Cranmer et al., 2020, arXiv/ICLR 2020 workshop

PolyGen: Autoregressive generative model of 3D meshes



Nash et al., 2020, ICML

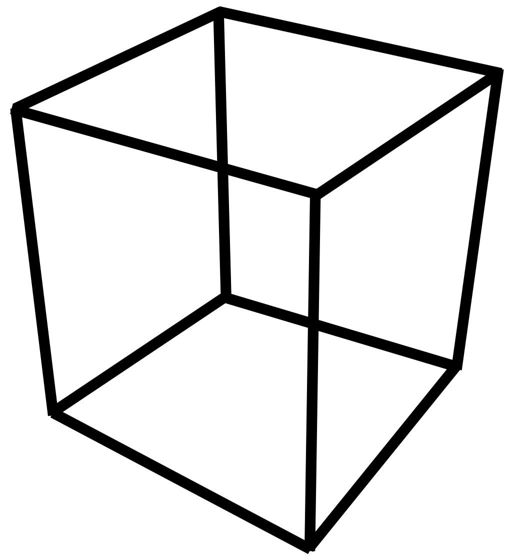
PolyGen: Autoregressive generative model of 3D meshes



.OBJ file: cube

```
v 0.000000 2.000000 2.000000
v 0.000000 0.000000 2.000000
v 2.000000 0.000000 2.000000
v 2.000000 2.000000 2.000000
v 0.000000 2.000000 0.000000
v 0.000000 0.000000 0.000000
v 2.000000 0.000000 0.000000
v 2.000000 2.000000 0.000000

f 1 2 3 4
f 8 7 6 5
f 4 3 7 8
f 5 1 4 8
f 5 6 2 1
f 2 6 7 3
```



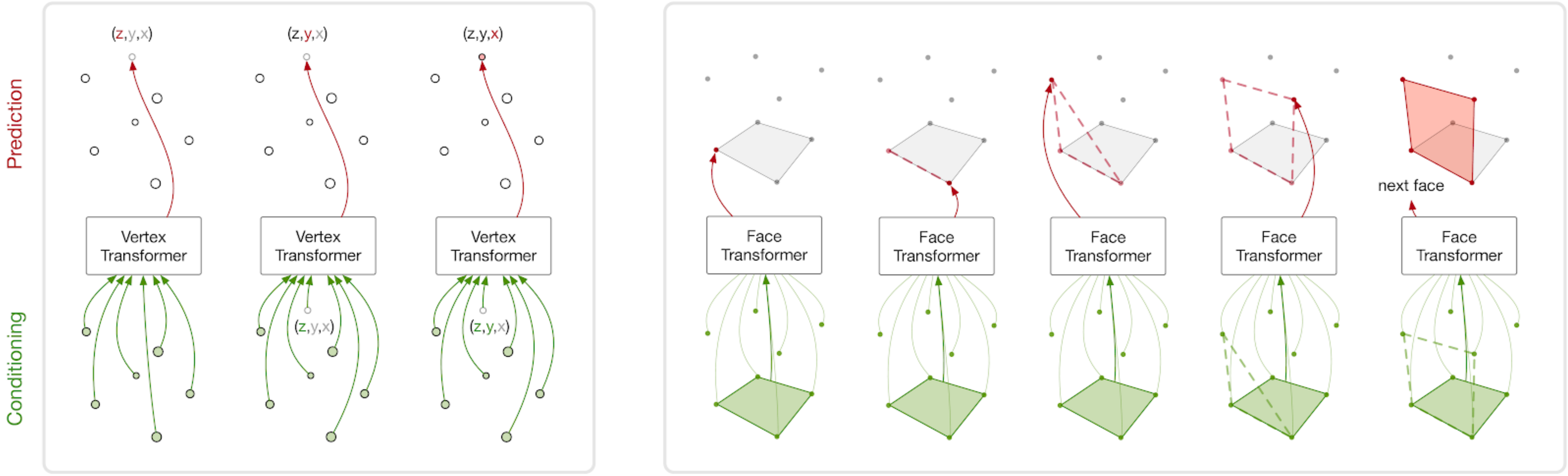
Nash et al., 2020, ICML

PolyGen: Autoregressive generative model of 3D meshes

Architecture: Transformer-based

Two phases:

