A COMPOSITIONAL OBJECT-BASED APPROACH TO LEARNING PHYSICAL DYNAMICS

Michael B. Chang[*], **Tomer Ullman**[**], **Antonio Torralba**[*], and **Joshua B. Tenenbaum**[**]
[*]Department of Electrical Engineering and Computer Science, MIT
[**]Department of Brain and Cognitive Sciences, MIT
{mbchang,tomeru,torralba,jbt}@mit.edu

ABSTRACT

We present the Neural Physics Engine (NPE), an object-based neural network architecture for learning predictive models of intuitive physics. We propose a factorization of a physical scene into composable object-based representations and also the NPE architecture whose compositional structure factorizes object dynamics into pairwise interactions. Our approach draws on the strengths of both symbolic and neural approaches: like a symbolic physics engine, the NPE 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 less structured architectures, we show that our model's compositional representation of the 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 a crucial part of learning, perception, planning, inference, and understanding in artificial intelligence, and can be leveraged to accelerate the learning of new tasks (Lake et al., 2016). Accurately modeling a scene involves reasoning about the spatial properties, identities and locations of objects (Eslami et al., 2016; Hinton et al., 2011; Jaderberg et al., 2015; Kulkarni et al., 2015), but also their physical properties, future dynamics, and causal relationships. Such a sense of intuitive physics can be seen as a program (Anderson, 1990; Goodman and Tenenbaum, 2016) that takes input provided by a physical scene and the past states of objects, and outputs the future states and physical properties of relevant objects. For example, humans have such programs that can simulate into the future how a stack of dishes would fall if pushed or how a billiard ball would bounce when colliding with others, and humans can also infer relative masses when observing how a tennis ball bounces off a moving a truck. The program must be general enough to express various arrangements and behavior in known worlds, and also flexible enough to adapt to unknown worlds with new configurations. For example, if the program can model three balls bouncing in a box, it should be able to apply the same model to four balls or eight balls. If the program can model balls bouncing around in a particular obstacle arrangement, it should be able to model balls bouncing around in a different obstacle arrangement without retraining. If the balls now have different mass, the program should have assumptions that are general enough to learn these new dynamics. This work presents a program that exhibits these desired properties.

At least two general approaches have emerged in the search for a program that captures commonsense physical reasoning. The top-down approach (Battaglia et al., 2013; Ullman et al., 2014; Wu et al., 2015) formulates the problem as inference over the parameters of a symbolic physics engine, while the bottom-up approach (Agrawal et al., 2016; Fragkiadaki et al., 2015; Lerer et al., 2016; Li et al., 2016; Mottaghi et al., 2015; 2016; Sutskever et al., 2009) learns to directly map physical observations to motion prediction or physical judgments. A program under the top-down approach can generalize across any scenario supported by the entities and operators in its description language. However, it may be 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, gradient-based bottom-up approaches can apply the same model architecture and learning algorithm to specific scenarios without requiring the physical dynamics of the scenario to be pre-specified. This often comes at the cost of reduced generality: transferring knowledge to new scenes may require extensive retraining, even in cases that seem trivial to human reasoning.

This paper takes a step toward bridging this gap between expressivity and adaptability by proposing the Neural Physics Engine (NPE), a differentiable physics program that combines rough symbolic structure with gradient-based learning for physical inference. This hybrid approach exhibits several strong inductive biases that are explicitly present in symbolic physics engines, such as a notion of objects-specific properties and object interactions. It is also end-to-end differentiable, and thus is able to flexibly tailor itself to the specific object properties and dynamics of a given world through training. It makes two strong, but natural, assumptions about a physical environment: 1) there exist objects and 2) these objects interact with each other in a factorized manner. By design, it can extrapolate to variable number of objects and variable scene configurations with only spatially and temporally local computation.

This paper's contribution links two levels of factorization and composition in learning physical dynamics. On the level of the physical scene, we factorize the scene into object-based representations (Sec. 2.1), and compose smaller building blocks to form larger objects (Sec. 3.5). This framework of representation adapts to complex scenes and configurations with variable number of objects. On the level of the physics program, the NPE architecture (Sec. 2.2) explicitly reflects a causal structure in object interactions by factorizing object dynamics into pairwise interactions. As a predictive model of physical dynamics, the NPE models the future state of a single object as a function composition of the pairwise interactions between itself and other neighboring objects in the scene. This structure serves to guide learning towards object-based reasoning as (Hinton et al., 2011) does, and is designed to scale to large numbers of objects anywhere in the scene with only local computation. This design allows physical knowledge to transfer across variable number of objects and for object properties to be explicitly inferred. 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 its architectural structure, and it extends and enriches this sketch to model the specifics of a particular scene by training on observed trajectories from that scene.

While previous bottom-up approaches (Sec. 4) have coupled learning vision and learning physical dynamics, we take a different approach, and for two reasons. First, we see that disentangling the visual properties of an object from its physical dynamics is a step toward achieving the generality of a physics engine. A program that learns to evolve objects through time can potentially be reused as a subprogram that can be composed under a modular framework with other programs for applications in areas such as model-based planning and model-based reinforcement learning. This is not to say that vision and dynamics should not be combined; both are necessary, but we believe that keeping these functionalities separate is important for common-sense generalization that is robust to cases where the visual appearance changes but the dynamics remain the same. Second, we are optimistic that those two components indeed can be decoupled, that a vision model can map visual input to an intermediate state space, and a dynamics model can evolve objects in that state space through time. For example, there is much work in object detection and segmentation for extracting position and velocity, as well as work for extracting latent object properties (Wu et al., 2015). Therefore this paper focuses on learning dynamics in that state space, taking a small step toward emulating a general-purpose physics engine (Lake et al., 2016), with the eventual goal of building a system that exhibits the compositionality, modularity, and generality of physics engine whose internal components can be learned through observation.

