

---

# Learning to Simulate Complex Physics with Graph Networks

---

**Alvaro Sanchez-Gonzalez<sup>\*1</sup> Jonathan Godwin<sup>\*1</sup> Tobias Pfaff<sup>\*1</sup> Rex Ying<sup>\*12</sup> Jure Leskovec<sup>2</sup>**  
**Peter W. Battaglia<sup>1</sup>**

ICML 2020

From Deepmind & Stanford

Presentor: Yuanhao Xiong, Xiangning Chen, Li-Cheng Lan

# Introduction

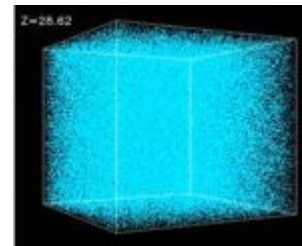
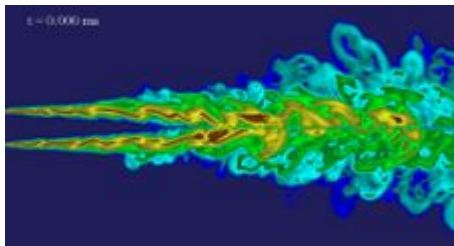
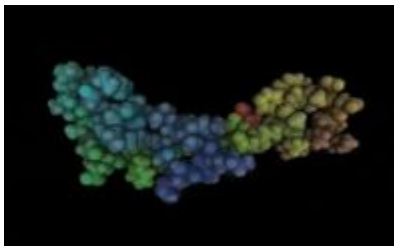
- Importance of simulation
  - Simulators of complex physics are invaluable to scientific and engineering disciplines
  - Largest supercomputers in the world

#1. "Summit" @ Oak Ridge: "A Sneak Peek at 19 Science **Simulations** for the Summit Supercomputer in 2019"

- |  |   |
|--|---|
| 1. <i>Evolution of the universe</i>          | 11. <i>Cancer data</i>                        |
| 2. <i>Whole-cell simulation</i>              | 12. <i>Earthquake resilience for cities</i>   |
| 3. <i>Inside a nuclear reactor</i>           | 13. <i>Nature of elusive neutrinos</i>        |
| 4. <i>Post-Moore's Law graphene circuits</i> | 14. <i>Extreme weather with deep learning</i> |
| 5. <i>Formation of matter</i>                | 15. <i>Flexible, lightweight solar cells</i>  |
| 6. <i>Cell's molecular machine</i>           | 16. <i>Virtual fusion reactor</i>             |
| 7. <i>Unpacking the nucleus</i>              | 17. <i>Unpredictable material properties</i>  |
| 8. <i>Mars landing</i>                       | 18. <i>Genetic clues in the opioid crisis</i> |
| 9. <i>Deep learning for microscopy</i>       | 19. <i>Turbulent environments</i>             |
| 10. <i>Elements from star explosions</i>     |   |

# Introduction

- Difficulties in building a high-quality simulator
  - Can be very expensive to create and use
  - Entail years of engineering effort
  - Trade off generality for accuracy in a narrow range of settings
- Turn to learned simulators
  - Shared architectures
  - Accuracy-efficiency trade off
  - As accurate as the available data
  - .....



# Introduction

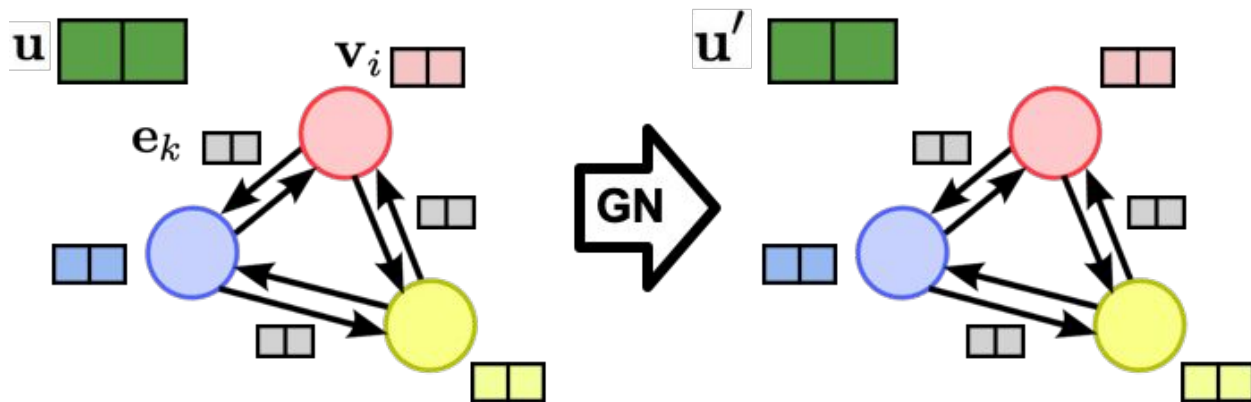
- Graph Network-based Simulators (GNS)
  - Rich physical states are represented by graphs of interacting particles
  - Complex dynamics are approximated by learned message-passing among nodes
  - Learn to accurately simulate a wide range of physical systems
    - Fluids
    - Rigid solids
    - Deformable materials
  - Generalization to larger systems and longer steps

# Related Work

- Particle-based simulation
  - States are represented as a set of particles, which encode mass, material, movement, etc. within local regions of space
  - Dynamics are computed on the basis of particles' interactions within their local neighborhoods
- Graph networks for learning forward dynamics
  - Interaction Networks
- Other baselines
  - DPI
  - CConv

# Preliminaries

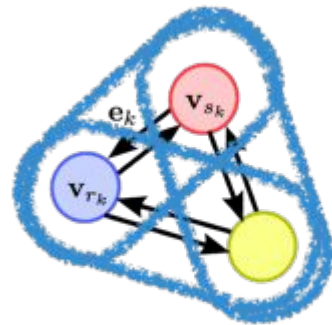
- Message passing network
  - Map an input graph to an output graph with the same structure but potentially different node, edge, and graph-level attributes



# Preliminaries

**Edge (message) function** (for every edge)

$$\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}) := \text{NN}_e \left( \begin{array}{c} \text{Edge features} \\ \text{Receiver node features} \\ \text{Sender node features} \\ \text{Global features} \end{array} \right)$$

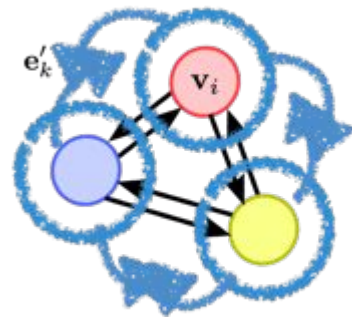


**Receiver edge aggregation** (Message pooling) (for every node)

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

**Node function** (for every node)

$$\mathbf{v}'_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}) := \text{NN}_v \left( \begin{array}{c} \text{Aggregated edge features} \\ \text{Node features} \\ \text{Global features} \end{array} \right)$$



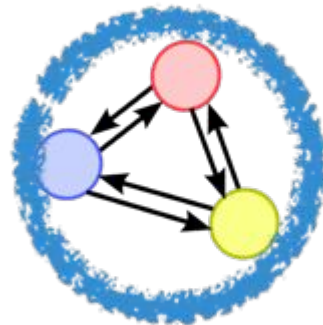
# Preliminaries

**Global node and edge aggregation**

$$\bar{\mathbf{v}}' \leftarrow \sum_i \mathbf{v}'_i \quad \bar{\mathbf{e}}' \leftarrow \sum_k \mathbf{e}'_k$$

**Global function**

$$\mathbf{u}' \leftarrow \phi^u (\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}) := \text{NN}_u ( \text{Aggregated edge features} \quad \text{Aggregated node features} \quad \text{Global features} )$$





# Preliminaries

$$\dot{\mathbf{p}}^t = f(\mathbf{p}^t) = \mathbf{v}^t, \quad \mathbf{p}^{t_0} = \mathbf{p}^0$$

