MinRNNs for Lagrangian-Based Simulations of Transient Flow Problems

Dody Dharma[1,2][0000−0003−1022−9346], Peter K. Jimack[2][0000−0001−9463−7595], and He Wang[3][0000−0002−2281−5679]

[1] School of EE and Informatics, Institut Teknologi Bandung, Bandung, Indonesia
dody.dharma@itb.ac.id
[2] School of Computer Science,University of Leeds, Leeds, UK
p.k.jimack@leeds.ac.uk
[3] AI Centre, Computer Science, University College London, London, UK
he_wang@ucl.ac.uk

Abstract. Motivated by the need for faster yet accurate surrogate modeling of continuum simulations, we investigate whether the recently proposed minimal recurrent networks (minLSTM and minGRU [1][4]) can benefit particle-based fluid and soft-solid simulations. To our knowledge, this is the first work applying these minimal RNNs to Lagrangian data from 2D continuum simulation, including single-phase fluids and multi-material interactions. We embed minLSTM and minGRU in an MLP-based encoder–decoder and compare them against (i) a classical LSTM, and (ii) an MLP baseline with no recurrent core. Where prior studies of minRNNs focused on simpler time-series tasks, our results show that minLSTM and minGRU remain highly effective in these physics-driven settings: they train approximately 350–400% faster than the standard LSTM or GRU, while matching—and often surpassing—their accuracy. Thus, for particle-based continuum simulations, minimal recurrent architectures offer a superior trade-off between computational overhead and predictive performance, thereby advancing real-time or high-fidelity simulation workflows in engineering and visual effects. We conclude that minimal RNNs are well-suited for surrogate modeling of fluid and soft-solid dynamics.

Keywords: Continuum simulation · Lagrangian · particle-based methods · Material Point Method · Surrogate modeling · Temporal learning · LSTM · minLSTM · minGRU · minRNNs· Minimal RNNs

1 Introduction

Physics-based continuum simulation techniques aim to enhance visual fidelity [2], simulate new phenomena [3], and improve computational efficiency [4]. Our work seeks to preserve visual realism while accelerating simulations via machine

[4] Also available at https://github.com/BorealisAI/minRNNs

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

learning, thereby enabling real-time interactive graphics and other applications. By leveraging machine learning algorithms, we aim to conserve critical flow features (e.g., density, kinetic energy, vorticity) while reducing the computational overhead of fluid simulations.

Simulating the behavior of fluids and deformable solids is essential in computer graphics, engineering, and scientific computing. Lagrangian representations (particle- or mesh-based) track positions, velocities, and other attributes over time, offering a versatile means of modeling complex behaviors. However, full physics-based simulations remain computationally expensive [5,6]. Consequently, machine learning-based surrogate models have emerged as an attractive alternative for rapidly predicting system evolution [7,8,9]. Traditional fluid-dynamics methods, which store velocity fields on regularly spaced grids [10], perform well for smooth flows but struggle with irregular phenomena in complex or continuous domains. Moreover, the curse of dimensionality, local discontinuities, and hidden physical constraints complicate modeling in large-scale, detailed space-time domains, highlighting the need for more compact and structured representation spaces.

Lagrangian continuum simulation [11] is especially popular for real-time applications (see, e.g.[13,14]). Nevertheless, relatively few efforts have developed particle-based or material point-based predictors for fluids or deformable solids. Recent advances, such as continuous convolution methods [15] and Graph Network simulators ([16]), are beginning to fill this gap. Additionally, momentum-conserving techniques have shown promise in learned physics simulations ([17]). Despite these strides, developing robust and scalable machine learning solutions for Lagrangian continuum simulation remains an open challenge, particularly for long-term temporal modeling ([9,18,19]).

Although machine learning has accelerated continuum simulations, trade-offs remain. Non-recurrent networks (e.g., [7,9]) are not well suited to capture long-term dependencies, while standard recurrent networks such as LSTMs [20] or GRUs [21] incur high training costs due to backpropagation through time. Furthermore, fluid simulations involve high-dimensional particle data and complex multi-phase interactions, requiring models that are both flexible enough to capture intricate dynamics and scalable enough to handle large amounts of data [6,22].

Building on insights from parallelizable RNNs [1], we hypothesize that minimal recurrent networks—which remove hidden-state dependencies from their gating mechanisms—can preserve key memory properties with lower parameter overhead and facilitate efficient, parallel training. Minimal variants such as *minLSTM* and *minGRU* [1] bypass the costly backpropagate through time (BPTT) process while capturing complex temporal dynamics, thereby addressing both the computational burden of large-scale surrogate modeling and the need for robust long-range memory.

In this work, we investigate neural network architectures for learning the temporal evolution of continuum simulations from per-particle attributes. We compare a baseline Multi-Layer Perceptron (MLP) that directly predicts the

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

next frame with recurrent architectures that integrate temporal information, focusing on classical LSTMs versus minimal recurrent architectures (minLSTM and minGRU, [1]). To the best of our knowledge, this is the first study to explore minimal recurrent architectures for predicting the temporal evolution of fluid and multi-material simulations, while evaluating trade-offs between accuracy and computational efficiency.

