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), a framework for learning simulators of intuitive physics that naturally generalize across variable object count and different scene configurations. We propose a factorization of a physical scene into composable object-based representations and a neural network architecture whose compositional structure factorizes object dynamics into pairwise interactions. Like a symbolic physics engine, the NPE is endowed with generic notions of objects and their interactions; realized as a neural network, it can 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 the NPE's compositional representation of the structure in physical interactions improves its ability to predict movement, generalize across variable object count and different scene configurations, and infer latent properties of objects such as mass.

1 INTRODUCTION

Endowing an agent with a program for physical reasoning constrains the agent's representation of the environment by establishing a prior on the environment's physics. The agent can leverage these constraints to rapidly learn new tasks, to flexibly adapt to changes in inputs and goals, and to naturally generalize reasoning to novel scenes (Lake et al., 2016).

For example, a foundational sense of intuitive physics is a prior that guides humans to decompose a scene into objects and carry expectations of object boundaries and motion across different scenarios (Spelke, 1990). Humans perceive balls on a billiard table not as meaningless patches of color but rather as impermeable objects. They expect balls moving toward each other to bounce a certain way after a collision rather than pass through each other, crumble into pieces, or disperse into smoke. Replace one billiard ball with a bowling ball and expectations for ball-to-ball interactions will differ, but the underlying sense of inertia and collisions remain. Arrange immovable wooden obstacles on the table and expectations for how a ball's surface interacts with wood remain constant regardless of how the obstacles are arranged. The ability to plan trajectories in this space without having to relearn physics from scratch each time, regardless of whether there are three balls or eight balls, whether there are obstacles or not, whether obstacles are arranged in one way or another, whether or not the configuration of objects has been seen before, suggests that humans leverage a prior on physics to reason at a level of abstraction where objects, relations, and events are primitive.

This paper explores the question of building this prior into an agent as a program. We view this program as a simulator 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 (Anderson, 1990; Battaglia et al., 2013; Goodman and Tenenbaum, 2016). Our goal is to design a program that naturally generalizes across variable object count and different scene configurations without additional retraining. Our proposed framework, the Neural Physics Engine (NPE), outlines several ingredients useful for realizing these two generalization capabilities. We describe these ingredients in the context of a specific instantiation of the NPE applied to two-dimensional worlds of balls and obstacles.

1.1 A HYBRID DESIGN

Two general approaches have emerged in the search for a program that captures common-sense physical reasoning. The top-down approach (Bates et al., 2015; Battaglia et al., 2013; Hamrick et al., 2011; 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., 2015b; Lerer et al., 2016; Li et al., 2016; Mottaghi et al., 2015; 2016; Sutskever et al., 2009) learns to directly map 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.

The NPE takes a step toward bridging the gap between expressivity and adaptability by combining the strengths of both approaches. The NPE framework is realized as a differentiable physics simulator that combines rough symbolic structure with gradient-based learning. It exhibits several strong inductive biases that are explicitly present in symbolic physics engines, such as a notion of objects-specific properties and object interactions. Implemented as a neural network, the NPE can also flexibly tailor itself to specific object properties and dynamics of a given world through training. By design, it can extrapolate to a variable number of objects and different scene configurations with only spatially and temporally local computation.

1.2 INGREDIENTS USEFUL FOR GENERALIZATION

Our framework proposes four key ingredients useful for generalization across variable object count and different scene configurations without additional retraining. The first ingredient is the view of objects as primitives of physical reasoning. The second is a mechanism for selecting context objects given a particular object. Together, these ingredients reflect two natural assumptions about a physical environment: There exist objects and these objects interact in a factorized manner.

