Graph network for simultaneous learning of forward and inverse physics

Sakthi Kumar Arul Prakash¹, Conrad Tucker¹,²,³,⁴,⁵

¹Department of Mechanical Engineering, Carnegie Mellon University
²Department of Machine Learning, Carnegie Mellon University
³The Robotics Institute, Carnegie Mellon University
⁴CyLab Security and Privacy Institute, Carnegie Mellon University
⁵Department of Biomedical Engineering, Carnegie Mellon University

5000 Forbes Avenue, Pittsburgh, Pennsylvania 15213
sarulpra@andrew.cmu.edu, conradt@andrew.cmu.edu

Abstract

In this work, we propose an end-to-end graph network that learns forward and inverse models of particle-based physics using interpretable inductive biases. Physics-informed neural networks are often engineered to solve specific problems through problem-specific regularization and loss functions. Such explicit learning biases the network to learn data-specific patterns and may require a change in the loss function or neural network architecture hereby limiting their generalizability. While recent studies have proposed graph networks to study forward dynamics, they rely on particle specific parameters such as mass, etc. to approximate the dynamics of the system. Our graph network is implicitly biased by learning to solve several tasks, thereby sharing representations between tasks in order to learn the forward dynamics as well as infer the probability distribution of unknown particle specific properties. We evaluate our approach on one-step next state prediction tasks across diverse datasets that feature different particle interactions. Our comparison against related data-driven physics learning approaches reveals that our model is able to predict the forward dynamics with at least an order of magnitude higher accuracy. We also show that our approach is able to recover multi-modal probability distributions of unknown physical parameters using orders of magnitude fewer samples.

1 Introduction

Across science and engineering, particle states often contain system and particle specific properties such as mass, density, velocity, particle type, etc. that are required to approximate the dynamics of a system. Depending on the problem domain and the required level of approximation, the dynamics of a system can be complex. However, on an abstract level, it is possible to decompose the dynamics of the system on the basis of particle interactions within their neighborhoods (Battaglia et al. 2016; Sanchez-Gonzalez et al. 2020) and neural network compositions (Hu et al. 2017; Ye et al. 2018). In the recent years, there has been a growing interest in learning physics with the help of deep learning coupled with other techniques such as inductive biases, physics informed loss functions and meta-learning (Wu et al. 2015; Fragiadaki et al. 2016; Battaglia et al. 2016; Xu et al. 2019; Hall et al. 2021). In general, such relational networks can be used to pose the following fundamental questions:

(1) Given the current state of a system of particles along with particle specific local properties, global system level properties, can the next state(s) of the system be predicted? For example, predicting the next state of a system of particles subject to Newtonian forces (Sanchez-Gonzalez et al. 2018, 2020) (2) Given the current state of a system of particles and/or the next/previous state(s), can system specific and particle specific properties be inferred? For example, inferring the unknown properties of objects in a scene or capacitor design (Wu et al. 2016; Pilozzi et al. 2018; Hall et al. 2021). The first question, often referred to as the forward problem, assumes knowledge about the physical properties of particles and therefore utilizes the observations to construct a suitable model that predicts the next state. The second question assumes that a suitable model already exists, and makes use of the known state of the particles to probabilistically estimate their unknown physical properties thereby allowing us to identify the particles along with the nature of their interaction. While it is evident that the nature of the questions are mutually dependent, so are their learning (Kubricht, Holyoak, and Lu 2017).

Depending on the complexity of the problem, the availability of an accurate physics simulator and the amount of data, a model that solves either or both of the above problems can include domain knowledge in various forms. Physics-informed neural networks explicitly include domain knowledge in the form of problem specific loss functions, often requiring diverse data obtained by varying all the properties that govern the dynamics of the problems which may be difficult to obtain (Raissi, Perdikaris, and Karniadakis 2019; Obiols-Sales et al. 2020; Zeng et al. 2020). Data-driven models do not use custom loss functions and may implicitly induce biases to discover the dynamics of the system (Battaglia, Hamrick, and Tenenbaum 2013; Battaglia et al. 2016; Sanchez-Gonzalez et al. 2018). However, such models require more data than the explicit learning counterparts in order to achieve comparable accuracy. Finally, there are hybrid models that combine explicit and implicit biasing strategies such as fusing differentiable physics simulators with neural networks, or incorporating physics intuition as implicit bias (Sanchez-Gonzalez et al. 2020; Silver et al. 2021).

This work introduces a hybrid graph network that jointly learns the forward dynamics and also infers the unknown
physical properties. We decompose Newtonian physics models into deterministic and probabilistic sub-components. We represent each of the sub-components as learnable graph networks; a Forward model using deterministic graph auto-encoders/decoders and an Inverse model using a differentiable probabilistic graphical model. The sub-components are composed together to form a single graph network. Together, they learn a shared encoded representation of the particles and their interactions using a supervised learning objective. While the inverse model uses the shared encoded representation as a learned prior to infer the distribution of the values of an unknown particle specific parameter, the forward model uses the same encoded representation to predict the next state of a system of particles. Apart from predicting the next state and particle specific unknown physical parameter, we explore other implicit biasing strategies in this paper such as particle classification and contact prediction. We hypothesize that composing inverse and forward models along with implicit learning tasks, induces a structural bias in the overall learning of problem dynamics, thereby avoiding the need for problem specific loss functions while using orders of magnitude fewer samples. We validate our model by evaluating on several particle systems that differ in the physics being learned. Our key contributions are summarized as follows:

• A differentiable multi-task approach to simultaneously learn the forward and inverse physics models
• An efficient inverse model that uses orders of magnitude fewer samples to estimate an arbitrary unknown physical parameter distribution

2 Related Work
2.1 Learning Physics using Domain Knowledge
Off-the-shelf physics simulators are implemented using analytical models that guarantee extrapolation to new samples. However, a caveat with analytical physics models is that they may not always yield numerically stable solutions, especially if the problem under consideration deals with complex interactions (e.g., discontinuity, collisions). To mitigate such challenges, constraints are often relaxed (Catto 2009). While learned simulators that use deep learning implementations offer simplicity and greatly favors rapid development (Obiols-Sales et al. 2020; Lu et al. 2021), the vast majority of deep learning techniques lack robustness and fail to provide guarantees of convergence (Raissi, Perdikaris, and Karniadakis 2019).