2 Related Work

Lagrangian Continuum Simulation is widely used for simulating fluids and deformable bodies. In these approaches, the continuum is discretized into particles that carry positions, velocities, and material properties. Classical solvers update these attributes based on physical conservation laws [23]. An example of a Lagrangian Approach is Lagrangian Finite Element Method [24]. In this method, the deforming computational domain is discretized by means of a finite element mesh that deforms with the flow [24]. The method allows simulations of suspensions under planar extensional flow to be conducted to large strains in a truly periodic cell. Another prominent example of the Lagrangian approach is SPH (Smoothed Particle Hydrodynamics) [11,27]. Initially used in astronomy to simulate gas dynamics on a large scale (astrophysics), but later it was also applied to the problem of incompressible flow such as simulations of ocean waves and liquid mud in tanks [12]. In the SPH method, the computational domain is discretized into a finite number of particles. The continuous integral representation of the field variable f(x) can then be approximated by summation over the neighbouring particles using the smoothing/kernel function W and smoothing length, r, defining the influence domain of W [27].

Deep Learning for Physics-Based Systems: The rise of Machine Learning Techniques, particularly Deep Neural Networks (DNN), is currently propelling research into accelerating physical simulation and modeling physical functions. Applications cover a range of areas such as rapid approximations for numerical fluid solvers [7,16], robotic control and planning [25,26], and more efficient cloth simulation [30,31].

Temporal modeling: Autoencoders have been used to compress fluid fields [18,19], and recurrent networks, especially LSTMs [20], have been employed to capture temporal dependencies. More recently, minimal recurrent architectures such as minLSTM and minGRU [1] have emerged as efficient alternatives that reduce parameter counts while maintaining performance. Autoencoders (AE) are used by [9,18,19] to compress the dimensionality of the simulation into latent space prior to temporal prediction. [18], in a separated training process from the AE, uses an LSTM network and time convolution in latent space to predict the temporal change of the fluid. This approach leads to large speed-ups relative to the tradional CPU-based solver. [28] developed a non-intrusive data-driven ROM using POD for reduced basis construction and LSTM for temporal evolution,

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

integrating CFD and DEM while employing a filtering procedure to reduce dimensionality. [29] developed a hybrid neural network combining LSTM and CNN to predict unsteady fluid flows with an eulerian representation, where the LSTM forecasts the lift coefficient over time, and the CNN reconstructs velocity and pressure fields based on CFD-simulated flow data around cylinders. [9] uses a generative approach with a CNN to re-synthesize / upscale the dynamic flow fields for both smoke and liquid.

3 Ground Truth for Continuum Simulation

In this section, we provide a detailed overview of the datasets and simulation environments used in our research on fluid and deformable solid simulations. The primary goal is to establish a comprehensive ground truth that serves as the basis for training and validating our models. All ground truth simulations are conducted using the Material Point Method (MPM) [33], implemented through the Taichi framework [34]. We adopt a time step size of $\Delta t = 8 \times 10^{-4}$ time units for these simulations, saving every 20th time step (referred to below as a "frame")—thus generating about 60 frames per second. This setup allows us to generate diverse scenarios that capture various physical interactions between different materials, such as liquids, soft solids, and deformable solids. Figure 1 illustrates examples of the different types of multi-material simulations used in our study. These include representations of homogeneous liquid, "snow", "rope", and "jelly" ball interactions, as well as mixed simulations where different materials interact dynamically ((see Appendix A for further details)).

3.1 Continuum Domain

These experiments involved two primary types of simulations: homogeneous fluid simulation and multi-material simulation. For multi-material simulations, we generated 60 two-dimensional scenes using predefined scenarios specified by different scenario IDs, each with unique initial configurations involving varying shapes, sizes, initial positions, and velocities of fluid and some material bodies within a square container of 1 by 1 unit. Similarly, for fluid-only simulations, we generated 40 distinct scenes, each focused solely on fluid dynamics, with varied initial configurations of fluid particles.

The duration of the multi-material scenes was uniformly set to 10 time units to capture the full range of material interactions. For the fluid-only simulations, scene durations varied from 5 to 25 time units. These simulations were designed to run until significant fluid dynamics had subsided, avoiding the bias of prolonged stationary scenarios in the training data. By ending the simulations when little activity remained, we ensured that the dataset represented a diverse set of dynamic behaviors without over-representing static configurations.