The third and fourth ingredients are factorization and compositionality, which are both applied on two levels: the scene and the network architecture. On the level of the physical scene, the NPE *factorizes* the scene into object-based representations, and *composes* smaller building blocks to form larger objects. This method of representation adapts to scene configurations of variable complexity and shape. On the level of the network architecture, the NPE explicitly reflects a causal structure in object interactions by *factorizing* object dynamics into pairwise interactions. The NPE models the future state of a single object as a function *composition* of the pairwise interactions between itself and other context objects in the scene. This structure serves to guide learning towards object-based reasoning and is designed for physical knowledge to transfer across variable number objects anywhere in the scene.

1.3 A STEP TOWARDS EMULATING A GENERAL-PURPOSE PHYSICS ENGINE

While previous bottom-up approaches (Sec. 4) have coupled learning vision and learning physical dynamics, we take a different approach 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. Both vision and dynamics 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 work in object detection and localization (e.g. Eslami et al. 2016) for extracting position and velocity, as well as work for extracting latent object properties (Wu et al., 2015; 2016). Therefore this paper focuses on learning dynamics in that state space, taking a small step toward emulating a general-purpose physics engine, with the eventual goal of building a system that exhibits the compo-

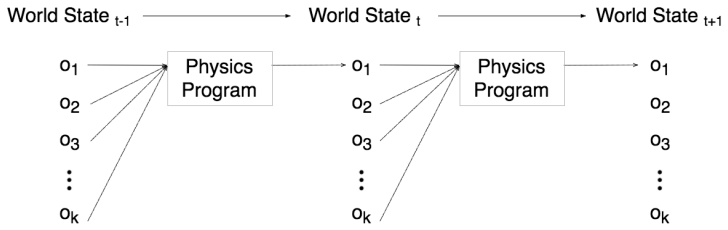

Figure 1: **Physics Programs:** We consider the space of physics programs over object-based representations under physical laws that are Markovian and translation-invariant. We consider each object in turn and predict its future state conditioned on the past states of itself and its context objects.

sitionality, modularity, and generality of a physics engine whose internal components can be learned through observation.

In Sec. 2 we present a specific instantiation of the NPE that uses a neighborhood mask to select context objects. In Sec. 3 we apply that instantiation to investigate variations on two-dimensional worlds of balls and obstacles from the matter-js physics engine (Brummitt, 2014) as a testbed for exploring the NPE's capabilities to model simple rigid-body dynamics. 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 that would be challenging to engineer into conventional physics engines. This paper establishes that important step.

2 Approach

2.1 Neural Physics Engine

We consider in detail a specific instantiation of the NPE that uses a neighborhood mask to select context objects. This section discusses each of the four ingredients of the NPE framework, that, when combined, comprise a neural network-based physics simulator that learns from observation.

Object-Based Representations 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. (2015b). Sec. 3.5 shows that when large structures are represented as a composition of smaller objects, a spatially local attention window helps achieve invariance to scene configuration. The second observation regards temporally local computation. Because physics is Markovian, this prediction need only be for the immediate next timestep, which we show in Sec. 3 is enough to predict physics effectively over long timescales. Given these two observations, 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.

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 focus object is a necessary input to the decoder because if there are no neighboring context objects, the summed encoder output would be zero. 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 predict velocity rather than position to help avoid memorizing the environment; 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.

Context Selection 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 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 uses a neighborhood threshold to select objects that might, but not necessarily will, collide an object. *Narrow phase* performs the actual collision detection on that smaller subset of objects and 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. This mask is 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. inertia) and their behavior with other objects (e.g. collisions). 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. The composition of f_{enc} and f_{dec} induce a causal structure on the pairs of objects. 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. This design 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 simulators that the NPE can learn, focusing on compositional programs that reflect pairwise causal structure in object interactions.

2.2 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 for comparison. 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. It is very similar to the NPE but does not compute pairwise interactions; otherwise its encoder and decoder are the same as the NPE's. 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. A possible mechanism for predicting dynamics with the NP is if the encoder's object encoding consists of an abstract object representation and a force field created by that object. Therefore the 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 would be interesting to see how the same architectural assumptions perform for the physics of moving objects.

[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.