1. Vertex model
2. Face model



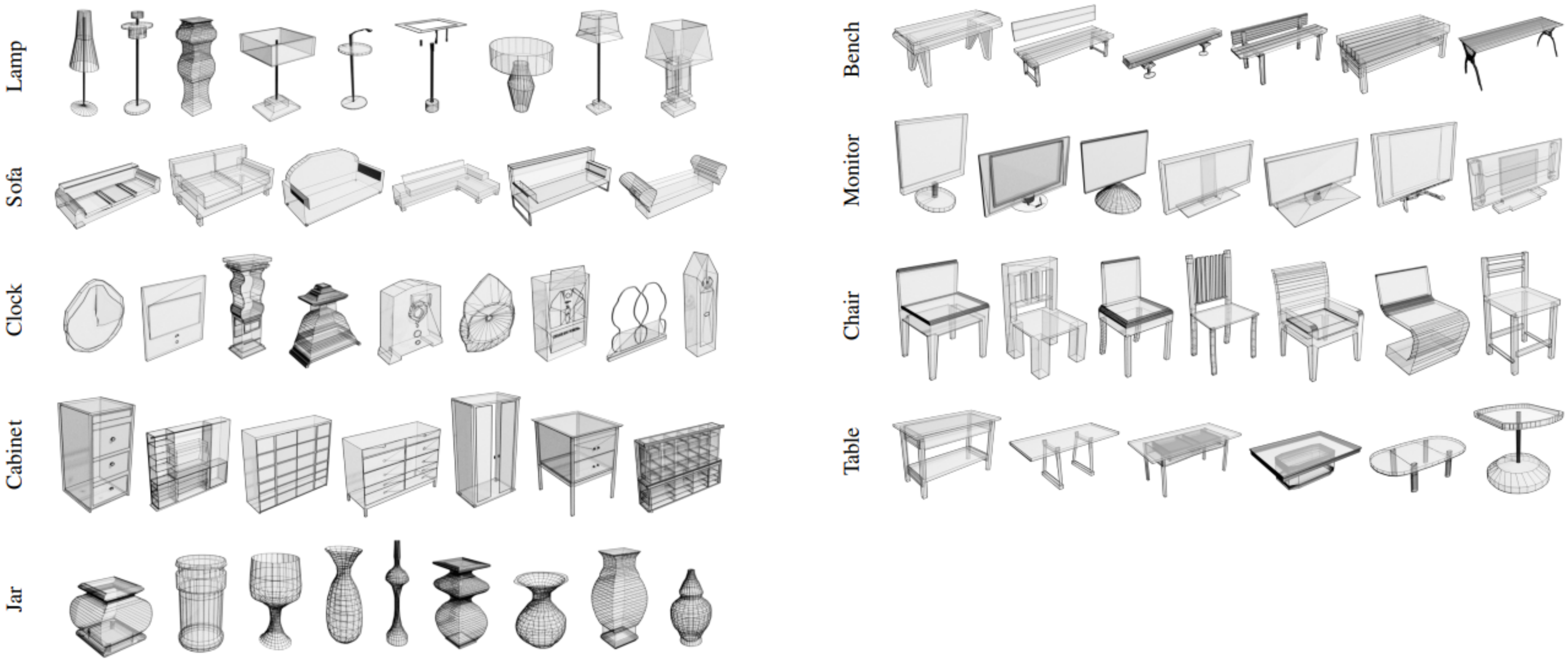
Vertex Model

Face Model

Nash et al., 2020, ICML

PolyGen: Autoregressive generative model of 3D meshes

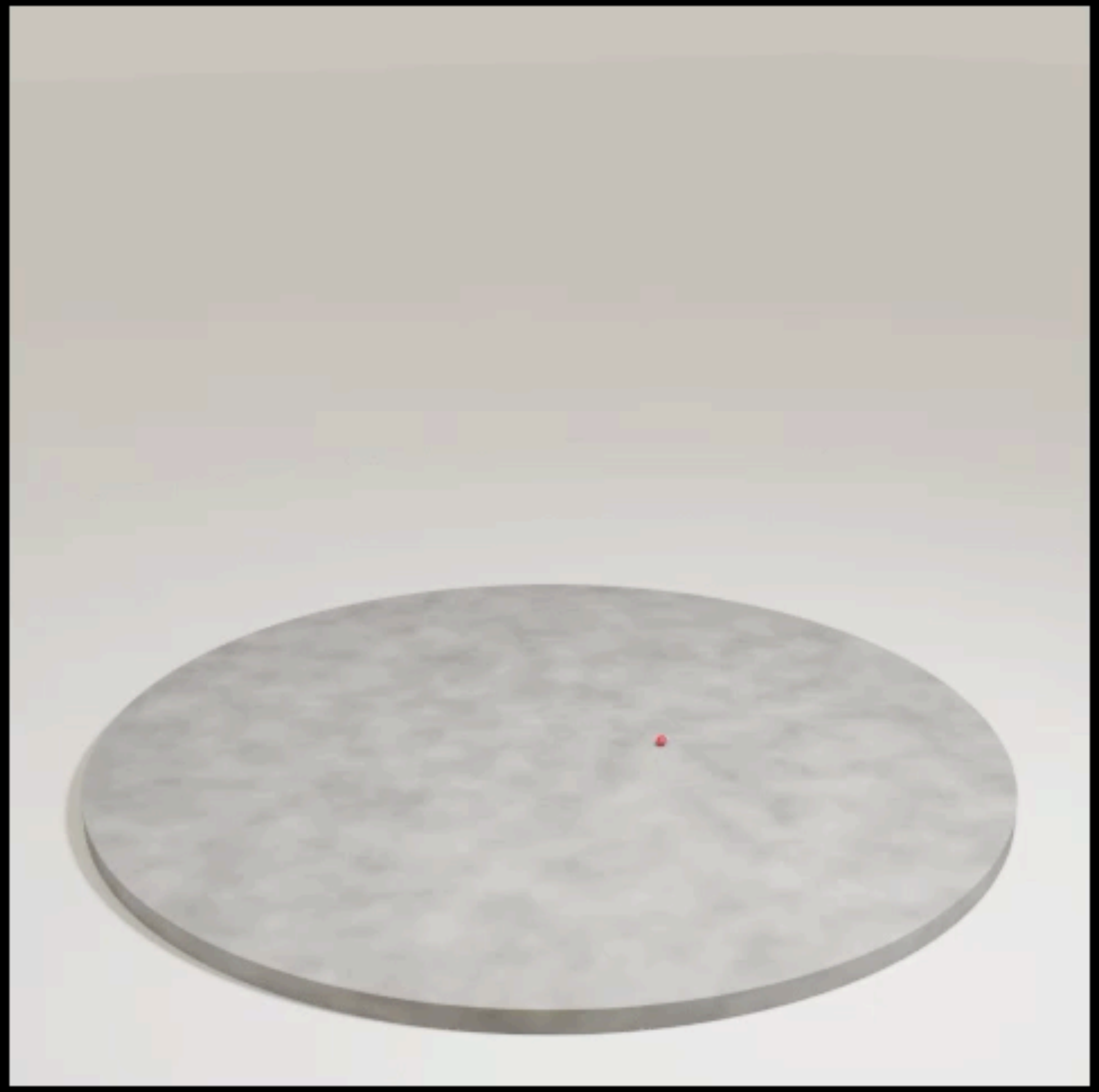
Class-conditional samples

