Learning Neural Constitutive Laws From Motion Observations for Generalizable PDE Dynamics

Pingchuan Ma [1] **Peter Yichen Chen** [1] **Bolei Deng** [1]
Joshua B. Tenenbaum [1 2 3] **Tao Du** [4 5] **Chuang Gan** [6 7] **Wojciech Matusik** [1]

Abstract

We propose a hybrid neural network (NN) and PDE approach for learning generalizable PDE dynamics from motion observations. Many NN approaches learn an end-to-end model that implicitly models both the governing PDE and constitutive models (or material models). Without explicit PDE knowledge, these approaches cannot guarantee physical correctness and have limited generalizability. We argue that the governing PDEs are often well-known and should be explicitly enforced rather than learned. Instead, constitutive models are particularly suitable for learning due to their data-fitting nature. To this end, we introduce a new framework termed "Neural Constitutive Laws" (NCLaw), which utilizes a network architecture that strictly guarantees standard constitutive priors, including rotation equivariance and undeformed state equilibrium. We embed this network inside a differentiable simulation and train the model by minimizing a loss function based on the difference between the simulation and the motion observation. We validate NCLaw on various large-deformation dynamical systems, ranging from solids to fluids. After training on *a single motion trajectory*, our method generalizes to new geometries, initial/boundary conditions, temporal ranges, and even multi-physics systems. On these extremely out-of-distribution generalization tasks, NCLaw is orders-of-magnitude more accurate than previous NN approaches. Real-world experiments demonstrate our method's ability to learn constitutive laws from videos.[1]

[1]MIT CSAIL [2]MIT BCS [3]Center for Brains, Minds and Machines [4]Tsinghua University [5]Shanghai Qi Zhi Institute [6]MIT-IBM Watson AI Lab [7]UMass Amherst. Correspondence to: Pingchuan Ma <pcma@csail.mit.edu>, Peter Yichen Chen <pyc@csail.mit.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]https://sites.google.com/view/nclaw

1. Introduction

Partial-differential-equation-governed dynamical systems are ubiquitous in physical, chemical, and biological settings (Haberman, 1998). Computationally solving these PDEs, e.g., using the finite element method (FEM) (Hughes, 2012), has enabled critical applications in science and engineering. Recently, research communities have explored the role of machine learning (ML) in advancing partial differential equation (PDE) modeling (Karniadakis et al., 2021). Some methods, e.g., physics-informed neural network (PINN) (Raissi et al., 2019), assume the entire PDE is known. Other methods, e.g., graph neural network (GNN) (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020), do not assume any knowledge about the underlying PDE. These methods learn the PDE-governed dynamics entirely from data. Without any knowledge about the PDE, these approaches are prone to violating physical laws and have limited generalizability. In this work, we argue that instead of learning the entire PDE with neural solutions, there are benefits of replacing only *parts* of the PDEs with neural components while keeping the rest of the PDEs intact. For example, we contend that commonly-agreed physical laws, such as the elastodynamics equation (Gurtin et al., 2010), do not need to be learned. Instead, these commonly-agreed physical laws should be explicitly enforced to guarantee physical correctness. By contrast, we champion that those parts of the PDEs where researchers traditionally struggle to model can particularly benefit from a learning solution. In particular, we explore an ML approach for an indispensable part of many PDEs: the constitutive laws.

A constitutive law describes the relationship between two physical quantities (Truesdell & Noll, 2004), e.g., stress and strain in solids (Gonzalez & Stuart, 2008), shear stress and shear rate in fluids (Tropea et al., 2007), the electric displacement field, and the electric field in electromagnetism (Landau et al., 2013). These constitutive laws play crucial roles in important PDEs, such as the elastodynamic, Navier-Stokes, and Maxwell's equations.

Traditionally, constitutive laws are manually designed by domain experts. While they should satisfy generally-accepted constraints, e.g., frame indifference (rotational equivari-

ance), the ultimate criterion of good constitutive laws is how well they match the experimentally observed data, which is often highly nonlinear. As such, to design constitutive relationships, researchers have relied on various data-fitting tools, ranging from high-order polynomials (Ogden, 1997) and splines (Xu et al., 2015) to exponential models (Hencky, 1933). Recently, neural-network-based models have been shown to achieve higher accuracies than their classic, expert-constructed counterparts, thanks to the neural network's large learning capacity (Ghaboussi et al., 1991; Le et al., 2015). Furthermore, these models alleviate the need to manually design function forms by hand since the same network architecture can be used to capture various mechanical behaviors (Liu et al., 2020). Indeed, we show in our work that a single neural network architecture can capture diverse constitutive behaviors, ranging from water to elastic solids, which previously required case-by-case, expert-designed constitutive models.

When it comes to learning constitutive laws via neural networks, two challenges stand out. First, how to enforce these neural constitutive laws to obey known physics priors, e.g., frame indifference? Second, how to obtain the labeled data for supervised training? For example, in the case of learning elasticity, one must obtain labeled stress-strain pairs (As' ad et al., 2022; Ellis et al., 1995; Furukawa & Yagawa, 1998). However, these stress-strain pairs are often either impossible to measure, e.g., *in vivo* medical imaging (Abulnaga et al., 2019), or require highly specialized devices (Bishop & Henkel, 1962).

To address the first question, we introduce the physical priors as an inductive bias in the network architecture and directly bake in these priors *by design*. In this way, our network naturally satisfies the necessary priors without tons of data augmentation. To address the challenge of labeled data, we embed the learnable constitutive model inside a differentiable PDE-based physical simulator (Murthy et al., 2020) and do supervised training on the output of this physical simulator, *not* on the output of the constitutive model itself. In particular, our approach trains on motion observation data (formally defined as kinematic information in the continuum mechanics literature), e.g., material point positions, which are significantly easier to measure than stress information.

With the proposed constitutive learning approach, we demonstrate generalizable PDE dynamics on which pure NN approaches, e.g., GNN, struggle. These generalization tasks include extreme out-of-distribution generalization over temporal ranges, initial/boundary conditions, geometries, and multi-physics systems while training on *a single motion trajectory*. We attribute our generalization advantage over pure NN approaches to the fact that we only learn one element of the PDE while keeping the PDE's widely-accepted elements intact (e.g., conservation of mass and momentum).

In summary, our work makes the following contributions:

1. We embed a learnable constitutive law inside a differentiable physics simulator and train the constitutive law directly on the simulator's output.
2. We guarantee the satisfaction of physical priors to constitutive laws by designing network architectures with matrix-wise rotation equivariance and undeformed state equilibrium.
3. We validate our hybrid NN-PDE approach on large-deformation PDE dynamics featuring one-shot generalization over unseen temporal ranges, initial conditions, boundary conditions, geometries, and multi-physics systems, on which previous pure NN approaches fail.

2. Related Work

Constitutive Laws Constitutive laws date back to 1676 when Robert Hooke, F.R.S. observed the linear relationship between spring force and deformation, i.e., $F_s = kx$ (Thompson, 1926). Since then, constitutive laws have been the staple of various branches of physics: elasticity (Treloar, 1943; Fung, 1967; Arruda & Boyce, 1993), plasticity (Mises, 1913; Drucker & Prager, 1952), fluids (Chhabra, 2006), and permittivity (Landau et al., 2013). While these constitutive laws have traditionally been constructed through nonlinear polynomial bases (Xu et al., 2015; Xu & Barbič, 2017), research communities have been embracing neural networks due to their versatility and robustness (Shen et al., 2005; Tartakovsky et al., 2018; Wang & Sun, 2018; Vlassis et al., 2020; Vlassis & Sun, 2022b; Fuchs et al., 2021; Sun et al., 2022; Vlassis & Sun, 2022a; 2021; Klein et al., 2022; Liu et al., 2022). Most of these neural-network-based approaches require labeled data for the input and output of the constitutive model. By contrast, our work does not require any such data. Instead, we embed the neural constitutive laws inside a differentiable-simulation-based training pipeline.

Differentiable Simulation implements traditional PDE simulations (e.g., FEM) in a differentiable manner such that the simulation can be employed in a gradient-based optimization workflow (Hahn et al., 2019; Liang et al., 2019; Ma et al., 2021; Hu et al., 2019b;a; Huang et al., 2021; Du et al., 2021b;a; Qiao et al., 2021a;b; Du et al., 2020; de Avila Belbute-Peres et al., 2018; Geilinger et al., 2020; Xian et al., 2023; Degrave et al., 2019). Notably, these differentiable physics simulations enable the recovery of constitutive parameters directly from videos and motion observations (Murthy et al., 2020; Ma et al., 2022; Chen et al., 2022), e.g., Young's modulus and Poisson's ratio of elasticity. Prior differentiable simulation works assume that an expert-designed constitutive law is known *a priori*.

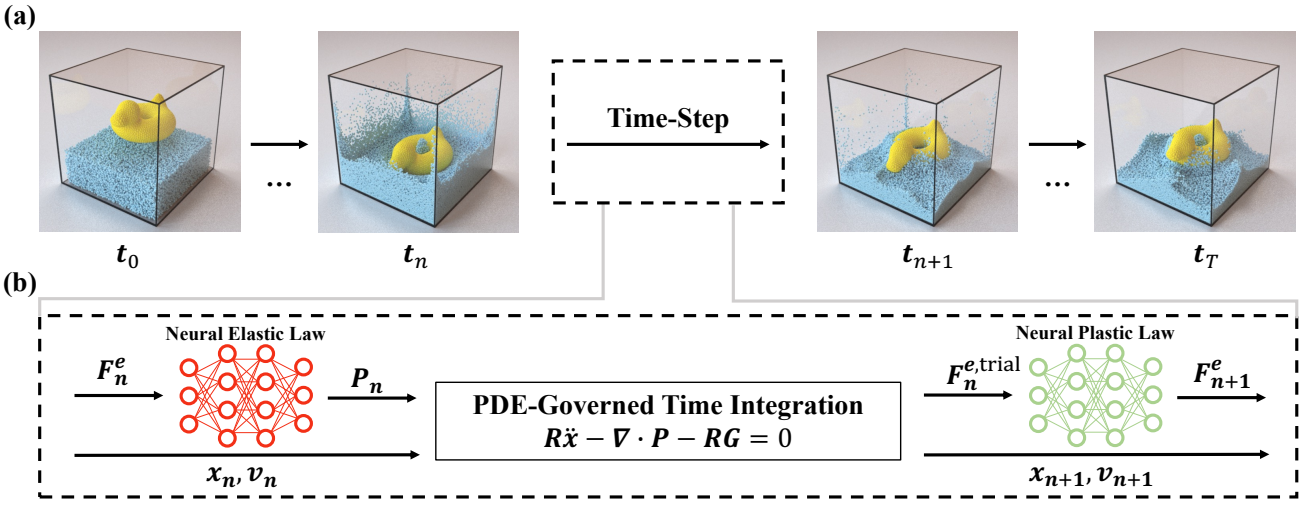


Figure 1. **Hybrid NN-PDE time-stepping**. (a) Our method "time-steps" sequentially to obtain the solution of the dynamical system (b) Inside the time-stepping algorithm, we (i) use the neural elastic constitutive law to obtain the stress, (ii) update the state by solving the governing PDE (Eqn. (1)), and (iii) obtain the new elastic deformation gradient via the neural plastic constitutive law. Algorithm 2 lists the corresponding pseudocode.

However, given a motion observation sequence of an arbitrary material, it is a non-trivial task for the practitioners to identify the underlying constitutive model, not to mention recovering their parameters. By contrast, we do not assume the availability of the material model. Instead, we parameterize the constitutive laws via generally-applicable neural network architectures. Wang et al. (2020) and Huang et al. (2020) also demonstrate promising results in training neural constitutive laws directly from motion observations, but their formulations are limited to isotropic elastic materials and 2D quasi-static loadings, respectively. Our generic formulation handles both isotropic and anisotropic materials as well as time-dependent 3D elastoplastic dynamics.

Machine Learning (ML) for PDE Dynamics Our work falls into the broader conversations on ML for PDEs. Physics-informed neural networks (PINNs) (Raissi et al., 2019) represent the PDE solution via a neural representation and introduce physics priors via PDE-informed loss. DeepONet (Lu et al., 2019) and neural operator (Li et al., 2020) learn mappings from the problem parameters to the solution with the goal of achieving fast surrogate models. Unlike these approaches, our work operates *entirely within* the classical numerical solver framework. We propose an ML solution for *one and only one component* of the classical solver: the constitutive model of the PDE. There are many great works that only replace one part of the PDE with the neural network, although not in a constitutive law context. For example, Bar-Sinai et al. (2019)'s data-driven discretizations, Yin et al. (2021)'s APHYNITY, and Um et al. (2020)'s solver-in-the-loop approach. Another robust ML-PDE framework is the graph neural network (GNN). GNN directly learns how much force is applied to a discretized particle (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020) at every time step. Implicitly, GNN learns *both* the constitutive law (material model) *and* the equation of motion (i.e., the elastodynamics equation). As such, our work differs from GNN by incorporating the equation-of-motion prior, which holds true in standard continuum mechanics settings (Gurtin et al., 2010), and only treating the constitutive model with an NN approach. Relatedly, Li et al. (2022) augment classic numerical solvers with neural-work-based plasticity energy, aiming to improve optimization-time-integrators. By contrast, our work focuses on constitutive laws, not time integrators.

