A Simulation Process

Environment and Setup We conducted our simulations using the MPM (Material Point Method) solver on a GPU with CUDA 10, running in an Ubuntu 22 environment. Table 5 outlines the hardware and software specifications used in this work.

The MPM solver operates through several key steps to accurately simulate the physical interactions of particles and materials. Depending on the type of simulation, different configurations are used to optimize the accuracy and relevance of the generated data.

For **multi-material simulations**, the simulation is initialized with 2,500 particles confined within a grid of 64×64 cells. This setup allows for the detailed modeling of interactions between different materials such as fluids, elastic materials (like jelly), and snow. Each material is characterized by distinct parameters, including Young's modulus, Poisson's ratio, and a dynamically adjusted **hardening coefficient**, which influences material stiffness based on deformation.

In contrast, **fluid-only simulations** employ a higher-resolution grid of 128×128 cells, also initialized with 2,500 particles. This increased grid resolution enhances the simulation's ability to capture fine details of fluid dynamics, focusing solely on the behavior of fluid particles without the complexity of multi-material interactions.

Table 5. Environment Specification for data generation and training on Temporal Learning Experiment

Environment	**Specifications**
CPU	6-Core Intel i7-5930K 64 bits Haswell, clock: 3.7GHz, L2: 15.0 MiB
RAM	64 GB DDR4 2133MHz
GPU	NVIDIA RTX A5000
Physics Simulator	MPM on Taichi 0.6.3 CUDA enabled
Training Framework	PyTorch

In both types of simulations, the grid resolution and particle count are determined by a quality factor, resulting in a spatial resolution with grid cells of size $\Delta x = \frac{1}{64}$ for multi-material simulations and $\Delta x = \frac{1}{128}$ for fluid-only simulations.

The simulation advances in time steps of $\Delta t = 8 \times 10^{-4}$ seconds, calculated to maintain numerical stability according to the Courant–Friedrichs–Lewy (CFL) condition:

$$\Delta t \leq \frac{C \cdot \Delta x}{v_{\max}}, \tag{3}$$

where C is the CFL number (typically less than 1 for stability), Δx is the grid cell size, and $v_{\max}$ is the maximum velocity in the simulation. With $C = 0.9$ and assuming a reasonable maximum velocity $v_{\max} = 10$ units/second, our chosen $\Delta t = 8 \times 10^{-4}$ seconds per sub-step remains well within the stability limit.

Each frame of the simulation is computed using 20 sub-steps, with each sub-step involving the transfer of particle state (position, velocity, material properties) to the grid. The grid serves as the computational domain where forces such as gravity and interactions with boundaries are applied. Boundary conditions are enforced to confine the particles within the simulation domain. Additionally, an attractor force may be applied before data recording begins to generate diverse initial conditions without affecting the subsequent dynamics.

B Selection of Activation Function

The choice of activation function is critical for the performance and efficiency of training deep neural networks. We experimented with several popular non-linear activation functions, such as *Sigmoid*, *Tanh*, *ReLU*, and *ELU* [32]. To determine the best activation function, we trained a simple deep neural network (MLP) with three layers, each containing 200 units, and an output layer with 10,000 units to predict the next simulation frame based on input data from the previous frame.

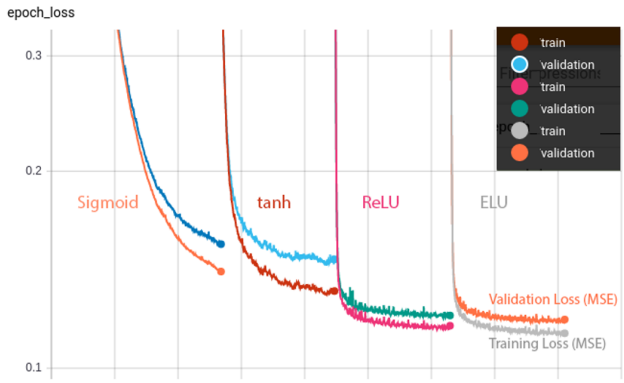

Fig. 5. Learning curve showing validation and training loss for different activation functions using the LSTM network. The small gap between the validation loss and training loss indicates that the network is not overfitted. The horizontal width of each plot corresponds to 500 epochs (unlabeled to simplify the comparison, as the four plots were merged into one image for easier side-by-side comparison).