We investigate variations on two-dimensional worlds of balls and obstacles 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 under our state space representation (Sec. 3). While these worlds are generated from a simplified physics engine, we believe that learning to model such simple physics under the NPE's framework is a first and necessary step towards emulating the full capacity of a general physics engine, while maintaining a differentiability that can allow it to eventually learn complex real-world physical phenomena (see Sec. 4) that would be challenging to engineer into these physics engines. This paper establishes that important step.

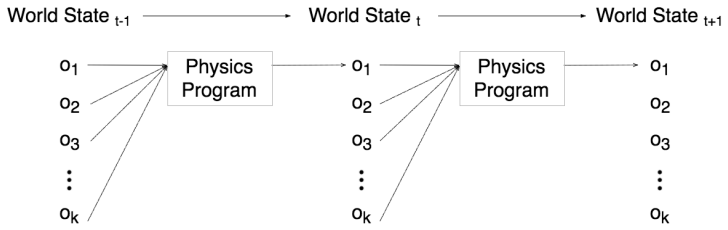

Figure 1: **Physics Programs:** We consider the space of physics programs over object-based representations under physical laws that are Markovian and translation-invariant. We consider each object in turn and predict its future state conditioned on the past states of itself and other context objects.

2 APPROACH

2.1 STATE SPACE

We make two observations (Fig. 1) in our factorization of the scene. The first regards spatially local computation. Because physics does 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 neighborhood, similar to Fragkiadaki et al. (2015). Sec. 3.5 describes using such local computation to achieve invariance to scene configuration by composing smaller objects to represent larger structures. The second regards temporally local computation. Because physics is Markovian, this prediction need only be for the immediate next timestep. Therefore it is natural to choose an object-based state representation. A state vector comprises extrinsic properties (position, velocity, orientation, angular velocity), intrinsic properties (mass, object type, object size), and global properties (gravitational, frictional, and pairwise forces) at a given time instance.

2.2 NEURAL PHYSICS ENGINE

Pairwise Factorization Letting a particular object be the *focus* object f and all other objects in the scene be *context* objects c, the NPE models the focus object's velocity $v_f^{[t+1]}$ as a composition of the pairwise interactions between itself and other neighboring context objects in the scene during time $t-1$ and t. This input is represented as pairs of object state vectors $\{(o_f, o_{c_1})^{[t-1,t]}, (o_f, o_{c_2})^{[t-1,t]}, ...\}$. As shown in Fig. 2b, the NPE composes an encoder function and a decoder function. The encoder function f_{enc} summarizes the interaction of a single object pair. The sum of encodings of all pairs is then concatenated with the focus object's past state as input to the decoder function. The decoder function then predicts the focus object's velocity $v_f^{[t+1]}$. In practice, the NPE predicts the change Δv between t and $t+1$ to compute $v^{[t+1]} = v^{[t]} + \Delta v$, and updates position using the velocity as a first-order approximation[1]. We choose to predict velocity rather than position because predicting velocity helps the network avoid memorizing the environment, whereas training the network to predict position conditions the network on the worlds in the training domain, making it more difficult to transfer knowledge across environments. We do not include acceleration in the state representation because position and velocity fully parametrize an object's state. Thus acceleration (e.g. collisions) can be learned by observing velocity for two consecutive timesteps, hence our choice for two input timesteps. We explored longer input durations as well and found no additional benefit.

Neighborhood Mask Each (o_f, o_c) pair is selected to be in the set of neighbors of f by the neighborhood masking function $\mathbb{1}\left[||p_c - p_f)|| < N(o_f)\right]$, which takes value 1 if the Euclidean distance

[1]The NPE as currently implemented also predicts angular velocity along with velocity, but for the experiments in this paper we always set angular velocity, as well as gravity, friction, and pairwise forces, to zero. We included these parameters in the implementation because in future work we are planning to test situations and scenarios in which angular velocity is important, such as block towers, magnetism. However, in the current work they are vestigial and set to zero and do not appear in the evaluation.

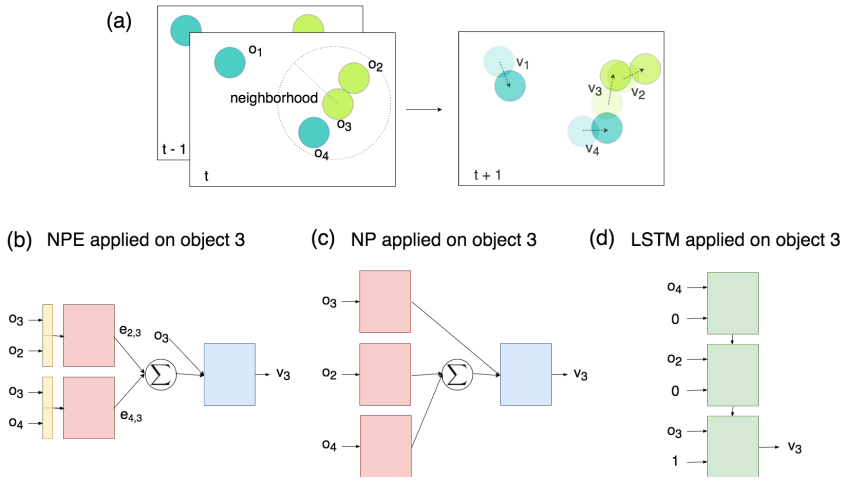

Figure 2: **Scenario and Models:** This figure compares the NPE, the NP and the LSTM architectures in predicting the velocity of object 3 for an example scenario [**a**] of two heavy balls (cyan) and two light balls (yellow-green). Objects 2 and 4 are in object 3's neighborhood, so object 1 is ignored. [**b**]: The NPE encoder comprises a pairwise layer (yellow) and a feedforward network (red) and its decoder (blue) is also a feedforward network. The input to the decoder is the concatenation of the summed pairwise encodings and the state of object 3. [**c**]: The NP encoder is the same as the NPE encoder, but without the pairwise layer. The NP decoder is the same as the NPE decoder. The input to the decoder is the concatenation of the summed context encodings and the encoding of object 3. [**d**]: We shuffle the context objects inputted into the LSTM and use a binary flag to indicate whether an object is a context or focus object.