3. Method

Given a motion sequence of materials undergoing deformations, our goal is to obtain their constitutive models, represented via neural networks. These constitutive laws can later be used to predict scenarios unseen in the training motion sequence. Sec. 3.1 first recaps the current state-of-the-art framework for identifying classic constitutive parameters from motions observations. Secs. 3.2 and 3.3 will introduce the proposed neural alternative.

3.1. Background

Continuous PDE In this work, we assume all materials in the experiments observe the elastodynamic equation (Gonzalez & Stuart, 2008):

$$\rho_0 \ddot{\boldsymbol{\phi}} = \nabla \cdot \boldsymbol{P} + \rho_0 \boldsymbol{b}. \tag{1}$$

The vector field of interest is the deformation map $\boldsymbol{\phi}$, which uniquely describes the current position $\boldsymbol{x}$ of an arbitrary

spatial point with an initial position $\boldsymbol{X}$ at time t, i.e., $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X}, t)$. Additionally, ρ_0 is the initial density, $\boldsymbol{P} = \widehat{\boldsymbol{P}}(\boldsymbol{F}^e)$ is the first Piola-Kirchhoff stress, $\boldsymbol{F}^e$ is the elastic part of the deformation gradient ($\nabla\boldsymbol{\phi}$), $\dot{\boldsymbol{\phi}}$ and $\ddot{\boldsymbol{\phi}}$ are the velocity and acceleration, and $\boldsymbol{b}$ is the body force. In principle, our approach can also be extended to other PDEs involving spatiotemporal gradients, e.g., the Navier-Stokes equations.

Constitutive laws For Eqn. (1) to be well-defined, constitutive relationships must be prescribed. The elastic constitutive law defines the relationship between $\boldsymbol{P}$ and $\boldsymbol{F}^e$: $\boldsymbol{P} = \widehat{\boldsymbol{P}}(\boldsymbol{F}^e)$. The plastic constitutive law defines an inequality constraint: $f_Y(\boldsymbol{P}) < 0$ (where f_Y is the yield function (Borja, 2013)), as well as a flow rule when this constraint is violated. Constitutive laws in these forms have been shown to capture a wide range of materials: from elastic solids to granular media.

Discretization In order to numerically solve Eqn. (1), we discretize it spatially and temporally, yielding a dynamical system $\mathcal{M}$:

$$\boldsymbol{s}_{n+1} = \mathcal{M}_\mu(\boldsymbol{s}_n), \forall n = 0, 1, \ldots, T, \tag{2}$$

where T is the total number of time steps in the simulation, and $\boldsymbol{s}_n$ and $\boldsymbol{s}_{n+1}$ are the state vectors at the corresponding time steps. The vector μ encodes all the system parameters, e.g., those of the constitutive laws.

To obtain the dynamical system $\mathcal{M}$, we may use any discretization technique, e.g., FEM, finite volume methods (Eymard et al., 2000), smoothed-particle hydrodynamics (Monaghan, 1992). In this work, we adopt the material point method (MPM) (Sulsky et al., 1995; Jiang et al., 2016; Hu et al., 2018) for its versatility in handling various materials. With MPM, we discretize the domain with Q material points, and $\mathcal{M}_\mu$ becomes Algorithm 1. Here, the state vector $\boldsymbol{s}_n = \{\boldsymbol{x}_n, \boldsymbol{v}_n, \boldsymbol{F}^e_n\}$ contains all the discretized material points' current positions, velocities, and elastic deformation gradients. The time integration scheme $\mathcal{I}$ follows standard MPM practices, derived from the weak form of Eqn. (1). Appendix A provides additional background on MPM.

We highlight that the main contribution of our work is orthogonal to the choice of discretization. The time integration scheme $\mathcal{I}$ can also be replaced by other PDE-governed, weak-form-derived numerical methods, e.g., FEM. Our main contribution is the *continuous* constitutive law, which is required by all numerical methods.

Indeed, the time-stepping scheme involves two constitutive laws (Eqn. (3)):

$$\begin{aligned} \mathcal{E}_\mu &: \boldsymbol{F}^e \mapsto \boldsymbol{P} & \mathcal{P}_\mu &: \boldsymbol{F}^{e,\text{trial}} \mapsto \boldsymbol{F}^{e,\text{new}} \\ &: \mathbb{R}^{3\times3} \to \mathbb{R}^{3\times3} & &: \mathbb{R}^{3\times3} \to \mathbb{R}^{3\times3}. \end{aligned} \tag{3}$$

The elastic constitutive law $\mathcal{E}_\mu$ computes the first Piola-Kirchhoff stresses $\boldsymbol{P}_n$ from the elastic deformation gradients $\boldsymbol{F}^e_n$. The plastic return-mapping scheme $\mathcal{P}_\mu$ (de Souza Neto et al., 2011) projects the trial elastic deformation gradient onto the plastic yield constraint f_Y. Examples of $\mathcal{E}_\mu$ and $\mathcal{P}_\mu$ are listed in Appendix B.

System Identification from Motion Observations Once we have the PDE-governed simulation, we can identify constitutive parameters directly from motion observations (Ma et al., 2022). In particular, we can define the loss function on the ground-truth observed particle positions $\boldsymbol{x}^{\text{gt}}_n$:

$$\min_\mu \mathcal{L}(\{\boldsymbol{x}^{\text{gt}}_n\}^T_{n=0}, \{\boldsymbol{x}_n\}^T_{n=0}), \tag{4}$$

where $\mathcal{L}$ is the loss measure. By implementing the simulation (Algorithm 1) in a differentiable manner, we can employ standard gradient-based optimization techniques to solve Eqn. (4) and optimize for the constitutive parameters that match the motion observations.

3.2. Neural Constitutive Laws

In this work, we replace the classic elastic and plastic constitutive treatments with neural-network-based models: $\mathcal{E}_{\theta_e}$ and $\mathcal{P}_{\theta_p}$, where θ_e and θ_p are the neural network weights. The time-stepping scheme now becomes Algorithm 2. Fig. 1 provides a schematic illustration of our hybrid NN-PDE time-stepping scheme. Notably, we only represent the constitutive laws with neural networks while keeping the classic PDE-based time integration $\mathcal{I}$ unchanged. The time integration scheme $\mathcal{I}$ computes the force from the divergence of stress and updates the particle positions and velocities via standard Euler methods (Ascher & Petzold, 1998). Since these steps follow well-known physical laws, we argue that they do not need to be learned. To train on motion observation data, we also adopt the same objective function as

Algorithm 1 Time-stepping, classic

Input: $\boldsymbol{s}_n = \{\boldsymbol{x}_n, \boldsymbol{v}_n, \boldsymbol{F}^e_n\}$
Output: $\boldsymbol{s}_{n+1} = \{\boldsymbol{x}_{n+1}, \boldsymbol{v}_{n+1}, \boldsymbol{F}^e_{n+1}\}$
1: for $i = 1 \ldots Q$, $\boldsymbol{P}_n(i) = \mathcal{E}_\mu(\boldsymbol{F}^e_n(i))$
2: $\boldsymbol{x}_{n+1}, \boldsymbol{v}_{n+1}, \boldsymbol{F}^{e,\text{trial}}_{n+1} = \mathcal{I}(\boldsymbol{x}_n, \boldsymbol{v}_n, \boldsymbol{F}^e_n, \boldsymbol{P}_n)$
3: for $i = 1 \ldots Q$, $\boldsymbol{F}^e_{n+1}(i) = \mathcal{P}_\mu(\boldsymbol{F}^{e,\text{trial}}_{n+1}(i))$

Algorithm 2 Time-stepping, neural

Input: $\boldsymbol{s}_n = \{\boldsymbol{x}_n, \boldsymbol{v}_n, \boldsymbol{F}^e_n\}$
Output: $\boldsymbol{s}_{n+1} = \{\boldsymbol{x}_{n+1}, \boldsymbol{v}_{n+1}, \boldsymbol{F}^e_{n+1}\}$
1: for $i = 1 \ldots Q$, $\boldsymbol{P}_n(i) = \mathcal{E}_{\theta_e}(\boldsymbol{F}^e_n(i))$
2: $\boldsymbol{x}_{n+1}, \boldsymbol{v}_{n+1}, \boldsymbol{F}^{e,\text{trial}}_{n+1} = \mathcal{I}(\boldsymbol{x}_n, \boldsymbol{v}_n, \boldsymbol{F}^e_n, \boldsymbol{P}_n)$
3: for $i = 1 \ldots Q$, $\boldsymbol{F}^e_{n+1}(i) = \mathcal{P}_{\theta_p}(\boldsymbol{F}^{e,\text{trial}}_{n+1}(i))$

Eqn. (4), but the optimized variable now becomes the neural network weights:

$$\min_{\theta_e,\theta_p} \mathcal{L}(\{\boldsymbol{x}_n^{\text{gt}}\}_{n=0}^T, \{\boldsymbol{x}_n\}_{n=0}^T). \tag{5}$$

We also emphasize that we do not assume a general backbone form for the constitutive law. Assuming a particular backbone (e.g., neo-Hookean elasticity) would prevent us from capturing the constitutive behaviors of another one. Therefore, we opt not to assume any backbone and represent the entire constitutive law with NNs, which are data-driven general representations with better expressiveness and versatility. However, the framework outside constitutive laws can be considered a general PDE-based backbone. In this sense, we share similar philosophy as Yin et al. (2021) by augmenting physical models with NNs.

3.3. Physics-Aware Network Architecture

Our neural constitutive laws must satisfy essential physics priors (Vlassis et al., 2022). Nevertheless, we also avoid being constrained by any overly strong prior, e.g., isotropic prior, which prevents our model from capturing anisotropic materials (Wang et al., 2020). In this work, we consider two generally applicable physical priors.

Rotation Equivariance The properties of most materials in the physical world remain invariable under rotations. This property is also known as frame indifference. Unlike previous works (Deng et al., 2021) focusing on vector data (e.g. point cloud), we have to enforce the rotation equivariance of deformation gradients, which are square matrices. To this end, we feed the neural networks with a selected set of rotation invariants, including the principal stretches $\boldsymbol{\Sigma}$ (singular values), the right Cauchy-Green tensor $\boldsymbol{F}^{eT}\boldsymbol{F}^e$, and the determinant of the elastic deformation gradient $\det(\boldsymbol{F}^e)$. Then, we take the rotation matrix $\boldsymbol{R}$ out of the deformation gradient via polar decomposition $\boldsymbol{F}^e = \boldsymbol{RS}$ and multiply it back to the output of the neural networks to ensure the rotation equivariance. Appendix C provides the detailed algorithm and proofs for rotation invariance of the neural networks and neural constitutive laws.

Undeformed State Equilibrium Another important constitutive prior is preserving the rest shape under zero load. We ensure the undeformed state equilibrium by (1) normalizing the input invariants so that they are zero under zero load and (2) removing all bias layers from the neural networks. Therefore, when zero loads are applied, the elasticity network will generate zero stress, while the plasticity network will preserve the trial elastic deformation gradient.

Table 1. **Reconstruction**. We show the reconstruction losses for different methods. For each environment, we indicate the top 1 with dark green and the top 2 with light green from all the baselines and our method. We also report the statistics (mean ± standard derivation) for each method in the last column.

Method	JELL-O	SAND	PLASTICINE	WATER	Overall
spline	2.4e-1	3.2e-1	3.0e-1	3.2e-1	2.9e-1±3.5e-2
neural	1.2e-5	1.2e-2	1.9e-1	2.9e-2	5.7e-2±8.7e-2
gnn	2.1e-2	1.1e-2	8.7e-3	3.2e-2	1.8e-2±1.0e-2
ours-init	1.1e-1	4.8e-2	4.6e-2	6.8e-2	6.8e-2±2.8e-2
ours	2.4e-4	2.6e-5	6.5e-5	2.0e-5	8.6e-5±1.0e-4
labeled	8.0e-9	7.1e-2	5.5e-6	1.6e-7	1.8e-2±3.5e-2
sys-id	1.7e-8	2.7e-7	5.8e-10	1.7e-8	7.6e-8±1.3e-7

4. Experiments

In this section, we first show our experimental setup (Sec. 4.1). We then compare our methods against baselines and oracles in reconstruction (Sec. 4.2) and generalization (Sec. 4.3). We further study our method in two advanced experiments: the multi-physics environments (Sec. 4.5) and a real-world experiment (Sec. 4.6). We refer the readers to our project website[2] where the results are best illustrated in videos.