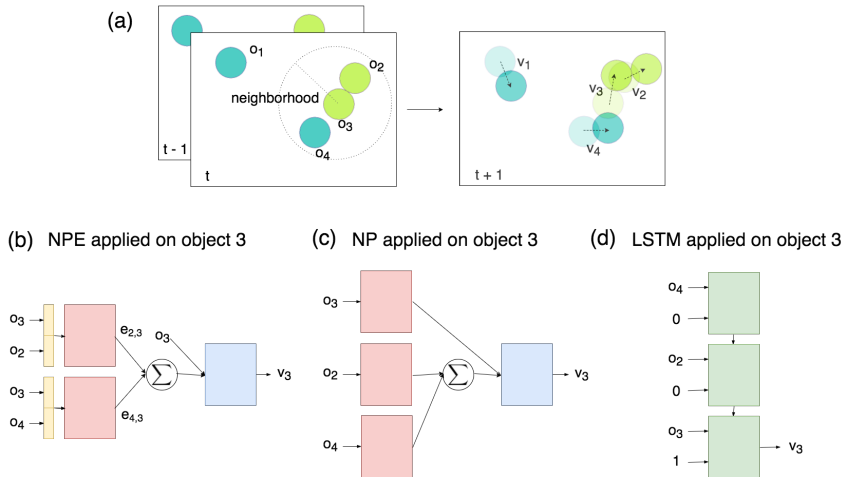

Figure 2: **Scenario and Models:** This figure compares the NPE, the NP and the LSTM architectures in predicting the velocity of object 3 for an example scenario [**a**] of two heavy balls (cyan) and two light balls (yellow-green). Objects 2 and 4 are in object 3's neighborhood, so object 1 is ignored. [**b**]: The NPE encoder consists of 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 input 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.

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 to all neighboring context objects equally, such that the NPE composes this repeated encoding function with the decoder function to predict velocity. 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.

3 Experiments

Object-based representations (ingredient 1) are necessary for the other three ingredients, and having explained the motivation for object-based representations in Sec. 1.3 and Sec. 2.1, we now analyze the other three ingredients in the context of several experiments. In the prediction task (Sec. 3.1), we first test if the NPE is even capable of predicting physics when the number of objects is held constant. In the generalization task (Sec. 3.2), we test the NPE's capability to generalize across variable object count. In the inference task (Sec. 3.3), we test if the NPE can be inverted to infer mass in both the prediction and generalization settings. In these experiments, we compare against the NPE-NN, a modified NPE without the neighborhood mask, to analyze the context selection

mechanism (ingredient 2), the NP to analyze factorization (ingredient 3), the LSTM to analyze compositionality (ingredient 4). Sec. 3.4 analyzes the neighborhood mask in depth. We test the NPE's capability to generalize across different scene configurations in Sec. 3.5.

Using the matter-js physics engine, we evaluate the NPE on worlds of balls and obstacles. These worlds exhibit nonlinear dynamics and support a wide variety of scenarios. Bouncing balls have 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 ≈ 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 those of Battaglia et al. (2016) the NPE predictions can be effective over long timescales even when the NPE is only trained to predict the immediate next time step. Randomly selected simulation videos can be found at `https://goo.gl/BWYuOF`. Plots show results over three independent runs averaged over held-out test data with different random seeds. As shown in the graphs in Fig. 3 (top two rows) and Fig. 5, both the NP and LSTM's predicted trajectories diverge from the ground truth, but for different reasons, which the videos illuminate. While the NP and 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.

3.1 Prediction task

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. 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 task

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. During simulation, 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 consistently outperforms the baselines by 0.5 to 1 order of magnitude in velocity prediction (Fig. 3b, bottom row).