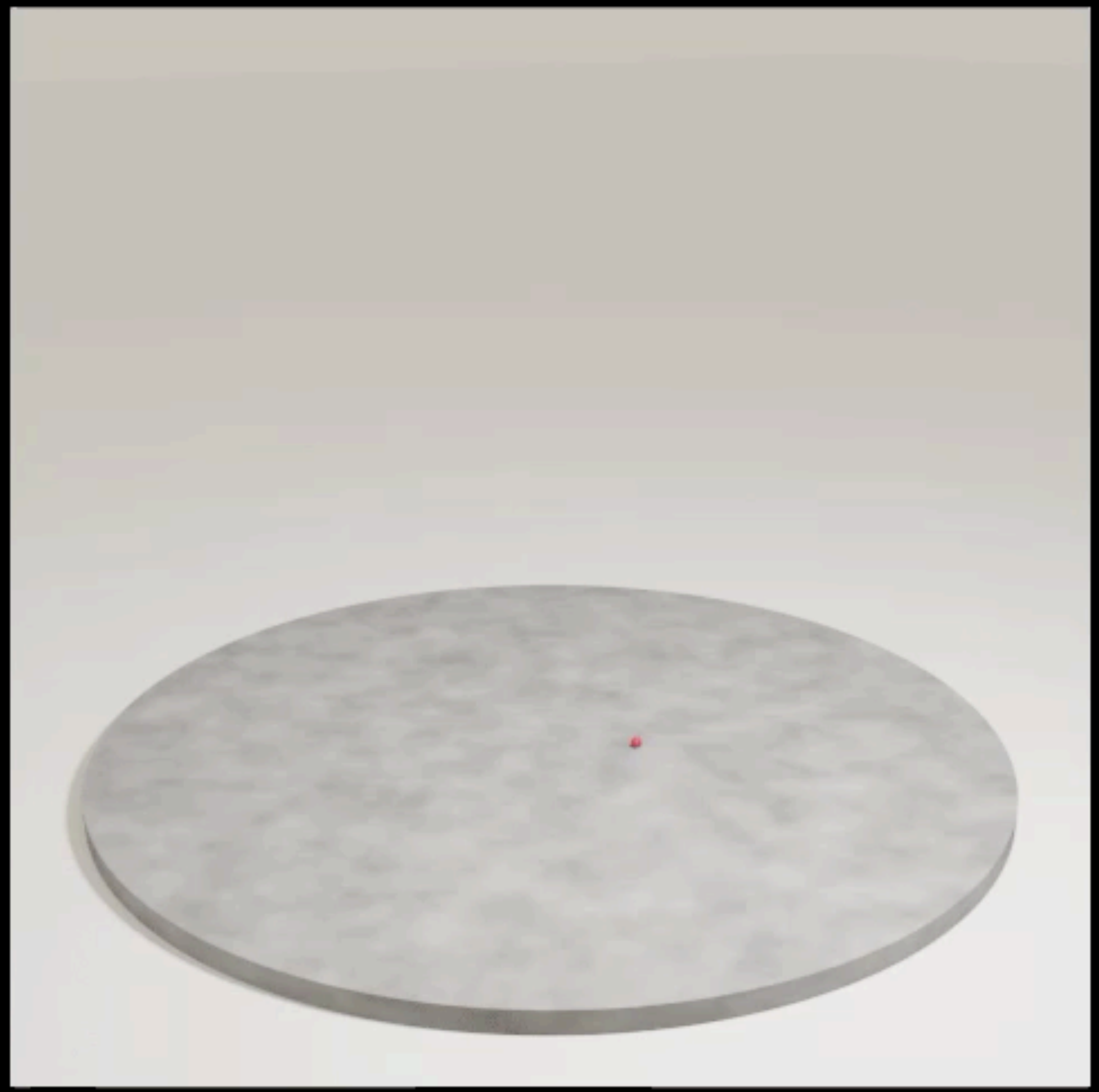


Nash et al., 2020, ICML

PolyGen

An Autoregressive Generative Model of 3D Meshes



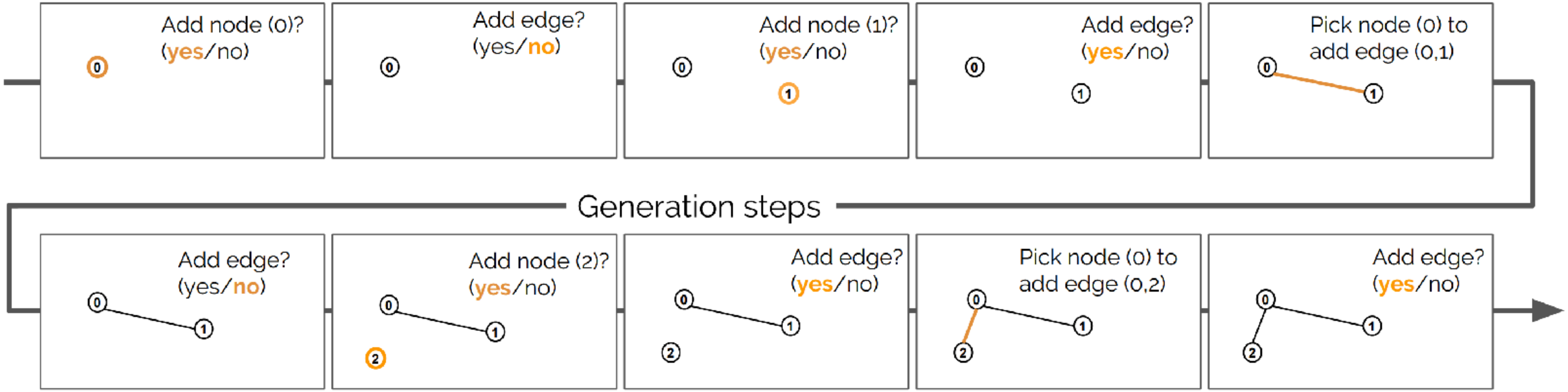


Class conditional samples

Nash et al., 2020, ICML

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