4.1. Experimental Setup

Environments We consider four distinct elastoplastic materials covering a diverse set of physical effects: purely elastic (JELL-O), elastic with the Drucker-Prager yield condition (SAND), elastic with the von Mises yield condition (PLASTICINE), and weakly compressible fluids (WATER). We detail the mathematical definition and implementation of these ground-truth material models in Appendix B.

Baselines We compare our method against three strong baselines. **spline** (Xu et al., 2015) models the constitutive laws of the elasticity using Bézier splines, where the control points are the parameters for optimization. **neural** (Wang et al., 2020) builds upon **spline** but replaces the Bézier spline with neural networks. **gnn** (Sanchez-Gonzalez et al., 2020) learns the particle dynamics with graph neural networks without any assumption about underlying PDEs.

Our Methods In addition to our results after a full training (**ours**), we also present the performance of our method at the initial stage (**ours-init**) without training using Eqn. (5) in order to emphasize the efficacy of learning.

Oracles We consider two oracles with extra knowledge that is normally inaccessible. **labeled** (As' ad et al., 2022) utilizes labeled ground truth of $\boldsymbol{P}$ and $\boldsymbol{F}^{e,\text{new}}$ for supervision instead of the motion observations. **sys-id** (Ma et al., 2022)

[2] https://sites.google.com/view/nclaw

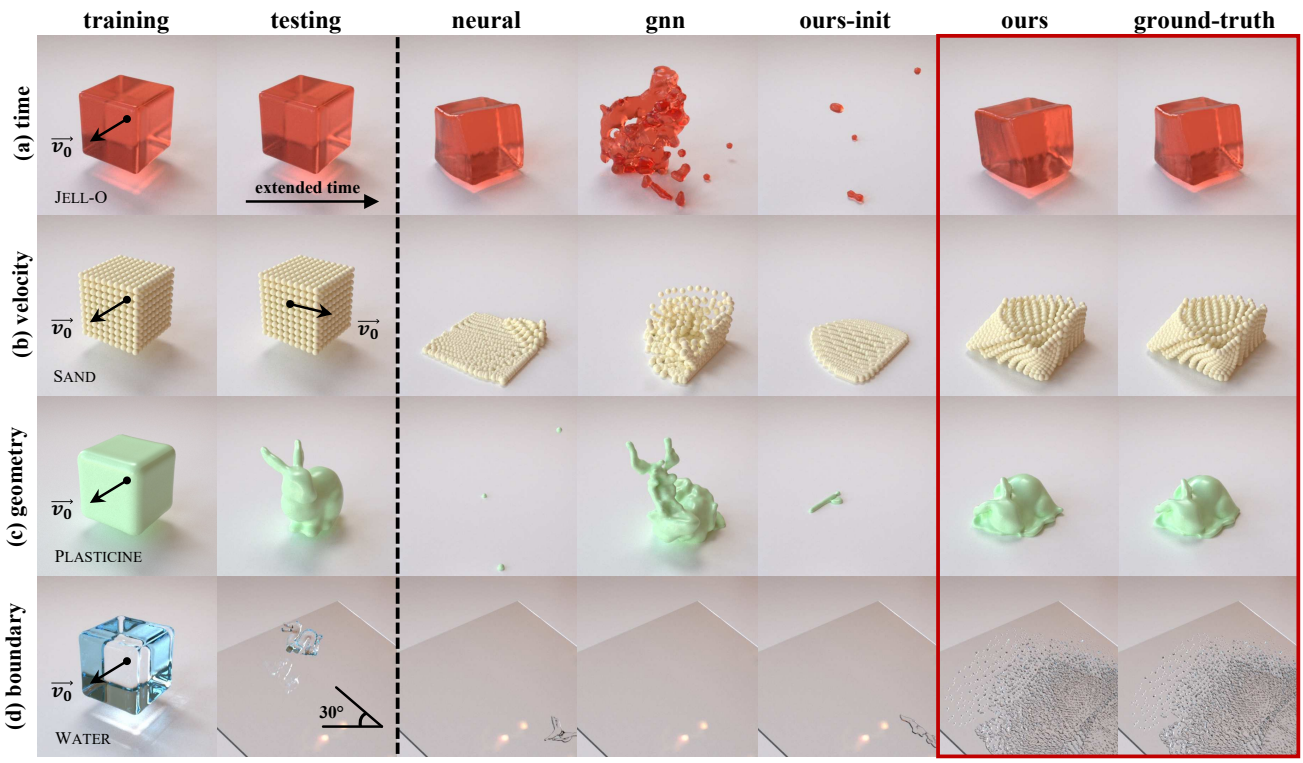


Figure 2. **Generalization**. We first train all methods in the environments specified in **training** with the initial velocities indicated using the black arrows. Then, we evaluate the generalization of all methods on four tasks specified in **testing**: **(a)** extended time, **(b)** unseen initial velocity, **(c)** challenging geometry, and **(d)** inclined boundary. The left side of the dashed line shows the initialization, and the right side shows the simulated results after a period of time. We compare our method (**ours**) against baselines (**neural** and **gnn**), our initialization (**ours-init**), and the ground-truth simulation (**ground-truth**). We highlight **ours** and **ground-truth** with a red box.

assumes a given constitutive model and identifies only the material parameters (e.g., Young's modulus).

Implementation Details For training, we load a cubic object with 1k homogeneous material points at the center of a box with a fixed linear and angular velocity. We simulate the cube's motion for a total of 1k time steps with a step size of 5e-4s. We generate *one single* trajectory for each environment as the training dataset. For our neural networks, we use a 3-layer multilayer perceptron (MLP) with 64 neurons per layer. Our quantitative results report the average mean square error every 5 frames since **gnn** is trained with 5× step size following (Sanchez-Gonzalez et al., 2020). We provide more implementation details in Appendix D.

4.2. Reconstruction

We first train all methods on a single trajectory in each environment to reconstruct the motion sequence. We quantitatively demonstrate the comparison of the reconstruction losses in Tab. 1. We find **spline** ends with constantly poor results in all environments, likely because of the optimization difficulty from discontinuous control point search and the limited expressiveness of the quadratic splines. The other baselines achieve reasonable performances: **neural** reconstructs JELL-O and WATER well but struggles in all other plasticity-heavy environments due to a lack of plasticity models; **gnn** generally has acceptable performance in all environments. In contrast to baselines' unstable or sub-optimal performances, **ours** generally reaches a reconstruction loss lower than 1e-3 and ranks top-1 except for JELL-O where **neural** performs the best. The reason is that **neural** assumes no plasticity, which is precisely the case with JELL-O, and thus eases the training. We also report the initial performance of our method in **ours-init** to emphasize that our method learns from scratch without ad-hoc tuning for individual environments. The two oracles **labeled** and **sys-id** achieve near-perfect results in almost all environments. The only exception is **labeled** performing sub-optimally in SAND, which has a particularly challenging pressure-dependent plasticity law. Because **labeled** is supervised directly from ground-truth $\boldsymbol{P}$ and $\boldsymbol{F}^{e,\text{new}}$, we attribute this performance drop to the lack of back-propagation through time (BPTT). BPTT automatically magnifies the accumulated simulation loss and guides the neural networks to learn long-term stability.

Table 2. **Generalization**. We show the generalization losses for different methods. We report the quantitative results on three main tasks from Fig. 2: **(a)** extended time, **(b)** unseen initial velocity, and **(c)** challenging geometry. Note that we evaluate with three random initial velocities for task **(b)** and report the average loss for more thorough validation. For each task in each environment, we indicate the top 1 with dark green and the top 2 with light green from all the baselines and our method. We also report the statistics (mean ± standard derivation) for each method in the last column.

Method	JELL-O			SAND			PLASTICINE			WATER			Overall
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)	
spline	2.6e-1	2.4e-1	3.5e-1	3.5e-1	3.1e-1	3.6e-1	3.0e-1	3.0e-1	3.1e-1	3.7e-1	3.2e-1	3.3e-1	3.1e-1±4.0e-2
neural	2.9e-5	1.4e-5	5.6e-5	4.7e-2	6.2e-3	2.3e-1	2.4e-1	4.1e-3	2.8e-1	4.6e-2	1.7e-2	5.1e-2	4.9e-2±8.8e-2
gnn	3.3e-2	1.5e-2	1.6e-1	1.5e-2	1.7e-2	3.4e-1	1.1e-2	6.8e-3	3.7e-2	1.1e+0	5.8e-2	2.9e-1	1.2e-1±2.6e-1
ours-init	1.5e-1	6.6e-2	1.2e-1	1.5e-1	2.7e-2	5.0e-2	1.3e-1	2.7e-2	5.8e-2	3.0e-1	4.1e-2	1.0e-1	7.7e-2±6.9e-2
ours	9.8e-4	2.4e-4	4.1e-4	4.2e-5	6.5e-5	3.6e-4	1.4e-4	4.6e-5	2.3e-4	3.5e-4	1.9e-5	2.4e-4	1.9e-4±2.3e-4
labeled	1.1e-7	7.8e-8	1.7e-4	2.2e-1	1.8e-3	3.6e-1	1.5e-5	7.5e-6	1.1e-4	2.1e-6	2.2e-7	1.8e-6	2.9e-2±9.2e-2
sys-id	4.0e-8	6.1e-9	1.1e-8	3.7e-7	4.4e-8	1.6e-7	7.2e-10	1.3e-10	1.4e-9	2.6e-7	7.2e-9	1.4e-8	5.1e-8±9.9e-8

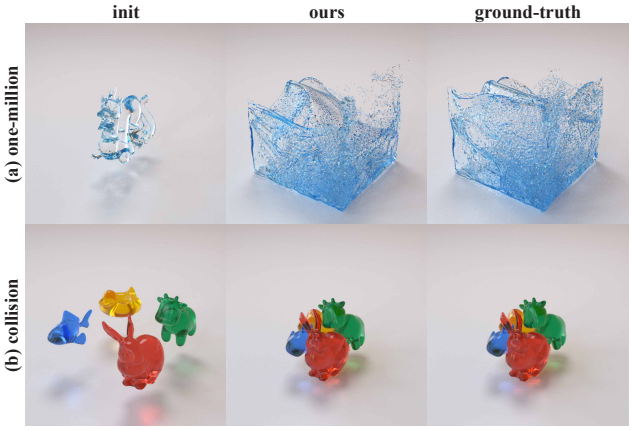


Figure 3. **Extreme Generalization**. We compare the initial state, our method, and the ground-truth results from the traditional simulation in two extreme generalization experiments: **(a)** a complex water dragon loaded with over 1 million particles, **(b)** four rubber toys vigorously colliding with each other and the floor.

4.3. Generalization

After training on a *single* trajectory in Sec. 4.2, we directly deploy the models on four tasks with out-of-distribution conditions to evaluate the generalizability of our method: **(a)** doubled time horizon, **(b)** unseen linear and angular velocity, **(c)** challenging geometry, and **(d)** inclined plane boundary. We also used a more challenging geometry in **(d)** and increased the number of particles in **(c)** and **(d)** to ∼30k (30× more than training). We emphasize that single-shot generalization is a non-trivial task where the training data are only the positions of 1k particles across 1k time steps. We expand the visualization in Appendix F.

We select a representative subset of the experiments and report the qualitative comparison in Fig. 2. **neural** is consistent with the reconstruction performance: generalizes well on JELL-O even when the time horizon is doubled but fails on all other tasks that heavily rely on plasticity. **gnn** cannot generate visually plausible simulation results with different conditions than training environments. We argue that training a generalizable **gnn** requires much more data than ours since it does not assume any knowledge about the underlying PDE. Indeed, Sanchez-Gonzalez et al. (2020) use at least 1k motion trajectories for training on similar setups while our approach uses only one trajectory. For all tasks, **ours** achieves similar visual results compared to **ground-truth**. We also visualize **ours-init** to indicate the significant improvement from random initialization to a trained model.

Tab. 2 quantitatively compares our approach and the baseline methods. Overall, **ours** ranks 1st on most environments and tasks and outperforms other baselines by orders of magnitudes. Consistent with reconstruction, our method has a small enough but worse loss than **neural** on JELL-O. However, as shown in Fig. 2, this gap is so small that it only introduces a negligible visual difference between them.

4.4. Extreme Generalization

To study the limits of our method, we design experiments stress-testing two important factors in physical simulation: the number of particles and the collision condition. We demonsstrate the results in Fig. 3. On the upper row, we adopt the WATER environment and apply a dragon geometry for the initial shape. We increase the number of particles in the simulation to *over 1 million* to stress-test the robustness of our method. As shown in the result, our method generates faithful prediction compared to the ground-truth results from traditional simulation even after a long horizon of time steps. We emphasize that the neural network deployed was trained on only 1k particles. On the lower row, we initialize four rubber toys made of JELL-O and throw them to each other vigorously with a large initial velocity. We depict the moment after the collision happened. Thanks to the disentanglement between the constitutive laws and the rest of the simulation, our method generalizes well to the collision condition even when it is totally unseen during training.

