The Neural Physics Engine: A Compositional Object-Based Approach to Learning Physical Dynamics

Abstract

This paper presents the Neural Physics Engine (NPE), an object-based neural network architecture for learning predictive models of intuitive physics. The NPE draws on the strengths of both symbolic and neural approaches: like a symbolic physics engine, it is endowed with generic notions of objects and their interactions, but as a neural network it can also be trained via stochastic gradient descent to adapt to specific object properties and dynamics of different worlds. We evaluate the efficacy of our approach on simple rigid body dynamics in two-dimensional worlds. By comparing to a less structured architecture, we show that NPE's compositional representation of the causal structure in physical interactions improves its ability to predict movement, generalize to different numbers of objects, and infer latent properties of objects such as mass.

1 Introduction

Physical reasoning is an important aspect of human and artificial intelligence, and it is also a crucial part of learning, perception and planning. Accurately modeling a visual scene involves reasoning about the spatial properties, identities and locations of objects (Eslami et al. 2016; Hinton, Krizhevsky, and Wang 2011; Jaderberg et al. 2015; Kulkarni et al. 2015), but also their physical properties, future dynamics, and causal relationships. The ability to predict real-world object dynamics – as well as hypothetical and counterfactual trajectories – is a crucial aspect of planning and inference for humans (Battaglia, Hamrick, and Tenenbaum 2013; Gerstenberg et al. 2015; Hamrick, Battaglia, and Tenenbaum 2011; Smith, Battaglia, and Vul 2013; Sanborn, Mansinghka, and Griffiths 2013), as well as for future human-level artificial intelligence (Lake et al. 2016).

Such a sense of intuitive physics can be seen as a program (Goodman and Tenenbaum 2016) that takes in the input provided by the scene and past states, and outputs the future states and physical properties of relevant objects. This program should flexibly scale to model worlds with variable numbers of objects, object properties, dynamics, interactions, and physical laws. It must be general enough to express various compositions, arrangements, and behavior in known worlds and also flexible enough to adapt to unknown worlds with new configurations.

A physics engine is an example of such a program. It symbolically represents objects and latent physical properties, and it approximately simulates the dynamics of physical systems using pre-specified physical laws and constraints. Battaglia, Hamrick, and Tenenbaum proposed a model for physical reasoning in humans as similar to game physics engines. While such a physics engine can easily express and generalize across any scenario with entities and operators supported by its description language, it is brittle under scenarios that are not supported. Adapting to these new scenarios requires modifying the code or generating new code for the physics engine itself. To do this manually is cumbersome, and to do this efficiently and automatically is not straightforward.

This paper takes a step toward [narrowing the gap/alleviating the tradeoff] between expressivity and adaptability by combining rough symbolic structure with gradient-based learning. We present the Neural Physics Engine (NPE), a predictive model of physical dynamics. It exhibits several strong inductive biases that are explicitly present in symbolic physics engines, such as a notion of objects and object interactions. Specifically, the NPE models the future state of a single object as a composition of the pairwise interactions between itself and other objects in the scene. Though the architecture exhibits strong structure, it is end-to-end differentiable and thus is also flexible to tailor itself to the specific object properties and dynamics of a given world through training. This approach – starting with a general sketch of a program and filling in the specifics – is similar to ideas presented by (Solar-Lezama 2008; Tenenbaum et al. 2011). The NPE's general sketch is the structure of its architecture, and it extends and enriches this sketch to [explain/model] the specifics of a particular scene by training on observed trajectories from that scene.

In this paper, we investigate variations on two-dimensional worlds of bouncing balls from the matter-js physics engine (Brummitt, `http://brm.io/matter-js/`) as a testbed for exploring the NPE's capabilities for modeling simple rigid body dynamics. We randomly generate trajectories for various number of objects with fixed and variable masses, and we use these trajectories as ground-truth observations for training our

predictive model.

The main contributions of this paper are as follows. [In Section 2 we review related work to learning physics.] In Section 3 we outline the problem of learning a predictive model of physical dynamics, and we present the NPE architecture as our proposal for learning such a predictive model. Unlike some other approaches that learn from pixels, the NPE operates on the state space of positions and velocities and therefore purely focuses on physics, giving us more uncluttered insight on how notions of compositionality and causality can help guide learning. To investigate the differences such notions make, we compare the NPE with a baseline that is very similar, but just lacks explicit modeling of pairwise interactions. In Section 4 we evaluate the NPE on tasks for predicting future trajectories, generalizing to greater numbers of objects than the NPE has encountered, and inferring mass in worlds where an object's mass is unknown.