- Euler method
  - Explicit Euler

$$\mathbf{p}^{t_{k+1}} = \mathbf{p}^{t_k} + \Delta t \cdot f(\mathbf{p}^{t_k}) = \mathbf{p}^{t_k} + \Delta t \cdot \dot{\mathbf{p}}^{t_k}$$

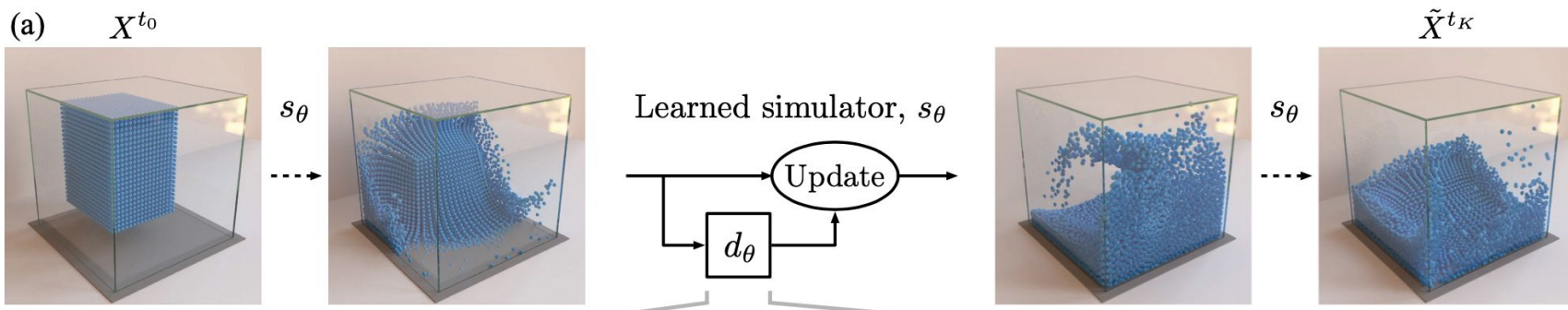
- Implicit Euler

$$\mathbf{p}^{t_{k+1}} = \mathbf{p}^{t_k} + \Delta t \cdot f(\mathbf{p}^{t_{k+1}}) = \mathbf{p}^{t_k} + \Delta t \cdot \dot{\mathbf{p}}^{t_{k+1}}$$

- Semi-implicit Euler

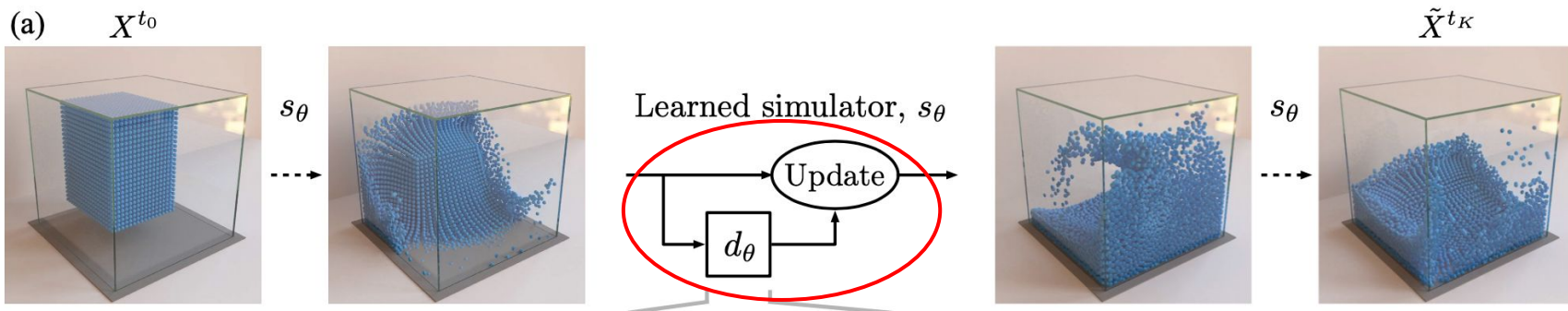
$$\begin{cases} \dot{\mathbf{p}}^t = f(\mathbf{p}^t) = \mathbf{v}^t, & \mathbf{p}^{t_0} = \mathbf{p}^0 \\ \dot{\mathbf{v}}^t = g(\mathbf{p}^t) = \ddot{\mathbf{p}}^t, & \mathbf{v}^{t_0} = \mathbf{v}^0 \end{cases} \quad \Rightarrow \quad \begin{aligned} \dot{\mathbf{p}}^{t_{k+1}} &= \dot{\mathbf{p}}^{t_k} + \Delta t \cdot \ddot{\mathbf{p}}^{t_k} \\ \mathbf{p}^{t_{k+1}} &= \mathbf{p}^{t_k} + \Delta t \cdot \dot{\mathbf{p}}^{t_{k+1}} \end{aligned}$$

# Formulation



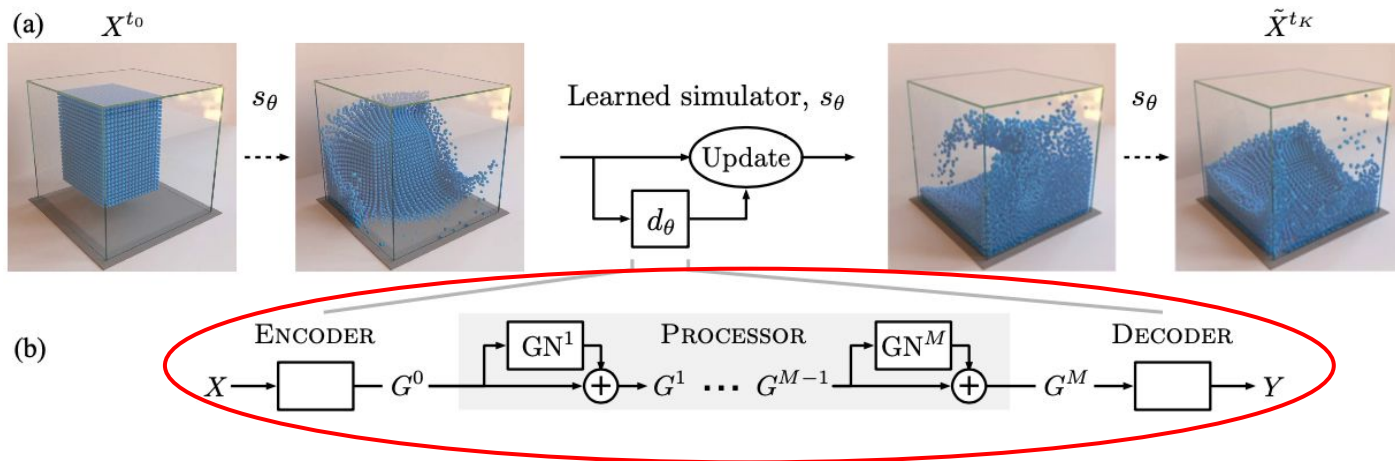
- Obtain a simulator  $s : \mathcal{X} \rightarrow \mathcal{X}$  to model the state dynamics ( $X^t \in \mathcal{X}$  is the state of the world at time  $t$ )
- Given an initial state  $X^{t_0}$  and physical dynamics over  $K$  timesteps, we need to simulate a trajectory of states  $\tilde{\mathbf{X}}^{t_0:K} = (X^{t_0}, \tilde{X}^{t_1}, \dots, \tilde{X}^{t_K})$
- Rollout: Computed iteratively by  $\tilde{X}^{t_{k+1}} = s(\tilde{X}^{t_k})$

# Pipeline



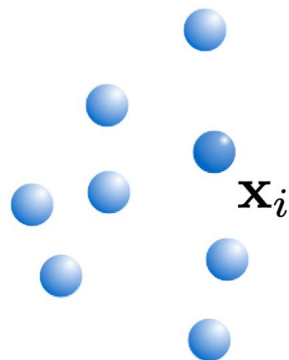
- First compute a function approximator  $d_\theta : \mathcal{X} \rightarrow \mathcal{Y}$  mapping state to dynamics (  $Y \in \mathcal{Y}$  represents the dynamics information, e.g., accelerations)
- Then the state is updated  $\tilde{X}^{t_{k+1}} = \text{Update}(\tilde{X}^{t_k}, d_\theta)$  following some update mechanism (Euler integrator in this paper)