To further characterize the flow in these experiments, we use the Reynolds number, defined as $Re = \frac{\rho U L}{\mu}$. Given the normalized parameters in our simulations ($\rho = 1$, $L \approx 1$, and $\mu = 0.5$), the Reynolds number ranges from approximately 0.2 to 40. In MPM, the dynamic viscosity μ is related to the density ρ and

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

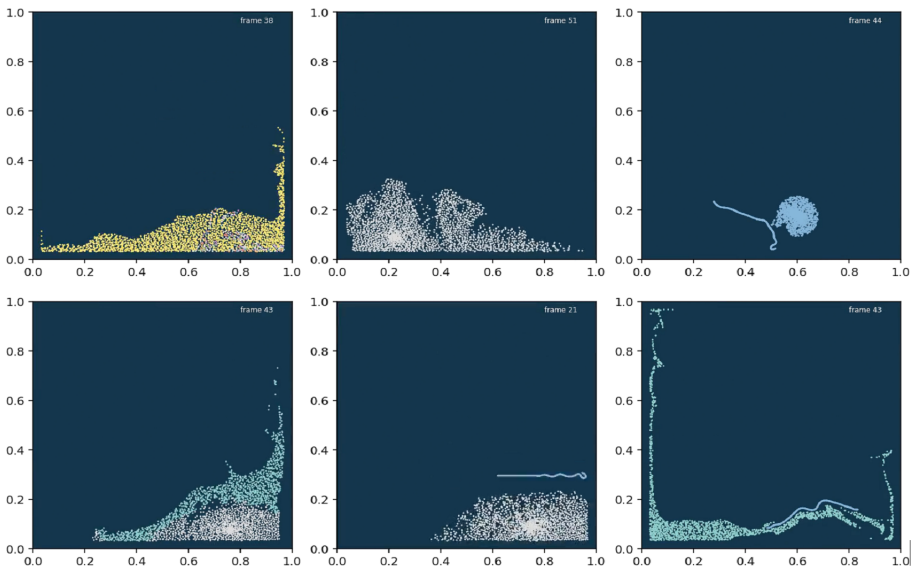

Fig. 1. Snapshots from groundtruth data for multi-material multi-phase simulation. Top-left: homogeneous liquid; top-middle: snow; top-right: rope and jelly ball; bottom-left: liquid and snow; bottom-middle: rope and snow; bottom-right: rope and liquid.

the kinematic viscosity ν by the relation $\mu = \rho\nu$. Despite dynamic variations, the flow remains predominantly laminar.

3.2 Data Generation and Description

Figure 2 illustrates examples from the dataset used for fluid simulation, while Figure 1 displays example datasets for multi-material simulations. For multi-material simulations, initial configurations were generated using a shell script that automated the creation of 60 unique scenarios by running the Python code of the MPM Simulation with varying scenario IDs. These scenarios included diverse initial conditions for fluids, and soft solids, ensuring a wide range of material interactions.

For fluid-only simulations, a separate script was used to automate the generation of 40 different scenarios, each focused solely on fluid dynamics. In both simulation types, we captured the position (p_x, p_y), velocity (v_x, v_y), and material parameter (m) (in integer) of each particle in every frame. Positions and velocities were normalized relative to the simulation domain and a predefined maximum velocity to ensure consistency across scenes. The resulting datasets provide a comprehensive collection of fluid and multi-material interactions, as depicted in Figures 2 and Figures 1.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

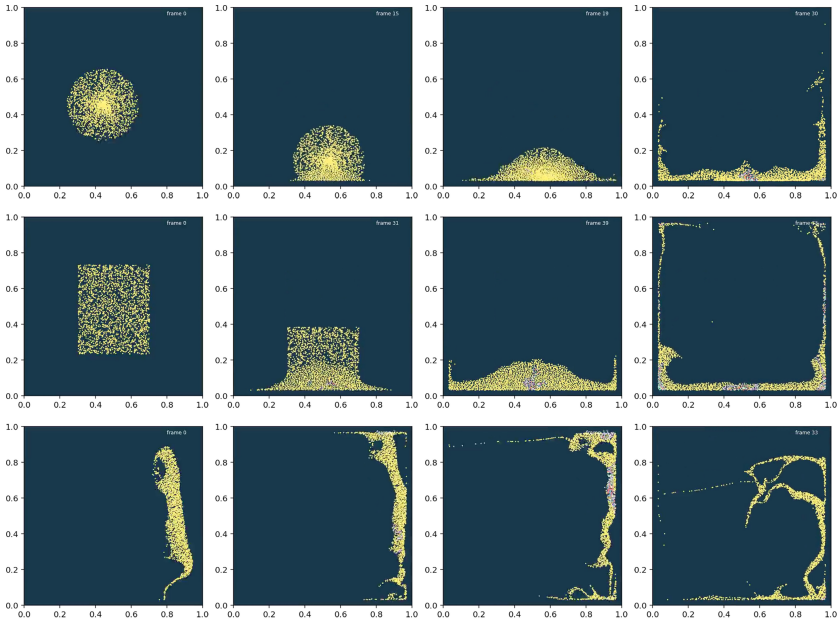

Fig. 2. Dataset of fluid simulations using 2,500 particles per domain, with each row showing a different initial configuration and its time evolution: **Top**: A circular fluid body spreading from the center. **Middle**: A rectangular fluid body deforming and dispersing. **Bottom**: A random-velocity splash under gravity.