2 Related Work

The search for a program that captures common-sense physical reasoning has been the focus of much recent work, with at least two general approaches being used. The first is a top-down approach that formulates the problem as inference over the parameters of a physics engine (Battaglia, Hamrick, and Tenenbaum 2013; Ullman, Stuhlmüller, and Goodman 2014; Wu et al. 2015). Physics engines, such as the matter-js engine that we use in this paper, symbolically represent objects, latent physical properties, and physical laws, and they compose these representations to approximately simulate the dynamics of physical systems. The top-down approach attempts to invert a physics engine by explaining scene dynamics and object trajectories as manifestations of physics engine parameters, including latent physical properties (e.g. mass and elasticity) and physical laws.

The second approach is a bottom-up data-driven approach that learns to directly map physical observations to motion prediction or physical judgments (Lerer et al. 2016; Li et al. 2016; Mottaghi et al. 2015; 2016). This approach often uses tools such as neural networks, and involves training on large datasets. Whereas the top-down approach usually explicitly assumes basic primitive concepts (such as force and mass) that can compose together to express a certain scenario, it is not as clear for these networks whether these concepts emerge as intermediate representations.

The top-down and bottom-up approaches have complementary advantages and disadvantages (Zhang et al. 2016), and scale in their respective ways. The programs under the top-down approach inherently use symbolic representations of the world and can easily express and generalize across any scenario supported by the entities and operators in its description language. For example, adding a new object to a given scene is trivial. However, inverting a physics engine is an expensive search problem over the space of physical parameters and laws that could have generated a particular set of observations. Moreover, a physics engine is brittle under scenarios not supported by its description language, and adapting to these new scenarios requires modifying the code or generating new code for the physics engine itself.

In contrast, the programs learned under gradient-based bottom-up approaches do not require hand-crafted specifications. The same architecture and learning algorithm of a neural network can be applied to new scenarios without requiring the physical dynamics of the scenario to be pre-specified. However, such models require extensive amounts of data, and oftentimes transferring knowledge to new scenes requires retraining, even in cases that seem trivial to human reasoning.

A particular form for a program for physical reasoning is one that emulates a general-purpose physics simulator (Lake et al. 2016). Battaglia, Hamrick, and Tenenbaum; Bates et al.; Ullman, Stuhlmüller, and Goodman investigate approximate probabilistic game engines as a computational model for physical reasoning in humans. Simulating object states forward into the future is not only useful as a predictive model that can be used for planning, but can also be used as a general-purpose mechanism for tasks such as inferring latent properties, highlighting causal relationships, and making hypothetical and counterfactual judgments.

Bottom-up approaches to these tasks (Finn, Goodfellow, and Levine 2016; Sutskever, Hinton, and Taylor 2009; Fragkiadaki et al. 2015; Lerer et al. 2016; Srivastava, Mansimov, and Salakhutdinov 2015; Agrawal et al. 2016) can bypass explicit representation used in top-down approaches. Due to how these models are trained, physical reasoning in the form of making judgments often need not be obtained through simulation. However, learning direct mappings from observation to output is often at the cost of reduced generality, such that the model may need to be retrained for different tasks. A couple recent work have begun to address this problem, such as by using a shared network for both predicting future frames and for making physical judgments (Lerer et al. 2016), or by decomposing the complexity of a scene by conditioning motion prediction on individual objects rather than the entire scene (Fragkiadaki et al. 2015).

3 Approach

3.1 Scenario

Consider a scene with K objects $o^{(1)}...o^{(K)}$. We adopt a simple parametrization for an object at any given time. An object's state $o^{(k)}$ at time t is represented by extrinsic properties position $p_t^{(k)}$, velocity $v_t^{(k)}$, angle $a_t^{(k)}$, angular velocity $\alpha_t^{(k)}$, by intrinsic properties mass $m^{(k)}$, type $t^{(k)}$, and size multiplier $s^{(k)}$, and by global properties gravity g, friction f, and pairwise forces p. An object's extrinsic properties specify its state through time and space, whereas its intrinsic properties do not change over time. Global properties are held constant.