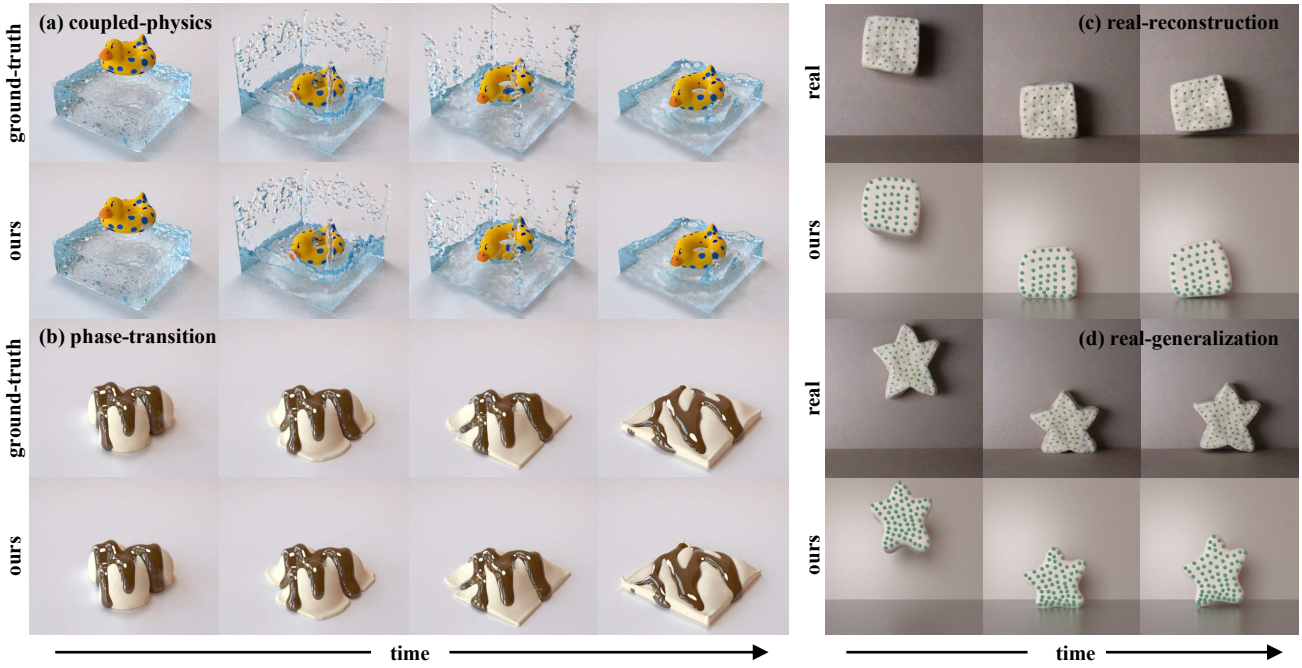


Figure 4. **Advanced Experiments**. Left: We compare our method with ground-truth results from the traditional simulation in two multi-physics environments: **(a)** a rubber duck falling into a swimming pool, and **(b)** a melting ice cream gradually changing from one material to another. Right: We train our method on the real-world data of dough dropping to learn dough's constitutive laws: **(c)** our method successfully reconstructs the training sequence, **(d)** our method generalizes well to *unseen* conditions, e.g., a complex geometry.

4.5. Multi-Physics Generalization

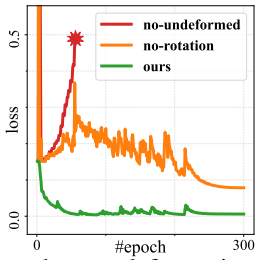
Since we train our model using a single trajectory containing only one material, it is non-trivial for it to predict the material responses in multi-physics environments. We push the boundary of this validation by constructing two challenging multi-physics environments as shown on the left side of Fig. 4. We illustrate the ground-truth motion generated using traditional simulations in the **ground-truth** row and compare it with the prediction of our method in the **ours** row. In Fig. 4 **(a)**, we model the rubber duck using the same material as JELL-O and drop it into a large pool of WATER. Their coupling is handled by MPM. The rubber duck is modeled with 8,086 particles and the water with 94,208 particles, combined to over *100k* particles. The negligible visual difference indicates the accuracy of our method in this challenging large-scale multi-physics environment. In Fig. 4 **(b)**, we initialize an ice cream made of PLASTICINE and gradually transit the constitutive laws to SAND from bottom to top to imitate a "melting" effect. Our method remains stable and accurate even with time-varying phase transition and yields little discrepancy compared to the ground truth.

4.6. Real-World Experiment

Real-world materials usually present complex behaviors which are challenging and tedious to model accurately. To further evaluate our method, we choose a real-world dough as our testing material, which has rich behaviors in both elastic and plastic sides. As shown on the right side of Fig. 4, we record a video clip of the movement of real dough dropped from high. We dotted the dough in green to track the particle-level deformation of the material. We build the simulation directly based on the labeled particles and train our method with supervision from this real-world data. Note that the environment is *truely 3D* in order to reflect the real material property of the dough. To train 3D NCLaw with only 2D supervision, we introduce an additional prior: a regularization loss minimizing the magnitude of velocities along the depth dimension. In Fig. 4 **(c)**, we experimentally show that our method reconstructs the real-world trajectory of dough impacting the ground. After training, we then directly apply the trained model on an *unseen* star-shaped dough made of the same material as shown in Fig. 4 **(d)**. Our method captures the non-trivial elastic and plastic deformation and achieves one-shot generalization on this new complex real-world geometry. Appendix D details our experimental setup and implementation.

4.7. Ablation Study

We then evaluate the role that rotation equivariance and undeformed state equilibrium play in training neural constitutive laws. We design two variants of our method for comparison: **no-undeformed** loosens the undeformed state equilibrium by removing the normalization of neural networks' input and adding back the bias layers. **no-rotation** loosens the rotation equivariance by directly feeding the raw deformation gradients to the neural networks. We train our method and its variants on PLASTICINE using our regular training configuration. As indicated by the training curve plot, our method achieves a better and faster convergence, whereas **no-rotation** is sub-optimal and **no-undeformed** blows up the simulation by introducing unstable constitutive laws.



In addition to the listed experiments, we also compare the runtime performance of our method against **gnn** and provide full visualization of the training dataset and generalization experiments in Appendix F.

5. Conclusions, Limitations, and Future Work

We present NCLaw as a novel hybrid NN-PDE approach toward learning generalizable PDE dynamics from motion observations. Our method enforces PDE constraints throughout the training and deployment stages while taking a data-driven approach toward constitutive modeling. Specifically, we embed NN-based constitutive laws inside a differentiable PDE-governed simulator. We then train the NN model via a loss function based on the difference between the simulator's output and the motion observation. Through a carefully designed network architecture based on physics, NCLaw guarantees common constitutive priors, such as rotational equivariance and rest state equilibrium. After training on a single motion trajectory, NCLaw achieves one-shot generalization over temporal ranges, initial/boundary conditions, complex geometries, and even multi-physics scenarios. Real-world experiments demonstrate NCLaw's ability to learn generalizable dynamics from videos.

In the near future, we aim to extend NCLaw to various physical phenomena, including heterogeneity, hysteresis, and hardening/softening. Enforcing the second law of thermodynamics (Gurtin et al., 2010) in constitutive design is also an exciting direction. In terms of applications, we plan to generalize it to tasks requiring minimal sim-to-real gaps ranging from robot locomotion/manipulation to general control in complex physics (Wang et al., 2023). We also envision extending NCLaw to other PDEs, such as the Reynold stress tensor in turbulence modeling (Ling et al., 2016). We also plan to generalize our framework to other discretization schemes, e.g., FEM. Furthermore, our NN approach can benefit from improved interpretability and facilitating modification after training. Currently, we assume the initial condition, including the geometry, is known. Additionally, we assume that we have motion observations of the entire volumetric object (in 2D, with the entire surface). Future work may consider modeling initial condition uncertainties and working with partial observations. NCLaw can also be extended to learn from pixel-level video data without manual tracking.

At a higher level, NCLaw benefits from *both* classic PDE *and* data-driven approaches. By enforcing classic PDE priors, NCLaw achieves orders-of-magnitude higher accuracy on generalization tasks than purely NN approaches that do not enforce PDE constraints (Sanchez-Gonzalez et al., 2020). Utilizing a single expressive neural architecture, NCLaw can capture many material properties, from solids to fluids, without hand-crafted, case-by-case, expert-designed models. Consequently, we believe NCLaw will open the gates for more forthcoming hybrid NN-PDE, best-of-both-world solutions.

Acknowledgements

We would like to thank Bohan Wang and Minghao Guo for the constructive discussion. This work was supported by MIT-IBM Watson AI Lab. Dr. Gan was supported by DSO grant DSOCO21072, and gift funding from MERL, Cisco, Sony, and Amazon.

References

Abulnaga, S. M., Abaci Turk, E., Bessmeltsev, M., Grant, P. E., Solomon, J., and Golland, P. Placental flattening via volumetric parameterization. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 39–47. Springer, 2019.

Amjid, M. R., Shehzad, A., Hussain, S., Shabbir, M. A., Khan, M. R., Shoaib, M., et al. A comprehensive review on wheat flour dough rheology. *Pakistan Journal of Food Sciences*, 23(2):105–123, 2013.

Arruda, E. M. and Boyce, M. C. A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials. *Journal of the Mechanics and Physics of Solids*, 41(2):389–412, 1993.

As' ad, F., Avery, P., and Farhat, C. A mechanics-informed artificial neural network approach in data-driven constitutive modeling. *International Journal for Numerical Methods in Engineering*, 123(12):2738–2759, 2022.

Ascher, U. M. and Petzold, L. R. *Computer methods for*

ordinary differential equations and differential-algebraic equations, volume 61. Siam, 1998.

Bar-Sinai, Y., Hoyer, S., Hickey, J., and Brenner, M. P. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Barbič, J. and James, D. L. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM transactions on graphics (TOG)*, 24(3):982–990, 2005.

Bishop, A. W. and Henkel, D. J. The measurement of soil properties in the triaxial test. 1962.

Borja, R. I. *Plasticity*, volume 2. Springer, 2013.

Chen, H.-y., Tretschk, E., Stuyck, T., Kadlecek, P., Kavan, L., Vouga, E., and Lassner, C. Virtual elastic objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15827–15837, 2022.

Chen, P. Y., Chantharayukhonthorn, M., Yue, Y., Grinspun, E., and Kamrin, K. Hybrid discrete-continuum modeling of shear localization in granular media. *Journal of the Mechanics and Physics of Solids*, 153:104404, 2021.

Chhabra, R. P. *Bubbles, drops, and particles in non-Newtonian fluids*. CRC press, 2006.

Crane, K. *Keenan's 3D Model Repository*, 2020. URL https://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/.

de Avila Belbute-Peres, F., Smith, K., Allen, K., Tenenbaum, J., and Kolter, J. Z. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.

de Souza Neto, E. A., Peric, D., and Owen, D. R. *Computational methods for plasticity: theory and applications*. John Wiley & Sons, 2011.

Degrave, J., Hermans, M., Dambre, J., et al. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, pp. 6, 2019.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., and Guibas, L. J. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12200–12209, 2021.

Drucker, D. C. and Prager, W. Soil mechanics and plastic analysis or limit design. *Quarterly of applied mathematics*, 10(2):157–165, 1952.

Du, T., Wu, K., Spielberg, A., Matusik, W., Zhu, B., and Sifakis, E. Functional optimization of fluidic devices with differentiable stokes flow. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

Du, T., Hughes, J., Wah, S., Matusik, W., and Rus, D. Underwater soft robot modeling and control with differentiable simulation. *IEEE Robotics and Automation Letters*, 6(3): 4994–5001, 2021a.

Du, T., Wu, K., Ma, P., Wah, S., Spielberg, A., Rus, D., and Matusik, W. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (TOG)*, 41(2):1–21, 2021b.

Ellis, G., Yao, C., Zhao, R., and Penumadu, D. Stress-strain modeling of sands using artificial neural networks. *Journal of geotechnical engineering*, 121(5):429–435, 1995.

Eymard, R., Gallouët, T., and Herbin, R. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

Fuchs, A., Heider, Y., Wang, K., Sun, W., and Kaliske, M. Dnn2: A hyper-parameter reinforcement learning game for self-design of neural network based elasto-plastic constitutive descriptions. *Computers & Structures*, 249: 106505, 2021.

Fung, Y. Elasticity of soft tissues in simple elongation. *American Journal of Physiology-Legacy Content*, 213(6): 1532–1544, 1967.

Furukawa, T. and Yagawa, G. Implicit constitutive modelling for viscoplasticity using neural networks. *International Journal for Numerical Methods in Engineering*, 43 (2):195–219, 1998.