# Structure of $d_\theta : \mathcal{X} \rightarrow \mathcal{Y}$



- Encoder:  $\mathcal{X} \rightarrow \mathcal{G}$ , embeds the raw state representations as a latent graph
- Processor:  $\mathcal{G} \rightarrow \mathcal{G}$ , message passing within the graph
- Decoder:  $\mathcal{G} \rightarrow \mathcal{Y}$ , extract the required dynamics information

# Input & Output Representation of $d_\theta : \mathcal{X} \rightarrow \mathcal{Y}$



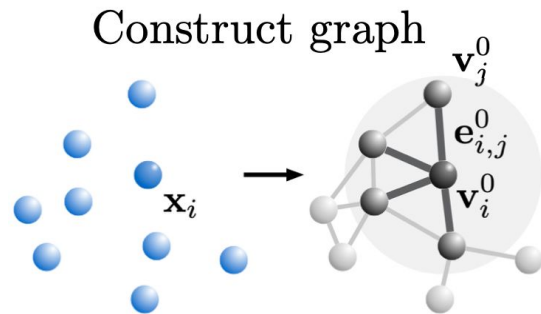
- **Input:**

- Particle-based representation  $X = (\mathbf{x}_0, \dots, \mathbf{x}_N)$
- Each of the  $N$  particles'  $\mathbf{x}_i$  denotes its state
- $\mathbf{x}_i^{t_k}$ : position +  $C$  previous velocities + static material properties, e.g. water, sand
- $\mathbf{x}_i^{t_k} = [\mathbf{p}_i^{t_k}, \dot{\mathbf{p}}_i^{t_k-C+1}, \dots, \dot{\mathbf{p}}_i^{t_k}, \mathbf{f}_i]$
- $\mathbf{r}_{i,j}$ , pairwise properties, e.g., spring constant,  $[(\mathbf{p}_i - \mathbf{p}_j), \|\mathbf{p}_i - \mathbf{p}_j\|]$
- Global properties of the system  $\mathbf{g}$ , e.g., external forces, global material properties

- **Outputs:**

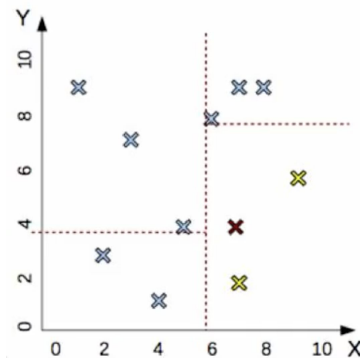
- Prediction targets: per-particle acceleration  $\ddot{\mathbf{p}}_i$
- $\dot{\mathbf{p}}_i$  and  $\ddot{\mathbf{p}}_i$  are computed by finite differences

# Encoder $\mathcal{X} \rightarrow \mathcal{G}$

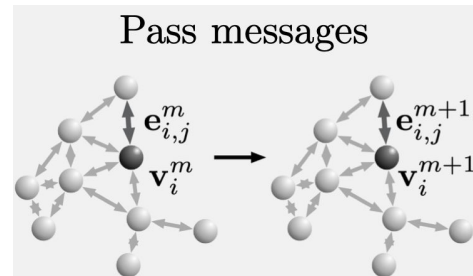


- Encodes the particle-based state representation as a latent graph  $G^0 = \text{ENCODER}(X)$
- $G^0$  is constructed by assigning a **node** to each particle and adding **edges** between particles within certain distance  $R$  (using k-d tree algorithm)
  - Pick random dimension, find median, split data, repeat

- $G = (V, E, \mathbf{u})$ ,  $\mathbf{v}_i \in V$ ,  $\mathbf{e}_{i,j} \in E$ ,  $\mathbf{u}$  represents the global properties
  - The node and vector embeddings are learned function
  - $\mathbf{v}_i = \varepsilon^v(\mathbf{x}_i)$ ,  $\mathbf{x}_i$  is the node representations
  - $\mathbf{e}_{i,j} = \varepsilon^e(\mathbf{r}_{i,j})$ ,  $\mathbf{r}_{i,j}$  is the pairwise properties



# Processor $\mathcal{G} \rightarrow \mathcal{G}$



- Compute the final graph  $G^M = \text{PROCESSOR}(G^0)$  via  $M$  steps of message-passing, where  $G^{m+1} = \text{GN}^{m+1}(G^m)$
- Simply uses existed Graph Networks (GN)

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{rk}, \mathbf{v}_{sk})$$

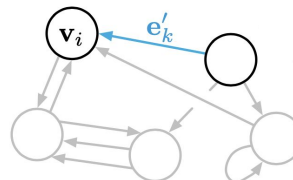
(1)

$$\bar{\mathbf{e}}'_i = \rho^{e \rightarrow v}(E'_i), E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i, k=1:N^e}$$

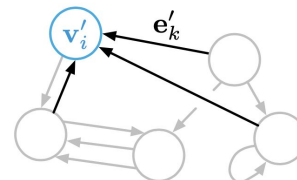
(2)

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

(3)



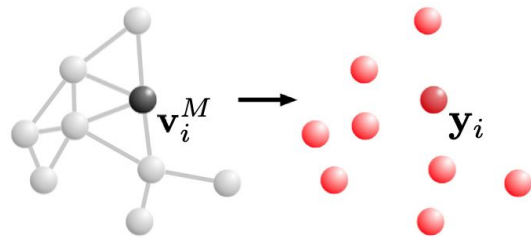
(a) Edge update



(b) Node update

## Decoder $\mathcal{G} \rightarrow \mathcal{Y}$

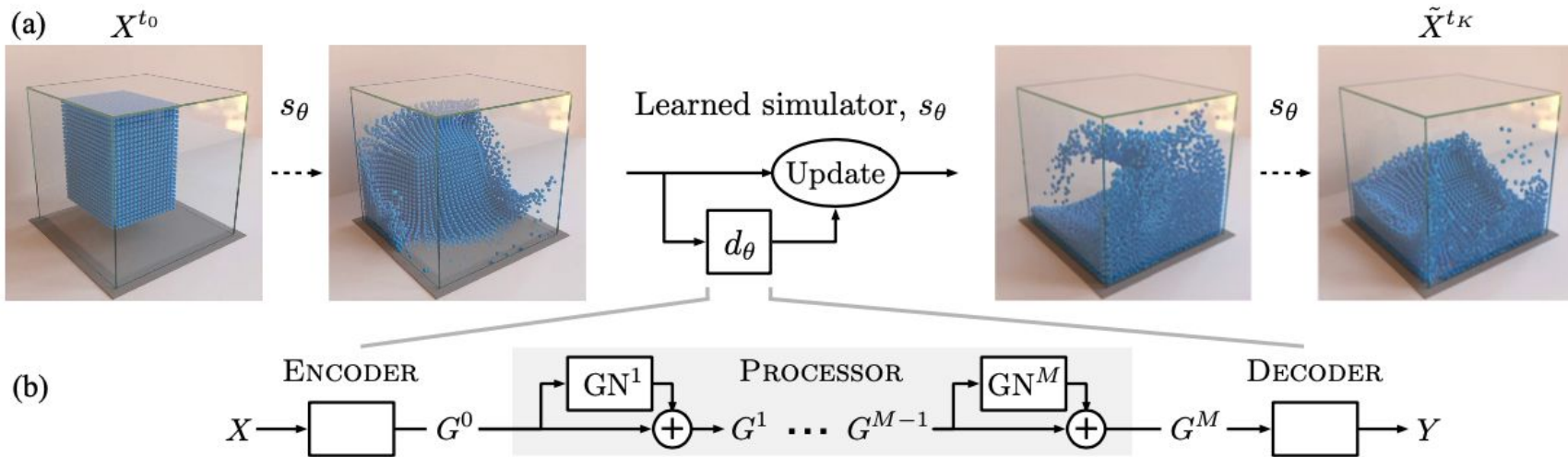
Extract dynamics info



- Extracts dynamics information from the nodes of the final latent graph  $\mathbf{y}_i = \delta^v(\mathbf{v}_i^M)$
- $\mathbf{y}_i$  is the per-node acceleration  $\ddot{\mathbf{p}}_i$
- Then the future position and velocity are updated using an Euler integrator
- $\delta^v$  is also implemented as a MLP



