See, feel, act: hierarchical learning for complex manipulation skills with multisensory fusion

Science Robotics
AAAS

Manuscript Template

FRONT MATTER

See, Feel, Act: Hierarchical Learning for Complex Manipulation Skills with Multi-sensory Fusion

Authors
N. Fazeli[1]*, M. Oller[1], J. Wu[2], Z. Wu[2], J. B. Tenenbaum[2], A. Rodriguez[1]

Abstract

Humans are able to seamlessly integrate tactile and visual stimuli with their intuitions to explore and execute complex manipulation skills. They not only see, but feel their actions. Most current robotic learning methodologies exploit recent progress in computer vision and deep learning to acquire data-hungry pixel-to-action policies. These methodologies do not exploit intuitive latent structure in physics or tactile signatures. Tactile reasoning is omnipresent in the animal kingdom, yet it is under-developed in robotic manipulation. Tactile stimuli are only acquired through invasive interaction, and interpretation of the data-stream together with visual stimuli is challenging. In this paper, we propose a methodology to emulate hierarchical reasoning and multi-sensory fusion in a robot that learns to play Jenga, a complex game that requires physical interaction to be played effectively. The game mechanics are formulated as a generative process using a temporal hierarchical Bayesian model, with representations for both behavioral arch-types and noisy block states. This model captures descriptive latent structures, and the robot learns probabilistic models of these relationships in force and visual domains through a short exploration phase. Once learned, the robot uses this representation to infer block behavior patterns and states as it plays the game. Using its inferred beliefs, the robot adjusts its behavior with respect to both its current actions and game strategy, similar to the way humans play the game. We evaluate the performance of the approach against three standard base-lines and show its fidelity on a real-world implementation of the game.

Summary

Robot leverage hierarchical Bayesian representation with multi-sensory observations to play the game of Jenga.

[1] Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 02139, USA.
* Corresponding author, email: nfazeli@mit.edu
[2] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 02139, USA.

MAIN TEXT

Introduction

Humans, even young children, learn complex tasks in the physical world by engaging richly with the physics of the world (1). This physical engagement starts with our perception, and extends into how we learn, perform, and plan actions. We seamlessly integrate touch and sight in coming to understand object properties and relations; allowing us to intuit structure in the physical world and build physics representations that are central to our ability to efficiently plan and execute manipulation skills (2).

While learning contact rich manipulation skills, we face two important challenges: active perception and hybrid behavior. In the former we ask: how do we use temporal tactile and visual information gathered by probing our environment through touch to learn about the world. In the latter we ask: how do we effectively infer and learn multi-modal behavior in order to control touch. These two challenges are central to mastering physical interactions. Jenga is a quintessential example of a contact rich task where we need to interact with the tower to learn and infer block mechanics and multi-modal behavior by combining touch and sight.

Current learning methodologies struggle with these challenges and have not exploited physics nearly as richly as we believe humans do. Most robotic learning systems still use purely visual data, without a sense of touch; this fundamentally limits how quickly and flexibly a robot can learn about the world. Learning algorithms that build on model-free reinforcement learning methods have little to no ability to exploit knowledge about the physics of objects and actions. Even the methods using model-based reinforcement learning or imitation learning have mostly used generic statistical models that do not explicitly represent any of the knowledge about physical objects, contacts, or forces that humans have from a very early age. As a consequence, these systems require far more training data than humans do to learn new models or new tasks, and they generalize much less broadly and less robustly.

In this work, we propose a hierarchical learning approach to acquiring manipulation skills. In particular, we pose a top-down bottom-up (7-10) learning approach to first build abstractions in the joint space of touch and vision that are then used to learn rich physics models. We use Jenga as a platform to compare and evaluate our approach. We have developed a simulation environment in which we compare the performance of our approach to three other state-of-the-art learning paradigms. We further show the efficacy of the approach on an experimental implementation of the game.

Our proposed approach draws from the notion of an "intuitive physics engine" in the brain that may be the cause of our ability to integrate multiple sensory channels, plan complex actions (3–5), and learn abstract latent structure (6) through physical interaction, even from an early age. Humans learn to play Jenga through physics-based integration of sight and touch: vision provides information about the location of the tower and current block arrangements, but not about block interactions. The interactions are dependent on minute geometric differences between blocks that are imperceptible to the human eye. Humans gain information by touching the blocks and combining tactile and visual senses to make inferences about their interactions. Coarse high-level abstractions such as “will a block move” play a central role in our decision making and are

possible precisely because we have rich physics-based representations. We emulate this hierarchical learning and inference in the robotic system depicted in Fig. 1A using the AI schematically shown in Fig. 1B. To learn the mechanics of the game, the robot builds its physics-based representation from the data it collects during a brief exploration phase. We show that the robot builds purposeful abstractions that yield sample-efficient learning of a physics model of the game that it leverages to reason, infer, and act to play the game.