Recently, differentiable particle-based simulators that avoid the need for black-box models have emerged (Hu et al. 2020; Schoenholz and Cubuk 2020). Such models preserve interpretability along with comparable computation and performance gains. However, end-to-end differentiable simulators may not be readily available for many problems of interest without making data dependent approximations (Cranmer et al. 2020b). Domain knowledge can also be encoded as physics priors in the form of loss function regularizations (Raissi, Perdikaris, and Karniadakis 2019) and data featurization (Meidani and Farmani 2021) in order to aid deep learning models extrapolate to new samples. Hybrid models have been proposed where data-driven approaches augment differentiable particle-based simulators at a fine-grained level to model parts of the dynamics unaccounted for by the analytic models (Hwangbo et al. 2019; Zeng et al. 2020). Depending on the size of the dataset and the nature of the problem, the contribution of explicit physics knowledge and physics-informed loss functions can often be tuned accordingly (Raissi, Perdikaris, and Karniadakis 2019; Obiols-Sales et al. 2020; Zeng et al. 2020). At the outset, these approaches rely on explicit domain knowledge or problem-dependent loss functions along with diverse data requirements that may limit their generalizability.

2.2 Learning Physics using Graphs
Engineering simulations developed for prediction and control of complex physical systems can be built based on empirical or theoretical findings. Compared to analytical models, a learned simulator can be more efficient at predicting complex phenomena (He et al. 2019; Sanchez-Gonzalez et al. 2020). While explicitly providing neural networks with physics knowledge can limit generalizability, implicit knowledge can be leveraged by such networks to understand object-object interactions and dependencies (Battaglia, Hamrick, and Tenenbaum 2013; Kubricht, Holyoak, and Lu 2017). Graphs are mathematical structures that model dependence between objects, imparting a dependence graph on the data that can hence induce structural bias on the learning process and enables knowledge-sharing and relational reasoning between tasks (Battaglia, Hamrick, and Tenenbaum 2013; Battaglia et al. 2016; Sanchez-Gonzalez et al. 2018; 2020; Silver et al. 2021). Previous works in the domain of intuitive physics learning (Battaglia, Hamrick, and Tenenbaum 2013; Fragkiadaki et al. 2016; Wu et al. 2016; Chang et al. 2017) have gained inspiration from cognitive sciences (Teglas et al. 2011; Tenenbaum et al. 2011) with the help of graphical models. Graphical models enable learning from high-dimensional data such as images or videos, thereby emulating mental models analogous to those humans develop from an early age to learn real-life physics. Recently, Graph Neural Networks have been proposed as alternative general learning models (Sanchez-Gonzalez et al. 2020; Cranmer et al. 2020b) that impose structure on the data in a non-Euclidean way as well as on the learning process (message-passing) in comparison to vision (Euclidean data) based models, thereby generalizing the notion of inductive bias.

Probabilistic graphical models offer ways to integrate symbolic reasoning and statistical inference to formalize correlation and causality in data (Battaglia, Hamrick, and Tenenbaum 2013; Cranmer et al. 2020a,b; Altosaar 2020). In parameter identification, optimization-based approaches are often utilized to obtain point estimates of the simulation parameters (Mahnken 2004; Hahn et al. 2019) by minimizing the model error from the dynamics of a real system. While a Monte-Carlo approach may accurately approximate the distribution, it can prove to be slow and computationally expensive. Other learning models such as Galileo (Wu et al. 2015) use a probabilistically inverted physics engine to recover physical parameters of objects from scenes by ini-
tializing MCMC (Markov chain Monte Carlo) methods with bottom-up predictions about objects and their physical properties. More recent approaches (Zheng et al. 2018; Cramer et al. 2020b) fit latent vectors that encode ground truth physical properties to principal components of actual ground-truth values. However, such latent properties are often sampled from a simple distribution such as log-normal with a single mode or a uniform distribution where every sample can be a mode. While a large number of samples are often required to perform efficient probabilistic inference, learnable approximators have greatly progressed to use few samples to approximate intractable distributions in a relatively short duration to MCMC while capturing the multi-modality of parameters of interest (Kingma and Welling 2014; Kingma et al. 2016; Rezende and Mohamed 2015; Huang et al. 2018).

For these reasons, the solution of an inverse problem can be difficult to acquire. Yet, a neural network model employed for learning the forward dynamics of physics simulations typically performs thousands of forward passes to compute only the next state. However, such a learned simulator always has access to the current state, the future state (in a supervised setting) and the rich hidden layer encodings that map the states. In this work, we present a flow-based latent variable model to infer the distribution of an unknown physical parameter using a learned prior that contains information pertaining to the current state and the next state. While a flow-based approach serves as our inverse model, we learn thee forward dynamics using a residual graph network. The following section will explain the proposed approach in detail.

3 Approach

We propose a model that improves the expressiveness of current Graph Network architectures as learnable simulators that predict the next state of a system of particles and also infers the probability distribution of unknown physical parameters.