4 Data Normalization

Simulation data, stored in HDF5 files (each frame having 2,500 particles), is loaded and reshaped from:

$$\text{(totalFrames, 2500, nFeatures)} \quad \rightarrow \quad \text{(totalFrames} \times 2500\text{, nFeatures-1)}.$$

This 2D representation allows MinMax scaling across all features—positions (x, y), velocities (v_x, v_y), and any material parameters. After scaling, the data is reshaped back to its original structure to preserve temporal and spatial relationships.

Because each feature can have different value ranges, they are normalized independently. Using MinMax scaling, each feature is mapped to the interval $[0, 1]$ by subtracting the feature's minimum value and dividing by its range: s defined as:

$$x_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{1}$$

This ensures no single feature disproportionately affects the model.

We apply Scikit-Learn's `MinMaxScaler` to fit and transform both input and target data, then reshape the transformed data back to its original format. The

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

normalized dataset is subsequently partitioned into training, validation, and test sets, following an 80:10:10 split ratio.

5 Temporal Modeling

Accurately capturing the time evolution of particle states is essential for building predictive surrogate models of Lagrangian simulations. Our approach compares different architectures to handle temporal information, focusing on the interplay between classical LSTM networks and their minimal variants.

5.1 Encoding and Decoding Particle States

We first encode raw particle attributes into a latent space using an MLP encoder, and then decode from that latent representation back into particle space with an MLP decoder. For a particle i with attributes $\mathbf{p}_i = (x_i, y_i, v_{x,i}, v_{y,i}, m_i)$, the encoder maps $\mathbf{p}_i$ to $\mathbf{z}_i$, and the decoder reconstructs $\hat{\mathbf{p}}_i$ from $\mathbf{z}_i$. Training these modules end-to-end, with Mean Squared Error (MSE) loss across all particles and frames, provides a compact representation of particle states.

5.2 Minimal RNN Variants

Although standard LSTM networks [20] effectively capture long-term dependencies in complex, non-linear sequences, they can be computationally intensive and contain many parameters. This is a limitation in scientific machine learning, as long training and inference times hinder real-time applications, and high parameter counts require significant computational resources, increasing the risk of overfitting when data are limited. To address these issues, *minLSTM* and *minGRU*, introduced in [1] (also presented in Apendix C Figure C), simplify the classical gating mechanisms while retaining core memory capabilities:

- **minLSTM** reduces the standard three-gate design to two gates (input and forget), removing the output gate but preserving a dedicated memory cell. This trimming decreases the parameter count and simplifies the updates.
- **minGRU** further streamlines by merging the reset and update gates into a single gate z_t. Instead of maintaining a separate cell state, it computes a candidate hidden state $\tilde{h}_t$ and forms the final output as a convex combination of h_{t-1} and $\tilde{h}_t$.

By reducing computational complexity and parameter overhead, these minimal recurrent variants are especially attractive for scientific machine learning applications where efficiency, scalability, and robustness are essential. Both minimal variants reduce the number of gating operations and the amount of sequential dependence at each time step, which in turn increases the portion of computations that can be parallelized by GPUs or other parallel hardware ([1]). This design leads to reduced training costs and makes the architectures attractive for large-scale simulations. By contrasting LSTM, minLSTM, and minGRU, we aim to identify the best balance of accuracy, model complexity, and computational efficiency for predicting particle states over extended temporal horizons.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

6 Network Architectures and Training Procedures

We study three classes of models:

MLP Baseline An MLP maps the concatenated per-particle attributes at time t directly to the attributes at time $t+1$. Although simple, this approach suffers from drift over long rollouts.

LSTM+MLP A standard LSTM processes a sequence of frames (or their latent representations) and predicts the next frame. While this model improves temporal continuity, it incurs computational overhead.

Minimal RNNs+MLP We combine an MLP encoder–decoder with a recurrent core. In our architecture, the encoder compresses per-particle data into a latent representation, the recurrent core evolves the latent state over time, and the decoder reconstructs full particle states. We report results separately for the model with a minLSTM core and the one with a minGRU core.

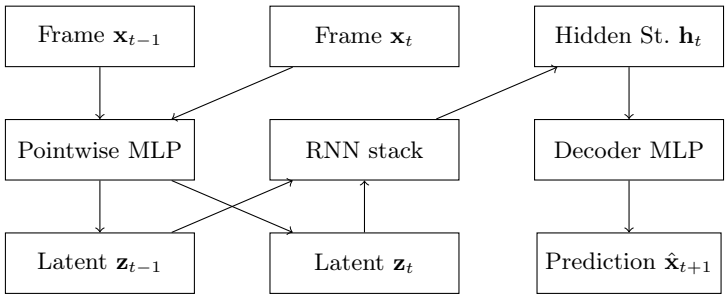

Fig. 3. Architecture for the proposed model with encoder-decoder and RNN Stack.