3.3 Inference task

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

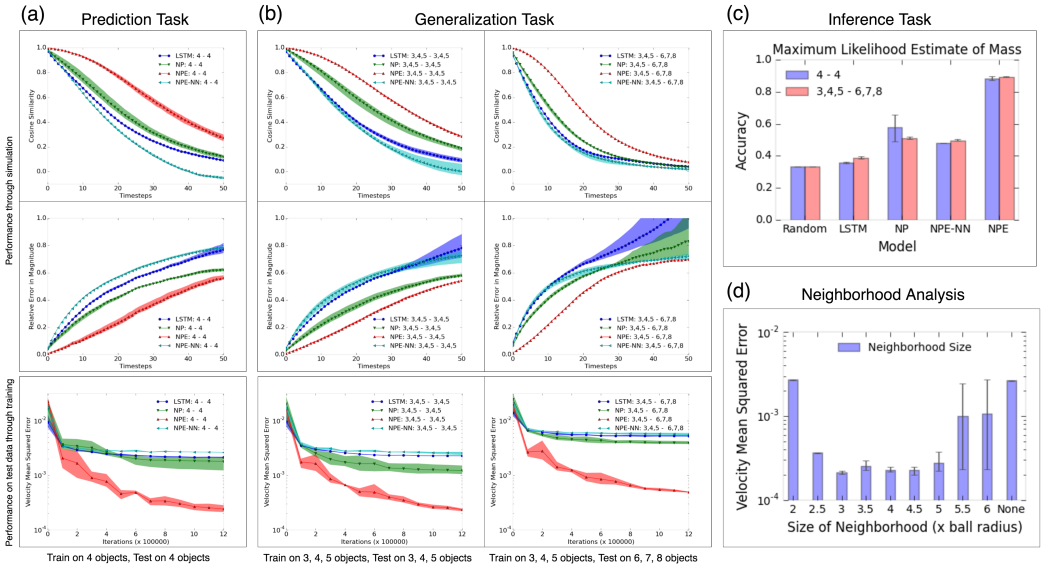

Figure 3: **Quantitative evaluation (balls):** [**a,b**]: Prediction and generalization tasks. *Top two rows*: The cosine similarity and the relative error in magnitude. *Bottom row*: The MSE of velocity on the test set over the course of training. Because these worlds are chaotic systems, it is not 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. 3.4), 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 achieves similar inference performance whether in the prediction or generalization settings, 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.

maximum likelihood estimate of the focus object's mass. Outperforming all baselines, the NPE achieves about 90% accuracy, meaning it has 90% probability of inferring the correct mass.

The NPE predicts outputs given inputs and infers inputs given outputs. 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 useful for inferring non-categorical values, such as positions of "invisible" objects, whose effects are felt but whose positions are unknown.

3.4 Neighborhood Mask

In Fig. 3d we vary the NPE's neighborhood threshold $N(o_f)$ and evaluate performance on the constant-mass prediction task. $N(o_f)$ is in units of ball radii, so $N(o_f) = 2$ 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.