Results

In this study, we demonstrate the efficacy and sample-efficiency of a novel hierarchical learning approach to manipulation on the challenging game of Jenga. To this end, we first outline the task and evaluation metric. Next, we present quantitative results for our method and three competitive state-of-the-art alternative approaches in simulation. We then demonstrate the fidelity of our approach on a real-world implementation of the game. We end the section with an analysis of the physics and abstractions learned using the proposed approach.

Evaluation Metric

Jenga is a particularly interesting instance of physical manipulation games where mastering contact interactions is key to game-play. Block mechanics are not discernible from just perception; rather, intrusive interaction together with tactile and visual feedback are required to infer and reason about the underlying latent structure, i.e. the different types of block mechanics.

The complex, partially observable, and multi-modal mechanics of the blocks in Jenga poses a significant challenge to robots learning to play the game. These challenges are not exclusive to this game and exist in many manipulation skills such as assembly and tool use. As such, progress in effective learning of these skills is important and necessitates rich models and policies.

In this study, we evaluate the robot's ability to play the game by counting the number of successful consecutive block extractions in randomly generated towers. This metric evaluates a model and/or policy's ability to account for the complex and time-varying mechanics of the tower as the game progresses. This metric, and our study, emphasizes physics modeling and does not explicitly evaluate the adversarial nature of the game.

Task Specifications

In this subsection, we specify the sensing modalities, actions, and rules by which the robot is allowed to play the game in both simulated and real environments:

- **Sensing:** The robot has access to its own pose, the pose of the blocks, and the forces applied to it at every time-step. The simulated robot observes these states directly, whereas the experimental robot has access to noisy estimates.
- **Action primitives:** The robot utilizes two “primitive” actions, push and extract/place. Using the push primitive, the robot first selects a block and moves to a collision-free configuration in plane. The robot then selects a contact location and heading, and pushes for a distance of 1 mm and repeats. The action is considered complete if either the robot chooses to retract or a maximum distance of 45 mm is reached. The extract/place primitive searches for a collision-free grasp of the block and places it on top of the tower at a

random unoccupied slot. The extract/place primitives are parametric and computed per call, as such they are not learned.

- **Base exploration policy:** The robot has access to a base exploration policy for data collection. This policy randomizes the push primitive by first selecting a block at random, then executing a sequence of randomized contact locations and headings.
- **Termination criteria:** A run, defined as an attempt at a new tower, is terminated when one of the following conditions is met: i) all blocks have been explored, ii) a block is dropped outside the tower or the tower has toppled.
- **Tower and robot specifications:** The simulated tower is composed of the same number and similar distribution of movable vs. immobile blocks as the real tower. This is due to slight perturbations to weight distribution resulting from small tolerances in the height of the blocks. The relative dimensions of the tower and the end-effector are consistent for both environments.

The robot's nominal execution loop is to select a block at random and attempt the push primitive. During the push primitive, the robot either choses push poses and headings or retracts. If the block is extracted beyond ¾ of its length, the extract/place primitive is invoked. During a run, the nominal execution loop is continued until a termination criterion is met. A key challenge is that movable vs immobile pieces are indistinguishable prior to contact and the robot needs to control the hybrid/multimodal interaction for effective extraction without causing damage to the tower. If the damage compounds then the tower loses integrity and termination criteria are met earlier. As such, this problem is a challenging example that motivates the need for abstract reasoning with a fusion of tactile and visual information with a rich representation of physics.

Simulation

Fig. 2A depicts our Jenga setup in the MuJuCo (11) simulation environment. We use the simulation environment to compare the performance of our proposed approach (Hierarchical Model Abstractions -- HMA) to several standard baselines, Fig. 2B. Specifically, we chose a feed-forward neural network as a representative non-hierarchical model-based approach, a mixture of regressions (MOR) model as a generic hierarchical model-based approach, and the proximal policy optimization (PPO) (12) implementation of reinforcement learning as a model-free approach. All models have access to the same set of states, actions, and model predictive controller (details in Materials and Methods section). Fig. 2C depicts the schematics of our proposed approach and the MOR models. The MOR model makes use of latent variables l_t, and our HMA model uses abstractions denoted with c_t. The states and noisy observations are denoted by s_t and z_t, respectively.