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:

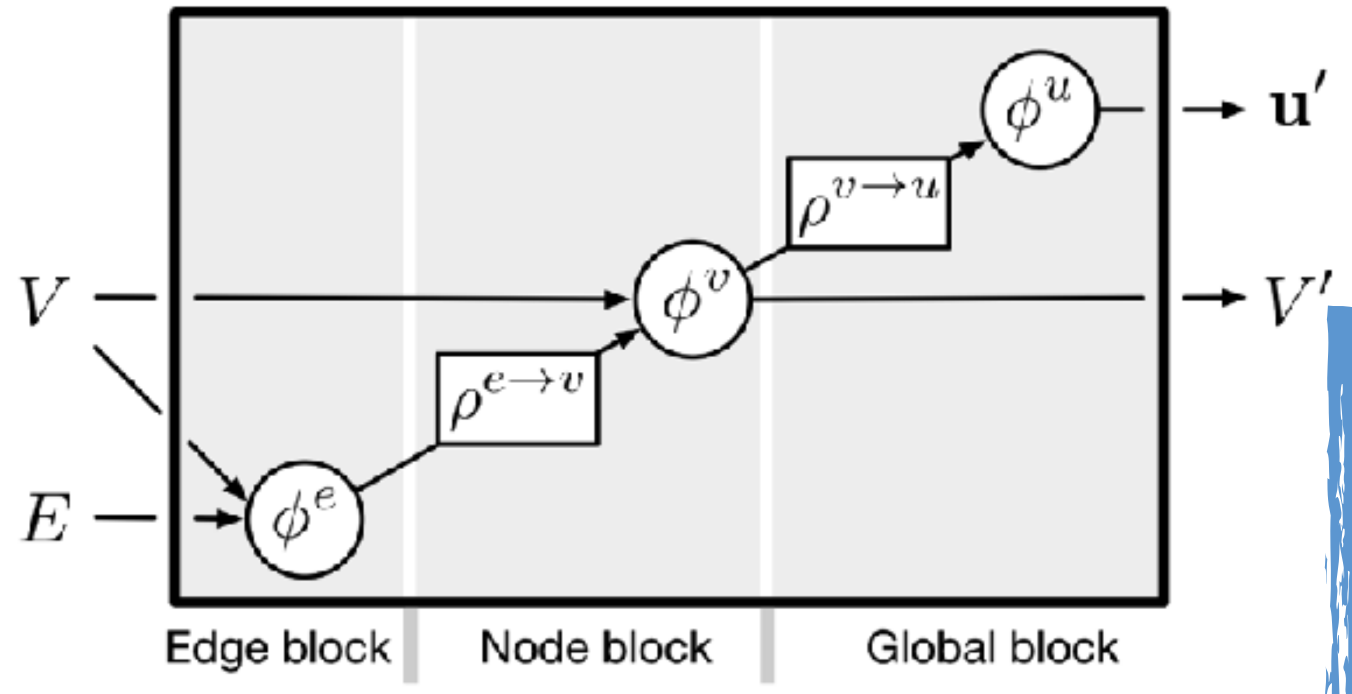
- pytorch_geometric: github.com/rusty1s/pytorch_geometric (for a GN analog, see `MetaLayer`)
- Deep Graph Library: github.com/dmlc/dgl