3.1 Learned Simulator Overview

Let \( \Omega \) be a continuous real space of all possible states a system of particles (\( P \)) can exist in at any given time \( t \). Consequently, since \( P^t \in \Omega \), the current state (\( P^t_i \in P^t \)) of each particle \( i \) describes a position vector (\( \mathbf{x}_t^i \in \mathbb{R}^N \)) and a velocity vector (\( \mathbf{v}_t^i \in \mathbb{R}^N \)) such that \( \mathbf{P}^t_i = [\mathbf{x}_t^i, \mathbf{v}_t^i, \mathbf{x}_{t-k}^i, \ldots, \mathbf{x}_{t-k}^i] \), where \( N \) represents the dimensionality of the problem and \( k \) is a hyper-parameter that denotes the number of previous velocity vectors to consider in the state representation of each particle. Since we adopt a particle-based representation of the physical system, we consider particles as nodes (\( V \)) in a graph (\( G \)) such that every particle has a local first-order neighborhood as well as hops of higher-order neighborhoods. Such an explicit graph structure further imposes inductive bias that minimizes the dependence of distant particles from target particles. Hence, every system state \( P^t \in \Omega \) has a corresponding adjacency matrix (\( A^t \)) which is a matrix representation of \( G^t = (V, E^t) \). The number of particles \( |V| \) stays a constant throughout the simulation since we assume that there is no new addition or removal of particles. In contrast, \( E^t \)--the edges of the graph, may vary as a function of time and hence the superscript \( t \).

A forward simulator \( S : P^t \mapsto P^{t+1} \), the next state is predicted as, \( P^{t+1}_i = S(P^t_i) \). We are interested in learning the dynamics of the system using a parameterized function approximator (\( S_{\theta} \)) as a surrogate to an actual physics simulator (\( S \)) which can typically be complex. In addition to learning a forward simulator, we learn unknown physical properties of particles such as mass by using a parameterizable latent variable model \( f_\phi \) that approximates the probability density function of a random variable \( Z \) that represents the unknown property of interest pertaining to all particle types from the simulation distribution. In this work, we assume that each particle has only one unknown physical parameter that needs to be estimated. The learnable parameter \( \phi \) is optimized using a variational inference training objective. Given multiple learning tasks that are related, we frame the problem to minimize multi-task learning objectives using the knowledge contained from all the tasks. \( S_{\theta} \) corresponds to four steps-- Encoding, Processing, Inverse parameter estimation and Decoding. The architecture of the proposed learned simulator is shown in Fig. 1.

3.2 Encoder definition and design

Following the established notations, we define a particle Encoder \( \mathcal{E}_V : [A, P^t] \mapsto H^t_V \) where \( H^t_V \) is an encoded particle representation. Next, we define a relation Encoder \( \mathcal{E}_E : [A, P^t_i - P^t_j] \mapsto H^t_E \) where \( (P^t_i - P^t_j) \) represents the relative position and velocity information between particles, and \( H^t_E \) is an encoded edge representation of the system of particles. The relative matrix \( (P^t_i - P^t_j) \) is a tensor of size \( N \times (k + 1) \times |V| \times |V| \). The computation graph of the absolute state encoder (\( \mathcal{E}_V \)) is made up of raw-residual graph convolution layers that have shown to prevent information smoothing (Zhang and Meng 2019) in deeper layers while capturing node specific graph structural features.

While prior approaches have used particle type as input to the Graph Networks, we consider learning particle types as one of the tasks. By mapping the input state to particle type, the encoding layers learn discriminative particle specific representations which summarize a patch of the graph centered around a particle \( i \). Such discriminative representations have helped downstream tasks such as node classification (Kipf and Welling 2017; Prakash and Tucker 2021) which is also what \( \mathcal{E}_V \) computes. A single layer of \( \mathcal{E}_V \) can mathematically be expressed as follows,

\[
H^{t+1}_V = g(H^t_V) = \sigma(\hat{A} H^t_V W^t_V + \hat{A}^t H^t_V)
\]

where, \( g(\cdot) \) is a learnable raw-residual graph convolution layer, \( \sigma(\cdot) \) is a non-linear activation function \( H^{t=0}_V = P^t \), \( H^t_V \) represents \( t \)th hidden layer, such that \( l = [1 \ldots L] \) and \( \hat{A} = A^t + I \), where \( I \) is an identity matrix.

Concurrently, while the computation graph of the relative state encoder (\( \mathcal{E}_E \)) is made up of similar hidden layers, it captures the edge specific graph structural features (Wang et al. 2020).
et al. (2019) that encode complex information such as contact, collision, friction, etc. We assume that complex phenomenon such as contact, collision or friction between particles can be summarized as a $|V| \times |V|$ matrix that denotes whether a direct contact has occurred using a binary matrix in the case of direct contact and a weighted matrix in case of indirect contact such as the influence of gravitational force. Hence, in order to create dependence between the other channels, we apply a $1 \times 1$ convolution layer on $(\mathbf{P}_i - \mathbf{P}_j)^t$ to transform it to a $|V| \times |V|$ matrix that is compatible with $g(\cdot)$ in order to compute $\mathbf{H}_E$.

### 3.3 Processor definition and design

The processor is an integral component in the proposed learnable simulator that biases edge and particle features through several tasks such as particle classification, contact detection, composing a learned prior for the inverse model and communicating with the forward simulation decoder. While we do not provide a theoretical description on the kinds of tasks that can be performed, we consider some example tasks and discuss their contribution in implicit learning. We briefly describe several tasks the processor performs in this paper. It takes the output of the encoder ($\mathcal{E}_V$) as input to predict the particle type ($\hat{Y}_i$) for all particles in the system as given by Eq.(2)

$$\hat{Y}_i = \alpha(H^L_V W^L)$$  \hspace{1cm} (2)

where, $\alpha$ can be a categorical softmax activation in case of several particle types or a sigmoid activation function in case of two particle types. The processor trains the classifier to minimize the following cross-entropy loss,$$
\mathcal{L}_\text{classification} = - \sum_{i \in V}[Y_i \log \hat{Y}_i] \text{ in a supervised setting.}
$$

Once the particle type has been identified, the current latent particle representations ($\mathbf{H}_E$) are updated as $\tilde{\mathbf{H}}_V = \mathbf{H}_E \| \hat{Y}_i$, where $\|$ indicates concatenation. The updated representation $\tilde{\mathbf{H}}_V$ will also serve as input to the inverse model.