Jointly predicting the future states of all objects in a scene quickly becomes unscalable as the number of objects grows large. We make two key observations to reduce the complexity of this problem. First, because physical laws do not change across inertial frames it suffices to separately predict the future state of each object conditioned on the past states of itself and the other objects in its local neighborhood. This approach has also been used in (Fragkiadaki et al. 2015).

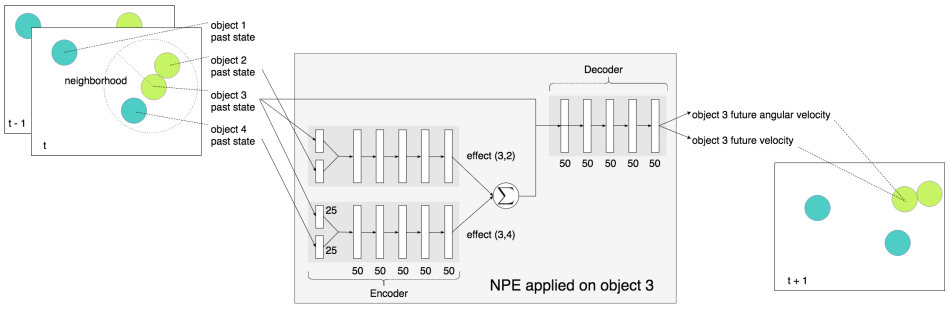

Figure 1: **NPE Architecture**: The above figure illustrates an example forward pass of the NPE. Here, the NPE is conditioned object 3 as the focus object and predicts object 3's velocity and angular velocity for timestep $t+1$. Other objects in the scene include objects 1, 2, and 4, but only objects 2 and 4 are within the neighborhood of object 3, so object 1 is ignored by the NPE. The NPE's shared encoder summarizes the pairwise interactions between object 3 and 2 and between objects 3 and 4 at times t and $t-1$. The encoder output is summed before fed into the decoder. The decoder takes the sum of the pairwise "effects" as well as objects 3's past state. Note that the neighborhood objects changes depending on the focus object, and the NPE predicts the trajectories of all objects by viewing each object in turn as a focus object.

Second, because physics is Markovian, this prediction need only be for the immediately next timestep. This spatiotemporal factorization of the scene is the basis of the NPE.

3.2 Neural Physics Engine

Letting a particular object be the *focus* object and all other objects in the scene be *context* objects, the NPE models the focus object's velocity $v_{t+1}^{(f)}$ and angular velocity $\alpha_{t+1}^{(f)}$ at time $t+1$ as a composition of pairwise interactions between the focus object and every other neighboring context object during time $t-1$ and t. For simplicity of notation, let $u = \{v_{t+1}^{(f)}, \alpha_{t+1}^{(f)}\}$. In practice, the NPE predicts Δu, the *change* in velocity and angular velocity between t and $t+1$, and it is trained via rmsprop to minimize the Euclidean loss between its prediction $\hat{u}$ and the ground truth u. To simulate an object forward in time, the model finds $u_{t+1} = u_t + \Delta u$, and updates $p_{t+1} = p_t + v_{t+1}$ and $a_{t+1} = a_t + \alpha_{t+1}$.

As shown in Figure 1, the NPE is a composition of two functions implemented as neural networks: the first is an encoder that summarizes the pairwise interaction between the focus object $o^{(f)}$ and a single context object $o^{(c)}$, and it is applied for each neighboring context object:

$$e^{(f,c)} = f_{enc}\left(o^{(f)}, o^{(c)}\right). \tag{1}$$

The second is a decoder that takes the sum of these pairwise interactions as input and predicts Δu_{t+1}:

$$\Delta \hat{u}_{t+1} = f_{dec}\left(o^{(f)}, \sum_{c \in N(f)} e^{(f,c)}\right). \tag{2}$$

Each $\left(o^{(f)}, o^{(c)}\right)$ pair is selected to be in the set of neighbors $N(f)$ by the neighborhood masking function $\mathbb{1}[||p^{(c)} - p^{(f)})|| < o_n^{(f)}]$, which takes value 1 if the Euclidean distance between the positions of the focus and context objects is less the threshold $o_n^{(f)}$.