# As a whole



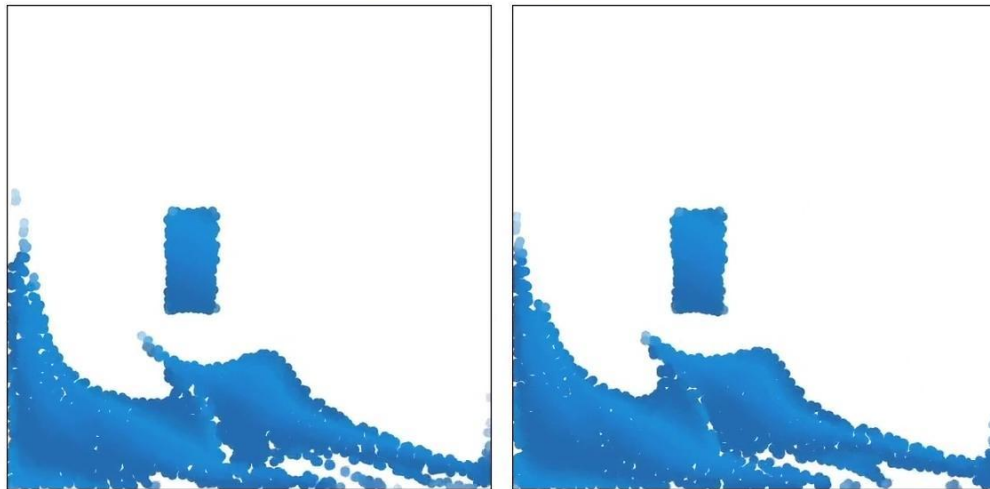
# Training

- Noise injection
  - Long rollouts can accumulate errors
  - Corrupt the input with Gaussian noise  $\sim \mathcal{N}(0, \sigma_v = 0.0003)$
  - Autoencoder
- Normalize all input and target vectors to 0 mean and 1 variance
- Randomly sample particle pairs  $(\mathbf{x}_i^{t_k}, \mathbf{x}_i^{t_{k+1}})$  from training trajectories
- Calculate  $\ddot{\mathbf{p}}^{t_k} = \mathbf{p}^{t_{k+1}} - 2\mathbf{p}^{t_k} + \mathbf{p}^{t_{k-1}}$  (omitting constant  $\Delta t$  for simplicity)
- Loss function:  $L(\mathbf{x}_i^{t_k}, \mathbf{x}_i^{t_{k+1}}; \theta) = \|d_\theta(\mathbf{x}_i^{t_k}) - \ddot{\mathbf{p}}_i^{t_k}\|^2$

# Physical Domains for Experiments

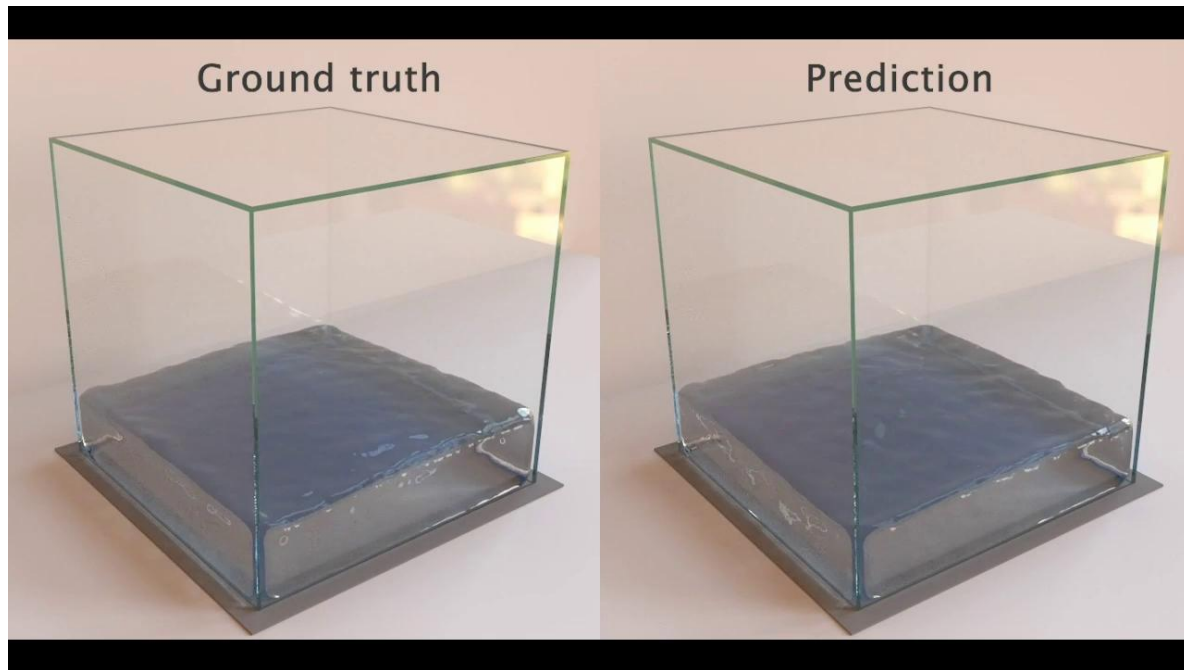
- WATER
  - 2k particles
  - 1000 steps
- WATER-3D
  - 14k particles
  - 800 steps
- BOXBATH
  - 1k particles
  - 150 steps

slow motion



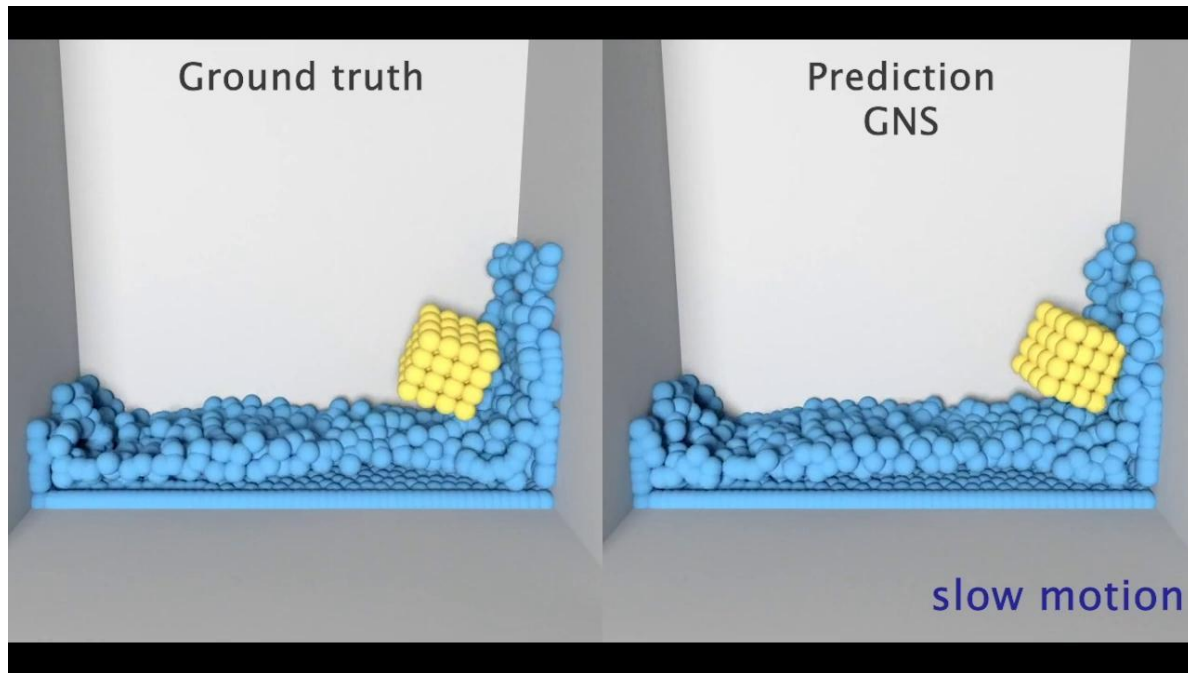
# Physical Domains for Experiments

- WATER
  - 2k particles
  - 1000 steps
- WATER-3D
  - 14k particles
  - 800 steps
- BOXBATH
  - 1k particles
  - 150 steps