Build Graph Nets in Tensorflow

github.com/deepmind/graph_nets

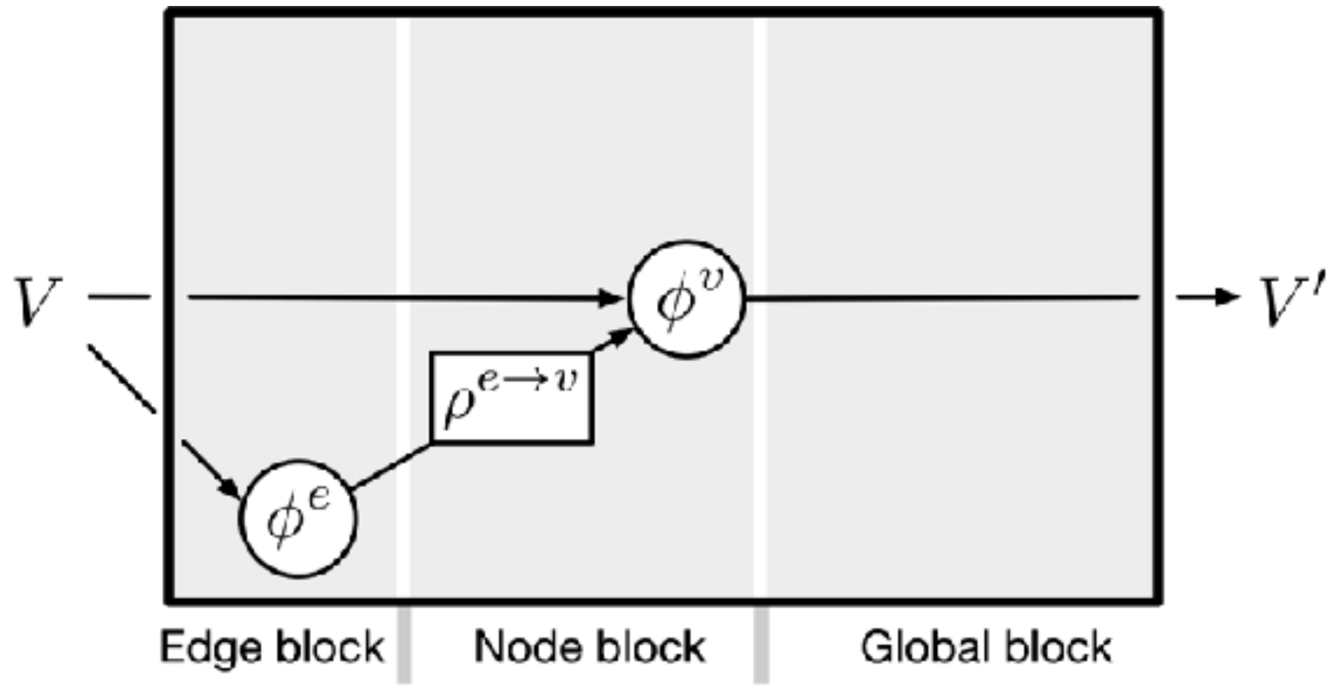
Message-Passing NN (eg. Interaction Net)