6.1 Architecture

We illustrate the full encoder–decoder pipeline with an RNN stack in Figure 3. The model receives two consecutive frames as input, and dropout is applied after the first and second RNN layers for regularization. In the first layer, the sequence of frame embeddings is passed through the RNN; its output is then processed by an MLP layer to transform the hidden dimensions. Finally, a decoder MLP generates the predicted frame. The layer-by-layer details of this RNN stack and hidden state appear in Table 1. By learning the encoder, RNN, and decoder weights simultaneously, the model constructs a latent-space representation on the fly that best suits the prediction task. The next section presents both qualitative and quantitative results showing how well this end-to-end approach captures long-term fluid and solid behavior.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

Table 1. Details layer by layer of the proposed Temporal Model

Layer	Input	Output	Details
(1) Input	$(B, T, N, 5)$	–	features per point (e.g., x, y, v_x, v_y, m).
(2) Encoder	$(B, N, 5)$	$(B, 1250)$	**Pointwise MLP**: 2×Linear + ELU (e.g., $[5 \to 256 \to 1250]$). Then mean/max pool over N.
(3) RNNStack	$(B, T, 1250)$	$(B, 1250)$	2× minRNN(1250) + MLP($1250 \to 1250$). Dropout after each RNN layer. Output is last time-step $(B, 1250)$.
(4) Decoder	$(B, 1250)$	$(B, N, 2)$	Single Linear layer ($1250 \to N \times 2$), then reshape to $(B, N, 2)$.
(5) Output	–	$(B, N, 2)$	Predicted frame (e.g., (dv_x, dv_y)).

6.2 Training Details

We train our network end-to-end in a supervised manner using particle trajectories generated by classical physics-based simulations as ground truth ([33]). Each training sample consists of a pair of input frames, $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, and the subsequent target frame, $\mathbf{x}_{t+1}$. Each frame contains $N = 2500$ particles, with each particle described by its 2D position (x, y) , velocity (v_x, v_y) and material parameter (m).

Our objective combines a standard mean-squared-error (MSE) loss on positions and velocities with an additional density-based loss to encourage globally consistent particle distributions. Formally, the temporal loss at time step $n+1$ is defined as:

$$\mathcal{L}^{n+1} = \frac{1}{N\,k} \sum_{i=1}^{N} \sum_{j=1}^{k} \left(y_{i,j}^{n+1} - \hat{y}_{i,j}^{n+1} \right)^2, \tag{2}$$

where N is the number of particles and k is the number of scalar attributes per particle (e.g., positions, velocities, densities, etc.). Here, $y_{i,j}^{n+1}$ denotes the ground truth value of the j-th attribute for the i-th particle, while $\hat{y}_{i,j}^{n+1}$ is the corresponding predicted value. In addition to this loss, a density map MSE is applied to enforce high quality particle distributions.

Data is processed in minibatches that capture partial sequences from the simulation. While early experiments used a batch size of 192, extensive testing led us to select a batch size of 4096 to balance training stability and speed. Training iterates over all T_{frames} of the simulation (from frame 0 to $T_{\text{frames}} - 1$), applying the same multilayer perceptron (MLP) pointwise to each particle without requiring any explicit dimensionality reduction.

Optimization is performed using the Adam optimizer with a learning rate of 0.001, and training proceeds for several hundred epochs. Real-time monitoring is facilitated by `torch.utils.tensorboard`, and the entire training pipeline is implemented using PyTorch.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

7 Results

7.1 Fluid Tests

We evaluate our models on single-phase fluid simulations. Figure 4 (left) compares the ground truth for a typical simulation with four methods: LSTM-based, GRU, minLSTM, and minGRU (all of which use an MLP as the hidden transformation) (complete figures for all fluid test scenes are available in appendix D). We also compare with a pure MLP architecture, though it is not shown here due to space limitations. Although the pure MLP captures the fluid's coarse shape, it drifts noticeably over time. The LSTM based model preserves fluid continuity but is computationally more expensive. By contrast, both minLSTM and minGRU produce stable, accurate predictions over long horizons, with slight differences in error metrics.

Table 2 reports numerical comparisons for six fluid test scenes, showing both average and final-frame MSE. Each table block highlights the lowest errors in bold among the tested methods.

Table 2. Comparisons of *Average frames rollout (20)* MSE and *frame(20th)* MSE for fluid scenes. The lowest value within each block is in bold.

Scene	Average frames rollout (1..20)				Frame (20th) Rollout MSE			
	minGRU	**minLSTM**	**LSTM**	**GRU**	**minGRU**	**minLSTM**	**LSTM**	**GRU**
54	0.1469	0.1465	0.1324	**0.1320**	0.2821	0.2814	**0.2632**	0.2632
55	0.1654	0.1652	0.1519	0.1491	0.3151	0.3148	**0.3052**	0.3052
56	0.1093	0.1090	0.1050	**0.0904**	0.2098	**0.2093**	0.2155	0.2155
57	0.1893	0.1892	**0.1882**	0.1935	0.3619	**0.3618**	0.4002	0.4002
58	0.1206	**0.1203**	0.1257	0.1231	0.2314	0.2310	**0.2310**	0.2356
59	0.1289	0.1287	0.1149	**0.1033**	0.2470	**0.2466**	0.2497	0.2497
Average	0.1343	0.1431	0.1364	**0.1320**	0.2746	**0.2741**	0.2764	0.2764

7.2 Multi-Material Simulations

We next evaluate scenes involving pairwise interactions among four material types: liquid, snow, rope, and jelly. These multi-material scenarios exhibit diverse physical behaviors (e.g., fluid-like spreading vs. elastic deformation) and thus present greater challenges for learning robust dynamics models.