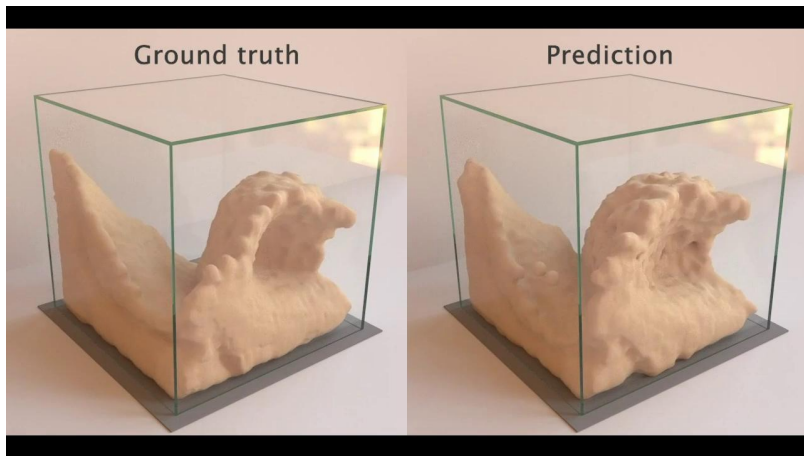
# Physical Domains for Experiments

- WATER
  - 2k particles
  - 1000 steps
- WATER-3D
  - 14k particles
  - 800 steps
- BOXBATH
  - 1k particles
  - 150 steps



# Experiment evaluation

- Edge length of the container = 1.0
- Metric: particle-wise MSE
  - 1-step
  - Rollout



Experimental domain	$N$	$K$	1-step ( $\times 10^{-9}$ )	Rollout ( $\times 10^{-3}$ )
<a href="#">WATER-3D (SPH)</a>	13k	800	8.66	10.1
<a href="#">SAND-3D</a>	20k	350	1.42	0.554
<a href="#">GOOP-3D</a>	14k	300	1.32	0.618
<a href="#">WATER-3D-S (SPH)</a>	5.8k	800	9.66	9.52
<a href="#">BOXBATH (PBD)</a>	1k	150	54.5	4.2
<a href="#">WATER</a>	1.9k	1000	2.82	17.4
<a href="#">SAND</a>	2k	320	6.23	2.37
<a href="#">GOOP</a>	1.9k	400	2.91	1.89
<a href="#">MULTIMATERIAL</a>	2k	1000	1.81	16.9
<a href="#">FLUIDSHAKE</a>	1.3k	2000	2.1	20.1
<a href="#">WATERDROP</a>	1k	1000	1.52	7.01
<a href="#">WATERDROP-XL</a>	7.1k	1000	1.23	14.9
<a href="#">WATERRAMPS</a>	2.3k	600	4.91	11.6
<a href="#">SANDRAMPS</a>	3.3k	400	2.77	2.07
<a href="#">RANDOMFLOOR</a>	3.4k	600	2.77	6.72
<a href="#">CONTINUOUS</a>	4.3k	400	2.06	1.06

*Table 1.* List of maximum number of particles  $N$ , sequence length  $K$ , and quantitative model accuracy (MSE) on the held-out test set. All domain names are also [hyperlinks to the video website](#).

# Simulating Complex Materials

- FLUIDSHAKE

- The container is being moved side-to-side
  - Causing splashes and irregular waves

Ground truth



Prediction



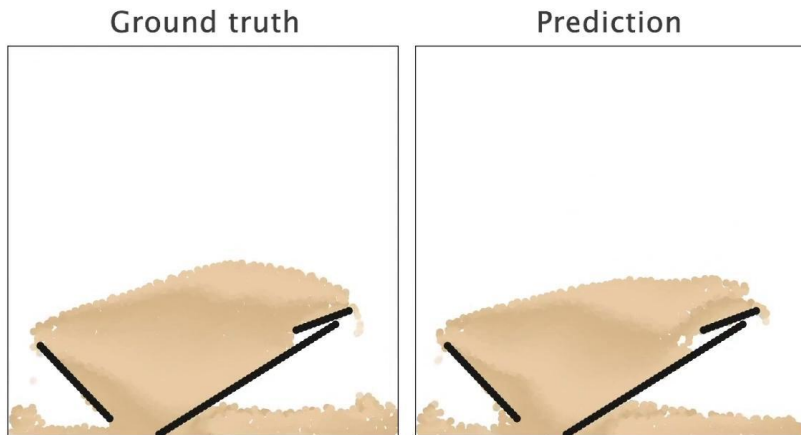
Experimental domain	$N$	$K$	1-step ( $\times 10^{-9}$ )	Rollout ( $\times 10^{-3}$ )
<a href="#">WATER-3D (SPH)</a>	13k	800	8.66	10.1
<a href="#">SAND-3D</a>	20k	350	1.42	0.554
<a href="#">GOOP-3D</a>	14k	300	1.32	0.618
<a href="#">WATER-3D-S (SPH)</a>	5.8k	800	9.66	9.52
<a href="#">BOXBATH (PBD)</a>	1k	150	54.5	4.2
<a href="#">WATER</a>	1.9k	1000	2.82	17.4
<a href="#">SAND</a>	2k	320	6.23	2.37
<a href="#">GOOP</a>	1.9k	400	2.91	1.89
<a href="#">MULTIMATERIAL</a>	2k	1000	1.81	16.9
<a href="#">FLUIDSHAKE</a>	1.3k	2000	2.1	20.1
<a href="#">WATERDROP</a>	1k	1000	1.52	7.01
<a href="#">WATERDROP-XL</a>	7.1k	1000	1.23	14.9
<a href="#">WATERRAMPS</a>	2.3k	600	4.91	11.6
<a href="#">SANDRAMPS</a>	3.3k	400	2.77	2.07
<a href="#">RANDOMFLOOR</a>	3.4k	600	2.77	6.72
<a href="#">CONTINUOUS</a>	4.3k	400	2.06	1.06

*Table 1.* List of maximum number of particles  $N$ , sequence length  $K$ , and quantitative model accuracy (MSE) on the held-out test set. All domain names are also [hyperlinks to the video website](#).