Gilmer et al. 2017

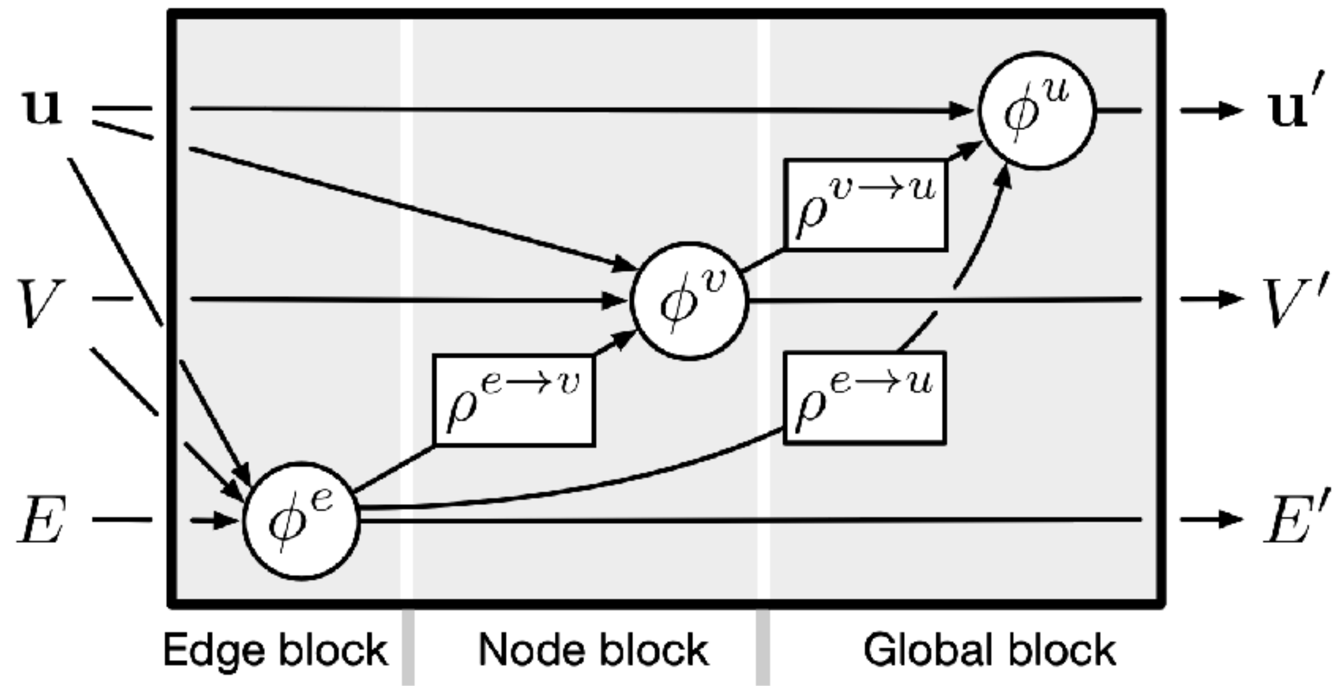


Non-Local NN (eg. Transformer)

Vaswani et al. 2017; Wang et al. 2017

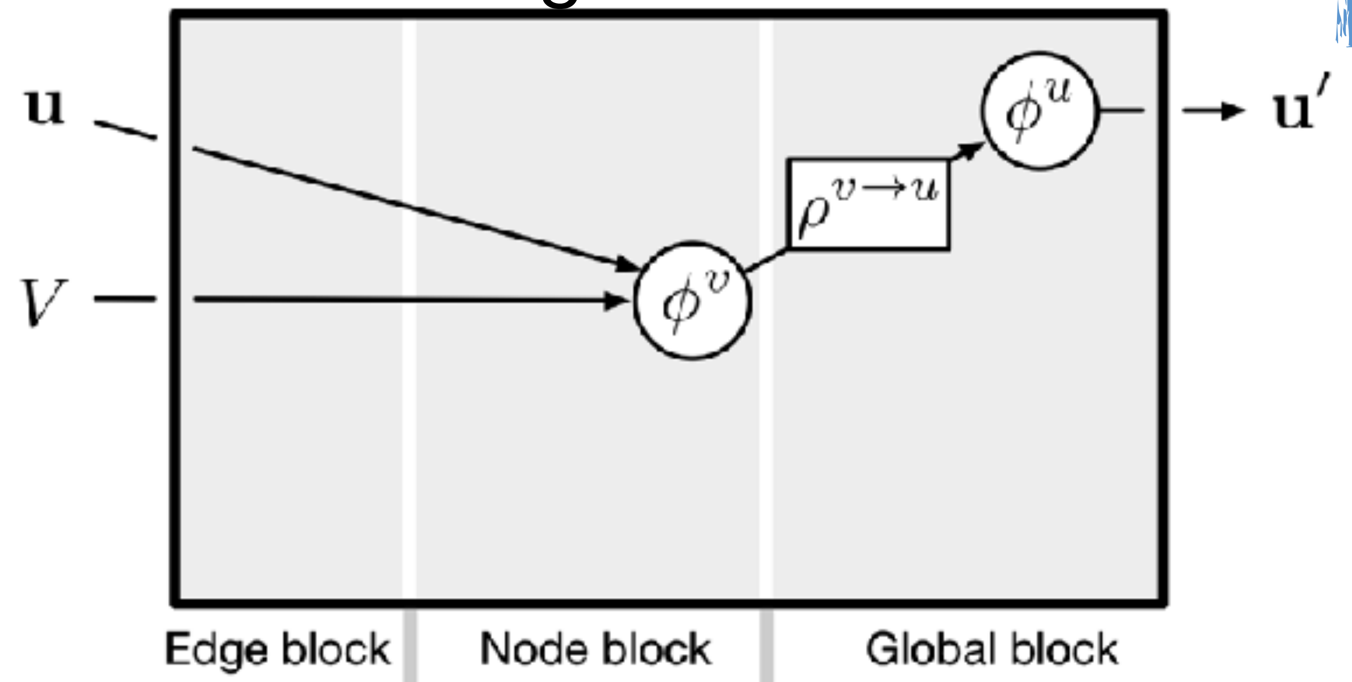


Graph Network
(a type of Graph Neural Network)