between the positions p_f and p_c of the focus and context object respectively at time t is less the neighborhood threshold $N(o_f)$. Many physics engines use a collision detection scheme with two phases. *Broad phase* is used for computational efficiency and use a neighborhood threshold to select objects that might, but not necessarily will, collide an object. The actual collision detection is done with *narrow phase* on that smaller subset of objects, which also resolves the collisions for the objects that do collide. Analogously, our neighborhood mask implements broad phase, and the NPE implements narrow phase. The mask only constrains the search space of context objects, and the network figures out how to detect and resolve collisions. In fact, one can view the mask as a specific case of a more general attention mechanism to select contextual elements of a scene.

Function Composition Symbolic physics engines evolve objects through time based on dynamics that dictate their independent behavior (e.g. friction, gravity, inertia) and their behavior with other objects (e.g. collision, support, attraction, repulsion). Notably, in a particular object's reference frame, the forces it feels from other objects are additive. The NPE architecture incorporates several inductive biases that reflect this recipe. f_{enc} and f_{dec} induce a causal structure on pairs of objects that can be composed when reasoning about how the object behaves. We provide a loose interpretation of the encoder output $e_{c,f}$ as the *effect* of object c on object f, and require that these effects are additive as forces are. By design, this allows the NPE to scale naturally to different numbers of 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, focusing on compositional programs that reflect pairwise causal structure in object interactions.

2.3 BASELINES

The purpose of contrasting the NPE with the following two baselines is to illustrate the benefit of pairwise factorization and function composition, which are the key architectural features of the NPE. As the architectures for both baselines have been shown to work well in similar tasks, it is not immediately clear whether the NPE's assumptions are useful or necessary, so these are good

baselines to compare with. Viewed in another way, comparing with these baselines is a lesion study on the NPE because each baseline lacks an aspect of the NPE structure.

No-Pairwise The No-Pairwise (NP) baseline is summarized by Fig. 2c. The NP is very similar to the NPE but has the pairwise layer removed; otherwise the NP's encoder and decoder are the same as the NPE. Therefore the NP most directly highlights the value of the NPE's pairwise factorization. The NP is also a Markovian variant of the Social LSTM (Alahi et al., 2016); it sums the encodings of context objects after encoding each object independently, similar to the Social LSTM's "social pooling." Information for modeling how objects interact would only be present after the encoding step. Thus one interpretation for how such a structure could predict dynamics is if the encoder's object encoding comprises an abstract object representation and a force field created by that object. Therefore the NP decoder could apply the sum of the force fields of all context objects to the focus object's abstract object representation to predict the focus object's velocity. As Alahi et al. (2016) has demonstrated the Social LSTM's performance in modeling human trajectories, it is interesting to see how the same architectural assumptions performs for the physics of moving objects.

LSTM Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) have been shown to sequentially attend to objects (Eslami et al., 2016), so it is interesting to test whether a LSTM is well-suited for modeling object interactions, when the object states are explicitly given as input. From a cognitive science viewpoint, an LSTM can be interpreted as a serial mechanism in object tracking (Pylyshyn and Annan, 2006). Our LSTM architecture (Fig. 2d) accepts the state of each context object until the last step, at which it takes in the focus object's state and predicts its velocity. Because the LSTM moves through the object space sequentially, its lack of factorized compositional structure highlights the value of the NPE's function composition of the independent interactions between an object and its neighbors. Our notion of compositionality treats each object and pairwise interaction as independently encapsulated in a separate computational entity that can be reused and rearranged; the NPE encoder is a function that is applied to each (o_f, o_c) pair. This function encapsulates this computation and can be repeatedly applied for all neighboring context objects equally, such that the NPE composes this repeated encoding function with the decoder function to predict velocity. Therefore the LSTM does not exhibit this notion of compositionality because it is not designed to take advantage of the factorized structure of the scene. Unlike the NPE and NP, the LSTM's structure does not differentiate between focus and context object, so we add a flag to the state representation to indicate to whether an object is a context or focus object. We shuffle the order of the context objects to account for an ordering bias.

2.4 IMPLEMENTATION

We trained all models using the rmsprop (Tieleman and Hinton, 2012) backpropagation algorithm with a Euclidean loss for 1,200,000 iterations with a learning rate of 0.0003 and a learning rate decay of 0.99 every 2,500 training iterations, beginning at iteration 50,000. We used minibatches of size 50 and used a 70-15-15 split for training, validation, and test data.

All models are implemented using the neural network libraries built by Collobert et al. (2011); Léonard et al. (2015). The NPE encoder consists of a pairwise layer of 25 hidden units and a 5-layer feedforward network of 50 hidden units per layer each with rectified linear activations. Because we use a binary mask to zero out non-neighboring objects, we implement the encoder layers without bias such that non-neighboring objects do not contribute to the network activations. The encoding parameters are shared across all object pairs. The decoder is a five-layer network with 50 hidden units per layer and ReLU activations after all but the last layer. The NP encoder architecture is the same as the NPE encoder, but without the pairwise layer. The NP decoder architecture is the same as the NPE decoder. The LSTM has three layers of 100 hidden units and a linear layer after the last layer. It has rectified linear activations after each layer of the LSTM.