# Simulating Complex Materials

- Environment with complicated static obstacles (WATERRAMPS and SANDRAMPS)



Experimental domain	$N$	$K$	1-step ( $\times 10^{-9}$ )	Rollout ( $\times 10^{-3}$ )
<a href="#">WATER-3D (SPH)</a>	13k	800	8.66	10.1
<a href="#">SAND-3D</a>	20k	350	1.42	0.554
<a href="#">GOOP-3D</a>	14k	300	1.32	0.618
<a href="#">WATER-3D-S (SPH)</a>	5.8k	800	9.66	9.52
<a href="#">BOXBATH (PBD)</a>	1k	150	54.5	4.2
<a href="#">WATER</a>	1.9k	1000	2.82	17.4
<a href="#">SAND</a>	2k	320	6.23	2.37
<a href="#">GOOP</a>	1.9k	400	2.91	1.89
<a href="#">MULTIMATERIAL</a>	2k	1000	1.81	16.9
<a href="#">FLUIDSHAKE</a>	1.3k	2000	2.1	20.1
<a href="#">WATERDROP</a>	1k	1000	1.52	7.01
<a href="#">WATERDROP-XL</a>	7.1k	1000	1.23	14.9
<a href="#">WATERRAMPS</a>	2.3k	600	4.91	11.6
<a href="#">SANDRAMPS</a>	3.3k	400	2.77	2.07
<a href="#">RANDOMFLOOR</a>	3.4k	600	2.77	6.72
<a href="#">CONTINUOUS</a>	4.3k	400	2.06	1.06

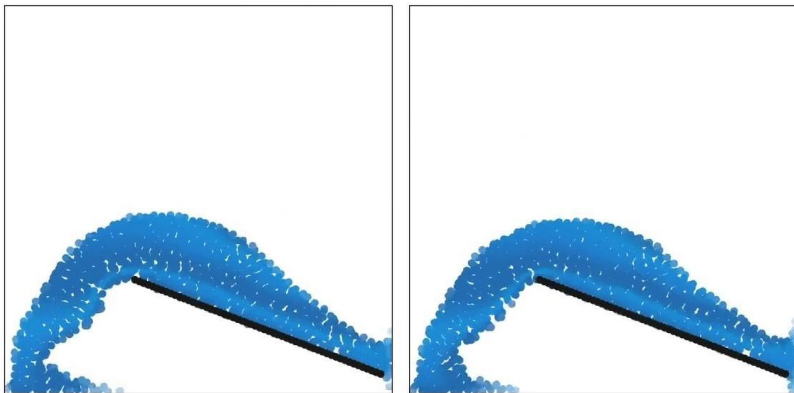
Table 1. List of maximum number of particles  $N$ , sequence length  $K$ , and quantitative model accuracy (MSE) on the held-out test set. All domain names are also [hyperlinks to the video website](#).



# Simulating Complex Materials

- Environment with complicated static obstacles (WATERRAMPS and SANDRAMPS)

slow motion

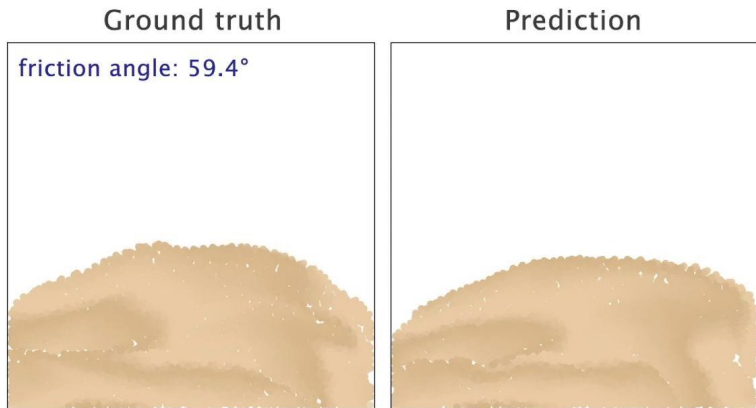


Experimental domain	$N$	$K$	1-step ( $\times 10^{-9}$ )	Rollout ( $\times 10^{-3}$ )
<a href="#">WATER-3D (SPH)</a>	13k	800	8.66	10.1
<a href="#">SAND-3D</a>	20k	350	1.42	0.554
<a href="#">GOOP-3D</a>	14k	300	1.32	0.618
<a href="#">WATER-3D-S (SPH)</a>	5.8k	800	9.66	9.52
<a href="#">BOXBATH (PBD)</a>	1k	150	54.5	4.2
<a href="#">WATER</a>	1.9k	1000	2.82	17.4
<a href="#">SAND</a>	2k	320	6.23	2.37
<a href="#">GOOP</a>	1.9k	400	2.91	1.89
<a href="#">MULTIMATERIAL</a>	2k	1000	1.81	16.9
<a href="#">FLUIDSHAKE</a>	1.3k	2000	2.1	20.1
<a href="#">WATERDROP</a>	1k	1000	1.52	7.01
<a href="#">WATERDROP-XL</a>	7.1k	1000	1.23	14.9
<a href="#">WATERRAMPS</a>	2.3k	600	4.91	11.6
<a href="#">SANDRAMPS</a>	3.3k	400	2.77	2.07
<a href="#">RANDOMFLOOR</a>	3.4k	600	2.77	6.72
<a href="#">CONTINUOUS</a>	4.3k	400	2.06	1.06

Table 1. List of maximum number of particles  $N$ , sequence length  $K$ , and quantitative model accuracy (MSE) on the held-out test set. All domain names are also [hyperlinks to the video website](#).

# Simulating Complex Materials

- Different friction angle (Continuous):
  - liquid ( $0^\circ$ ), sand ( $45^\circ$ ), or gravel ( $> 60^\circ$ )
  - Train on  $[0, 30]$ ,  $[55, 80]$
  - Test on  $[0, 90]$
  - Can still be accurate on unseen angle

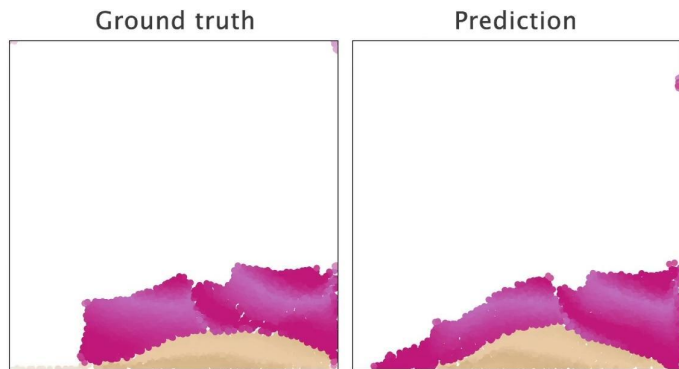


Experimental domain	$N$	$K$	1-step ( $\times 10^{-9}$ )	Rollout ( $\times 10^{-3}$ )
<a href="#">WATER-3D (SPH)</a>	13k	800	8.66	10.1
<a href="#">SAND-3D</a>	20k	350	1.42	0.554
<a href="#">GOOP-3D</a>	14k	300	1.32	0.618
<a href="#">WATER-3D-S (SPH)</a>	5.8k	800	9.66	9.52
<a href="#">BOXBATH (PBD)</a>	1k	150	54.5	4.2
<a href="#">WATER</a>	1.9k	1000	2.82	17.4
<a href="#">SAND</a>	2k	320	6.23	2.37
<a href="#">GOOP</a>	1.9k	400	2.91	1.89
<a href="#">MULTIMATERIAL</a>	2k	1000	1.81	16.9
<a href="#">FLUIDSHAKE</a>	1.3k	2000	2.1	20.1
<a href="#">WATERDROP</a>	1k	1000	1.52	7.01
<a href="#">WATERDROP-XL</a>	7.1k	1000	1.23	14.9
<a href="#">WATERRAMPS</a>	2.3k	600	4.91	11.6
<a href="#">SANDRAMPS</a>	3.3k	400	2.77	2.07
<a href="#">RANDOMFLOOR</a>	3.4k	600	2.77	6.72
<a href="#">CONTINUOUS</a>	4.3k	400	2.06	1.06

*Table 1.* List of maximum number of particles  $N$ , sequence length  $K$ , and quantitative model accuracy (MSE) on the held-out test set. All domain names are also [hyperlinks to the video website](#).