As an example of another possible task, the processor takes the output of the encoder ($\mathcal{E}_E$) as input to a Decoder $\mathcal{D}_\theta : \mathcal{E}_E \rightarrow \mathbf{C}$ where $\mathbf{C}$ is a contact matrix that estimates contact magnitude between particles at time $t$. The contact matrix $\mathbf{C}^t$ induces an inductive bias on the feature matrix $\mathbf{H}_V$ such that the updated representations capture absolute and relative information. $\mathcal{D}_\theta$ is a simple inner-product decoder that can be represented as follows,

$$\hat{C}^t = \mathbf{H}_E^L (\mathbf{H}_E^L)^T$$ \hspace{1cm} (3)

The contact detector module within the processor is trained to reconstruct a matrix by minimizing the following mean squared error,$$
\mathcal{L}_\text{collision} = \frac{1}{|V|} \sum_{i=1}^{|V|} \|\mathbf{C}_t^i - \hat{\mathbf{C}}_t^i\|^2 \hspace{1cm} (4)
$$

The purpose of $\mathcal{E}_E$ and $\mathcal{D}_E$ is to emulate contact estimation that several Newtonian physics models possess (Cant,
of computing the posterior log-likelihood of \( \tilde{\mathcal{C}} \). The normalizing flows approach \cite{rezende2015variational,glatt2019flow} such as the posterior predictive or the unknown physical parameter as a particle samples. A potential work-around is to use an expressive model to approximate an over-smoothed picture of the distribution, a potential issue when the flows in order to quickly sample the unknown physical parameter from the posterior. To accomplish this we use flow layers proposed by \cite{huang2020bayesian} that have shown to be expressive in recovering all modes of an unknown density function. While the flow direction implicitly fits a reverse KL divergence, the learning objective as given by Eq. (8) minimizes the forward KL divergence objective as follows:

\[
\mathcal{L}_{KL}(\tilde{\mathcal{C}}||p_{z_K}) = \mathbb{E}_{m\sim\tilde{\mathcal{M}}} [\log \tilde{\mathcal{M}}] - \mathbb{E}_{m\sim\tilde{\mathcal{M}}} [\log p(z_K)]
\]

and are then further passed through several linear layers to predict the relative masses of all particles. Thus the final output of the decoder of the inverse model \( D_{\theta_{inv}} : Z_K \rightarrow \mathbb{R} \) where \( \mathbb{R} \) is a reconstruction of the relative physical parameter matrix \( \mathbb{R} \) that can be expressed using the following expression,

\[
\hat{\mathbb{R}} = D_{\theta_{inv}}((\sigma((Z_K^T \mathbb{W}_{\text{inv}})Z_K))W_{\text{inv}}_{\text{Dec}})W_{\text{inv}}_{\text{Dec}}
\]

We minimize the reconstruction loss \( \mathcal{L}_{\text{Dec}} = \frac{1}{\mathcal{V}} \sum_{i=1}^{\mathcal{V}} \| R_i - \hat{R}_i \|_2^2 \) along with the inverse model encoder loss \( \mathcal{L}_{KL}(\tilde{\mathcal{M}}||p_{z_K}) \) such that maximizes the following log-likelihood,

\[
\log p_{\theta}(\mathbb{R}) = -\mathcal{L}_{KL}(\tilde{\mathcal{M}}||p_{z_K}) + \mathbb{E}_{m\sim\tilde{\mathcal{M}}} [\log p(|z_K; \theta)]
\]

Finally, we put together the combined loss function from all the steps as follows:

\[
\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Inverse}} + \mathcal{L}_{\text{Dec}} + \mathcal{L}_{\text{collision}} + \mathcal{L}_{\text{classification}}
\]

where, \( \mathcal{L}_{\text{Inverse}} = -\log p_{\theta}(\mathbb{R}) \).
verse set of experiments. In order to evaluate the rigorosity of the proposed approach, we keep a number of hyper-parameter choices such as the number of hidden layers, hidden neurons, etc. constant (please refer to appendix for more details) and further compare with related work (Sanchez-Gonzalez et al. 2020; Cranmer et al. 2020b).

4.1 Dataset
Our datasets are simulations of common Newtonian dynamics that describes the dynamics of particles according to Newton’s law of motion. The motion of particles are predicted using their complex interactions with neighboring particles that cause a change in their position, velocity and accelerations. We evaluate our approach on simulations that involve direct contact namely, elastic rigid body collisions, and indirect contacts wherein particles are subject to spring and gravitational forces. The elastic rigid-body simulation was written using numpy and scipy libraries while the other simulations that deal with spring and gravitational particle interactions (Cranmer et al. 2020b) were written using the JAX library. The variable parameters of these analytic simulation models include the number of simulations, number of particles, time-steps and frame-rate (step-size). Each simulation per analytic model is integrated over 1000 time-steps which is typically the simulation length used by related work such as (Cranmer et al. 2020b; Sanchez-Gonzalez et al. 2020), and our dataset comprises of 10 simulations per analytic model. We set the number of particles = 200 in the case of 2D elastic rigid-body collisions and a standard size of 4 in the case of 2D spring and gravitational force simulations.