Each frame consists of 2,500 particles, with each particle characterized by four properties: (px, py, vx, vy) for the fluid model and five properties (px, py, vx, vy, m) for the multi-material model. Thus, both the input and output layers have 10,000 units. We compared the Mean Squared Error (MSE) after 500 epochs for different activation functions, as shown in figure 5. The results indicated that the network with ELU activation achieved the lowest MSE (0.1131 for training and 0.1187 for validation).

The exponential linear unit (ELU) is essentially a smoothed ReLU function. ELU reached the lowest MSE among the activation functions within 500 epochs, indicating faster training. Figure 5 further illustrates the relationship between training time and the minimum loss achieved with different activation functions. The plot also shows a small gap between the validation loss and training loss, which reflects that the network is not overfitted. From this comparison, it is evident that ELU and ReLU reduce the loss much faster than Sigmoid and Tanh, with ELU achieving the lowest final training and validation loss. Additionally, ELU maintains stable training dynamics throughout, with minimal fluctuation in the loss curve. Given its superior performance in terms of rapid convergence, low final loss, and training stability, we decided to use ELU activation in most of the subsequent experiments.

C minRNN Architecture

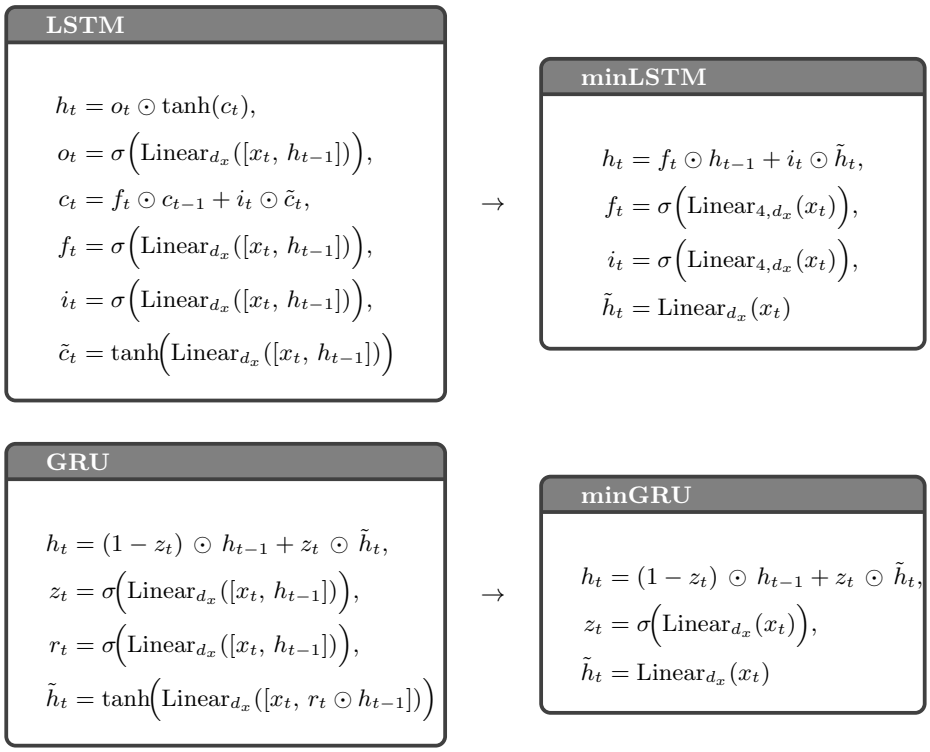

Fig. 6. Classical LSTM/GRU versus minimal LSTM/GRU proposed by [1]. Merging or removing gates reduces parameters and accelerates training without sacrificing long-term temporal modeling.