The design choice behind how f_{enc} and f_{dec} are defined and composed reflects the high-level formulation of many symbolic physics engines. The general recipe evolves objects through time based on dynamics that dictate their independent behavior (e.g. friction, gravity, inertia) and dynamics that dictate their behavior with other objects (e.g. collision, support, attraction, repulsion). Notably, in the reference frame of a particular object, the forces it feels from other objects are additive.

The NPE architecture incorporates several inductive biases that reflects this recipe. In particular, we provide a loose interpretation the encoder output $e^{(f,c)}$ as the *effect* of object c on object f, and that these effects are additive as forces are. This design thus scales flexibly with the number of objects in a scene: for an object with no context objects nearby, $\sum_{c=\in N(f)} e^{(f,c)} = 0$, and this sum grows and shrinks based on effects from neighboring context objects. These inductive biases have the effect of strongly constraining the space of possible programs of predictive models that the NPE can learn, allowing the NPE to waste less time exploring this space during training, as shown in the learning curves in Section 4.

Baseline: No Pairwise Structure To highlight the benefit of the inductive biases mentioned above, we focus our attention on a particular component of the NPE that exemplifies a compositional and causal factorization of the underlying physics of a scene. We compare the NPE to a baseline **??** that is very similar, but lacks the pairwise layer of the encoder. Instead, each object is encoded independently through a shared encoder. Information for modeling how objects interact would only be present after the encoder. Similar to the "social pooling" discussed in (Alahi et al.), the NP sums the encodings for neighboring context objects. This sum is concatenated with the encoding of the focus object before fed into the decoder. We name this baseline No-Pairwise (NP) because it most directly highlights the usefulness of explicitly modeling pairwise interactions between objects, a strong

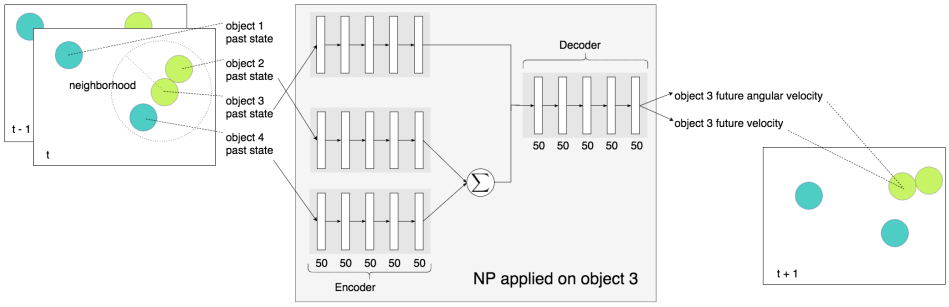

Figure 2: **NP Baseline**: Like the NPE, the No-Pairwise (NP) baseline also is conditioned on a particular focus object. Note that instead of having explicit structure of causal pairwise relationships as the NPE has, the NP encodes all relevant objects (the focus object and its neighbors) independently through the same encoder. Effects from neighboring context objects only is present at the concatenation right before the decoder. The decoder is the same as that of the NPE.

modeling assumption to make.

3.3 Implementation

State Representation Our worlds are defined in a rectangular box 600 pixels in height and 800 pixels in width. We represent the kth object's state, $o^{(k)}$ as a 22-dimensional vector. Elements 1 and 2 of $o^{(k)}$ represent the x and y coordinate positions of the center of the object. All positional coordinates are normalized by dividing their values by 800, the width of the world window. Elements 3 and 4 represent the x and y components of the object's velocity. Units are defined such that an object moves a number of pixels equivalent to its velocity over one timestep. The velocity is normalized by dividing by 60, the maximum velocity any object can have in our worlds. Elements 5 and 6 are the angle and angular velocity of an object. The angle takes on values in $(-\pi, \pi)$, and both angle and angular velocity are normalized by 2π. Elements 7 through 10 represent a one-hot encoding of mass, with possible mass values 1, 5, 25, and 10^{30}. The mass of 10^{30} is reserved for stationary objects. Elements 11 through 13 represent a one-hot encoding of the object type, which can be a circular ball, a square obstacle, or a rectangular block with a 3:1 height-width ratio. Each object type has a default size. A ball has a default radius of 60 pixels, an obstacle has a default side length of 80 pixels, and the long side of the block has a default length of 60 pixels. Elements 14 through 16 are a one-hot encoding of a multiplicative constant (0.5, 1, or 2) that scales the object's default size by that amount. Elements 17 and 18 are a one-hot encoding of the presence of gravity. Elements 19 and 20 are a one-hot encoding of the presence of frictional forces. Elements 21 and 22 are a one-hot encoding of the presence of pairwise forces. In the bouncing ball worlds that we investigate, gravitational, frictional and pairwise forces are set to zero. We concatenate $o_{t-1}^{(k)}$ and $o_t^{(k)}$ together into a 44-dimensional vector, and this is the input representation for a single object into our models.