4.2 Experiments and Discussion
Simulators that employ explicit and semi-implicit methods of integration for updating particle states are error bounded based on step-size or frame rate. We evaluate the forward and inverse estimation performance of our proposed approach across varying frame rates. Frame-rate or also referred to as step-size is an important parameter in physics simulations since they directly influence the stability of the simulator as well as the accuracy of the simulator’s estimations (Arnold 2001; Lei and Hongzhou 2012). We evaluate Cranmer et al. (Cranmer et al. 2020b) and Sanchez-Gonzalez et al. models on simulator specific hand-tuned frame rates. Further, we train and test their approaches on $10^5$ samples as reported by the authors in order to present a fair comparison. We also investigate how our approach is able to estimate an unknown physical parameter’s probability distribution - the mass distribution, when the frame rate and the complexity of the distribution varies. We consider 5 frame rates ranging from 10-200 as a factor and compare our model performance on all the datasets and against related work.

Performance of forward model  The graph network architectures proposed by Cranmer et al. (Cranmer et al. 2020b) and Sanchez-Gonzalez et al. (Sanchez-Gonzalez et al. 2020), hereon referred to as baseline 1 (BS 1) and baseline 2 (BS 2) respectively differ in the design of their encoder and processor. While their approaches are designed to predict the forward dynamics, we note that they are a special case of our approach such that removing the classification step, contact decoder and the inverse decoder from our approach tends to theirs. We consider them as baseline models to predict the forward dynamics. We find that BS 2 predicts the forward dynamics of elastic rigid-body collisions with an order of magnitude higher accuracy than BS 1. BS 2 features a parallel (assuming unshared parameters) particle and edge model that computes hidden layer representation of particle and edge features that are concatenated to predict the next state of the system of particles. BS 1 on the other hand features a serial graph network wherein the edge features are transformed to hidden representations and then concatenated with raw particle features to predict the next state. As a consequence of this distinction, we find that when there are multiple particle type interactions, BS 2 predicts the next state with approximately an order of magnitude higher accuracy than BS 1. This can be observed by comparing their forward prediction errors on the 2D elastic collisions dataset. However, in the absence of multiple particle types, we find that BS 1 performs an order of magnitude better than BS 2. This difference shows that in the absence of particle types, encoded edge features play a dominant role in forward dynamics prediction.

In an effort to unify current work such that they are agnostic to the presence of particle types and various particle interactions, we explore the following implicit biasing strategies. The processor in graph network models usually contain rich particle and edge information that we leverage by learning to predict 1) the contact matrix of particles and 2) particle type. In physics simulations involving direct elastic contacts (collisions) between particles, every particle continues moving with their current velocity and acceleration until they experience a collision. We detect collisions between particles by computing a sigmoid activation on the elements of the contact matrix ($C^t$). A value of 1 indicates collision between two particles while a value of 0 indicates absence of collision. We compute the activated sigmoid matrix and concatenate it with node-level features to then predict the next state. This computation introduces an inductive bias in the case of direct contact between elastic rigid-body particles that satisfies an implicit assumption that rigid-body simulators follow. From Table. 1 (2D elastic collisions), it is evident that implicit biasing improves the performance of predicting the next state of the particles across varying frame rate. Additionally, we compare the receiver operating characteristics score of the learned simulator across the various frame rates and find that the node classification accuracy is close to 96% and remains consistent with change in frame-rates. In the absence of multiple particle-types and a system of directly colliding particles, we directly append raw particle features with relative particle features ($C^t$). The 2D gravity with varying edge potential along with 2D spring like edge interaction experiments are several examples of systems that exhibit such phenomenon.

Behavior of the inverse model  Across our datasets, the unknown physical parameter of interest is the problem spe-
cific absolute particle mass. Each of the datasets pertaining to an analytic simulation model differs in particle mass distribution with varying complexity as shown by the ground-truth mass distribution. Figs. [2] and [3] show the ground-truth kernel density estimation of the mass distribution of particles along with their approximations. We observe that our inverse model has captured the modes (0.001 and 0.4 in Fig. [2]) as well as the standard deviation of particle mass distributions with a high probability. We perform a t-test to compare the average values of the ground-truth mass distribution with the estimated mass distribution and we find with a p-value > 0.8 that the estimation of the inverse model and the ground-truth are not statistically significant across the datasets. While elastic rigid-body collision between particles comprises of a balanced dataset such that two modes that have similar density, particle masses subject to spring and gravitational forces are sampled from an imbalanced mass distribution with several low probability modes. While the approximation as shown by Fig. [3] covers the target mode, the approximation is much smoother and certain low probability regions have been assigned more support while some high probability regions have been assigned less support. We perform additional analysis (please refer to appendix for more details) wherein we compute the $R^2$ value of the relative mass prediction using samples sampled from the posterior distribution of the relative mass distribution.

Finally, we note that we have used at least 3 orders of magnitude fewer samples in comparison to the other works that we have bench-marked in this paper [Cranmer et al., 2020b; Sanchez-Gonzalez et al., 2020]. Similarly, compared with [Zheng et al., 2018], our graph-based approach requires at least 1 order of magnitude fewer samples.

### Figure 2: Mass distribution of particles subject to elastic rigid-body collisions. [left: KDE of ground-truth mass distribution, right: KDE of approximate mass distribution]

### Figure 3: Mass distribution of particles subject to spring-force. [left: KDE of ground-truth mass distribution, right: KDE of approximate mass distribution]

### 5 Conclusion

Currently, learnable physics simulators make use of explicit physics knowledge in the form of problem specific loss functions or implicit knowledge in the form of physics intuition and graph message-passing networks. In an effort to improve the functionality of existing graph networks we propose a data-driven graph network to simultaneously predict the forward state and the unknown physical parameter of a system of particles using multi-task learning objectives. We evaluate our approach on a diverse set of Newtonian physics models that simulate various interactions between particles. We explore biasing strategies such as particle classification and contact estimation, two related tasks that induce implicit learning of forward dynamics and inverse model. We find that our biasing strategies greatly improve the learning of forward dynamics while successfully inferring the probability distribution of an unknown particle specific physical parameter with orders of magnitude fewer samples. As future work, the proposed inverse model can be extended to discover multivariate as well as multiple particle-specific physical parameters. Finally, we highlight that our approach has considered an unweighted multi-task objective that equally gives importance to all tasks. This may not be ideal since all tasks may not carry equal importance and we speculate that additional objective function regularization might help the model converge to a better solution.
Acknowledgments
References

Altosaar, J. 2020. Probabilistic Modeling of Structure in Science: Statistical Physics to Recommender Systems. Academic dissertations (ph.d.), Princeton University.