Gao, M., Wang, X., Wu, K., Pradhana, A., Sifakis, E., Yuksel, C., and Jiang, C. Gpu optimization of material point methods. *ACM Transactions on Graphics (TOG)*, 37(6): 1–12, 2018.

Geilinger, M., Hahn, D., Zehnder, J., Bächer, M., Thomaszewski, B., and Coros, S. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

Ghaboussi, J., Garrett Jr, J., and Wu, X. Knowledge-based modeling of material behavior with neural networks. *Journal of engineering mechanics*, 117(1):132–153, 1991.

Gonzalez, O. and Stuart, A. M. *A first course in continuum mechanics*, volume 42. Cambridge University Press, 2008.

Gurtin, M. E., Fried, E., and Anand, L. *The mechanics and thermodynamics of continua*. Cambridge University Press, 2010.

Haberman, R. *Mathematical models: mechanical vibrations, population dynamics, and traffic flow*. SIAM, 1998.

Hahn, D., Banzet, P., Bern, J. M., and Coros, S. Real2sim: Visco-elastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019.

Hencky, H. The elastic behavior of vulcanized rubber. *Rubber Chemistry and Technology*, 6(2):217–224, 1933.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Hu, Y., Fang, Y., Ge, Z., Qu, Z., Zhu, Y., Pradhana, A., and Jiang, C. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4): 1–14, 2018.

Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J., and Durand, F. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019a.

Hu, Y., Liu, J., Spielberg, A., Tenenbaum, J. B., Freeman, W. T., Wu, J., Rus, D., and Matusik, W. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pp. 6265–6271. IEEE, 2019b.

Huang, D. Z., Xu, K., Farhat, C., and Darve, E. Learning constitutive relations from indirect observations using deep neural networks. *Journal of Computational Physics*, 416:109491, 2020.

Huang, Z., Hu, Y., Du, T., Zhou, S., Su, H., Tenenbaum, J. B., and Gan, C. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *ICLR*, 2021.

Hughes, T. J. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

Jiang, C., Schroeder, C., Selle, A., Teran, J., and Stomakhin, A. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.

Jiang, C., Schroeder, C., Teran, J., Stomakhin, A., and Selle, A. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, pp. 1–52. 2016.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klár, G., Gast, T., Pradhana, A., Fu, C., Schroeder, C., Jiang, C., and Teran, J. Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics (TOG)*, 35(4): 1–12, 2016.

Klein, D. K., Fernández, M., Martin, R. J., Neff, P., and Weeger, O. Polyconvex anisotropic hyperelasticity with neural networks. *Journal of the Mechanics and Physics of Solids*, 159:104703, 2022.

Landau, L. D., Bell, J., Kearsley, M., Pitaevskii, L., Lifshitz, E., and Sykes, J. *Electrodynamics of continuous media*, volume 8. elsevier, 2013.

Le, B., Yvonnet, J., and He, Q.-C. Computational homogenization of nonlinear elastic materials using neural networks. *International Journal for Numerical Methods in Engineering*, 104(12):1061–1084, 2015.

Li, X., Cao, Y., Li, M., Yang, Y., Schroeder, C., and Jiang, C. Plasticitynet: Learning to simulate metal, sand, and snow for optimization time integration. *Advances in Neural Information Processing Systems*, 35:27783–27796, 2022.

Li, Z. and Farimani, A. B. Graph neural network-accelerated lagrangian fluid simulation. *Computers & Graphics*, 103: 201–211, 2022.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Liang, J., Lin, M., and Koltun, V. Differentiable cloth simulation for inverse problems. *Advances in Neural Information Processing Systems*, 32, 2019.

Ling, J., Kurzawski, A., and Templeton, J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

Liu, B., Kovachki, N., Li, Z., Azizzadenesheli, K., Anandkumar, A., Stuart, A. M., and Bhattacharya, K. A learning-based multiscale method and its application to inelastic impact problems. *Journal of the Mechanics and Physics of Solids*, 158:104668, 2022.

Liu, M., Liang, L., and Sun, W. A generic physics-informed neural network-based constitutive model for soft biological tissues. *Computer methods in applied mechanics and engineering*, 372:113402, 2020.

Lu, L., Jin, P., and Karniadakis, G. E. Deeponet: Learning nonlinear operators for identifying differential equations

based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Ma, P., Du, T., Zhang, J. Z., Wu, K., Spielberg, A., Katzschmann, R. K., and Matusik, W. Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021.

Ma, P., Du, T., Tenenbaum, J. B., Matusik, W., and Gan, C. Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. *arXiv preprint arXiv:2205.05678*, 2022.

Macklin, M. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, March 2022. NVIDIA GPU Technology Conference (GTC).

Marzec, A., Kowalska, J., Domian, E., Galus, S., Ciurzyńska, A., and Kowalska, H. Characteristics of dough rheology and the structural, mechanical, and sensory properties of sponge cakes with sweeteners. *Molecules*, 26(21):6638, 2021.

Mises, R. v. Mechanik der festen körper im plastisch-deformablen zustand. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1913:582–592, 1913.

Monaghan, J. J. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992.

Murthy, J. K., Macklin, M., Golemo, F., Voleti, V., Petrini, L., Weiss, M., Considine, B., Parent-Lévesque, J., Xie, K., Erleben, K., et al. gradsim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, 2020.

Ogden, R. W. *Non-linear elastic deformations*. Courier Corporation, 1997.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Qiao, Y., Liang, J., Koltun, V., and Lin, M. Differentiable simulation of soft multi-body systems. *Advances in Neural Information Processing Systems*, 34:17123–17135, 2021a.

Qiao, Y.-L., Liang, J., Koltun, V., and Lin, M. C. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pp. 8661–8671. PMLR, 2021b.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Shen, Y., Chandrashekhara, K., Breig, W., and Oliver, L. Finite element analysis of v-ribbed belts using neural network based hyperelastic material model. *International Journal of Non-Linear Mechanics*, 40(6):875–890, 2005.

Stomakhin, A., Howes, R., Schroeder, C. A., and Teran, J. M. Energetically consistent invertible elasticity. In *Symposium on Computer Animation*, volume 1, 2012.

Stomakhin, A., Schroeder, C., Chai, L., Teran, J., and Selle, A. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

Stomakhin, A., Schroeder, C., Jiang, C., Chai, L., Teran, J., and Selle, A. Augmented mpm for phase-change and varied materials. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014.

Sulsky, D., Zhou, S.-J., and Schreyer, H. L. Application of a particle-in-cell method to solid mechanics. *Computer physics communications*, 87(1-2):236–252, 1995.

Sun, X., Bahmani, B., Vlassis, N. N., Sun, W., and Xu, Y. Data-driven discovery of interpretable causal relations for deep learning material laws with uncertainty propagation. *Granular Matter*, 24(1):1–32, 2022.

Tampubolon, A. P., Gast, T., Klár, G., Fu, C., Teran, J., Jiang, C., and Museth, K. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., and Barajas-Solano, D. Learning parameters and constitutive relationships with physics informed deep neural networks. *arXiv preprint arXiv:1808.03398*, 2018.

Thompson, J. O. Hooke's law. *Science*, 64(1656):298–299, 1926.

Treloar, L. The elasticity of a network of long-chain molecules. i. *Transactions of the Faraday Society*, 39: 36–41, 1943.

Tropea, C., Yarin, A. L., Foss, J. F., et al. *Springer handbook of experimental fluid mechanics*, volume 1. Springer, 2007.

Truesdell, C. and Noll, W. The non-linear field theories of mechanics. In *The non-linear field theories of mechanics*, pp. 1–579. Springer, 2004.

Um, K., Brand, R., Fei, Y. R., Holl, P., and Thuerey, N. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

Vlassis, N. N. and Sun, W. Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering*, 377:113695, 2021.

Vlassis, N. N. and Sun, W. Component-based machine learning paradigm for discovering rate-dependent and pressure-sensitive level-set plasticity models. *Journal of Applied Mechanics*, 89(2), 2022a.

Vlassis, N. N. and Sun, W. Geometric deep learning for computational mechanics part ii: Graph embedding for interpretable multiscale plasticity. *arXiv preprint arXiv:2208.00246*, 2022b.

Vlassis, N. N., Ma, R., and Sun, W. Geometric deep learning for computational mechanics part i: anisotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering*, 371:113299, 2020.

Vlassis, N. N., Zhao, P., Ma, R., Sewell, T., and Sun, W. Molecular dynamics inferred transfer learning models for finite-strain hyperelasticity of monoclinic crystals: Sobolev training and validations against physical constraints. *International Journal for Numerical Methods in Engineering*, 2022.

Wang, B., Deng, Y., Kry, P., Ascher, U., Huang, H., and Chen, B. Learning elastic constitutive material and damping models. In *Computer Graphics Forum*, volume 39, pp. 81–91. Wiley Online Library, 2020.

Wang, K. and Sun, W. A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning. *Computer Methods in Applied Mechanics and Engineering*, 334:337–380, 2018.

Wang, T.-H., Ma, P., Spielberg, A. E., Xian, Z., Zhang, H., Tenenbaum, J. B., Rus, D., and Gan, C. Softzoo: A soft robot co-design benchmark for locomotion in diverse environments. In *The Eleventh International Conference on Learning Representations*, 2023.

Xian, Z., Zhu, B., Xu, Z., Tung, H.-Y., Torralba, A., Fragkiadaki, K., and Gan, C. Fluidlab: A differentiable environment for benchmarking complex fluid manipulation. *arXiv preprint arXiv:2303.02346*, 2023.

Xu, H. and Barbič, J. Example-based damping design. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

Xu, H., Sin, F., Zhu, Y., and Barbič, J. Nonlinear material design using principal stretches. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015.

Yin, Y., Le Guen, V., Dona, J., de Bézenac, E., Ayed, I., Thome, N., and Gallinari, P. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124012, 2021.

Yue, Y., Smith, B., Batty, C., Zheng, C., and Grinspun, E. Continuum foam: A material point method for shear-dependent flows. *ACM Transactions on Graphics (TOG)*, 34(5):1–20, 2015.

Yue, Y., Smith, B., Chen, P. Y., Chantharayukhonthorn, M., Kamrin, K., and Grinspun, E. Hybrid grains: Adaptive coupling of discrete and continuum simulations of granular media. *ACM Transactions on Graphics (TOG)*, 37(6): 1–19, 2018.

A. The Material Point Method

This section's goal is to demonstrate how MPM (i.e., the time integration scheme in $\mathcal{I}$ from Algorithm 1) is systematically derived from the governing PDEs. Additional derivation details can be found in the works by Sulsky et al. (1995); Stomakhin et al. (2013); Yue et al. (2015); Jiang et al. (2016; 2015).

To derive MPM, we start with the Eulerian forms for the conservation of mass and conservation of momentum,

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\boldsymbol{v}, \tag{6}$$

$$\rho\boldsymbol{a} = \nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{b}, \tag{7}$$

where ρ is the density, $\boldsymbol{a}$ is the acceleration, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, and $\boldsymbol{b}$ is the body force. Note that mass conservation (Eqn. (6)) does not require special treatments since it is automatically satisfied by advecting Lagrangian MPM particles.

A.1. Weak Form

We obtain the weak form of the equation of motion (Eqn. (7)) by multiplying a test function $\boldsymbol{w}$ on both sides and integrating it over an arbitrary domain Ω:

$$\int_\Omega \rho\boldsymbol{a}\cdot\boldsymbol{w}d\Omega = \int_\Omega(\nabla\cdot\boldsymbol{\sigma})\cdot\boldsymbol{w}d\Omega + \int_\Omega\rho\boldsymbol{b}\cdot\boldsymbol{w}d\Omega, \tag{8}$$

This should be satisfied for any test function $\boldsymbol{w}$ and any domain Ω.

With integration by parts, the equation above becomes

$$\int_\Omega \rho\boldsymbol{a}\cdot\boldsymbol{w}d\Omega = -\int_\Omega\boldsymbol{\sigma}:\nabla\boldsymbol{w}d\Omega + \int_{\partial\Omega_{\boldsymbol{T}}}\boldsymbol{w}\cdot\boldsymbol{T}dS + \int_\Omega\rho\boldsymbol{b}\cdot\boldsymbol{w}d\Omega, \tag{9}$$

where $\boldsymbol{T}$ is the traction and $\partial\Omega_{\boldsymbol{T}}$ is the boundary where traction is applied.