We informally explored several hyperparameters, varying the number of layers from 2 to 5, the hidden dimension from 50 to 100, and learning rates in $\{10^{-5}, 3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$. Though this is far from an exhaustive search, we found that the above hyperparameter settings work well.

3 EXPERIMENTS

Using the matter-js physics engine, we evaluate the NPE on worlds of balls and obstacles. These worlds exhibit self-evident dynamics and support a wide set of scenarios that reflect everyday physics. Bouncing balls have also been of interest in cognitive science to study causality and counterfactual reasoning, as in Gerstenberg et al. (2012). We trained on 3-timestep windows in trajectories of 60 timesteps (10 timesteps $\approx$ 1 second). For a world of k objects, we generate 50,000 such trajectories. For experiments where we train on multiple worlds together, we shuffle the examples across all training worlds and train without a curriculum schedule. All worlds have a vertical dimension of 600 pixels and a horizontal dimension of 800 pixels, and we constrain the maximum velocity of an object to be 60 pixels/second. We normalize positions to $[0, 1]$ by dividing by the horizontal dimension, and we normalize velocities to $[-1, 1]$ by dividing by the maximum velocity.

Like Battaglia et al. (2016) our predictions can be effective for a large number of time steps even though we only train to predict the immediate next time step. Plots show results over three independent runs averaged over held-out test data with different random seeds. Randomly selected simulation videos are at `https://drive.google.com/drive/folders/0BxCJLi4FnT_6QW4tcF94d1doLWs?usp=sharing`. As can be seen by the graphs in Fig. 3 (top two rows) and Fig.5, both the NPE and the NP/LSTM's predicted trajectories diverge from the ground truth, but for different reasons. As can be seen by the videos, while the NP/LSTM fail to predict plausible physical movement entirely, the NPE's predictions initially adhere closely to the ground truth, then slowly diverge due to the accumulation of subtle errors, just as the human perceptual system also accumulates errors (Smith and Vul, 2013). However, the NPE preserves the general intuitive physical dynamics that may roughly be consistent with people's intuitive expectations. Quantitative error analysis is in Fig. 6.

3.1 PREDICTION

First we consider simple worlds of four balls of uniform mass (Fig. 3a). To measure performance in simulation, we visualize the cosine similarity between the predicted velocity and the ground truth velocity as well as the relative error in magnitude between the predicted velocity and the ground truth velocity over 50 timesteps of simulation (about 5 seconds). The models take timesteps 1 and 2 as initial input, and then use previous predictions as input to future predictions. To measure progress through training, we also display the Mean Squared Error (MSE) on the normalized velocity.

3.2 GENERALIZATION AND KNOWLEDGE TRANSFER

We test whether learned knowledge of these simple physics concepts can be transferred and extrapolated to worlds with a number of objects previously unseen (Fig. 3b). The unseen worlds (6, 7, 8 balls) in the test data are combinatorially more complex and varied than the observed worlds (3, 4, 5 balls) in the training data. All objects have equal mass. The NPE's predictions are more consistent, whereas the NP and LSTM's prediction begin to diverge wildly towards the end of 50 timesteps of simulation (Fig. 3b, middle row). The NPE's performance on this extrapolation task suggests that its architectural inductive biases are useful for generalizing knowledge learned in Markovian domains with causal structure in object interactions.

3.3 NEIGHBORHOOD MASK

In Fig. 3d we vary the NPE's neighborhood threshold $N(o_f)$ and evaluate performance on the constant-mass prediction task. $N(o_f) = 2$ means that a context object is only detected if it is exactly touching the focus object. Because ball radii are 60 pixels and the maximum velocity is 60 pixels per timestep, the maximum distance two balls can initially be before touching at the next timestep is 4 ball radii. Given that velocities were sampled uniformly, it makes sense that the NPE performs well in and is robust[2] to the range $N(o_f) \in [3, 5]$, but performance drops off with smaller and larger $N(o_f)$. It is important to note that different $N(o_f)$ may work better for different domains and object geometries.

[2]The results reported in this paper were with $N(o_f) = 3.5$ ball radii, which we found initially with a coarser search than the results in Fig. 3d, although any threshold in the range $N(o_f) \in [3, 5]$ performs similarly.