Architectures Both the NPE and the NP are implemented as multilayer neural networks using the neural network libraries built by Collobert, Kavukcuoglu, and Farabet; Léonard, Waghmare, and Wang. The NPE encoder is consists of a pairwise layer and a 5-layer fully connected network with rectified linear activations after every layer. The pairwise layer transforms the focus object and a single other context object each into a 25-dimensional hidden representation. These are concatenated into a 50-dimensional vector as input the encoder, which has 50 units in every layer. The NP encoder is the same as the NPE encoder, but without the pairwise layer. Thus, the NPE encoder takes in one (focus, context) pair as input, while the NP encoder takes in a single object. The encoder is replicated many times, for each object pair in the case of the NPE, or for each object in the case of the NP. Thus, the NPE and NP can model a variable number of objects. Both models share the same decoder architecture.

Training Details We trained both models using the rmsprop (Tieleman and Hinton 2012) backpropagation algorithm for 1,200,000 iterations with a learning rate decay of 0.99 every 2,500 training iterations, beginning at iteration 50,000. We used a 70-15-15 split for training, validation, and test data. In our experiments, we empirically found that 3.5 times the focus object's size worked well for the neighborhood threshold $o_n^{(f)}$.

TODO cosine similarity

4 Evaluation

We compare the NPE and the NP on worlds of bouncing balls. We choose bouncing balls because it exhibits self-evident dynamics and support a wide set of rich scenarios that reflect everyday physics. Bouncing balls have also been used in cognitive science to study causality and counterfactual reasoning, as in Gerstenberg et al.. Experiments are summarized in Table 1 and randomly selected model predictions from the test data are shown in 3. Angle and angular velocity are clamped to zero in these worlds. We generate 50000 trajectories of 60 timesteps (10 timesteps $\approx$ 1 second), and trained our models on 3-timestep windows in these trajectories. Our plots show results over three independent runs of each model.

4.1 Prediction

We visualize the cosine distance between the predicted velocity and the ground truth velocity as well as the relative error in magnitude of the predicted velocity and the ground truth velocity, over 50 timesteps of simulation (about 5 seconds). Both the NPE and the NP take timesteps 1 and 2 as initial input, and then use previous predictions as input to future predictions. As can be seen from Figure 4, the NPE produces significantly lower error than the NP, and Figure 3 illustrates an example scenario where the NPE predicts a collision that the NP does not. Though both the NPE and the NP begin to diverge from the ground truth as time passes (the NP much more so), Figure 3 shows that the models still learn simple Netwonian concepts such as inertial movement and collisions with other objects and the world boundaries.

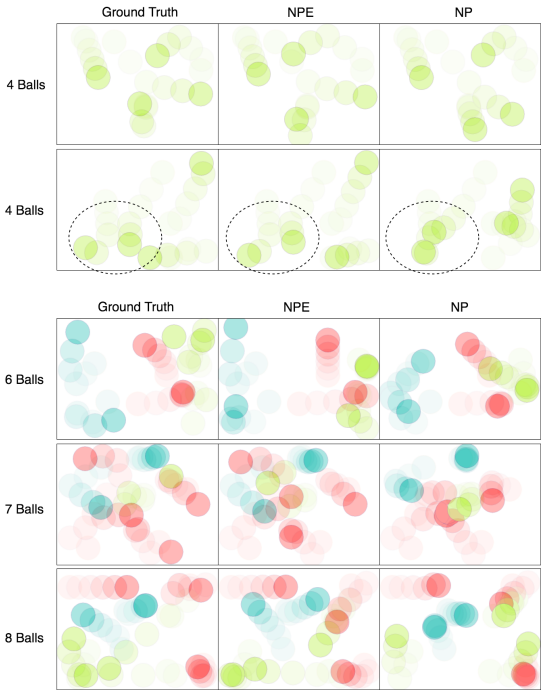

Figure 3: **Predicted Trajectories** *Top two rows:* Two examples of trajectories from the BS experiments. Note that not only does the NPE adhere more closely to the ground truth, but it is able to successively resolve object-object collisions. This is demonstrated with the bottom left balls in the second row (circled). Rather than predicting that collision, the NP allows them to overlap. *Bottom three rows:* Examples of trajectories in worlds with 6 balls, 7 balls, and 8 balls of variable mass, as described in 4.3. Note that the models have only seen 3, 4, and 5 balls during training, so this task carries complexity not only with the variable mass but also with the number of balls. Both networks learn about inertial movement, but the NPE handles collisions better, suggesting an advantage of its pairwise structure.