Arnold, D. N. 2001. A concise introduction to numerical analysis. Institute for Mathematics and its Applications, Minneapolis.

Battaglia, P.; Pascanu, R.; Lai, M.; Rezende, D.; and Kavukcuoglu, K. 2016. Interaction networks for learning about objects, relations and physics. Advances in Neural Information Processing Systems, 4509–4517.

Catto, E. 2009. Modeling and solving constraints. In Game Developers Conference, 16.

Cant, R.; Langensiepen, C.; and Foster, L. 2008. A composite convex hull collision detector for the open dynamics engine. In Tenth International Conference on Computer Modeling and Simulation (uksim 2008), 122–125. IEEE. ISBN 0769531148.

Catto, E. 2009. Modeling and solving constraints. In Game Developers Conference, 16.

Chang, M.; Ullman, T.; Torralba, A.; and Tenenbaum, J. B. 2017. A Compositional Object-Based Approach to Learning Physical Dynamics. In 5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.

Cranmer, M.; Greydanus, S.; Hoyer, S.; Battaglia, P.; Spergel, D.; and Ho, S. 2020a. Lagrangian neural networks. arXiv, 1–9.

Cranmer, M. D.; Sanchez-Gonzalez, A.; Battaglia, P. W.; Xu, R.; Cranmer, K.; Spergel, D. N.; and Ho, S. 2020b. Discovering Symbolic Models from Deep Learning with Inductive Biases. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.-F.; and Lin, H.-T., eds., Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Fragniadaki, K.; Agrawal, P.; Levine, S.; and Malik, J. 2016. Learning Visual Predictive Models of Physics for Playing Billiards. In Bengio, Y.; and LeCun, Y., eds., 4th International Conference on Learning Representations, {ICLR} 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.

Glatt, M.; Kull, D.; Ravani, B.; and Aurich, J. C. 2019. Validation of a physics engine for the simulation of material flows in cyber-physical production systems. Procedia CIRP, 81: 494–499.

Hahn, D.; Banzet, P.; Bern, J. M.; and Coros, S. 2019. Real2Sim: Visco-elastic parameter estimation from dynamic motion. ACM Transactions on Graphics (TOG), 38(6): 1–13.

Hall, E. J.; Taverniers, S.; Katsoulakis, M. A.; and Taratkovsky, D. M. 2021. GINNs: Graph-Informed Neural Networks for multiscale physics. Journal of Computational Physics, 433: 1–20.

He, S.; Li, Y.; Feng, Y.; Ho, S.; Ravabakhsh, S.; Chen, W.; and Póczos, B. 2019. Learning to predict the cosmological structure formation. Proceedings of the National Academy of Sciences, 116(28): 13825–13832.

Hu, R.; Andreas, J.; Rohrbach, M.; Darrell, T.; and Saenko, K. 2017. Learning to reason: End-to-end module networks for visual question answering. In Proceedings of the IEEE International Conference on Computer Vision, 804–813.

Hu, Y.; Anderson, L.; Li, T.-M.; Sun, Q.; Carr, N.; Ragan-Kelley, J.; and Durand, F. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. In 8th International Conference on Learning Representations, {ICLR} 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Huang, C.-W.; Krueger, D.; Lacoste, A.; and Courville, A. C. 2018. Neural Autoregressive Flows. In Dy, J. G.; and Krause, A., eds., Proceedings of the 35th International Conference on Machine Learning, {ICML} 2018, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, 2083–2092. PMLR.

Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsouannis, V.; Koltun, V.; and Hutter, M. 2019. Learning agile and dynamic motor skills for legged robots. Science Robotics, 4(26).

Kingma, D. P.; Salimans, T.; Józefowicz, R.; Chen, X.; Sutskever, I.; and Welling, M. 2016. Improving Variational Autoencoders with Inverse Autoregressive Flow. In Lee, D. D.; Sugiyama, M.; von Luxburg, U.; Guyon, I.; and Garnett, R., eds., Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, 4736–4744.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In Bengio, Y.; and LeCun, Y., eds., 2nd International Conference on Learning Representations, {ICLR} 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In 5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.

Kubricht, J. R.; Holyoak, K. J.; and Lu, H. 2017. Intuitive physics: Current research and controversies. Trends in cognitive sciences, 21(10): 749–759.

Lei, Z.; and Hongzhou, J. 2012. Variable step euler method for real-time simulation. In Proceedings of 2012 2nd International Conference on Computer Science and Network Technology, 2006–2010. IEEE. ISBN 1467329649.

Li, Y.; Ni, Y.; Croft, R. A. C.; Di Matteo, T.; Bird, S.; and Feng, Y. 2021. AI-assisted superresolution cosmological simulations. Proceedings of the National Academy of Sciences, 118(19).
Lu, Y.; and Huang, B. 2020. Structured Output Learning with Conditional Generative Flows. In The Thirty-Fourth {AAAI} Conference on Artificial Intelligence, {AAAI} 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, {AAAI} 2020, The Tenth {AAAI} Symposium on Educational Advances in Artificial Intelligence, {EAAI} 2021, 5005–5012. {AAAI} Press.