# Multiple Interacting Materials

- Visually, their model's performance in MULTIMATERIAL is comparable to its performance when trained on those materials individually



Experimental domain	$N$	$K$	1-step ( $\times 10^{-9}$ )	Rollout ( $\times 10^{-3}$ )
<a href="#">WATER-3D (SPH)</a>	13k	800	8.66	10.1
<a href="#">SAND-3D</a>	20k	350	1.42	0.554
<a href="#">GOOP-3D</a>	14k	300	1.32	0.618
<a href="#">WATER-3D-S (SPH)</a>	5.8k	800	9.66	9.52
<a href="#">BOXBATH (PBD)</a>	1k	150	54.5	4.2
<a href="#">WATER</a>	1.9k	1000	2.82	17.4
<a href="#">SAND</a>	2k	320	6.23	2.37
<a href="#">GOOP</a>	1.9k	400	2.91	1.89
<a href="#">MULTIMATERIAL</a>	2k	1000	1.81	16.9
<a href="#">FLUIDSHAKE</a>	1.3k	2000	2.1	20.1
<a href="#">WATERDROP</a>	1k	1000	1.52	7.01
<a href="#">WATERDROP-XL</a>	7.1k	1000	1.23	14.9
<a href="#">WATERRAMPS</a>	2.3k	600	4.91	11.6
<a href="#">SANDRAMPS</a>	3.3k	400	2.77	2.07
<a href="#">RANDOMFLOOR</a>	3.4k	600	2.77	6.72
<a href="#">CONTINUOUS</a>	4.3k	400	2.06	1.06

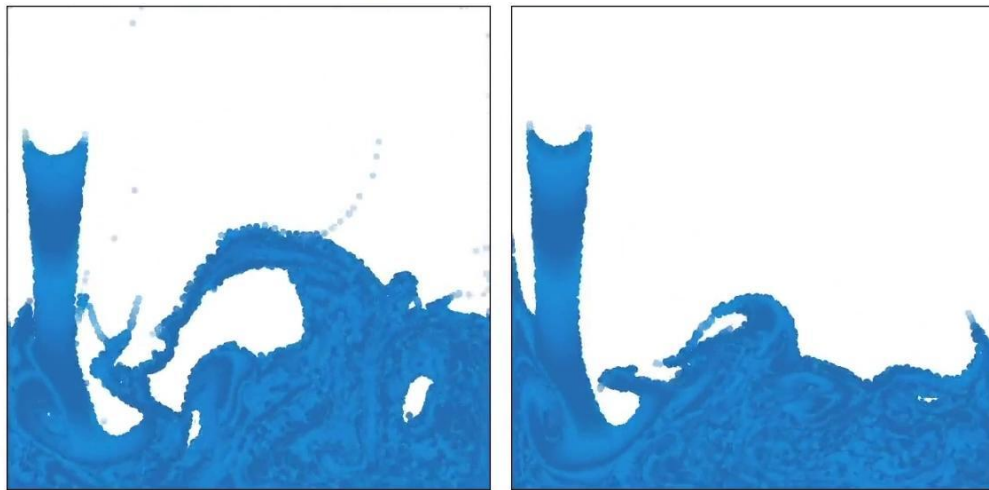
Table 1. List of maximum number of particles  $N$ , sequence length  $K$ , and quantitative model accuracy (MSE) on the held-out test set. All domain names are also [hyperlinks to the video website](#).