Experiment	Description
BS	$4 \rightarrow 4$, fixed mass
BG	$3, 4, 5 \rightarrow 6, 7, 8$, fixed mass
BSM	$4 \rightarrow 4$, variable mass
BGM	$3, 4, 5 \rightarrow 6, 7, 8$, variable mass

Table 1: Two ways to increase complexity are to model a previously unseen number of objects and to vary latent properties such as mass. Our experiments begin with the Balls Simple (BS) experiment. From there we investigate extrapolation capability in Balls Generalization (BG). We explore further complexity in the worlds with the Balls Simple Mass (BSM) and the Balls Generalization Mass (BGM) experiments are variants of BS and BG that have variable masses. $3, 4, 5 \rightarrow 6, 7, 8$ in an experiment description means that we train on 3, 4, and 5 objects and test on 6, 7, and 8 objects.

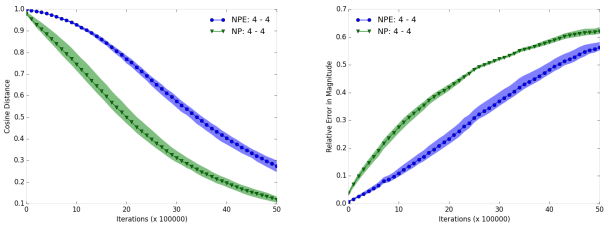

Figure 4: **BS Experiments** *left:* Cosine distance as a function of simulation time between the predicted velocity and the ground truth velocity. *right:* Relative error in the magnitude of the predicted velocity. Both the NPE and the NP have strong predictions early on, but the NP's adherence to the ground truth falls off quicker than the NPE's. These plots are on the test data.

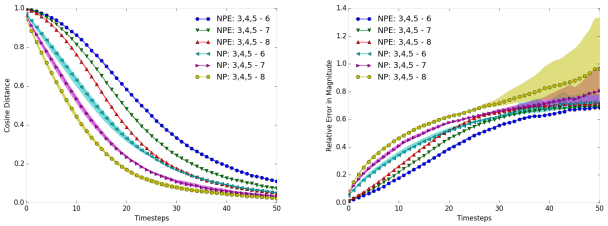

Figure 5: **BG Experiments: Simulation** These plots show the cosine distance (*left*) and relative magnitude (*right*) of the predicted vs ground truth velocity over time on the test data. Note that the NPE's behavior is quite consistent over the three independent runs compared to the NP's behavior, which diverges wildly for the relative magnitude towards 50 seconds of simulation. Compared to the BS Experiments, the models' performance drops much sooner for this extrapolation task.

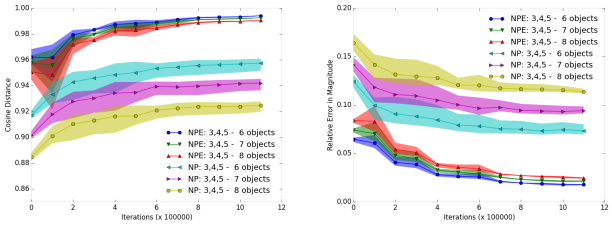

Figure 6: **BG Experiments: Training** The NPE is consistently superior to the NP in cosine distance error and relative magnitude error for velocity throughout training. This plot shows test data results.

