# Applications of Graph Neural Networks in Simulation vs. Animation

Liam Shatzel

University of Victoria, Computer Science

2025-03-10

## Introduction

- Simulation is the modelling of dynamic systems using physics (House & Keyser, 2016). It is often used as a technique to exactly replicate physical systems.
- Animation, on the other hand, is a sequence of images which are played sequentially to depict motion (Wells, 2006). Applications of animations include animated films, games, and even GIFs.
- Simulations and animations have different users with different use-cases
  - A game developer might want to be able to quickly iterate over different animations or animation styles.
  - A physicist might want a simulator that models a single phenomena which is as accurate as possible.
- Graph neural networks (GNNs) are a type of neural network that allow data which is better suited for a graph-based representation to be learned (Sanchez-Gonzalez et al., 2020). They provide a framework in which both a simulation or animation can be learned.

## Background

- Graphs are well suited for data which is not easily structured in a matrix or linear fashion, therefore GNNs are effective for particle-based systems.
- After the data is transformed into its graph-based representations, features of the GNN need to be learned. There are separate neural networks for the nodes, edges, and the overall graph itself.
- To learn the relationships between the particles, their features are aggregated, this is called a message passing step. The message passing step also has its own network which learns the relationships between nodes and edges, along with sibling nodes (Fig. 1).

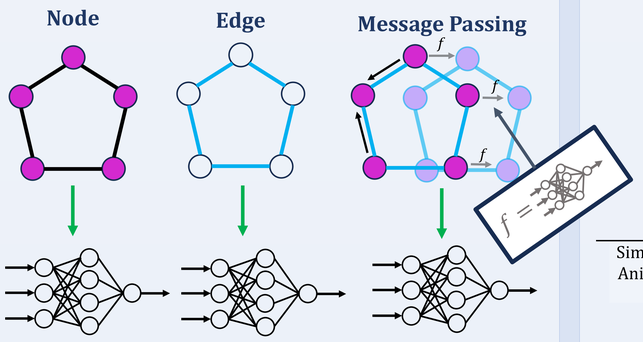


Figure 1: Elements of a Graph Neural Network

## Materials and Methods

- Training was done on Intel E5-2650 v4 Broadwell @ 2.2GHz CPUs and NVIDIA P100 Pascal GPUs hosted on Digital Research Alliance of Canada's Cedar cluster.
- Training data was generated using the material point method (MPM), which numerically simulates continuum materials such as liquids, fluids, and gasses (Jiang et al., 2016).
- Experiments were run using three cases of MPM: Sand, water, and high viscosity goop.
- Rollouts were created by testing hyperparameter combinations to minimize the mean squared error of the training dataset.
  - This was done using randomized and Bayesian hyperparameter optimization with Weights and Biases.

### Water Dataset

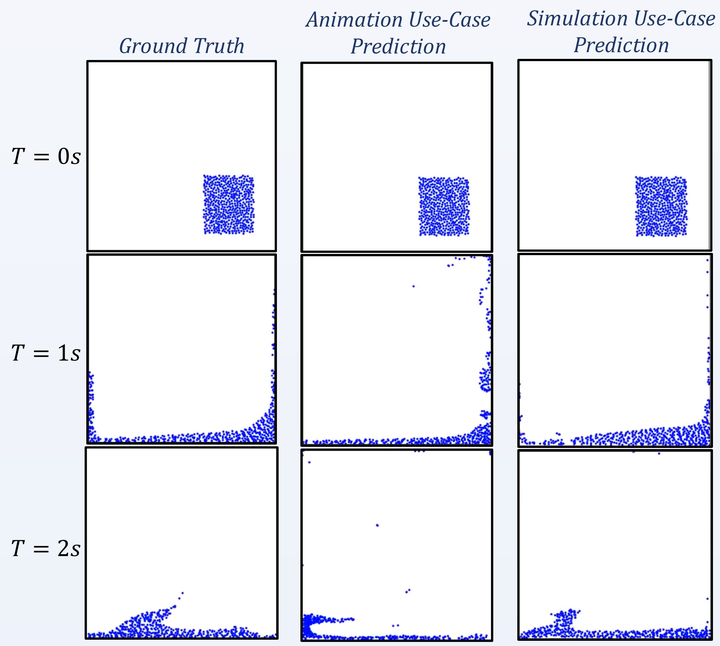


Figure 2: Qualitative Comparison of Rollouts from the Water Dataset

### Training Statistics

| *Use-Case* | Train Time (Mins) | Num. Epochs | Train Error |
|---|---|---|---|
| Simulation Use-Case | 294 | 287573 | 0.0340 |
| Animation Use-Case | 100 | 96191 | 0.0558 |

| *Use-Case* | Eval. Error | Rollout Time (s) | Rollout FPS | MPM Time (s) |
|---|---|---|---|---|
| Simulation Use-Case | 0.0508 | 23.370 | 43 | 5.780 |
| Animation Use-Case | 0.0785 | 17.850 | 58 | 5.780 |