Mahnken, R. 2004. Identification of material parameters for constitutive equations. Encyclopedia of computational mechanics.

Meidani, K.; and Farimani, A. B. 2021. Data-driven identification of 2D Partial Differential Equations using extracted physical features. Computer Methods in Applied Mechanics and Engineering, 381: 113831.

Obiols-Sales, O.; Vishnu, A.; Malaya, N.; and Chandramowliswaran, A. 2020. CFDNet: A deep learning-based accelerator for fluid simulations. In Proceedings of the 34th ACM International Conference on Supercomputing, 12. ACM. ISBN 9781450379830.

Pilozzi, L.; Farrelly, F. A.; Marcucci, G.; and Conti, C. 2018. Machine learning inverse problem for topological photonics. Communications Physics 2018 1:1, 1(1): 1–7.

Prakash, S. K. A.; and Tucker, C. S. 2021. Node classification using kernel propagation in graph neural networks. Expert Systems with Applications, 174: 114655.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. 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.

Rezende, D. J.; and Mohamed, S. 2015. Variational Inference with Normalizing Flows. In Bach, F. R.; and Blei, D. M., eds., Proceedings of the 32nd International Conference on Machine Learning, {ICML} 2015, Lille, France, 6-11 July 2015, volume 37 of {JMLR} Workshop and Conference Proceedings, 1530–1538. JMLR.org.

Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; and Battaglia, P. 2020. Learning to simulate complex physics with graph networks. In International Conference on Machine Learning, 8459–8468. PMLR. ISBN 2640-3498.

Sanchez-Gonzalez, A.; Heess, N.; Springenberg, J. T.; Merel, J.; Riedmiller, M.; Hadsell, R.; and Battaglia, P. 2018. Graph networks as learnable physics engines for inference and control. 35th International Conference on Machine Learning, ICML 2018, 10: 7097–7117.

Schoenholz, S.; and Cubuk, E. D. 2020. Jax md: a framework for differentiable physics. Advances in Neural Information Processing Systems, 33.

Silver, T.; Chitnis, R.; Curtis, A.; Tenenbaum, J. B.; Lozano-Pérez, T.; and Kaelbling, L. P. 2021. Planning with Learned Object Importance in Large Problem Instances using Graph Neural Networks. In Thirty-Fifth {AAAI} Conference on Artificial Intelligence, {AAAI} 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, {IAAI} 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, {EAAI} 2021, Vir, 11962–11971. {AAAI} Press.

Téglaïs, E.; Vul, E.; Girotto, V.; Gonzalez, M.; Tenenbaum, J. B.; and Bonatti, L. L. 2011. Pure Reasoning in 12-Month-Old Infants as Probabilistic Inference. Science, 332(6033): 1054–1059.

Tenenbaum, J. B.; Kemp, C.; Griffiths, T. L.; and Goodman, N. D. 2011. How to grow a mind: Statistics, structure, and abstraction. science, 331(6022): 1279–1285.

Trippe, B. L.; and Turner, R. E. 2017. Conditional Density Estimation with Bayesian Normalising Flows. In Second workshop on Bayesian Deep Learning (NIPS 2017).

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog), 38(5): 1–12.

Wu, J.; Lim, J. J.; Zhang, H.; Tenenbaum, J. B.; and Freeman, W. T. 2016. Physics 101: Learning Physical Object Properties from Unlabeled Videos. In Wilson, R. C.; Hancock, E. R.; and Smith, W. A. P., eds., Proceedings of the British Machine Vision Conference 2016, {BMVC} 2016, York, UK, September 19-22, 2016. {BMVA} Press.

Ye, T.; Wang, X.; Davidson, J.; and Gupta, A. 2018. Interpretable Intuitive Physics Model. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., Computer Vision - {ECCV} 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part {XII}, volume 11216 of Lecture Notes in Computer Science, 89–105. Springer.

Zeng, A.; Song, S.; Lee, J.; Rodriguez, A.; and Funkhouser, T. 2020. Tossingbot: Learning to throw arbitrary objects with residual physics. IEEE Transactions on Robotics, 36(4): 1307–1319.

Zhang, J.; and Meng, L. 2019. GResNet: Graph Residual Network for Reviving Deep GNNs from Suspended Animation. CoRR, abs/1909.0.

Zheng, D.; Luo, V.; Wu, J.; and Tenenbaum, J. B. 2018. Unsupervised Learning of Latent Physical Properties Using Perception-Prediction Networks. In Globerson, A.; and Silva, R., eds., Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, {UAI} 2018, Monterey, California, USA, August 6-10, 2018, 497–507. {AUAI} Press.
A.1 Implementation details:

We implemented our approach using the Pytorch library. All models were trained and tested on a NVIDIA DGX Station A100 for 100 train/test epochs with a batch size of 1 using 5-fold cross-validation.

**Optimization and Hyper-parameters:** We use Adam optimizer with an exponential learning rate decay of $\gamma = 0.9$ with an initial learning rate of $1e^{-4}$ and weight decay (L2 Norm) as $1e^{-8}$. We use Xavier initialization for all weight matrices. While our model can train in significantly less time-steps per dataset, we find that higher learning rates tend to destabilize the training. We keep the initial learning rate and $\gamma$ constant throughout our experiments. We use dropout with a dropout rate = 0.5 and layer norm only while training on the elastic collisions dataset. We find that adding dropout and layer norm layers to the model for the 2D spring and gravity datasets causes testing instability, even though the training remained stable. In order to rectify this issue, we removed dropout and layer norm for those datasets and instead normalized all input and target vectors elementwise to zero mean and unit variance using dataset statistics. Finally, we set the number of hidden layers in the node, edge and processor network sub-components as 6, decoder hidden layers as 2 and inverse model hidden layers as 4. Since there are only 5 particle interactions per time-step in the 2D spring and gravity datasets, we observe an over-fitting due to model complexity and change the number of hidden layers of our model by half for these datasets.