D Further Visualization of Result

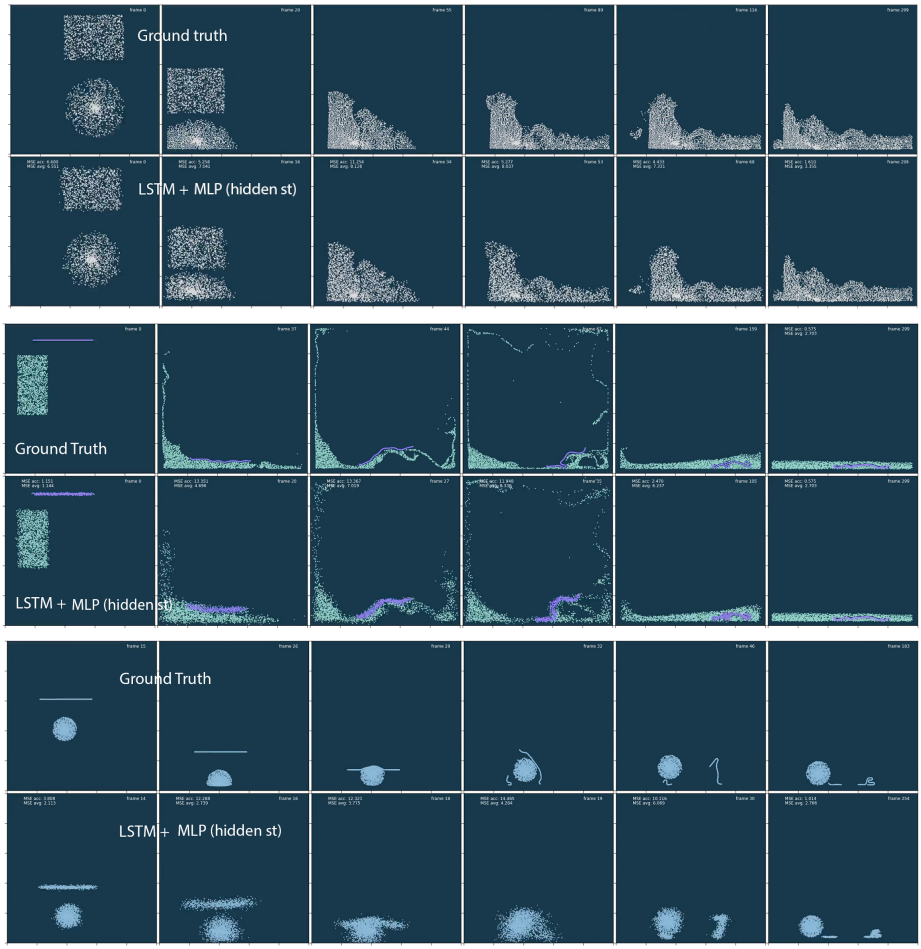

Fig. 7. Snapshots of Qualitative comparisons for Multi-material simulations: (top) Liquid-Snow, (second) Snow-Snow, (third) Liquid-Rope, and (bottom) Jelly-Rope. Different architectures exhibit varying degrees of stability and accuracy throughout each sequence, highlighting the challenges posed by multi-material dynamics.