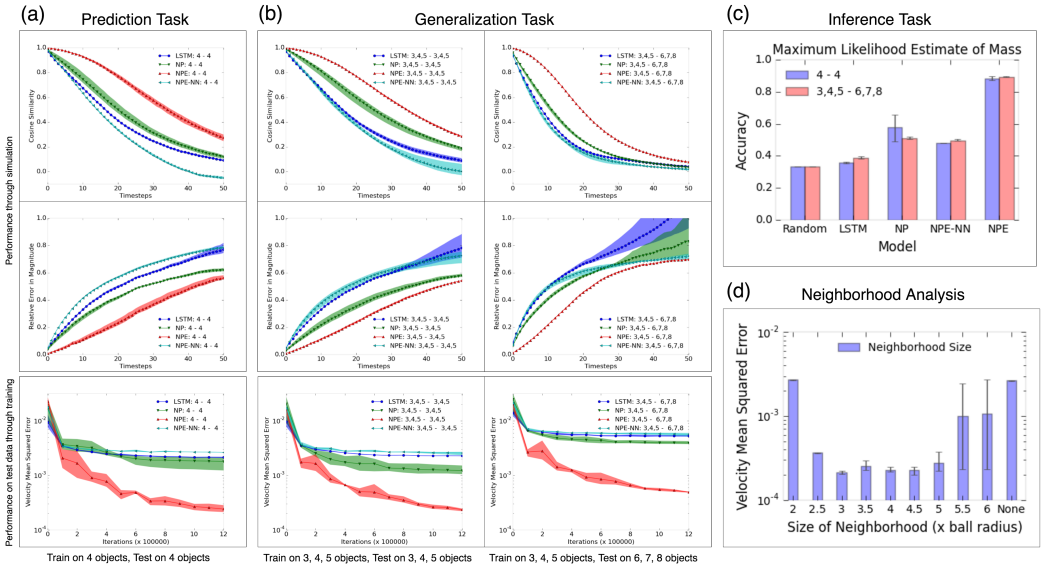

Figure 3: **Quantitative evaluation (balls):** [**a,b**]: Prediction and generalization tasks. *Top two rows*: The cosine similarity and the relative error in magnitude. *Bottom row*: The MSE of velocity on the test set over the course of training. Because these worlds are chaotic systems, it is not surprising that all predictions diverge from the ground truth with time, but NPE consistently outperforms the other two baselines on all fronts, especially when testing on 6, 7, and 8 objects in the generalization task. The NPE's performance continues to improve with training while the NPE-NN (an NPE without a neighborhood mask, see Sec. 3.3), NP and LSTM quickly plateau. We hypothesize that the NPE's structured factorization of the state space guides it from wasting time exploring suboptimal programs. [**c**]: The NPE's accuracy is significantly greater than the baseline models' in mass inference. Notably, the NPE performs just as well in worlds with greater number of objects it has trained on similarly well whether in a world it has seen before or in a world with a number of objects it hasn't trained on, further showcasing its strong generalization capabilities. The LSTM performs poorest, reaching just above random guessing (33% accuracy). [**d**]: We analyze the effectiveness of different neighborhood thresholds for the NPE on the constant-mass prediction task. The neighborhood threshold is quite robust from 3 to 5 ball radii.

We include analysis in the prediction and generalization tasks on an NPE without the neighborhood mask, the NPE-NN (NN = No Neighborhood). The neighborhood mask gives the NPE about an order of magnitude improvement in velocity prediction loss (Fig. 3a,b: bottom row and Fig. 6). While the NPE loss continues to improve through training, the NPE-NN loss quickly plateaus. It is interesting that the NPE-NN performs no better than both the NP and LSTM in predictive error, but outperforms the LSTM in mass inference. These two observations suggest that computing the interactions the focus object shares with each context object is more effective for inferring a property of the focus object than disregarding these factorized effects. They also suggest that the additional spatial structure from constraining the context space with the neighborhood mask prevents the NPE from naively finding associations with objects that cannot influence the focus object.

In our experiments, the neighborhood mask has the additional practical benefit of reducing computational complexity from $O(k)$ to $O(1)$, where k is the number of objects in the scene, because the number of context-focus object pairs the NPE considers is bounded above by the neighborhood mask. Though beyond the scope of this work, to extend the functionality of such a mask to include worlds that contain forces that act from a distance, future iterations of the NPE may investigate a more general attention mechanism that can be learned jointly with the other model parameters.

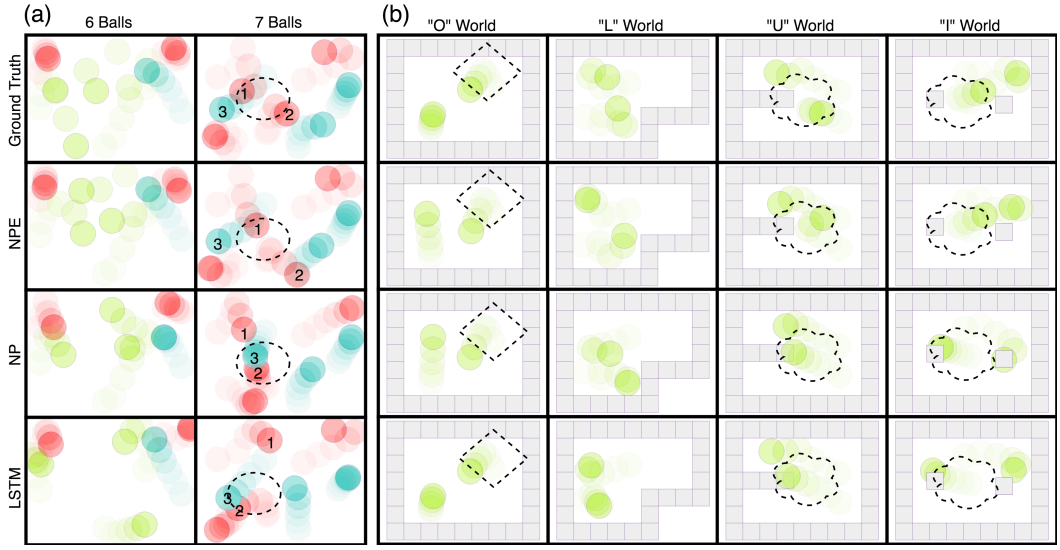

Figure 4: **Visualizations:** The NPE scales to complex dynamics and world configurations while the NP and LSTM cannot. The masses are visualized as: cyan = 25, red = 5, yellow-green = 1. [**a**] Consider the collision in the 7 balls world (circled). In the ground truth, the collision happens between balls 1 and 2, and the NPE correctly predicts this. The NP predicts a slower movement for ball 1, so ball 2 overlaps with ball 3. The LSTM predicts a slower movement and incorrect angle off the world boundary, so ball 2 overlaps with ball 3. [**b**] At first glance, all models seem to handle collisions well in the "O" world (diamond), but when there are internal obstacles (cloud), only the NPE can successfully resolve collisions. This suggests that the NPE pairwise factorization handles object interactions well, letting it generalize to different world configurations, whereas the NP and LSTM have only memorized the geometry of the "O" world.