A.2. Spatial Discretization

With two sets of basis functions, one for the material points and the other for the grid, MPM discretizes the weak form spatially and obtains the following discretized system:

$$\sum_{b=1}^{G} M_{ab}\boldsymbol{a}_b = -\sum_{i=1}^{Q} V_i^0\boldsymbol{\tau}_i\nabla N_a(\boldsymbol{x}_i) + \sum_{i=1}^{Q} M_i N_a(\boldsymbol{x}_i)\boldsymbol{b}_i, \tag{10}$$

where $M_{ab} = \sum_{i=1}^{Q} M_i N_a(\boldsymbol{x}_i)\cdot N_b(\boldsymbol{x}_i)$ is the full mass matrix; $V_i^0, M_i, \boldsymbol{\tau}_i, \boldsymbol{x}_i, \boldsymbol{b}_i$ are the initial volume, mass, Kirchhoff stress, current position, the external force of material point i; Q is the number of material points; $N_b, \boldsymbol{a}_b$ are the basis function and acceleration on the Eulerian grid node b; G is the number of Eulerian basis functions. We refer to Sulsky et al. (1995) for additional details on this derivation.

A.3. Temporal Discretization

Using the explicit Euler scheme with a time step size Δt, we can discretize in time,

$$\sum_{b=1}^{G} M_{ab}\frac{\boldsymbol{v}_b^{n+1}-\boldsymbol{v}_b^n}{\Delta t} = -\sum_{i=1}^{Q} V_i^0\boldsymbol{\tau}_i^n\nabla N_a(\boldsymbol{x}_i^n) + \sum_{i=1}^{Q} M_i N_a(\boldsymbol{x}_i^n)\boldsymbol{b}_i^n. \tag{11}$$

A.4. MPM Pseudocode

From these spatial and temporal discretizations, we arrive at the pseudocode in Algorithm 3. We implement this algorithm under the MLS-MPM framework by Hu et al. (2018).

This completes the background on MPM and the time integration scheme in $\mathcal{I}$ from Algorithm 1. Note that neural constitutive laws introduce in our work contribute to time integration in two ways. First, the neural elastic constitutive law $\mathcal{E}_{\theta_e}$ computes the first Piolar-Kirchhoff stress $\boldsymbol{P}$ (equivalently, the Cauchy stress $\boldsymbol{\sigma}$ or the Kirchhoff stress $\boldsymbol{\tau}$). This stress is used for computing the forces of the grid node. Second, the neural plastic constitutive law $\mathcal{P}_{\theta_p}$ post-processes the trial elastic deformation gradient by projecting back to the admissible elastic region.

B. Material Models for Training

Below we list the material models for generating ground truth *training data*. For each material, we list both the elastic constitutive law ($\mathcal{E}_\mu$) and the plastic constitutive law ($\mathcal{P}_\mu$).

We emphasize that our method can capture all these expert-designed classic constitutive models using the same network architecture and the same training strategy.

Purely Elastic Examples of purely elastic materials include rubber, biological tissues, and jelly. We use the fixed corotated hyperelastic model by Stomakhin et al. (2012).

$$\boldsymbol{P} = 2\mu(\boldsymbol{F}-\boldsymbol{R}) + \lambda J(J-1)\boldsymbol{F}^{-T}, \tag{12}$$

where $\boldsymbol{R}$ is the rotation matrix from the polar decomposition of $\boldsymbol{F} = \boldsymbol{RS}$. There is no plasticity in this material. Equivalently, the plastic return mapping is an identity map,

$$\mathcal{P}(\boldsymbol{F}) = \boldsymbol{F}. \tag{13}$$

Elastic with the Drucker-Prager Yield Condition We use the Saint Venant–Kirchhoff elastic model. Computing

Algorithm 3 MPM Algorithm

Input: Position $\boldsymbol{x}_n^i$, velocity $\boldsymbol{v}_n^i$, and elastic deformation gradient $\boldsymbol{F}_n^{e,i}$ of each material point, $i = 1, \ldots, Q$ at time instance t_n

Output: Position $\boldsymbol{x}_{n+1}^i$, velocity $\boldsymbol{v}_{n+1}^i$, trial elastic deformation gradient $\boldsymbol{F}_{n+1}^{e,\text{trial},i}$, $i = 1, \ldots, Q$ at time instance t_{n+1}

1: Transfer Lagrangian kinematics to the Eulerian grid by performing a "particle to grid" transfer: Compute for $b = 1, \ldots, G$

$$
\begin{aligned}
m_{b,n} &= \sum_{i=1}^{Q} N_b(\boldsymbol{x}_n^i) M_i \\
m_{b,n}\boldsymbol{v}_{b,n} &= \sum_{i=1}^{Q} N_b(\boldsymbol{x}_n^i) M_i \boldsymbol{v}_n^i \\
\boldsymbol{f}_{b,n}^{\boldsymbol{\sigma}} &= -\sum_{i=1}^{Q} \frac{J(\boldsymbol{F}_n^{e,i})}{\rho_0} \boldsymbol{\sigma}(\boldsymbol{F}_n^{e,i}) \nabla N_b(\boldsymbol{x}_n^i)\ M_i \\
\boldsymbol{f}_{b,n}^{e} &= \sum_{i=1}^{Q} \frac{J(\boldsymbol{F}_n^{e,i})}{\rho_0} \boldsymbol{b}(\boldsymbol{x}_n^i) N_b(\boldsymbol{x}_n^i)\ M_i
\end{aligned}
$$

2: Solve Eulerian governing equations by computing for $b = 1, \ldots, G$

$$
\begin{aligned}
\dot{\boldsymbol{v}}_{b,n+1} &= \frac{1}{m_{b,n}} (\boldsymbol{f}_{b,n}^{\boldsymbol{\sigma}} + \boldsymbol{f}_{b,n}^{e}) \\
\Delta\boldsymbol{v}_{b,n+1} &= \dot{\boldsymbol{v}}_{b,n+1} \Delta t \\
\boldsymbol{v}_{b,n+1} &= \boldsymbol{v}_{b,n} + \Delta\boldsymbol{v}_{b,n+1}
\end{aligned}
$$

3: Update the Lagrangian velocity and deformation gradient by performing a "grid to particle" transfer: Compute for $i = 1, \ldots, Q$

$$
\begin{aligned}
\boldsymbol{v}_{n+1}^i &= \sum_{i=1}^{G} N_i(\boldsymbol{x}_n^i) \boldsymbol{v}_{b,n+1} \\
\boldsymbol{F}_{n+1}^{e,\text{trial},i} &= (\boldsymbol{I} + \sum_{i=1}^{G} \boldsymbol{v}_{i,n+1} \otimes \nabla N_i(\boldsymbol{x}_n^p) \Delta t) \boldsymbol{F}_n^{e,i}
\end{aligned}
$$

4: Update Lagrangian positions for $i = 1, \ldots, Q$

$$
\boldsymbol{x}_{n+1}^i = \boldsymbol{x}_n^i + \Delta t \boldsymbol{v}_{n+1}^i
$$

the singular value decomposition of $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and the Hency strain $\boldsymbol{\epsilon} = \log(\boldsymbol{\Sigma})$, we have (Barbič & James, 2005)

$$
\boldsymbol{P} = \boldsymbol{U}(2\mu\boldsymbol{\epsilon} + \lambda\,\mathsf{tr}(\boldsymbol{\epsilon}))\boldsymbol{U}^T. \tag{14}
$$

In addition, we apply the Drucker-Prager yield condition (Drucker & Prager, 1952; Klár et al., 2016; Yue et al., 2018; Chen et al., 2021):

$$
\mathsf{tr}(\boldsymbol{\epsilon}) > 0 \quad \text{or} \quad \delta\gamma = \|\hat{\boldsymbol{\epsilon}}\| + \alpha \frac{(3\lambda + 2\mu)\,\mathsf{tr}(\boldsymbol{\epsilon})}{2\mu} > 0, \tag{15}
$$

where $\alpha = \sqrt{\frac{2}{3}} \frac{2\sin\theta}{3-\sin\theta}$ and θ is the friction angle of the granular media.

With this yield criterion, we then define two types of plastic projection. When the material undergoes tension, we project the stress unto the cone tip; When the material undergoes compression and violates the cone constraint, we project the stress onto the cone in an isochoric manner. Mathematically, we have

$$
\mathcal{P}(\boldsymbol{F}) = \begin{cases} \boldsymbol{U}\boldsymbol{V}^T & \mathsf{tr}(\boldsymbol{\epsilon}) > 0, \\ \boldsymbol{F} & \mathsf{tr}(\boldsymbol{\epsilon}) \le 0 \ \ \& \ \ \delta\gamma \le 0, \\ \boldsymbol{U}\exp(\epsilon - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|})\boldsymbol{V}^T & \mathsf{tr}(\boldsymbol{\epsilon}) \le 0 \ \ \& \ \ \delta\gamma > 0. \end{cases} \tag{16}
$$

Elastic with the von Mises Yield Condition We adopt the same Saint Venant–Kirchhoff elastic model detailed previously,

$$
\boldsymbol{P} = \boldsymbol{U}(2\mu\boldsymbol{\epsilon} + \lambda\,\mathsf{tr}(\boldsymbol{\epsilon}))\boldsymbol{U}^T. \tag{17}
$$

However, in this case, we adopt the von Mises yield condition (Mises, 1913; Hu et al., 2018; Huang et al., 2021),

$$
\delta\gamma = \|\hat{\boldsymbol{\epsilon}}\| - \frac{\tau_Y}{2\mu}, \tag{18}
$$

where $\boldsymbol{\epsilon}$ is the normalized Hencky strain computed from the deformation gradient $\boldsymbol{F}$. τ_Y is the yield stress and describes how easily the material undergoes the plastic flow. A positive $\delta\gamma$, i.e., $\delta\gamma > 0$, suggests the material violates the yield constraint. For stress that violates the yield criterion, we will project it back into the elastic region via an isochoric (volume-preserving) projection:

$$
\mathcal{P}(\boldsymbol{F}) = \begin{cases} \boldsymbol{F} & \delta\gamma \le 0, \\ \boldsymbol{U}\exp(\epsilon - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|})\boldsymbol{V}^T & \delta\gamma > 0. \end{cases} \tag{19}
$$

Weakly Compressible Fluids Following Stomakhin et al. (2014); Tampubolon et al. (2017), we model fluids using the aforementioned fixed corotated elastic model with $\mu = 0$, effectively modeling the fluid with no shearing resistance while penalizing any volume change,

$$
\boldsymbol{P} = \lambda J(J-1)\boldsymbol{F}^{-T}. \tag{20}
$$

In addition, we adopt the plasticity model by Stomakhin et al. (2014); Gao et al. (2018),

$$
\mathcal{P}(\boldsymbol{F}) = J^{\frac{1}{3}}\boldsymbol{I}. \tag{21}
$$

Algorithm 4 Neural Constitutive Laws

Input: $\boldsymbol{F}$ ($\boldsymbol{F}^e$ for elasticity; $\boldsymbol{F}^{e,\text{trial}}$ for plasticity)
Output: $\boldsymbol{P}$ or $\boldsymbol{F}^{e,\text{new}}$

$\boldsymbol{F} \overset{\text{SVD}}{=} \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ // singular value decomposition
$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^T$ // rotation equivariant
$\boldsymbol{T}_1 = \text{NN}(\boldsymbol{\Sigma}, \boldsymbol{F}^T\boldsymbol{F}, \det(\boldsymbol{F}))$ // rotation invariant
$\boldsymbol{T}_2 = \frac{1}{2}(\boldsymbol{T}_1 + \boldsymbol{T}_1^T)$ // ensure symmetry
$\boldsymbol{Y} = \boldsymbol{R}\boldsymbol{T}_2$ // ensure rotation equivariance
if elasticity **then**
  $\boldsymbol{P} = \boldsymbol{Y}$
end if
if plasticity **then**
  $\boldsymbol{F}^{e,\text{new}} = \boldsymbol{F} + \alpha\boldsymbol{Y}$
end if

Effectively, this flow rule projects the deformation gradient onto the hydrostatic axis in an isochoric manner.

In this work, we focus on exploring the aforementioned four types of materials. We leave it as future work to explore other elastoplastic materials, e.g., non-Newtonian fluids (Yue et al., 2015).

C. Algorithm of Neural Constitutive Laws

C.1. Algorithm

We demonstrate the algorithm pipeline in Algorithm 4. Note that we can easily implement polar decomposition that $\boldsymbol{F} = \boldsymbol{R}\boldsymbol{S}$ by singular value decomposition (SVD) that $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and get $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^T$ and $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T$. We use a highly-optimized 3×3 SVD implementation on GPUs (Gao et al., 2018). We use $\alpha = 0.001$ in all our experiments.

C.2. Proof of Rotation Invariance

Proof. Our network takes three inputs, we prove they are rotation-invariant with respect to an arbitrary rotation $\boldsymbol{R}^* \in \text{SO}(3)$ one by one:

1. The singular values $\boldsymbol{\Sigma}$. We have:

$$\begin{aligned}\boldsymbol{F}_{\text{new}} &= \boldsymbol{R}^*\boldsymbol{F} && (22)\\ &= \boldsymbol{R}^*\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T && (23)\\ &= (\boldsymbol{R}^*\boldsymbol{U})\boldsymbol{\Sigma}\boldsymbol{V}^T, && (24)\end{aligned}$$

   where $\boldsymbol{R}^*\boldsymbol{U}$ is still a unitary matrix because both $\boldsymbol{R}^*$ and $\boldsymbol{U}$ are unitary matrices. As a result, when $\boldsymbol{R}^*$ applied, $\boldsymbol{\Sigma}$ is invariant.