Table 3 summarizes each method's performance across six multi-material scenes, reporting both *average MSE* over frames 1–20 and the *final-frame* MSE at frame 20. Four neural architectures (LSTM, minGRU, minLSTM, and GRU) are compared, with the lowest value in each row highlighted in bold. Although every approach excels for certain material combinations, *minLSTM* consistently achieves lower error across most interactions.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

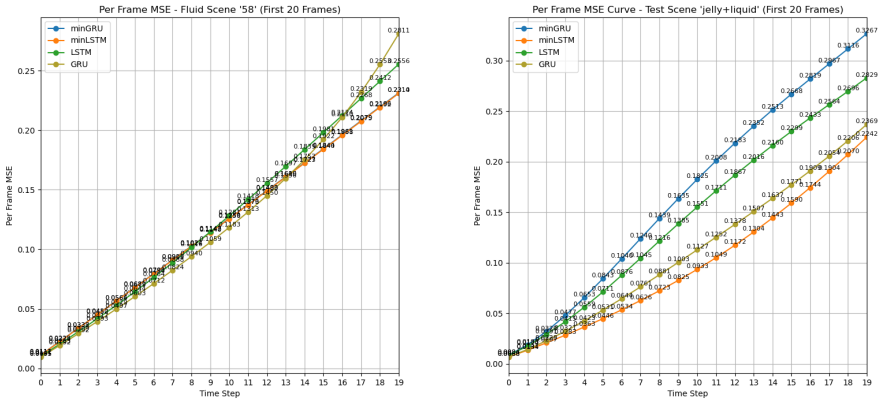

Fig. 4. Left: Raw per-frame MSE curves for fluid scene 58, which underpin the numerical comparisons in Table 2. **Right**: MSE curves for the multimaterial scene "jelly+liquid", as reported in Table 3. Each plot displays MSE values over 20 frames for the evaluated models. Complete figures for all test scenes are available in Appendix D.

Figure 4 (right) and Figure D in Appendix D reinforces these MSE trends on a per-scene basis, while Figure D in the same appendix shows visual snapshots of four representative multi-material simulations. We observe that the more challenging pairs (e.g., *rope+liquid* or *rope+jelly*) produce higher errors overall, underscoring the difficulty of accurately modeling multi-material coupling.

Table 3. Comparisons of *Average frames rollout (20)* MSE and *frame(20th)* MSE for multi-material scenes. The lowest value within each block is in bold.

Scene	**Average frames rollout (1..20)**				**Frame (20th) Rollout MSE**			
	minGRU	**minLSTM**	**LSTM**	**GRU**	**minGRU**	**minLSTM**	**LSTM**	**GRU**
liquid+snow	0.2273	**0.2269**	0.2557	0.3210	**0.4700**	0.5076	0.6026	0.7888
rope+snow	0.2362	**0.1657**	0.3193	0.1789	0.6143	**0.4314**	0.8193	0.4861
jelly+liquid	0.1682	**0.0983**	0.1444	0.1111	0.3267	**0.2242**	0.2829	0.2369
snow+snow	0.3553	0.3843	**0.3360**	0.4263	**0.6687**	0.8077	0.8330	1.0047
rope+jelly	0.1980	**0.0529**	0.1832	0.2336	0.6679	**0.1646**	0.5421	0.6328
rope+liquid	0.2552	**0.1886**	0.6009	0.3924	0.7185	**0.4610**	1.6143	1.2627
Average	0.2400	**0.1861**	0.3066	0.2772	0.5777	**0.4327**	0.7824	0.7353

7.3 Training Speed

To assess efficiency, we measured training times for each recurrent architecture on a dataset with 23,856 samples of multi-material scenario. Table 4 summa-

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

rizes the key metrics. The minimal variants (minLSTM and minGRU) converge much faster, both in total training time and average time per epoch, than their standard LSTM and GRU counterparts.

Table 4. Training speed comparison for minLSTM, minGRU, LSTM, and GRU. Total training time is the wall-clock duration until completion (or early stopping), and average epoch time is the total divided by the final epoch count.

Model	Epochs	Time (s)		Time/Epoch (s)	
		Total	per sample (μs)	Avg.	Max.
minLSTM	99	61.50	36.68	0.62	0.88
minGRU[5]	85	54.04	36.05	0.54	0.86
LSTM	99	286.96	139.84	2.87	3.34
GRU	99	266.94	129.07	2.67	3.08

As shown above, minLSTM and minGRU both converge in significantly less time (~ 0.6 s per epoch) compared to standard LSTM and GRU ($\sim$ 2.7-2.9 s per epoch), demonstrating the computational advantages of minimal RNN architectures. Despite faster training, minLSTM and minGRU also match or exceed the predictive performance of their standard counterparts (see Table 2), indicating a superior trade-off between speed and accuracy.