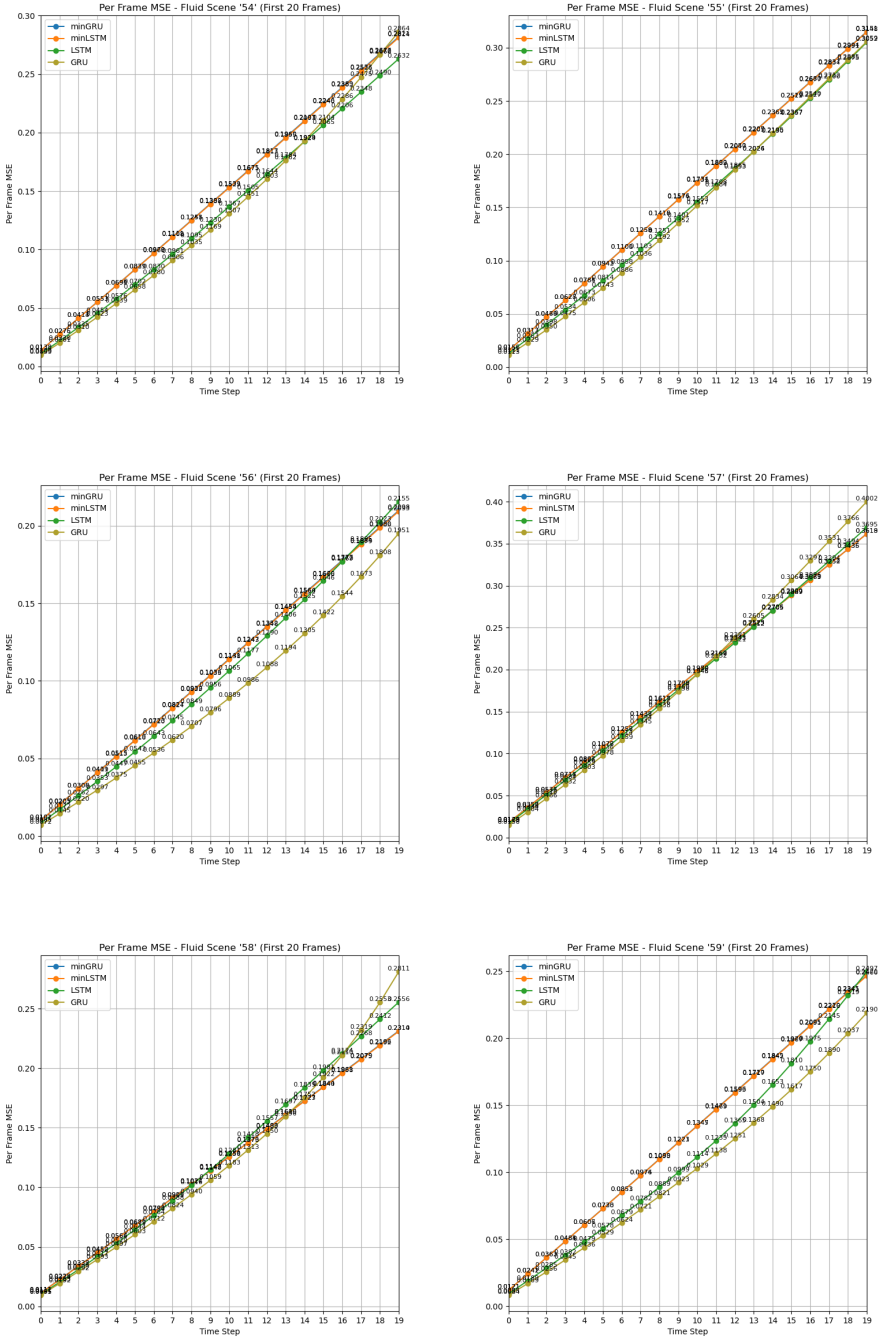

Fig. 8. Raw per-frame MSE curves for six fluid scenes (54–59). These curves underlie the numerical comparisons in Table 2. Each plot shows MSE values over 20 frames for the tested models.