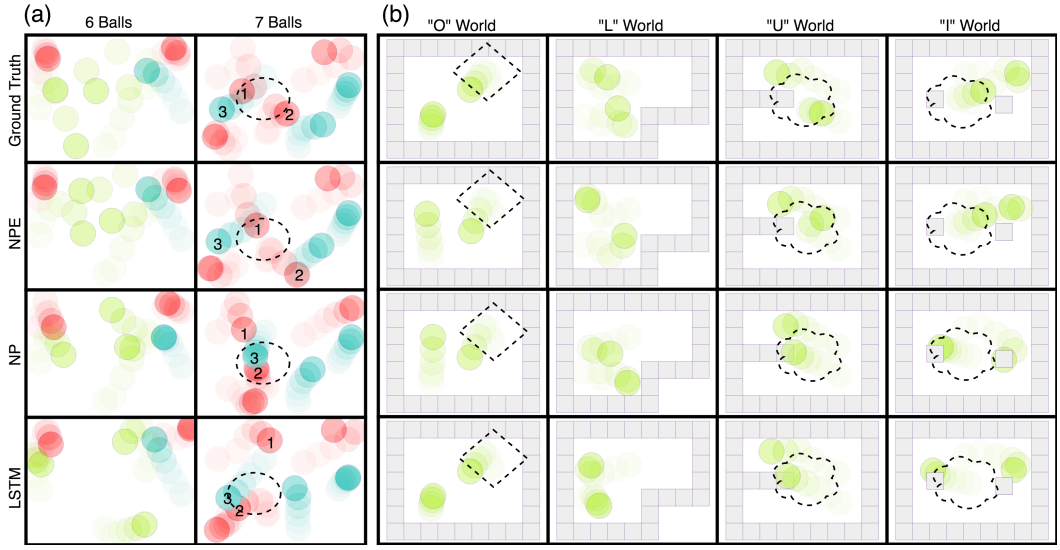

Figure 4: **Visualizations:** The NPE scales to complex dynamics and world configurations while the NP and LSTM cannot. The masses are visualized as: cyan = 25, red = 5, yellow-green = 1. [**a**] Consider the collision in the 7 balls world (circled). In the ground truth, the collision happens between 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.

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 at a constant number. Though beyond the scope of this work, to extend the functionality of such context selection mechanism to include worlds that contain forces that act from a distance, future instantiations of the NPE may investigate a more general context selection mechanism that can be learned jointly with the other model parameters.

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"

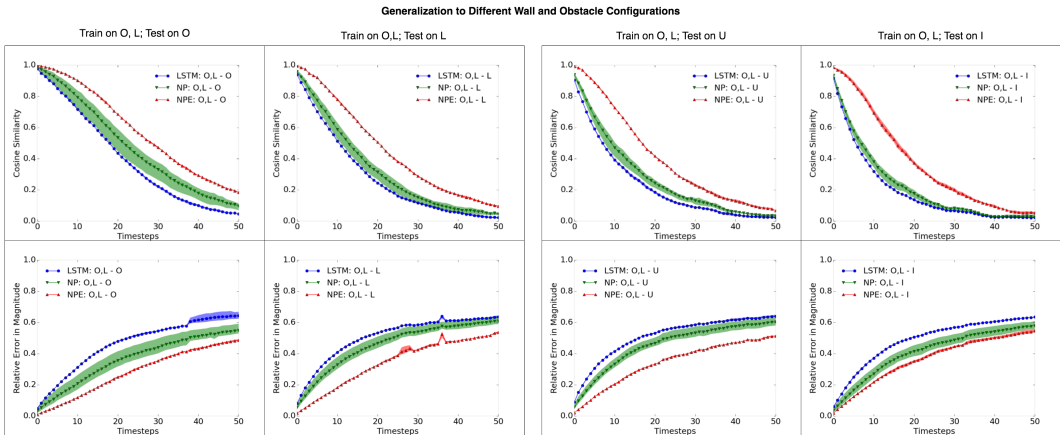

Figure 5: **Quantitative evalution (walls and obstacles):** The compositional state representation simplifies the physical prediction problem to only be over local arrangements of context balls and obstacles, even when the wall geometries are more complex and varied on a macroscopic scale. Therefore, 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.

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. 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, the NPE is robust to scenes with internal obstacles, even when it has not observed such scenes during training.

3.6 Analysis

We explain the NPE's superior performance in generalization from the perspective of context selection, factorization, and compositionality. By design, all three ingredients transform the testing data distribution to be similar to the training data distribution, such that generalization across variable object count and different scene configurations happens naturally.

Consider generalizing across variable object count. The neighborhood mask selects context objects such that the NPE need only focus on a bounded subset of the objects regardless of the total number of objects. Factorizing the scene into pairwise interactions induces a causal structure between each context object and the focus object, such that no matter the object count, this causal structure remains consistent because the input is merely a set of object pairs. Composing these pairwise interactions together with a summation encourages the encoder output to be additive, such that the decoder receives the appropriate net effect from the context objects, regardless of how many there are.