2. The right Cauchy-Green tensor $\boldsymbol{F}^T\boldsymbol{F}$. We have:

$$\begin{aligned}\boldsymbol{F}_{\text{new}}^T\boldsymbol{F}_{\text{new}} &= (\boldsymbol{R}^*\boldsymbol{F})^T(\boldsymbol{R}^*\boldsymbol{F}) && (25)\\ &= \boldsymbol{F}^T(\boldsymbol{R}^*)^T\boldsymbol{R}^*\boldsymbol{F} && (26)\\ &= \boldsymbol{F}^T\boldsymbol{F}. && (27)\end{aligned}$$

3. The determinant of the deformation gradient $\det(\boldsymbol{F})$. We have:

$$\begin{aligned}\det(\boldsymbol{F}_{new}) &= \det(\boldsymbol{R}^*\boldsymbol{F}) && (28)\\ &= \det(\boldsymbol{R}^*)\det(\boldsymbol{F}) && (29)\\ &= \det(\boldsymbol{F}), && (30)\end{aligned}$$

   because the determinant of a unitary matrix is 1.

As a result, all inputs to the neural networks are rotation invariant, which implies the output of the neural networks is rotation invariant. We also have that $\text{NN}(\boldsymbol{R}^*\boldsymbol{F}) = \text{NN}(\boldsymbol{F})$, which makes both $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ rotation invariant since they solely depend on the output of neural networks. □

C.3. Proof of Rotation Equivariance

Proof. With an arbitrary rotation $\boldsymbol{R}^* \in \text{SO}(3)$ applied, the input to neural constitutive laws will be transformed into $\boldsymbol{R}^*\boldsymbol{F}$. By the definition of polar decomposition:

$$\boldsymbol{F} \overset{\text{PD}}{=} \boldsymbol{R}\boldsymbol{S} \quad (31)$$

$$\boldsymbol{R}^*\boldsymbol{F} = \boldsymbol{F}_{\text{new}} \overset{\text{PD}}{=} \boldsymbol{R}_{\text{new}}\boldsymbol{S} = (\boldsymbol{R}^*\boldsymbol{R})\boldsymbol{S}. \quad (32)$$

As a result, given $\boldsymbol{F}_{\text{new}}$ as the input, the output of neural constitutive laws is

$$\boldsymbol{Y}_{\text{new}} = \boldsymbol{R}_{\text{new}}\boldsymbol{T}_2 = \boldsymbol{R}^*\boldsymbol{R}\boldsymbol{T}_2 = \boldsymbol{R}^*\boldsymbol{Y}, \quad (33)$$

which proves the rotation equivariance of $\boldsymbol{Y}$. For elasticity, the rotation equivariance automatically inherits. For plasticity, we have:

$$\begin{aligned}\boldsymbol{F}_{\text{new}}^{e,\text{new}} &= \boldsymbol{F}_{\text{new}}^{e,\text{trial}} + \alpha\boldsymbol{Y}_{\text{new}} && (34)\\ &= \boldsymbol{R}^*\boldsymbol{F}^{e,\text{trial}} + \alpha\boldsymbol{R}^*\boldsymbol{Y} && (35)\\ &= \boldsymbol{R}^*(\boldsymbol{F}^{e,\text{trial}} + \alpha\boldsymbol{Y}) && (36)\\ &= \boldsymbol{R}^*\boldsymbol{F}^{e,\text{new}}, && (37)\end{aligned}$$

which proves the rotation equivariance of neural plasticity constitutive laws. □

D. Implementation Details

D.1. Simulation

For each training environment, we load material points inside a 0.5^3 m cube. We set the grids in the MPM simulator

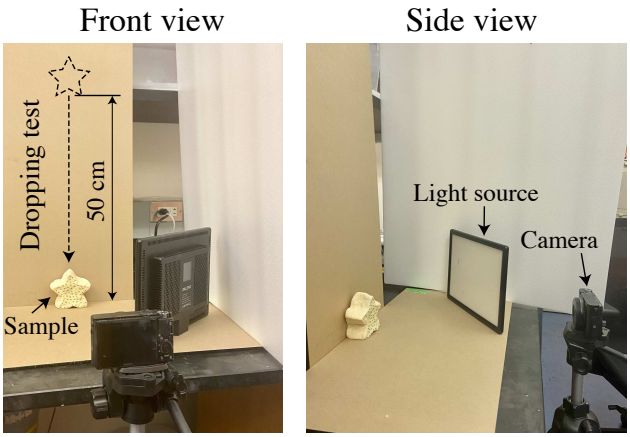


Figure 5. **Real Experiment**. We illustrate the front and side view of our experiment setup. We indicate the dropping position of the dough with a dashed outline and the direction with a dashed arrow.

to be $20 \times 20 \times 20$ to speed up the simulation. We use a threshold of 3 grids to detect the collision and set a free-slip boundary condition within a 1.0^3 m box. Our simulation always applies a gravity of -9.8 m/s^2. We uniformly use the same hyper-parameters for MPM simulation for all our training environments. We implement our differentiable MPM simulator on GPUs using Warp (Macklin, 2022).

D.2. Network and Training

Our neural constitutive laws contain two neural networks with equal sizes each for elasticity and plasticity, a total of 11,008 parameters. The neural networks use GELU (Hendrycks & Gimpel, 2016) as non-linearity and contain no normalization layers. We use the Adam optimizer (Kingma & Ba, 2014) with learning rates of 1.0 and 0.1 for elasticity and plasticity, respectively. We train the neural constitutive laws for 300 epochs and decay the learning rates of both elasticity and plasticity using a cosine annealing scheduler. We also clip the norm of gradients to a maximum of 0.1. We also utilize a "teacher-forcing" scheme that restarts from the ground-truth position periodically. We increase the period of teacher forcing from 25 to 200 steps by a cosine annealing scheduler. We train all our experiments on one NVIDIA RTX A6000. All experiments share the same training configurations and hyper-parameters without ad-hoc tuning.

We implement the neural networks using PyTorch (Paszke et al., 2019), which can share CUDA memory with Warp in order to reduce the overhead interacting with the simulator.

D.3. Real-World Experiment

We use dough as our testing material. Dough is a malleable and elastic paste made from a mixture of flour and water. Dough is famous for its complex and hard-to-predict mechanical properties (Marzec et al., 2021; Amjid et al., 2013) and, therefore, a good test for our algorithm. Our dough is fabricated by mixing 125 g flour with 75 g water and kneading for around 5 minutes. A variety of 2D shapes are then formed out of the dough for training and testing purposes.

In physical experiments (See Fig. 5), the dough is dropped from a height of 50 cm and collides with a rigid surface. This process is recorded by a high-speed camera (SONY RX100) at 960 frames per second with an exposure time of 1/4000 second. The dough is marked with around 50 green dots on the front surface, which are later tracked to provide precise information about the deformation history.

E. Extra Discussion

E.1. Generalization to Mesh-Based Simulation

Among all components in the physical simulation, our method only replaces the manually tuned constitutive laws with a carefully designed neural network, which is a data-driven representation with better expressiveness, uniformity, and versatility. Thus, neural constitutive laws are agnostic to discretization and simulation frameworks. In the current work, we select the material point method (MPM) as the backbone due to its compatibility with various materials, easing our implementation. That said, our framework is general and discretization-independent. We expect that our method also applies to mesh-based or hybrid mesh/particle simulation or any discretization framework. However, it will require significant engineering work to implement the supporting differentiable simulation for mesh and mesh/particle coupling, and we believe such an endeavor would be best served by its own dedicated manuscript.

F. Extra Experiments

F.1. Runtime Comparison

We evaluate the runtime performance of our method and compare it with baselines in Tab. 3. We conduct this experiment by running for a certain period so that the simulation covers 0.5s in the real world. Because of the difference in time step size, it takes 1k steps for **ours** and analytical, while it takes 200 steps for **gnn**. We rerun the same trajectory 10 times and take the average to reduce the variance. Even though **gnn** takes much larger steps, we find our method run at a comparable speed with **gnn**. We suspect the reason is (1) we develop all components of our method on GPUs, and (2) the radius graph network is time-consuming. Since we did not focus on runtime optimization, there is still room for our method to improve, including (1) removing redundant SVD in elasticity and plasticity, (2) reducing the communication between Warp and PyTorch, and (3) replacing PyTorch with

Table 3. **Runtime comparison**.

Method	JELL-O	SAND	PLASTICINE	WATER
ours	2.56s	2.55s	2.62s	2.56s
gnn	2.35s	2.10s	2.19s	2.26s

Table 4. **More recent comparison**.

Task	ours	gnn	Li & Farimani (2022)
reconstruction	**6.5e-5**	8.7e-3	9.3e-4
time	**1.4e-4**	1.1e-2	5.2e-3
velocity	**4.6e-5**	6.8e-3	1.0e-3
geometry	**2.3e-4**	3.7e-2	4.7e-2

Table 5. **Parameter identification**.

Term	ours	labeled
$\boldsymbol{P}$	6.1e-1	**2.6e-3**
$\boldsymbol{F}^{e,\text{new}}$	2.388e-6	**2.381e-6**

Table 6. **Data efficiency**.

Method	#Trajectories	Loss
gnn	1	5.8e-2
gnn	10	2.5e-2
gnn	100	1.2e-2
ours	1	**1.9e-5**

light-weight neural network libraries in deployment.

F.2. More Recent Comparison

We select one of the related previous works (Li & Farimani, 2022) and evaluate it on a subset of our experiments. We use PLASTICINE as the environment and report the performance comparison on four tasks as shown in Tab. 4. According to the results, Li & Farimani (2022) is better than the original **gnn** baseline in our manuscript on 3 out of 4 tasks. However, our method is still stronger than these GNN-based methods by orders of magnitude. There are two potential reasons: (1) our method incorporates physics-aware network architectures ensuring rotation equivariance and undeformed state equilibrium, and (2) our method disentangles the learning of constitutive laws from the simulation framework and promotes data efficiency.

F.3. Parameter Identification

We generate a trajectory using WATER and randomly select 1k deformation gradients as the input to the neural networks. We measure the mean squared errors (MSE) of **ours** and **labeled** versus the ground truth and report the results in Tab. 5. Note that the error of $\boldsymbol{P}$ is naturally larger than deformation gradients because of the physical meaning of the stress tensor (often orders of magnitude larger than the deformation gradient). As shown in the table, our method achieves a similar result in $\boldsymbol{F}^{e,\text{new}}$ but falls behind in $\boldsymbol{P}$ compared to **labeled**. There are a few reasons behind it. First, **labeled** actually uses extra supervision of ground-truth $\boldsymbol{P}$ and $\boldsymbol{F}^{e,\text{new}}$, which are not easily accessible in the real world. Also, we train our method by back-propagation through time (BPTT). By contrast, **labeled** employs traditional training without time stepping. Our approach risks the training efficacy to some extent but is more natural to apply to training time-dependent physical systems. Finally, in continuum mechanics, different constitutive laws may describe the same kinematics (e.g., $\boldsymbol{F}$) with different stresses (e.g., $\boldsymbol{P}$). The goal of our method is to recover the kinematics ($\boldsymbol{x}$, $\boldsymbol{F}$) while being generalizable to new scenarios.

Despite not being as accurate as **labeled** when it comes to $\boldsymbol{P}$, our method predicts precise, generalizable kinematics. Our approach is also more stable than **labeled** across all environments, as shown in Tab. 1.

F.4. Data Efficiency

For the **gnn**, we keep as much as possible the training details in Sanchez-Gonzalez et al. (2020). Due to the loss of implementation details in their papers, we adapt the training procedure for spline and neural to be compatible with our method. In order to ablate the number of training trajectories to study how the amount of data affects the performance, we select **gnn**, which is the baseline with the strongest reliance on data, as the backbone and redo the training with 10 and 100 data trajectories. We present the loss comparison using the WATER environment on the velocity generalization task, where we randomly sample 3 initial velocities and average the losses between the evaluation and the ground truth in Tab. 6. The gradually decreasing loss with respect to the growing number of trajectories indicates **gnn** generalizes better with more data. However, even with 100x fewer data, **ours** still outperforms **gnn** by orders of magnitude, reflecting the efficiency of our method.

F.5. Auxiliary Loss

Here we study the impact of another important state variable: velocity. We additionally add a penalty on the dissimilarity between simulated velocities and the ground-truth velocities. We show the loss comparison of position using WATER with or without velocity dissimilarity penalty in Tab. 7. As indicated by the results, 3 out of 4 tasks slightly benefit from the velocity dissimilarity penalty. We argue that incorporating complementary losses could help the training to some extent, but it might be enough for a quick start to train with

Table 7. **Auxiliary loss**.