3.4 Mass Inference

We now show that the NPE can infer latent properties such as mass. This proposal is motivated by the experiments in Battaglia et al. (2013), which uses a probabilistic physics simulator to infer various properties of a scene configuration. Whereas the physical rules of their simulator were manually pre-specified, the NPE learns these rules from observation. We train on the same worlds used in both the prediction and generalization tasks, but we uniformly sampled the mass for each ball from the log-spaced set $\{1, 5, 25\}$. We chose to use discrete-valued masses to simplify our qualitative understanding of the model's capacity to infer. For future work we would like to investigate continuously valued masses and evaluate with binary comparisons (e.g. "Which is heavier?").

As summarized by Fig. 3c and Fig. 4a, we select scenarios exhibiting collisions with the focus object, fix the masses of all other objects, and score the NPE's prediction under all possible mass hypotheses for the focus object. The prediction is scored against the ground-truth under the same MSE loss used in training. The hypothesis whose prediction yields the lowest error is the NPE's maximum likelihood estimate of the focus object's mass. The NPE achieves about 90% accuracy, meaning it has 90% probability of inferring the correct mass.

We see that a simulator with a more structured model of the world performs more accurate inferences, and that this simulation capability can be learned from observation. Though we adopted a particular parametrization of an object, the NPE is not limited to the semantic meaning of the elements of its input, so we expect other latent object properties can be inferred this way. Because the NPE is differentiable, we expect that it can also infer object properties by backpropagating prediction error to its a randomly sampled input. This would be especially useful for inferring non-categorical values, such as positions of "invisible" objects, whose effects are felt but whose position is unknown.

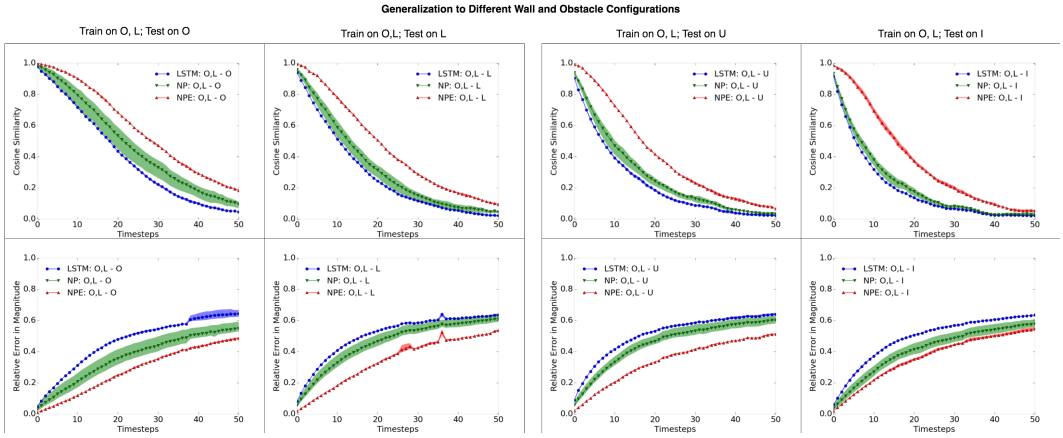

Figure 5: **Quantitative evalution (walls and obstacles):** By design, our compositional state representation simplifies the physical prediction problem to be over a focus object and differently arranged context balls and obstacles, even when the wall geometries are more complex and varied on a macroscopic scale. Therefore, it is not surprising that the models perform consistently across wall geometries. Note that the NPE consistently outperforms the other models, and this gap in performance increases with more varied internal obstacles for the cosine similarity of the velocity angle. This gap is more prominent in "L" and "U" geometries for relative error in magnitude.

3.5 Different Scene Configurations

We demonstrate representing large structures as a composition of smaller objects as building blocks. This is important for testing the NPE's invariance to scene configuration; the scene configuration should not matter if the underlying physical laws remain the same. These worlds contain 2 balls bouncing around in variations of 4 different wall geometries. "O" and "L" geometries have no internal obstacles and are in the shape of a rectangle and "L" respectively. "U" and "I" have internal obstacles. Obstacles in "U" are linearly attached to the wall like a protrusion, while obstacles in "I" have no constraint in position. We randomly vary the position and orientation of the "L" concavity and the "U" protrusion. We randomly sample the positions of the "I" internal obstacles.

We train on conceptually simpler "O" and "L" worlds and test on more complex "U" and "I" worlds. Variations in wall geometries adds to the difficulty of this extrapolation task. However, our state space representation was designed to be flexible to this variation, by representing walls as composed of uniformly-sized obstacles, just as many real-world objects are composed of smaller components. At most 12 context objects are present in the focus object's neighborhood at a time. The "U" geometries have 33 objects in the scene, the most out of all the wall geometries. As shown in Fig. 4b and 5, using such a compositional representation of the scene allows the NPE to scale to different configurations, which would not be straightforward to do without such a representation.