Consider generalizing across different scene configurations. Our state representation composes larger structures from smaller objects, just as many real-world objects are composed of smaller components. Therefore, even when wall geometries are complex and varied on a macroscopic scale, the input distribution to the NPE remains roughly the same, because the prediction problem still remains only over objects in a local glimpse the entire scene.

4 Related Work

Top-down and bottom-up approaches A recent set of top-down approaches investigate probabilistic game physics engines as computational models for physical simulation in humans (Bates et al., 2015; Battaglia et al., 2013; Hamrick et al., 2011; Ullman et al., 2014). However, these models

require a full specification of the physical laws and object geometries. Given such a specification, inferring *how* physical laws compose and apply to a given scenario are their strength, but automatically inferring from visual data *what* physical laws and object properties are present requires more work in inverse graphics (Chen et al., 2016; Kulkarni et al., 2014; 2015a;b; Whitney et al., 2016) and physics-based visual understanding (Brand, 1997; Wu et al., 2015; 2016). The NPE builds on top of the key structural assumptions of these top-down approaches, but its differentiable architecture opens a possible path for joint training with a vision model that can automatically adapt to the specific physical properties of the scene.

Bottom-up approaches attempt to bypass the intermediate step of finding physics representations and directly map 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., 2015b) motion prediction. Because these work historically have not been compositional in nature, they have had limited flexibility to transfer knowledge to conceptually similar worlds where the physics remain the same, but the number of objects or complexity of object configurations varies. Moreover, these approaches above do not infer latent properties as the NPE does.

Other work have taken similar hybrid approaches as the NPE, such as the NeuroAnimator (Grzeszczuk et al., 1998), one of the first work to train a neural network to emulate a physics simulator, and the interaction network (Battaglia et al., 2016), which learns to simulate physics over a graph of objects and their relations.

Sketching The NPE combines a symbolic structure that assumes generic objects and interactions with a differentiability that allows the specific nature of these interactions to be learned from training. This approach of starting with a general sketch of a program and filling in the specifics is inspired by ideas from the program synthesis community (Ellis et al., 2015; Gaunt et al., 2016; Solar-Lezama, 2008). Examples of other work that combine symbolic with neural approaches via sketching include graph-based neural networks (Jain et al., 2016; Li et al., 2015; Scarselli et al., 2009) and transforming autoencoders (Hinton et al., 2011).

Composing functions for reuse Just as the NPE repeatedly applies the same encoder to each object pair, iteratively applies itself to each object in the scene as a focus object, and recursively predicts future timesteps using predictions from previous timesteps, employing function reuse to achieve generalization is also featured in work such as Abelson et al. (1996); Andreas et al. (2016); Lake et al. (2015); Reed and de Freitas (2015); Socher et al. (2011). These work all assemble small subprograms to form larger programs. The NPE also dynamically composes its internal modules (encoder and decoder) based on the number of objects and the arrangement of context objects.

Object-based approaches Fragkiadaki et al. (2015b) and Battaglia et al. (2016) are two notably similar work in the sense that our work and theirs all take an object-based approach to model the bouncing balls environment. Our work was inspired by Fragkiadaki et al. (2015b)'s iterative approach to predicting the motion of each object in turn, conditioned on a context. The key contrast is that their model assumes no relational structure between objects beyond a visual attention window centered around the focus object, whereas ours explicitly processes the interaction between the focus and each context object.

If we compare their simulation videos (Fragkiadaki et al., 2015a) to ours, we see some specific and significant improvements evident in our approach. For example, in their work, the balls appear attracted to each other and to the walls; the balls appear to bounce along the walls even when no attractive force should be present. The balls rarely touch during collisions, but magnetically repel each other when at a short distance. The NPE does not exhibit these behaviors and tends to preserve the intuitive physical dynamics of colliding balls. 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, Battaglia et al. (2016) independently and in parallel developed an architecture that they call the *interaction network* for learning to model physical systems. They show how such an architecture can apply to several different kinds 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 the interaction network, 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 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. Their work also exhibits the four ingredients in our framework, and we view the similarities between their and our work as converging evidence for the utility of object-based representations and compositional model architectures in learning to emulate general-purpose physics engines.