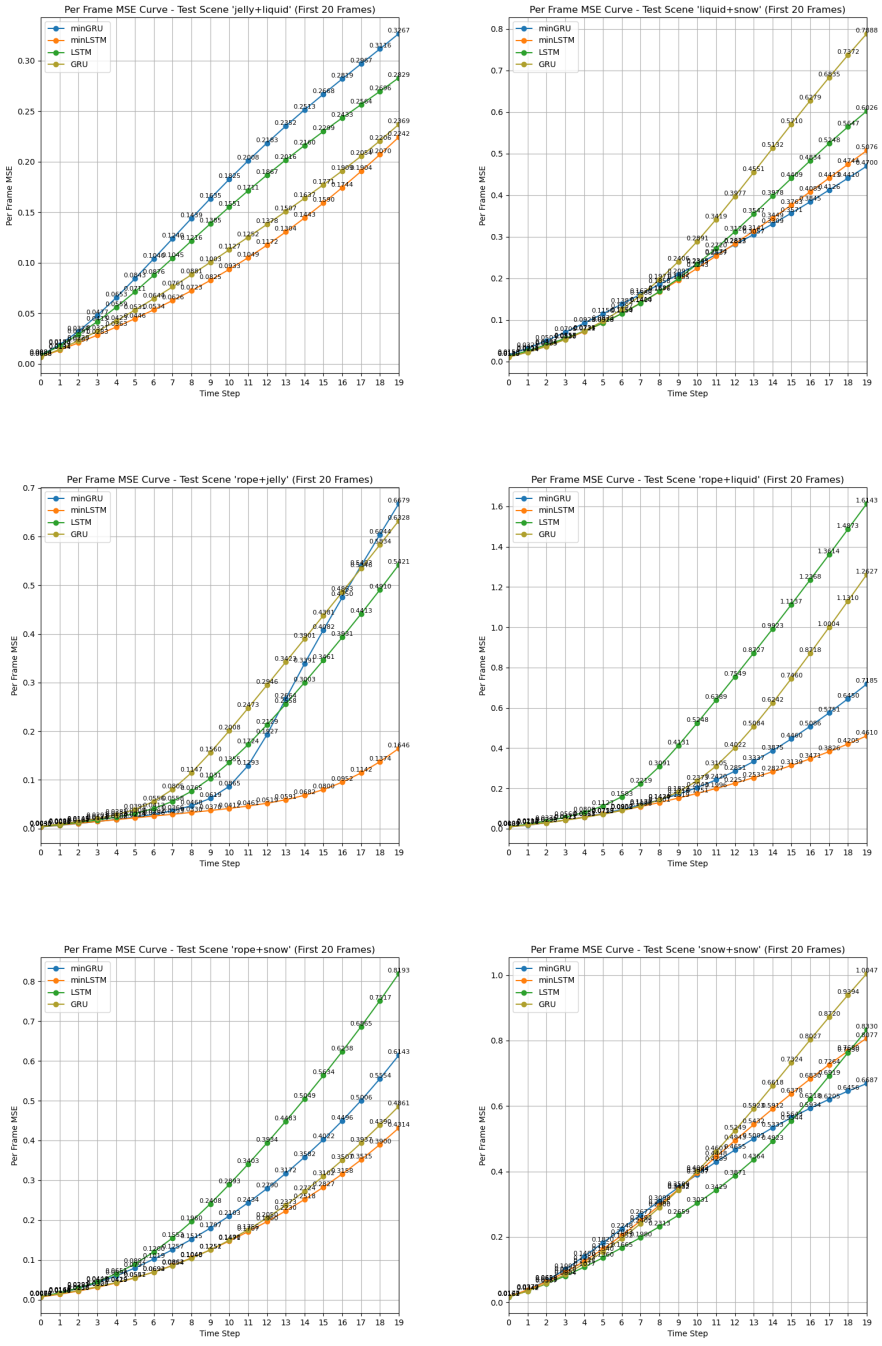

Fig. 9. Raw per-frame MSE curves for six multi material scenes. These curves underlie the numerical comparisons in Table 3. Each plot shows MSE values over 20 frames for the tested models.