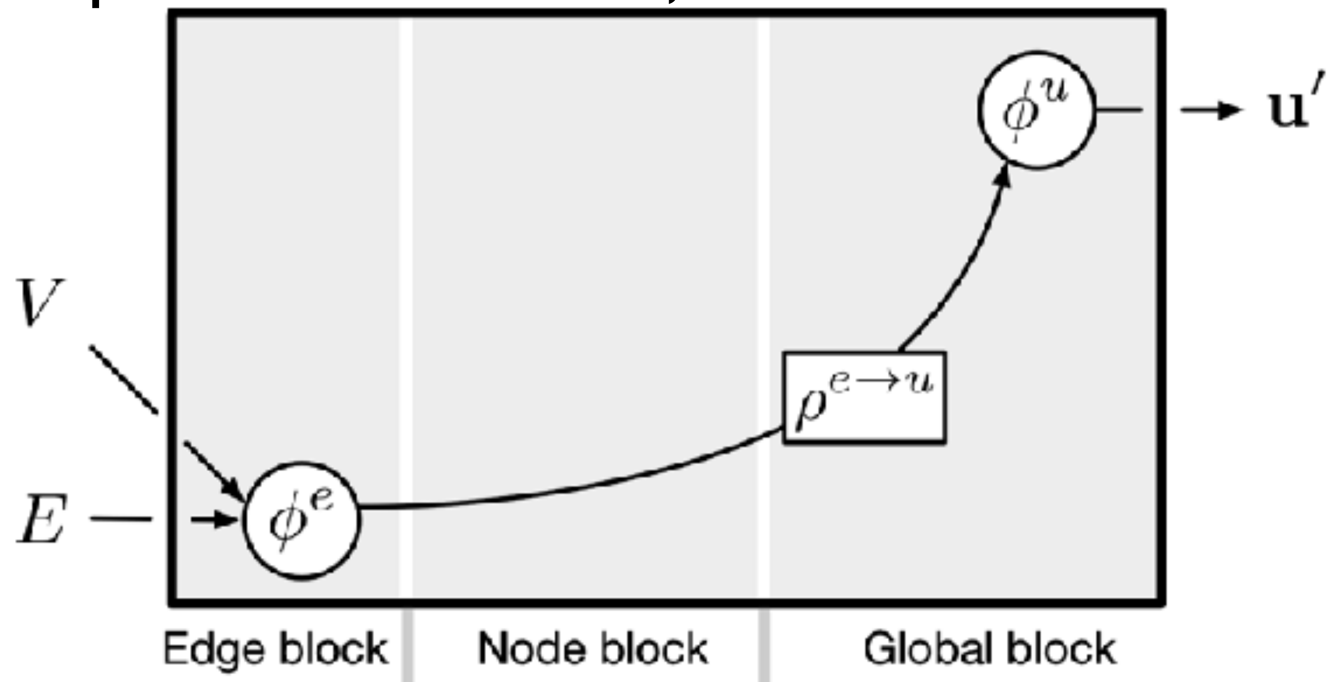
Deep Sets

Zhang et al. 2017



Relation Network

Raposo et al. 2017; Santoro et al. 2017



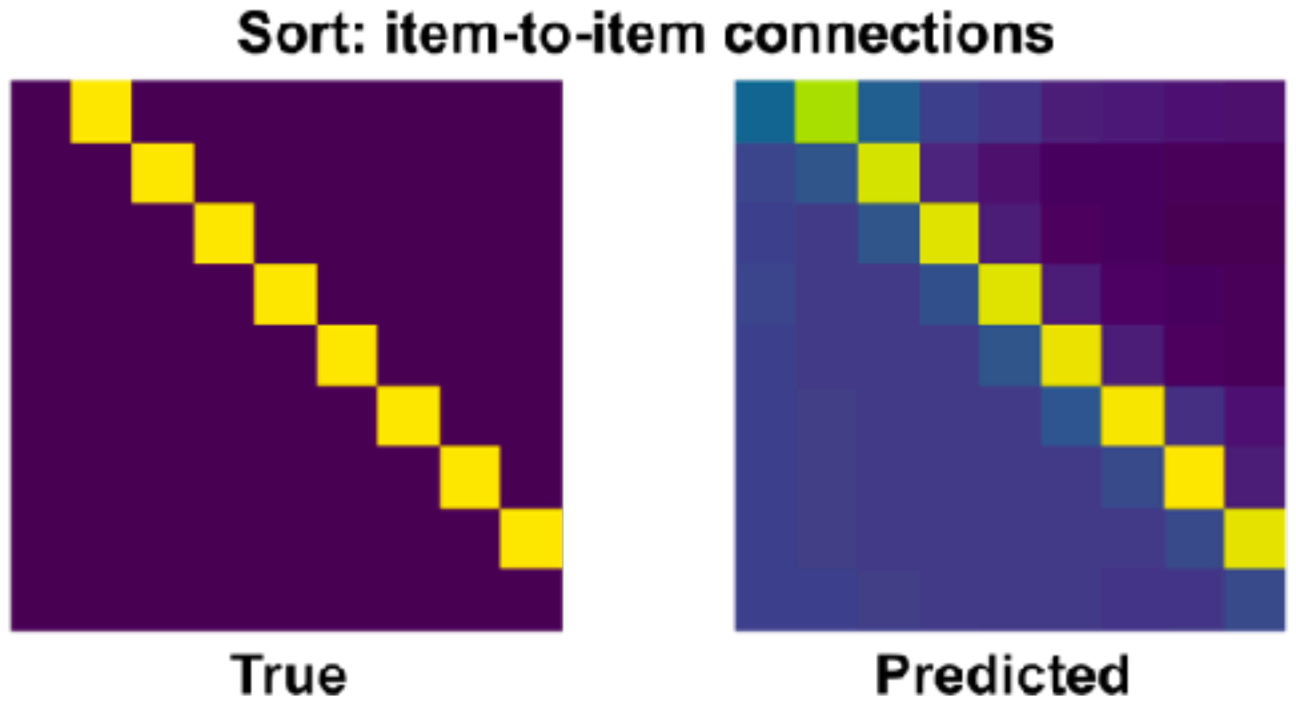
Build Graph Nets in Tensorflow

github.com/deepmind/graph_nets

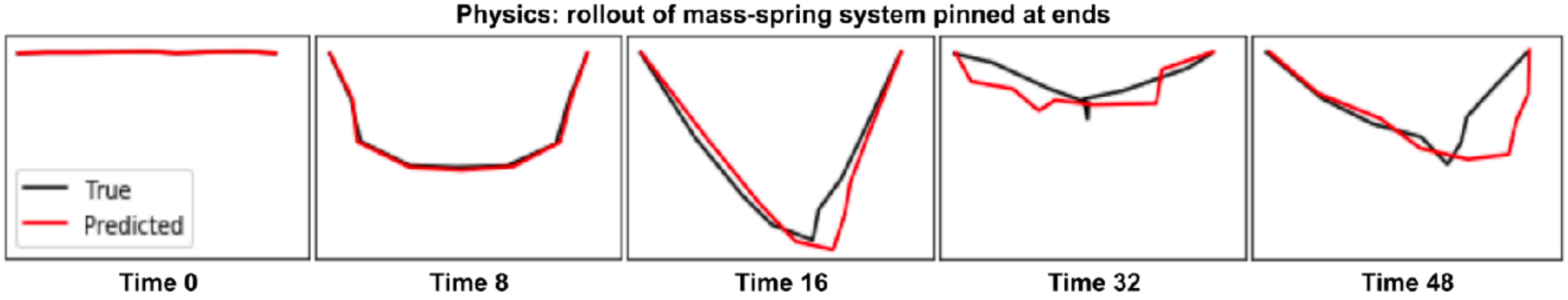
IPython Notebook demos
(All use same architecture)

Shortest path:

Sorting:



Predicting physics:



Conclusions

- Graph neural networks: a first-class member of the deep learning toolkit.
- Learned message-passing on graphs can capture complex physical knowledge.
- "Graph Nets" support learning simulation, as well as other forms of structured reasoning and decision-making.
- Build Graph Nets in Tensorflow: `github.com/deepmind/graph_nets`.
- For a recent review of GNNs in HEP, please check out:
  Shlomi, Battaglia, Vlimant (2020) "Graph Neural Networks in Particle Physics".
  arXiv 2007.13681 and 2 weeks ago in Machine Learning: Science and Technology

Key collaborators

Alvaro Sanchez-Gonzalez
Jonny Godwin
Tobi Pfaff
Meire Fortunato
Rex Ying
Jure Leskovec
Charlie Nash
Yaroslav Ganin
Ali Eslami
Miles Cranmer
Shirley Ho
David Spergel
Sam Greydanus
Stephan Hoyer
Jessica Hamrick
Victor Bapst
Razvan Pascanu
Nicholas Heess
Yujia Li
Oriol Vinyals

References

[Battaglia et al. 2018 arXiv]
[Battaglia et al., 2016, NeurIPS]
[Sanchez-Gonzalez et al., 2018, ICML]
[Sanchez-Gonzalez et al., 2020, ICML]
[Pfaff et al., 2020, arXiv/under review]
[Sanchez-Gonzalez et al., 2019, arXiv/NeurIPS workshop]
[Cranmer et al., 2020, arXiv]
[Cranmer et al., 2020, NeurIPS]
[Li et al., 2018, arXiv]
[Nash et al., 2020, ICML]