4 Relation to Previous Work

A recent set of top-down approaches investigate probabilistic (Battaglia et al., 2013; Bates et al., 2015; Ullman et al., 2014) and deterministic (Wu et al., 2015) game physics engines as computational models for physical simulation in humans. However, inverting a physics engine, as the Intuitive Physics Engine (IPE) (Battaglia et al., 2013) does, requires a full specification of the physical laws and object geometries. Inferring how physical laws compose and apply to a given scenario is the IPE's strength, but the nature of these physical laws are pre-defined rather than learned from observation. To manually define, for example, the asymmetrical geometry of a hammer, the fluid motion of a tape dispenser dispensing tape, or the multiple stages of opening a door by its door handle, and in addition the geometry and physical behavior of every other object in a scene, requires extensive hand engineering that is difficult to automate.

Experiments	Train - Test	LSTM		NP		NPE-NN		NPE	
Prediction Task	4 - 4	2.177e-03	2.276e-02	1.822e-03	1.923e-02	2.684e-03	2.283e-02	**2.469e-04**	**4.362e-03**
Prediction Task Variable Mass	4 - 4	3.521e-03	2.725e-02	2.534e-03	1.829e-02	4.278e-03	2.562e-02	**5.312e-04**	**6.379e-03**
Generalization Task	345 - 3	1.783e-03	1.872e-02	5.844e-04	8.118e-03	1.667e-03	1.700e-02	**1.651e-04**	**3.523e-03**
	345 - 4	2.237e-03	2.336e-02	1.172e-03	1.329e-02	2.554e-03	2.222e-02	**2.372e-04**	**4.508e-03**
	345 - 5	2.839e-03	2.909e-02	1.944e-03	1.959e-02	3.543e-03	2.810e-02	**3.069e-04**	**5.514e-03**
	345 - 6	3.757e-03	3.636e-02	2.897e-03	2.665e-02	4.542e-03	3.381e-02	**4.066e-04**	**6.676e-03**
	345 - 7	5.085e-03	4.546e-02	3.894e-03	3.395e-02	5.654e-03	3.944e-02	**4.951e-04**	**7.858e-03**
	345 - 8	6.943e-03	5.595e-02	5.091e-03	4.182e-02	6.913e-03	4.604e-02	**5.992e-04**	**9.174e-03**
Generalization Task Variable Mass	345 - 3	2.663e-03	2.218e-02	2.228e-03	1.638e-02	2.785e-03	1.913e-02	**3.546e-04**	**4.790e-03**
	345 - 4	3.588e-03	2.784e-02	3.486e-03	2.375e-02	4.291e-03	2.563e-02	**5.393e-04**	**6.215e-03**
	345 - 5	4.719e-03	3.472e-02	4.918e-03	3.164e-02	5.848e-03	3.273e-02	**6.983e-04**	**7.719e-03**
	345 - 6	6.389e-03	4.302e-02	6.733e-03	3.982e-02	7.927e-03	4.092e-02	**9.414e-04**	**9.398e-03**
	345 - 7	8.581e-03	5.276e-02	8.746e-03	4.853e-02	1.012e-02	4.998e-02	**1.196e-03**	**1.130e-02**
	345 - 8	1.153e-02	6.469e-02	1.086e-02	5.724e-02	1.244e-02	5.967e-02	**1.592e-03**	**1.367e-02**
Different Scene Configurations	OL - O	5.967e-03	5.546e-02	1.010e-03	1.358e-02	N/A	N/A	**3.338e-04**	**5.921e-03**
	OL - L	8.658e-03	6.995e-02	2.680e-03	2.663e-02	N/A	N/A	**7.117e-04**	**1.019e-02**
	OL - U	1.083e-02	7.765e-02	4.152e-03	3.201e-02	N/A	N/A	**8.193e-04**	**1.141e-02**
	OL - I	1.201e-02	7.947e-02	6.206e-03	3.565e-02	N/A	N/A	**1.605e-03**	**1.482e-02**

Figure 6: **Error analysis on velocity and position:** We summarize the error in velocity and position for each train-test variant of each experiment. Normalized velocity MSE is shown in the gray columns (multiplying these values by the maximum velocity of 60 would give the actual velocity in pixels/timestep, where each timestep is about 0.1 seconds). The white columns show the error in Euclidean distance between the predicted position and the ground truth position of the ball. These have been normalized by the radius of the ball (60 pixels), so multiplying these values by 60 would give the actual Euclidean distance in pixels. The NPE consistently outperforms all baselines by 0.5 to 1 order of magnitude, and this is also reflected in the bottom row of Fig. 3a,b. Notice that experiments with variable mass exhibit only slightly higher error than their constant-mass variants, even when the variable mass experiments contain masses that differ by a factor of 25. For the experiments with different scene configurations, we do not report error for NPE-NN; the unnecessary computational complexity of operating on over 30 objects, and the degradation in performance without this mask, evident from the other experiments, make the need for the neighborhood mask clear.

Adapting to a scene with a differentiable model such as a neural network offers a possible path to automatically adapt a general architecture to the specific physical properties of a scene without prior human specification. Thus, we view the NPE as a differentiable model that complements and builds on top of the key structural assumptions in these top-down approaches.

Bottom-up approaches have mapped visual observations to physical judgments (Lerer et al., 2016; Li et al., 2016; Mottaghi et al., 2015; 2016) or passive (Lerer et al., 2016; Srivastava et al., 2015; Sutskever et al., 2009) and action-conditioned (Agrawal et al., 2016; Finn et al., 2016; Fragkiadaki et al., 2015) motion prediction. With the exception of works as (Wu et al., 2015), the bottom-up approaches mentioned above do not infer latent properties as our model does. Because the visual and physical are not disentangled, knowledge transfer to conceptually similar worlds, such as those with different numbers of objects, has been a challenge without retraining.