Task	w/o vel loss	w/ vel loss
reconstruction	2.0e-5	**1.3e-5**
time	3.5e-4	**2.1e-4**
velocity	1.9e-5	**1.8e-5**
geometry	**2.4e-4**	3.0e-4

only positional supervision, which is usually the handiest ground truth in the real world.

F.6. More Visualization

- We visualize the training dataset in Fig. 6.
- We compare the generalizability of our method with baselines and ground truth about different geometries in Figs. 7 and 8.

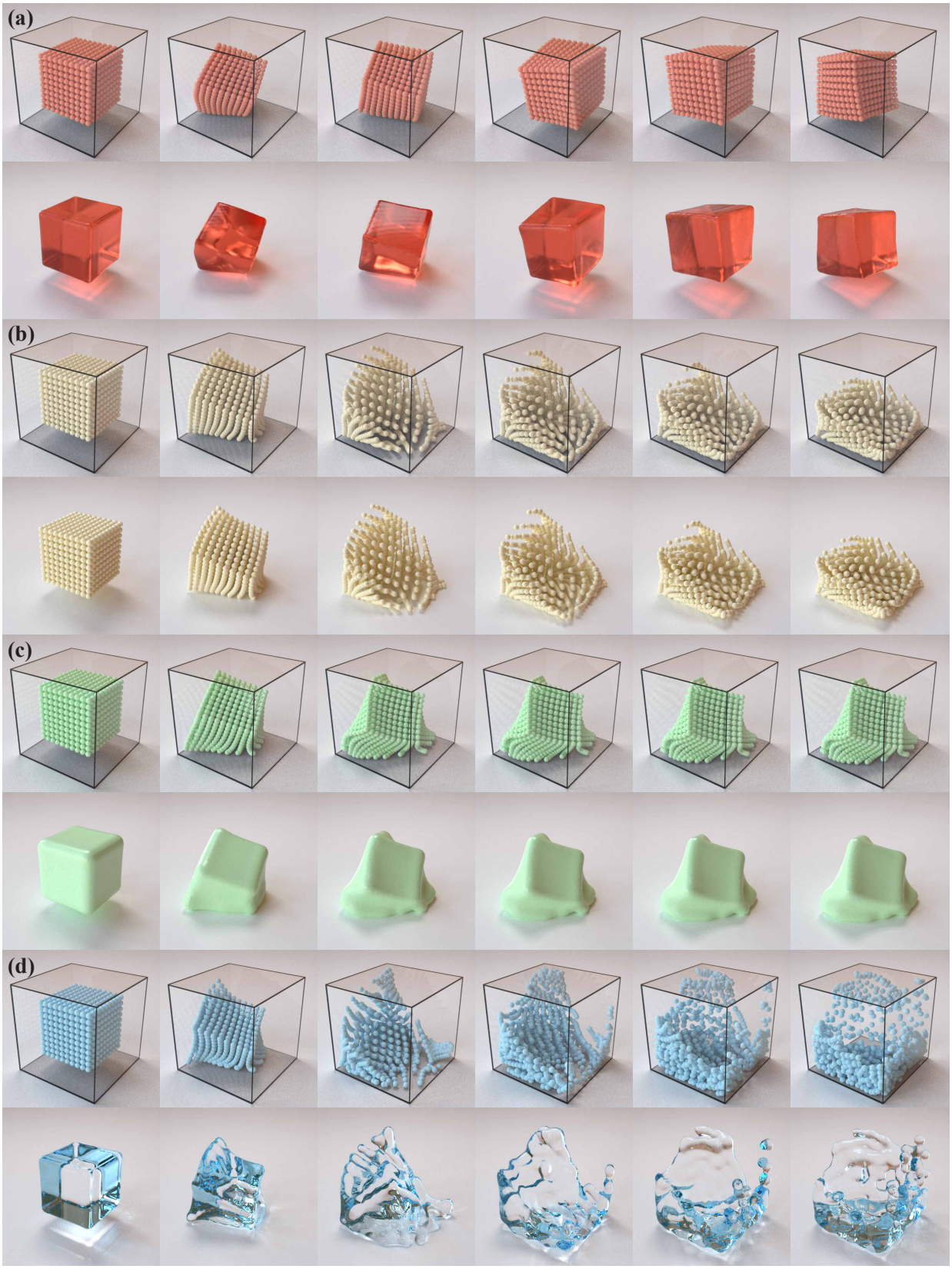


Figure 6. **Training Data**. We show granular and realistic rendering for **(a)** JELL-O, **(b)** SAND, **(c)** PLASTICINE, and **(d)** WATER.

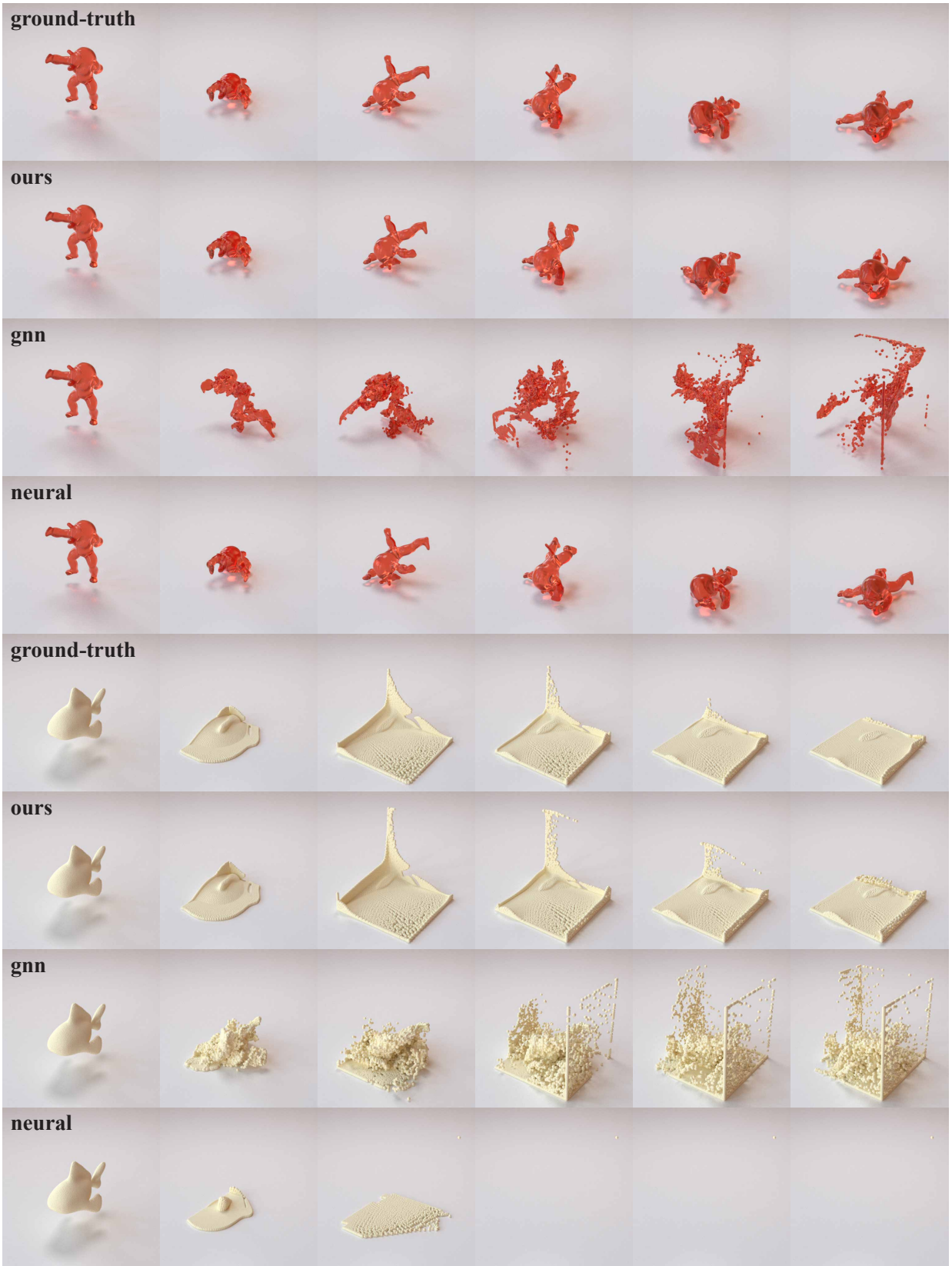


Figure 7. **Geometry Generalization for** JELL-O **and** SAND. We set an armadillo geometry for JELL-O and a goldfish (Crane, 2020) geometry for SAND. We compare **ours** with **ground-truth** and baselines including **gnn** and **neural**.

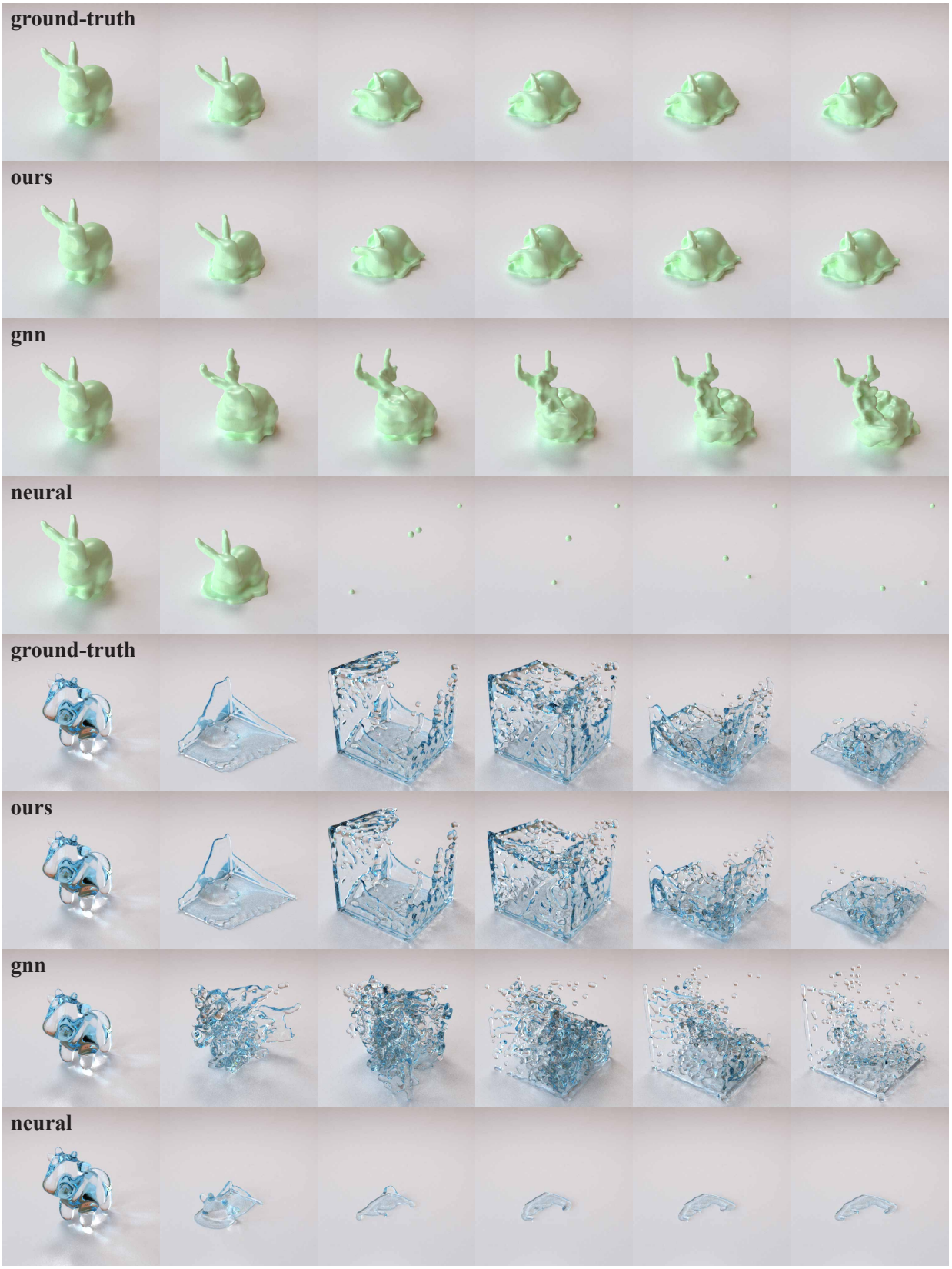


Figure 8. **Geometry Generalization for** PLASTICINE **and** WATER. We set a bunny geometry for PLASTICINE and a cow (Crane, 2020) geometry for WATER. We compare **ours** with **ground-truth** and baselines including **gnn** and **neural**.