8 Conclusion

We introduced an end-to-end surrogate modeling approach for particle-based continuum simulations using neural networks, incorporating a recurrent core to capture both local particle dynamics and global temporal dependencies. Our experiments spanned homogeneous fluid and multi-material scenarios, providing a broad view of each method's strengths and weaknesses.

In particular, we compared a classical LSTM against more compact RNN variants (minLSTM and minGRU). Across all tested scenes, these minimal architectures consistently exhibited accuracy on par with, or better than, the standard LSTM and GRU, often achieving the lowest mean-squared errors. Moreover, our training speed benchmarks showed minLSTM and minGRU to be roughly 350–400% faster, substantially reducing wall-clock time and computational overhead. This efficiency boost not only cuts overall training costs but also makes more frequent or comprehensive hyperparameter searches feasible.

Overall, these results highlight that minimal RNNs can provide a superior trade-off between accuracy and efficiency for physics-based simulation tasks. By reducing overhead without compromising predictive quality, practitioners can accelerate high-fidelity simulation workflows in engineering, visual effects, and other real-time applications. Looking ahead, we plan to extend these methods to three-dimensional simulations and more complex, multi-modal material interactions, while also exploring graph-based neural networks to better capture local particle neighborhoods and interactions.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

Acknowledgments. We gratefully acknowledge the LPDP – Indonesia Endowment Fund for Education Agency for supporting this research, and the NVIDIA Hardware Grant Program for providing the NVIDIA RTX A5000 GPU used in our experiments.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Appendix

This section contains supplementary material for the paper. You can access the full supplementary materials PDF online at the following link:

– github.com/dodydharma/minRNNsFlow/blob/main/docs/appendix.pdf

A Simulation Process

This section provides details on the simulation process used to generate the datasets. For complete information, please refer to the supplementary PDF at: appendix A.

B Selection of Activation Function

The choice of activation function is critical for the performance and efficiency of deep neural networks. In this section, we describe our experiments with various nonlinear activation functions (e.g., Sigmoid, Tanh, ReLU, and ELU) and explain the rationale for our selection. For full details, please refer to the supplementary PDF at: appendix B.

C minRNN Architecture

This section describes the architecture of the minimal RNN variants (minLSTM and minGRU) and compares them with classical LSTM/GRU models. Merging or removing gates reduces parameters and accelerates training without sacrificing long-term temporal modeling. For further illustration, please see the supplementary PDF at: appendix C.

D Further Visualization of Results

This section provides additional visualizations of the experimental results, including qualitative comparisons and per-frame MSE curves. For a more comprehensive set of images and charts, please refer to the supplementary PDF at: appendix D.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

References

1. Feng, L., Tung, F., Ahmed, M. O., Bengio, Y., Hajimirsadeghi, H.: *Were RNNs All We Needed?* arXiv preprint arXiv:2410.01201 (2024). https://arxiv.org/abs/2410.01201
2. Qu, Z., Zhang, X., Gao, M., Jiang, C., Chen, B.: *Efficient and Conservative Fluids Using Bidirectional Mapping*. ACM Trans. Graph. **38**(4), 128 (2019).
3. Huang, L., Hadrich, T., L, D., Michels.: *On the Accurate Large-scale Simulation of Ferrofluids*. ACM Trans. Graph. **38**(4), 93 (2019).
4. Goldade, R., Wang, Y., Aanjaneya, M., Batty, C.: *An Adaptive Variational Finite Difference Framework for Efficient Symmetric Octree Viscosity*. ACM Trans. Graph. (TOG) **38**(4), 94 (2019).
5. Dharma, D., Jonathan, C., Kistidjantoro, A. I., Manaf, A.: *Material point method based fluid simulation on GPU using compute shader*. In: 2017 Int. Conf. Advanced Informatics, Concepts, Theory, and Applications (ICAICTA) (2017).
6. Gao, M., Wang, X., Wui, K., Pradhana, A., Sifakis, E., Yuksel, C., Jiang, C.: *GPU Optimization of Material Point Methods*.umen ACM Trans. Graph. **37**(6) (2018).
7. Tompson, J., Schlachter, K., Sprechmann, P., Perlin, K.: *Accelerating Eulerian Fluid Simulation With Convolutional Networks*. In: Proc. 34th Int. Conf. Machine Learning (2017).
8. Chu, M., Thuerey, N.: *Data-Driven Synthesis of Smoke Flows with CNN-based Feature Descriptors*. ACM Trans. Graph. **36**(4) (2017).
9. Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., Solenthaler, B.: *Deep Fluids: A Generative Network for Parameterized Fluid Simulations*. In: Comput. Graph. Forum (Proc. Eurographics 2019) (2019).
10. Wang, Y., Jimack, P. K., Walkley, M. A.: *One-field monolithic fictitious domain method for fluid-structure interactions*. Comput. Methods Appl. Mech. Eng. **317**, 1168–1196 (2017).
11. Monaghan, J. J.: *Smoothed Particle Hydrodynamics*. Annu. Rev. Astron. Astrophys. **30**, 543 (1992).
12. Auer, S.: *Realtime Particle-based Fluid Simulation*. Tech. Rep. (2009). Technische Universität München.
13. Macklin, M., Müller, M.: *Position Based Fluids*. ACM Trans. Graph. **32**(4) (2013).
14. Müller, M., Charypar, D., Gross, M.: *Particle-based Fluid Simulation for Interactive Applications*. In: Proc. 2003 ACM SIGGRAPH/Eurographics Symp. Comput. Anim. 154–159 (2003).
15. Ummenhofer, B., Prantl, L., Thuerey, N., Koltun, V.: *Lagrangian Fluid Simulation with Continuous Convolutions*. In: Proc. Int. Conf. Learn. Representations (ICLR) (2019).
16. Ladický, Ľ., Jeong, S., Solenthaler, B., Pollefeys, M., Gross, M.: *Data-driven Fluid Simulations using Regression Forests*. ACM Trans. Graph. (TOG) **34**(6), 199 (2015).
17. Prantl, L., Ummenhofer, B., Koltun, V., Thuerey, N.: *Guaranteed Conservation of Momentum for Learning Particle-Based Fluid Dynamics*. arXiv preprint arXiv:2210.06036 (2022). https://arxiv.org/abs/2210.06036
18. Xie, W., Yang, Z., Yu, F., Jiang, C.: *Latent Space Physics: Towards Learning the Temporal Evolution of Fluid Flow*. Comput. Graph. Forum **38**(2), 71–82 (2019). https://doi.org/10.1111/cgf.13661
19. Wiewel, S., Kim, B., Azevedo, V. C., Solenthaler, B., Thuerey, N.: *Latent Space Subdivision: Stable and Controllable Time Predictions for Fluid Flow*. In: Proc. Int. Conf. Machine Learning (ICML) (2020).