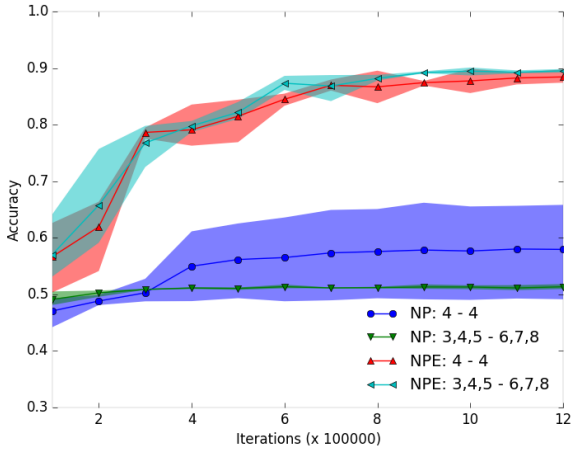

Figure 7: **BSM and BGM Experiments** Results show that the NPE's accuracy in mass inference is significantly greater than that of the NP. Moreover, the NPE performs similarly well whether in a world it has seen before or in a world with a different number of objects it has seen, further showcasing its strong generalization capabilities. These plots show accuracy on the test data. Accuracy across 6, 7, and 8 balls are averaged together. In this task, the models must infer one out of three masses, so random guessing would have an accuracy of 33%.

4.2 Strong Generalization

We next test whether learned knowledge of these simple physics concepts can be transferred to worlds with a number of objects previously unseen. The NPE demonstrates successful transfer of physics concepts in the extrapolation experiment BG. As the training curves Figure 7 shows, the NPE's performance on unseen worlds is comparable to its performance on observed worlds. The unseen worlds (6, 7, 8 objects) are structurally more complex and varied than the observed worlds (3, 4, 5 objects). The NPE's performance of this generalization task suggests that its compositionality and its architectural inductive biases are useful for generalizing knowledge learned in Markovian domains with causal structure in object interactions.

4.3 Mass Inference

We now show that the NPE can infer latent properties such as mass. This proposal is motivated by the experiments in (Battaglia, Hamrick, and Tenenbaum 2013), which uses a probabilistic physics simulation engine, the Intuitive Physics Engine (IPE), to infer various properties of a scene configuration. Whereas the rules of the IPE for modeling physical dynamics were manually pre-specified, the NPE learns these rules from observation.

We trained the NPE on the balls worlds, except we uniformly sampled the mass for each ball from the log-spaced set $\{1, 5, 25\}$. For evaluation, we fix the masses of all objects except that of the focus object, and simulate the NPE's prediction under all possible mass hypotheses for the focus object. Because collisions can inform mass judgments and inertial movement cannot, we consider only examples where the focus object collides with another object in the world. The prediction is scored against the ground-truth under the same Euclidean loss used in training. The mass hypothesis whose prediction yielded the lowest error is the NPE's maximum likelihood estimate of the mass for the focus object. Figure 7 visualizes the NPE's accuracy as the fraction of examples where the NPE chooses the correct mass. A model that chooses randomly would have an accuracy of 33%. When we compare the NPE's accuracy to the accuracy of NP, we see that a simulator with a more accurate model of the world performs more accurate inferences, and that this simulation capability can be learned from observation. Because the NPE is differentiable, we expect that it can also infer latent properties such as mass by backpropagating prediction error to its a randomly sampled input. This would be especially useful for inferring the positions of "invisible" objects, whose effects are felt but whose position is unknown. Furthermore, though we adopted a particular parametrization of an object, the NPE is not limited to the semantic meaning of the elements of its input.

5 Discussion

We have demonstrated a compositional object-based approach to learning physical dynamics in worlds of bouncing balls. The NPE achieves low prediction error, scales to various number of objects, and can infer latent properties such as mass. We compared the NPE's performance with that of the NP baseline, and showed that the NPE architectural structure is more well suited for modeling worlds of bouncing objects. Further work includes applying the NPE architecture to different objects and physical behavior, such as to immovable obstacles and stacked block towers. We've shown that the NPE performs well in inferring mass, and because the NPE is differentiable, we hope to investigate in future work whether such inference can also be possible by backpropagation to the inputs. We've shown that though the NPE is can be trained from observation, its general structure is inspired by the formulation of symbolic physics engines, and this structure has helped the NPE perform well in various tasks of physical reasoning. Our results invite ques-

tions on how much prior information and structure should and could be given to bottom-up neural networks, and what can be learned without inducing such structure. The NPE is itself a program that predicts future object states from past object states, but it would interesting to explore how similar models to the NPE can be used as subprograms that can be called by parent programs to evolve entity states through time for applications in areas such as model-based planning and model-based reinforcement learning.

References

Agrawal, P.; Nair, A.; Abbeel, P.; Malik, J.; and Levine, S. 2016. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR* abs/1606.07419.

Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; and Savarese, S. Social lstm: Human trajectory prediction in crowded spaces.

Bates, C. J.; Yildirim, I.; Tenenbaum, J. B.; and Battaglia, P. W. 2015. Humans predict liquid dynamics using probabilistic simulation.

Battaglia, P. W.; Hamrick, J. B.; and Tenenbaum, J. B. 2013. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences* 110(45):18327–18332.

Brummitt, L. http://brm.io/matter-js.

Collobert, R.; Kavukcuoglu, K.; and Farabet, C. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376.

Eslami, S.; Heess, N.; Weber, T.; Tassa, Y.; Kavukcuoglu, K.; and Hinton, G. E. 2016. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*.

Finn, C.; Goodfellow, I.; and Levine, S. 2016. Unsupervised learning for physical interaction through video prediction. *arXiv preprint arXiv:1605.07157*.

Fragkiadaki, K.; Agrawal, P.; Levine, S.; and Malik, J. 2015. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*.

Gerstenberg, T.; Goodman, N.; Lagnado, D. A.; and Tenenbaum, J. B. 2012. Noisy newtons: Unifying process and dependency accounts of causal attribution. In *In proceedings of the 34th*. Citeseer.

Gerstenberg, T.; Goodman, N. D.; Lagnado, D. A.; and Tenenbaum, J. B. 2015. How, whether, why: Causal judgments as counterfactual contrasts.

Goodman, N. D., and Tenenbaum, J. B. 2016. Probabilistic models of cognition.

Hamrick, J.; Battaglia, P.; and Tenenbaum, J. B. 2011. Internal physics models guide probabilistic judgments about object dynamics.

Hinton, G. E.; Krizhevsky, A.; and Wang, S. D. 2011. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning–ICANN 2011*. Springer. 44–51.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2008–2016.

Kulkarni, T. D.; Kohli, P.; Tenenbaum, J. B.; and Mansinghka, V. 2015. Picture: A probabilistic programming language for scene perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4390–4399.

Lake, B. M.; Ullman, T. D.; Tenenbaum, J. B.; and Gershman, S. J. 2016. Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289*.

Léonard, N.; Waghmare, S.; and Wang, Y. 2015. rnn: Recurrent library for torch. *arXiv preprint arXiv:1511.07889*.

Lerer, A.; Gross, S.; Fergus, R.; and Malik, J. 2016. Learning physical intuition of block towers by example. *arXiv preprint arXiv:1603.01312*.

Li, W.; Azimi, S.; Leonardis, A.; and Fritz, M. 2016. To fall or not to fall: A visual approach to physical stability prediction. *arXiv preprint arXiv:1604.00066*.

Mottaghi, R.; Bagherinezhad, H.; Rastegari, M.; and Farhadi, A. 2015. Newtonian image understanding: Unfolding the dynamics of objects in static images. *arXiv preprint arXiv:1511.04048*.

Mottaghi, R.; Rastegari, M.; Gupta, A.; and Farhadi, A. 2016. " what happens if..." learning to predict the effect of forces in images. *arXiv preprint arXiv:1603.05600*.

Sanborn, A. N.; Mansinghka, V. K.; and Griffiths, T. L. 2013. Reconciling intuitive physics and newtonian mechanics for colliding objects. *Psychological review* 120(2):411.

Smith, K.; Battaglia, P.; and Vul, E. 2013. Consistent physics underlying ballistic motion prediction.

Solar-Lezama, A. 2008. *Program synthesis by sketching*. ProQuest.

Srivastava, N.; Mansimov, E.; and Salakhutdinov, R. 2015. Unsupervised learning of video representations using lstms.

Sutskever, I.; Hinton, G. E.; and Taylor, G. W. 2009. The recurrent temporal restricted boltzmann machine. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 21*. Curran Associates, Inc. 1601–1608.

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.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Ullman, T.; Stuhlmüller, A.; and Goodman, N. 2014. Learning physics from dynamical scenes.

Wu, J.; Yildirim, I.; Lim, J. J.; Freeman, B.; and Tenenbaum, J. 2015. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems*, 127–135.

Zhang, R.; Wu, J.; Zhang, C.; Freeman, W. T.; and Tenenbaum, J. B. 2016. A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding. *arXiv preprint arXiv:1605.01138*.