5 DISCUSSION

While this paper is not the first to explore learning a physics simulator, 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 the NPE framework this paper proposes.

We showed that object-based representations, a context selection mechanism, factorization, and compositionality are useful ingredients for learning a physics simulator that generalizes across variable object count and different scene configurations with only spatially and temporally local computation. This generalization is possible because these ingredients transform the testing data distribution to be similar to the training data distribution.

The NPE 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 to model physical phenomena in terms of objects 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. We compared against several baselines designed to test the ingredients of the NPE framework and found superior performance when all these ingredients are combined in the NPE. Though we demonstrated the NPE in the balls environment with nonlinear dynamics and complex scene configurations, the state representation and NPE architecture we propose are quite general-purpose because they assume little about the specific dynamics of a scene.

This paper works toward emulating a general purpose physics engine under a framework where visual and physical aspects of a scene are disentangled. Next steps include linking the NPE with perceptual models that extract properties such as position and mass from visual input. Learning to simulate is unsupervised learning of the structure of the environment. When a simulator like the NPE is incorporated into an agent in the context of model-based planning and model-based reinforcement learning, it becomes a prior on the environment that guides learning and reasoning. By combining the expressiveness of physics engines and the adaptability of neural networks in a compositional architecture that supports generalization in fundamental aspects of physical reasoning, the Neural Physics Engine is an important step towards lifting an agent's ability to think at a level of abstraction where the concept of physics is primitive.

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. This work was supported MIT's SuperUROP and UROP programs, and by the Center for Minds, Brains and Machines under NSF STC award CCF-1231216 and an ONR grant N00014-16-1-2007.

REFERENCES

H. Abelson, G. J. Sussman, and J. Sussman. *Structure and interpretation of computer programs*. Justin Kelly, 1996.

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.

J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT*, pages 1545–1554, 2016.

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.

M. Brand. Physics-based visual understanding. *Computer Vision and Image Understanding*, 65(2): 192–205, 1997.

L. Brummitt. http://brm.io/matter-js, 2014. URL `http://brm.io/matter-js`.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

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

K. Ellis, A. Solar-Lezama, and J. Tenenbaum. Unsupervised learning by program synthesis. In *Advances in Neural Information Processing Systems*, pages 973–981, 2015.

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. Intuitive physics. `https://sites.google.com/site/intuitivephysicsnips15/`, 2015a. (Accessed on 03/03/2017).

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

A. L. Gaunt, M. Brockschmidt, R. Singh, N. Kushman, P. Kohli, J. Taylor, and D. Tarlow. Terpret: A probabilistic programming language for program induction. *arXiv preprint arXiv:1608.04428*, 2016.

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`.

R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 9–20. ACM, 1998.

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

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.

A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

T. D. Kulkarni, V. K. Mansinghka, P. Kohli, and J. B. Tenenbaum. Inverse graphics with probabilistic cad models. *arXiv preprint arXiv:1407.1339*, 2014.

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, 2015a.

T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2530–2538, 2015b.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 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.

Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

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.

S. Reed and N. de Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

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

R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. 2011.

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

E. S. Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990.

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.

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.

W. F. Whitney, M. Chang, T. Kulkarni, and J. B. Tenenbaum. Understanding visual concepts with continuation learning. *arXiv preprint arXiv:1602.06822*, 2016.

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.

J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *British Machine Vision Conference*, 2016.

A 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 encoder activations. The encoding parameters are shared across all object pairs. The decoder is a five-layer network with 50 hidden units per layer and rectified linear 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.

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.

B QUANTITATIVE ANALYSIS

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.