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17

20. Hochreiter, S., Schmidhuber, J.: *Long Short-Term Memory*. Neural Comput. **9**(8), 1735–1780 (1997).
21. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: *Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation*. In: Proc. EMNLP (2014).
22. Nie, X., Chen, L., Xiang, T.: *Real-Time Incompressible Fluid Simulation on the GPU*. Int. J. Comput. Games Technol. (2015).
23. Bridson, R.: *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press (2008).
24. Ahamadi, M., Harlen, O. G.: *A Lagrangian Finite Element Method for Simulation of a Suspension under Planar Extensional Flow*. J. Comput. Phys. **227**, 7543–7560 (2008).
25. Hu, Y., Liu, J., Tenenbaum, J. B., Freeman, W. T., Wu, J., Rus, D., Matusik, W.: *ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics*. In: Proc. 32nd Conf. Neural Information Processing Systems (NIPS) (2018).
26. Schenck, C., Fox, D.: *SPNets: Differentiable Fluid Dynamics for Deep Neural Networks*. 2nd Conf. Robot Learning (CoRL) (2018).
27. Wang, C., Wang, Y., Peng, C., Meng, X.: *Smoothed Particle Hydrodynamics Simulation of Water-Soil Mixture Flows*. J. Hydraul. Eng. **142**(10) (2016).
28. Hajisharifi, A., Halder, R., Girfoglio, M., Beccari, A., Bonanni, D., Rozza, G.: *An LSTM-enhanced surrogate model to simulate the dynamics of particle-laden fluid systems*. Comput. Fluids **280**, 106361 (2024). https://doi.org/10.1016/j.compfluid.2024.106361
29. Portal-Porras, K., Fernandez-Gamiz, U., Zulueta, E., Irigaray, O., Garcia-Fernandez, R.: *Hybrid LSTM+CNN architecture for unsteady flow prediction*. Mater. Today Commun. **35**, 106281 (2023). https://doi.org/10.1016/j.mtcomm.2023.106281
30. Lee, T. M., Oh, Y. J., Lee, I.: *Efficient Cloth Simulation using Miniature Cloth and Upscaling Deep*. ACM Trans. Graph. **38**(1) (2019).
31. Tan, Q., Pan, Z., Gao, L., Manocha, D.: *Realtime Simulation of Thin-Shell Deformable Materials using CNN-Based Mesh Embedding*. arXiv preprint arXiv:1909.12354 (2019).
32. Nwankpa, C. E., Ijomah, W., Gachagan, A., Marshall, S.: *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning*. In: Proc. Int. Conf. Learn. Representations (ICLR) (2018).
33. Yan, X., Li, C.-F., Chen, X.-S., Hu, S.-M.: *MPM simulation of interacting fluids and solids*. In: ACM SIGGRAPH/Eurographics Symp. Comput. Anim. (2018).
34. Hu, Y., Li, T., Anderson, L., Ragan-Kelley, J., Durand, F.: *Taichi: a language for high-performance computation on spatially sparse data structures*. ACM Trans. Graph. **38**(6), 1–6 (2019).

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97554-7_17