Table 1: Statistics of Training on Water Dataset for Simulation and Animation Use-Cases (Averaged Across 10 Runs)

### Hyperparameter Transferability

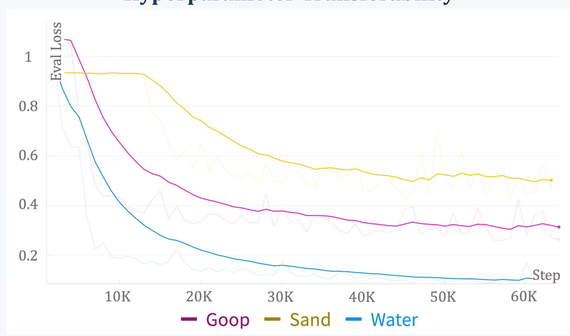


Figure 3: Convergence Rate of Training on Different Materials (Goop, Sand, Water)

### Sand and Goop Datasets

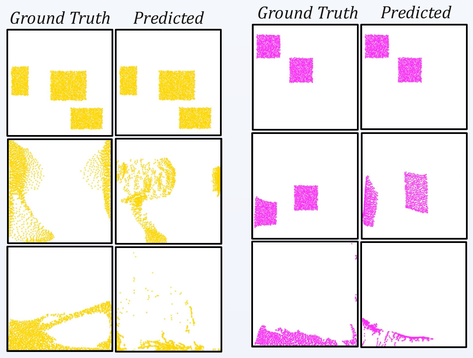


Figure 4: Sand and Goop Rollouts

### Model Eval. Loss vs Layer Size

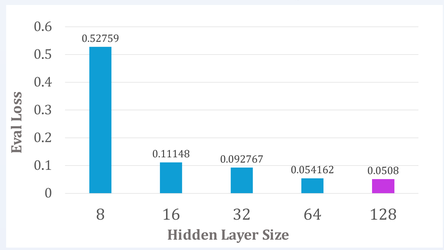


Figure 5: The Mean Squared Error of the Training Dataset Versus the Hidden Layer Size

### Rollout FPS vs Eval Loss

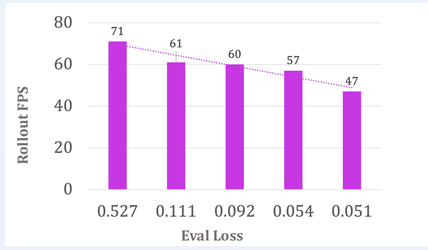


Figure 6: FPS of the Rollout versus the Mean Squared Error of the Validation Dataset

## Results

- Both the animation and simulation use-cases produced visually accurate fluid behaviour (Fig. 2).
- The animation use-case had an evaluation mean squared error (MSE) of 0.0785 whereas the simulation use-case had 0.0508. Approximately a 42% difference (Table 1).
- Animation use-case rollouts were done in ~17 seconds for 1000 frames (60 FPS), whereas the MPM took ~6 seconds or 173 FPS (Table 1).
- The water material began to converge in about 10k steps. The goop and sand materials began to converge closer to 20k steps (Fig. 3).

## Conclusion & Discussion

- When compared visually, the animation and simulation use-cases appear similar; but the simulation use-case very closely replicates the expected fluid behaviour, whereas the animation use-case is less precise.
- The animation use-case still creates visually convincing fluid behaviour and can do so up to a reduced hidden layer size of 32.
- As expected, the training error of the animation use-case was higher than that of the simulation use-case, given the shorter training time and smaller hidden layer size.
- Changing simulated material type while maintaining a fixed set of hyperparameters has a different learning rate but still converges.
- The results show that users are able to tune the behaviour of their animation or simulator, and thus GNNs can be effectively used as a framework to learn both animation and simulation.
- Future work includes blending trained GNNs to allow for further customizability in material properties and applying GNNs to different domains.

## Acknowledgements

This research was supported by the Jamie Cassels Undergraduate Research Award, University of Victoria, and Supervised by Dr. Brandon Haworth. Special thanks to Steven Bobyn and Digital Research Alliance.

## References

House, D., & Keyser, J. C. (2016). *Foundations of Physically Based Modeling and Animation*. A K Peters/CRC Press.

Jiang, C., Schroeder, C., Teran, J., Stomakhin, A., & Selle, A. (2016). The material point method for simulating continuum materials. ACM SIGGRAPH 2016 Courses, 1–52.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. (2020). Learning to Simulate Complex Physics with Graph Networks. Proceedings of the 37th International Conference on Machine Learning, 8459–8468.

Sanchez-Lengeling, B., Reif, E., Pearce, A., & Wiltschko, A. B. (2021). A Gentle Introduction to Graph Neural Networks. Distill, 6(9), e33.

Wells, P. (2006). The Fundamentals of Animation. AVA Publishing.