**Graph structure:** For the 2D elastic body collisions, we consider a sparse graph wherein we use k-nearest neighbors algorithm (kNN) radius = 0.07. We observe that any value of radius > 0.07, up-to a fully connected graph yields no improvement in performance. We hypothesize that this is due to the nature of the problem wherein change in state of a particle is affected only upon direct contact/interaction with other particles. On the other hand, we observe that sparsifying the graph connectivity of 2D spring and gravity datasets significantly affects the performance of the output. We hypothesize that this is also caused by the nature of problems wherein change in state of a particle is affected by all other particles (Example:gravitational force imparted by planets and the sun in our solar-system affects the position and trajectory of all planets). In order to maintain consistency with the experimental setting of BS 1, we impart a fully-connected graph structure on the 2D spring and gravity datasets. However, while we find a significant improvement in conditioning the model on no previous velocity ($k = 0$) to $k > 1$, we find no significant improvement in performance when $k > 3$. Since BS 2 have used $k = 5$ in their approach, for the sake of consistency, we have used $k = 5$ in our approach as well.

A.2 Supplementary Experimental Results

**Performance of forward model** Table 2 reports the mean target test prediction errors across frame rates for all models. While the error values reported for our approach in Table 1 is based on each dataset comprising of 10,000 time-steps, we were unable to notice any significant improvement in performance when we increase the number of samples over 10000. However, we observe a significant drop in performance for BS 1 and BS 2. Hence, the test prediction errors reported for BS 1 and BS 2 were trained on 1,000,000 time-steps per dataset per frame rate. Table 2 simply serves as an extension to Table 1 wherein we additionally report the test prediction error across non hand-tuned frame rates (Cranmer et al. 2020b) for BS 1 and BS 2. We observe that BS 1 in particular had to be trained for more 350 epochs and at a learning rate of $1e^{-3}$ to achieve the results reported in Table 2. Additionally, both BS 1 and BS 2 were producing NaN and throwing huge error values for 2D gravity ($1/r^2$) data set across most frame rates.

![Comparison of ROC-AUC](image)

**Figure 4:** ROC-AUC comparison between different frame rates

**Behavior of the inverse model** Table 4 describes the proportion of the variation in the dependent variable ($R$) that is predictable from the independent variable ($Z$) -mass of each particle. Relative mass per particle pair ($\hat{R}_{ij} = 2m_i/m_i + m_j$). We find that our approach is more accurate and consistent at predicting the relative mass of particles at the step size of 100 fps. We observe worst prediction performance for 2D gravity data set when step size = 10 fps.
Table 2: Mean one-step target velocity/acceleration prediction error for varying frame rate

| Datasets                  | This Paper | (Cranmer et al. 2020b) | (Sanchez-Gonzalez et al. 2020) |
|---------------------------|------------|-------------------------|---------------------------------|
|                           | 10 30 50 100                      | 10 30 50 100                      | 10 30 50 100                      |
| 2D Elastic Collisions     | 0.003 0.003 0.003                     | 0.006 0.119 0.501                  | 0.625 0.122 0.053                  |
| 2D Spring                 | 0.0007 0.093 0.0009 0.08            | 0.047 0.089 0.082                  | 0.099 0.734 0.874                  |
| 2D Gravity ($\frac{1}{r^2}$) | 1.023 1.363 1.306 0.804           | ** ** **                         | ** 1.634 ** ** **                   |
| 2D Gravity ($\frac{1}{r}$)  | 0.129 0.561 0.588 0.836          | 0.088 0.093 0.077                  | 1.478 1.393 1.340                  |

Table 3: Mean Squared Error ($N$ denotes # of particles and $K$ denotes trajectory length, W. denotes with and W/O denotes without)

| Datasets       | N   | K  | Ours (W/O Collision and W/O Node embedding) | Ours (W. Collision and W/O Node embedding) | Ours (W. Collision and W. Node embedding) | Cranmer et al. | Sanchez-Gonzalez et al. |
|----------------|-----|----|---------------------------------------------|---------------------------------------------|---------------------------------------------|----------------|------------------------|
| Water          | 212 | 1000| 2.52e-8                                    | 2.3e-8                                      | 4.43e-8                                    | 6.56e-2        | 1.68e-7                |
| Water Drop     | 678 | 1000| 1.46e-8                                    | 1.254e-8                                   | 9.402e-9                                  | 4.92e-1        | 2.89e-7                |
| Sand           | 542 | 320 | 1.2e-7                                     | 2.34e-7                                   | 6.06e-8                                   | 5e-2           | 1.31e-6                |

Table 4: Coefficient of determination ($R^2$) score in predicting relative mass

| Datasets                  | 10 fps | 30 fps | 50 fps | 100 fps |
|---------------------------|--------|--------|--------|---------|
| 2D Elastic Collisions     | 0.91 ± 0.0003 | 0.89 ± 0.0004 | 0.89 ± 0.00034 | 0.88 ± 0.0002 |
| 2D Spring                 | 0.70 ± 0.052  | 0.977 ± 0.024   | 0.937 ± 0.04   | 0.949 ± 0.0003 |
| 2D Gravity ($\frac{1}{r^2}$) | 0.15 ± 0.05  | 0.737 ± 0.019  | 0.443 ± 0.04  | 0.947 ± 0.03 |
| 2D Gravity ($\frac{1}{r}$)  | 0.547 ± 0.002 | 0.949 ± 0.0003 | 0.9 ± 0.01   | 0.949 ± 0.0003 |