# Generalization

Training domain



slow motion

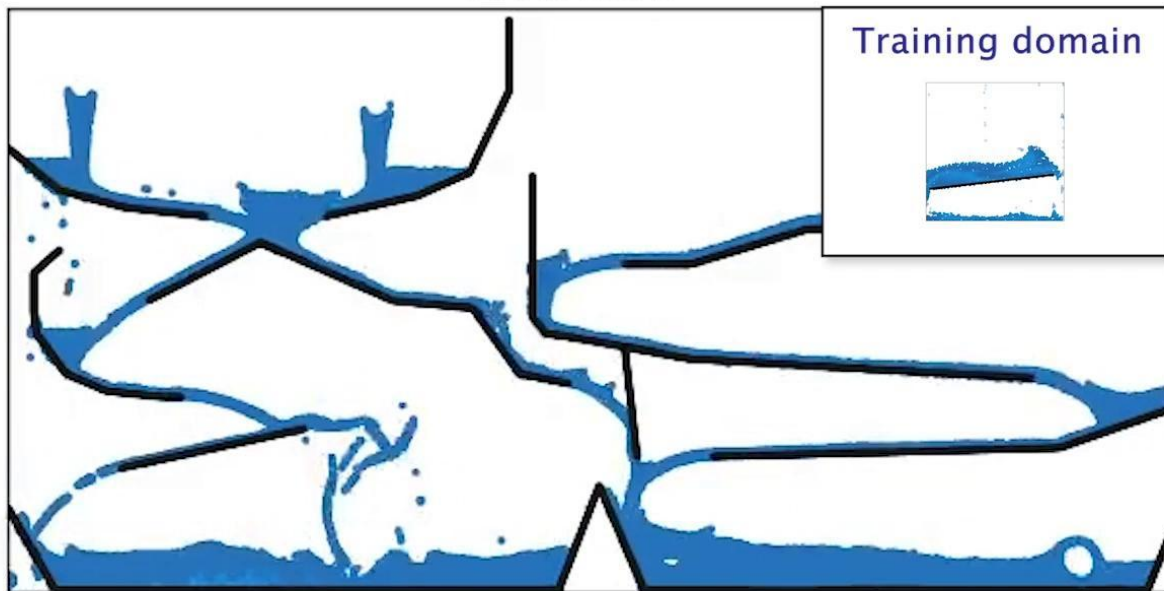


# Generalization

Training domain



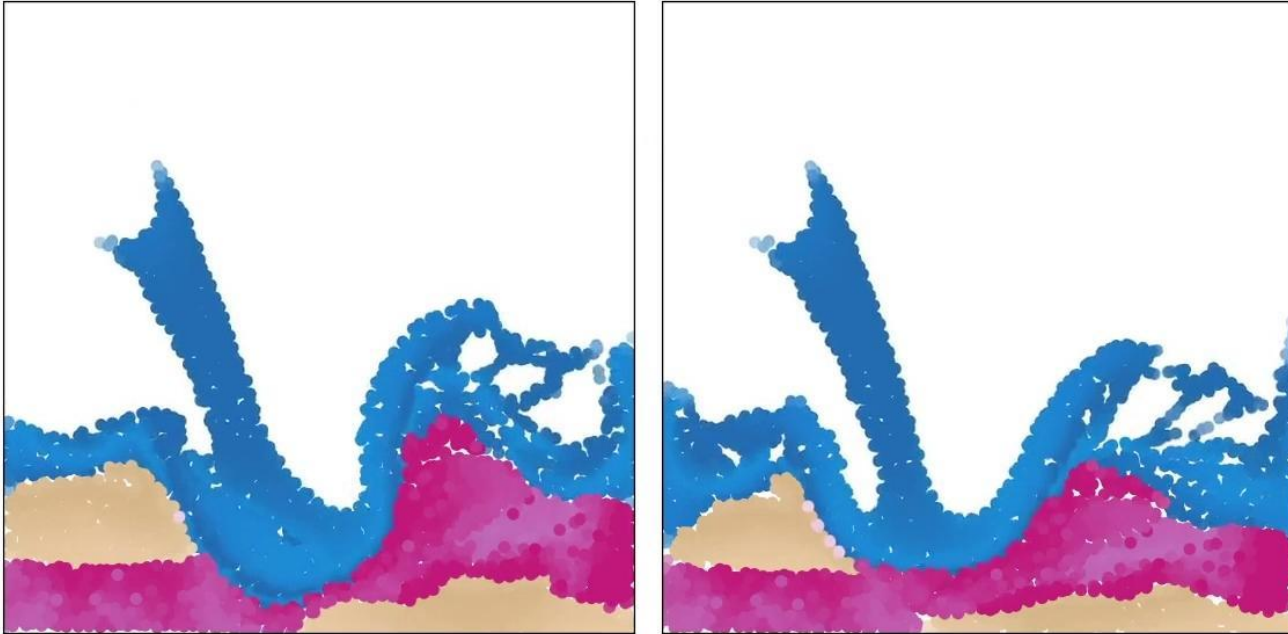
Prediction



# Generalization

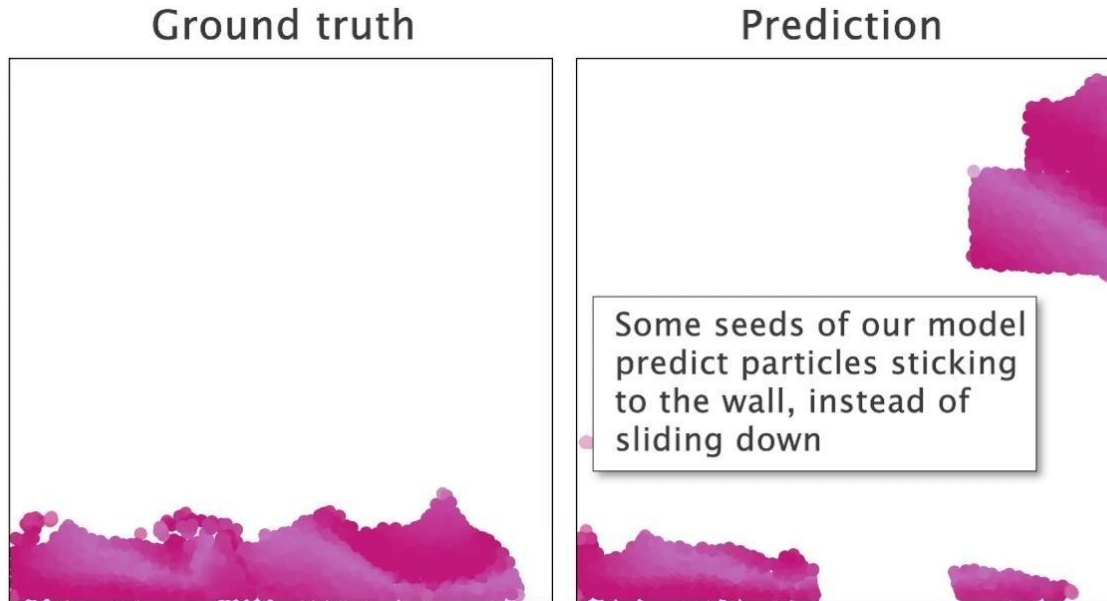
---

slow motion

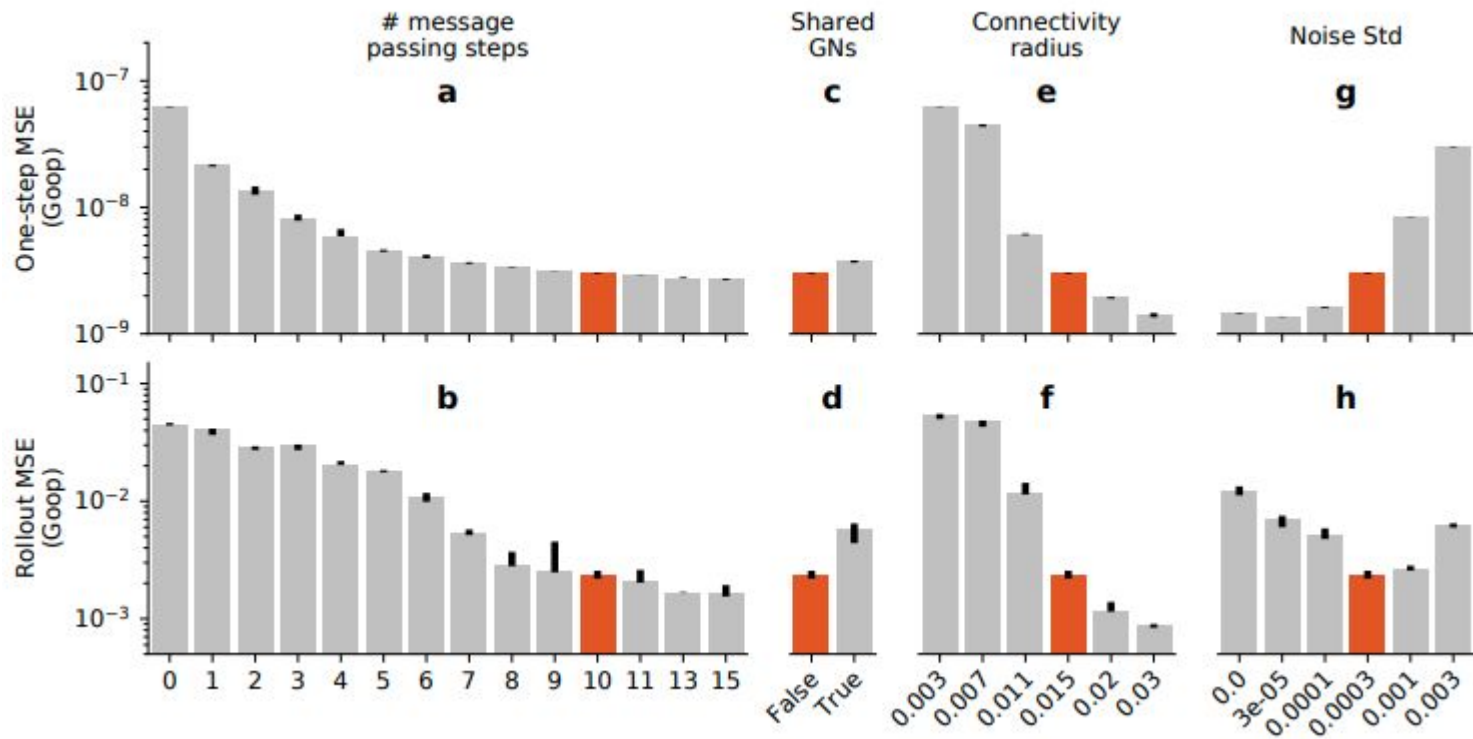




# Fail example



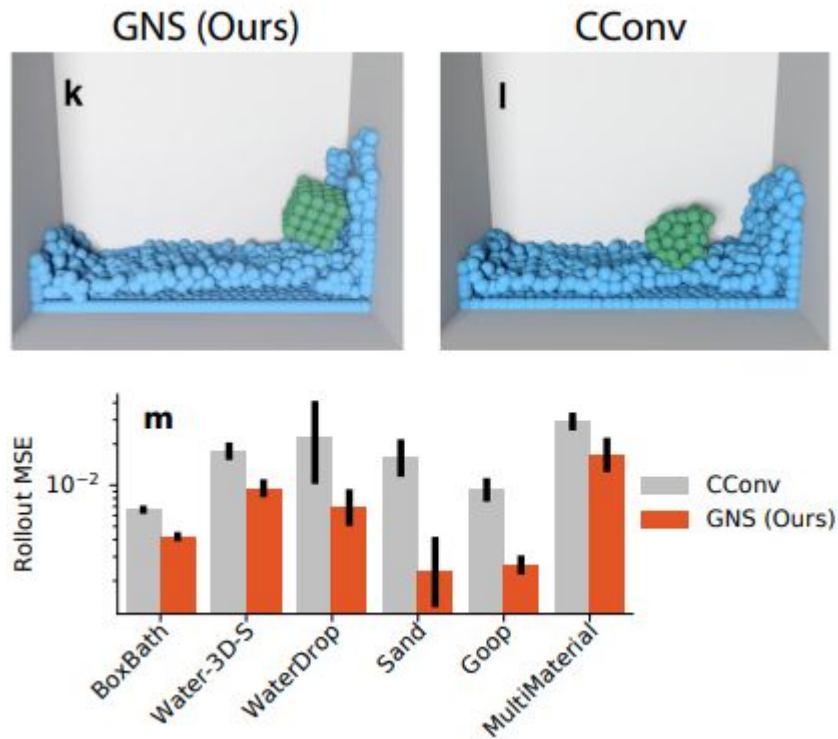
# Analysis





# Comparisons to Previous Models

- They implement CConv with noise and multiple input states



# Conclusion

- A powerful machine learning framework based on particle-based representations of physics and learned message-passing on graphs
- A simple architecture, but can learn to simulate dynamics of complex physics with tens of thousands of particles over thousands time steps
- More accurate, and has better generalization than previous approaches