Our work brings a different perspective to recent work that have begun to address this problem. Lerer et al. (2016) used a shared network for both predicting future frames and for making physical judgments. Similarly, our work presents a single model that makes physical judgments beyond frame prediction. However, while their network used a specifically designed branch to predict tower stability, our network is less task-specific and infers mass using the same architecture used for prediction. Translation invariance is an explicit assumption in our model that allows our model to exhibit greater ability to extrapolate to larger numbers of objects.

This assumption is also core to Fragkiadaki et al. (2015), which conditions motion prediction on individual objects rather than the entire scene. However, a key contrast between their and our work is that the NPE's representations operate over a more abstract symbolic state space while theirs operate over visual attention windows. They argued that the combinatorial structure from a variable number of objects with variable properties and nonlinear behavior such as collisions creates a depth

in complexity across which it is difficult to define a single state space. We presented a factorization in 2.1 that allows us to define such a state space.

If we compare the simulation videos in Fragkiadaki et al. (2015) to ours, we see some specific and significant improvements evident in our approach. NPE preserves the intuitive physical dynamics of colliding balls, while their balls exhibit less realistic behavior. For example, their model exhibits random forces acting on the balls, causing the balls to sometimes magnetically attract each other. During collisions, their balls rarely touch, but magnetically repel each other at a short distance. Their balls appear attracted to the walls and appear to bounce along the walls even when no attractive force should be present. The NPE does not exhibit these behaviors. In addition to these differences, we show strong predictive performance on generalizing to eight balls, five more than the balls in their videos. We also crucially show this performance under stronger generalization conditions, variable mass, and more complex scene configurations.

Recently, in work we only learned about after implementing our approach, Battaglia et al. (2016) in parallel and independently developed a similar architecture that they call Interaction Networks for learning to model physical systems. They show how such architectures can apply to several different kind of physical systems, including n-body gravitational interactions and a string falling under gravity. Like their work, our model can simulate over many timesteps very effectively when only trained for next-timestep prediction, and can generalize to different world configurations and different numbers of objects.

Compared to Interaction Networks, a main difference in our architecture is that ours does not take object relations as explicit input, but instead learns the nature of these relations by constraining attention to a neighborhood set of objects. Another difference is in function reuse: we demonstrated that a trained NPE model can automatically infer properties of its input such as mass without further retraining. In contrast, they train an additional classifier on top of their model to do inference. Our network produces output given inputs during prediction, but infers inputs given outputs during inference. We view the similarities between their and our work as converging evidence for the utility of object-based representations and factorized model architectures in learning to emulate general-purpose physics engines.

5 Discussion

While this paper is not the first to explore predictive models of physics, here we take the opportunity to highlight the value of this paper's contributions. We hope these contributions can seed further research that builds on themes of factorization and composition this paper proposes.

We have presented a factorization of a physical scene into composable object-based representations and also a model architecture whose compositional structure factorizes object dynamics into pairwise interactions. The architecture is designed to generalize to variable object count and variable scene configurations with only spatially and temporally local computation. It makes few but strong assumptions about the nature of objects in a physical environment. These assumptions are inductive biases that not only give the NPE enough structure to help constrain it from naively exploring suboptimal programs but also are general enough for the NPE to learn physical dynamics almost exclusively from observation.

We applied the NPE to simple two-dimensional worlds of bouncing balls ranging in complexity. We showed that NPE achieves low prediction error, extrapolates learned physical knowledge to previously unseen number of objects and world configurations, and can infer latent properties such as mass. Though we demonstrated the NPE in the balls environment with highly nonlinear dynamics and complex scene configurations, the state representation and NPE architecture we propose are quite general-purpose because they assumes little about the specific dynamics of a scene. Further work includes generalizing beyond object count to new object types and physical laws, such as in worlds with stacked block towers and liquids.

Our results invite questions on how much prior information and structure should and could be given to bottom-up neural networks, and what can be learned without building in such structure. It will be interesting to see whether similar model assumptions work for both objects and agents. This paper works toward emulating a general purpose physics engine under a framework where visual and physical aspects of a scene are disentangled. Future work also includes linking the NPE with

perceptual programs that extract properties such as position and mass from visual input. We believe the expressiveness of physics engines and the adaptability of neural networks are ingredients that will become increasingly important for modeling larger and more complex physical systems. The Neural Physics Engine proposes a possible path to link those two ingredients.

ACKNOWLEDGMENTS

We thank Tejas Kulkarni for insightful discussions and guidance. We thank Ilker Yildirim, Erin Reynolds, Feras Saad, Andreas Stuhlmüller, Adam Lerer, Chelsea Finn, Jiajun Wu, and the anonymous reviewers for valuable feedback. We thank Liam Brummit, Kevin Kwok, and Guillermo Webster for help with matter-js. M. Chang was graciously supported by MIT's SuperUROP and UROP programs.

REFERENCES

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

A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. 2016.

J. R. Anderson. *Cognitive psychology and its implications* . WH Freeman/Times Books/Henry Holt & Co, 1990.

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

P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, and K. Koray. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, 2016.

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

L. Brummitt. http://brm.io/matter-js. URL `http://brm.io/matter-js`.

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

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

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

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

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

N. D. Goodman and J. B. Tenenbaum. Probabilistic models of cognition, 2016. URL `http://probmods.org`.

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

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

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

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

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

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

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

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

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

Z. W. Pylyshyn and V. Annan. Dynamics of target selection in multiple object tracking (mot). *Spatial vision*, 19(6):485–504, 2006.

K. A. Smith and E. Vul. Sources of uncertainty in intuitive physics. *Topics in cognitive science*, 5 (1):185–199, 2013.

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

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

I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2009.

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

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

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

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