Fig. 2B shows the number of blocks successfully extracted in sequence as a function of the number of samples used to learn either a model (HMA, MOR, NN) or a policy (RL). A sample is an incremental slice of the push trajectory. For the model-based approaches, the samples are collected during an exploration phase in which the robot interacts with the tower and collects states and actions. The exploration phase follows the robot's nominal execution loop using the exploration policy. Once data collection is complete, a model is trained and its fidelity is evaluated in a model predictive control framework per number of samples over an unseen set of test towers. For the purpose of evaluation, the test towers were uniform for all models. For

reference, a complete push is composed of 45 steps where a sample is measured at each step. A single sample takes approximately 2 seconds to collect experimentally and 0.3 seconds in simulation. The robot can interact with a total of 45 blocks for a new tower. We found empirically that approximately 47% of pieces move in a real-world random tower and emulated this ratio in the simulation environment. Consequently, the robot can extract 21 blocks in expectation for an unperturbed tower of 3 by 18 (the last 3 layers are prohibited by the rules). This value provides a reasonable goal against which performance can be evaluated.

The proposed approach reaches the expected number of successful consecutive extractions within 100 samples. The mixture of regressions model is next to achieve the maximum score, requiring an order of magnitude more samples. Interestingly, the feed-forward neural network saturated in performance, falling short of the expected maximum extractions. Upon closer inspection, we found that this model was unable to reliably predict the multi-modal interactions and either behaved too conservatively or too recklessly. The RL algorithm was the slowest in convergence. Per run, all approaches were presented with new towers at random which explains, in part, why the RL algorithm takes a large number of samples to converge as the space of possible tower configurations is very large.

Experiments

In our experimental setup, the robot has access to an Intel RealSense D415 camera (RGB) and an ATI Gamma 6-axis Force-Torque sensor mounted at the wrist, Fig. 1A. These two modalities provide noisy approximations of the current pose of the pieces in the tower and the forces applied to the robot (details in the methods section). We use the robots forward kinematics to estimate the pose of the gripper. To estimate the pose of the fingertips, we compute the deflection of the fingers with respect to the gripper using the measured force applied to the finger and known compliance parameters. We use the experimental setup to demonstrate the fidelity of the proposed approach.

The failure criteria in the experimental setting is expanded to include tower rotation or displacements exceeding 15 degrees and 10 mms respectively. This criterion is imposed by the poor predictions made in the vision systems beyond these values. The exploration strategy is also modified to include a hand-coded supervisory algorithm that attempts to mitigate damage to the tower using measurements of poses and forces but is tuned to allow mistakes. Table 1 shows the robot's performance prior to and post exploration. Here a successful push is one in which the robot is able to push a block to a desired end-goal without dropping it outside the tower or causing excessive damage. A successful extraction is one in which the robot is able to pull the block free after a push without damaging the tower and a successful placement is placing the block on top of the tower without damage.

The robot shows an appreciable improvement in block extraction, from 56% to 88%. The most noticeable gain is in side-block extraction, doubling the success rate from 42.6% to 78.3%. The results suggest that middle-block extraction is considerably easier than the side blocks due to its constrained motion and the favorable weight distribution of the tower. The robot is able to displace 42.7% of the blocks, close to the empirical average of a random tower.

The two main failure modes for the extraction of blocks are: i) excessive forces applied to the tower (characteristic of failing to identify block behaviors), and ii) poorly controlled extraction of blocks (characteristic of poor predictive ability). The first failure mode often results in either a large tower perturbation such that the vision system is no longer reliable or the tower collapses. The second failure mode often leads to either blocks dropping outside the tower or ending in configurations that are either difficult to grasp or occluded to the camera.

Model Learning

In this study, we represent the physics of the tower using a hierarchical probabilistic model (Fig. 2C) with the structural composition similar to that of Dynamic Bayesian Networks. We use a top-down bottom-up learning approach to learn abstractions and physics for the tower. This methodology is inspired by models of cognition, in particular “concept learning", and the intuitive abstractions that humans develop to facilitate complex manipulation skills (see Discussion section).

In top-down learning, the objective is to build abstractions from the physics of the tower. This approach is an instance of latent variable learning, where the variable identifies the type of macro behavior. Specifically, in top-down learning abstractions are acquired prior to learning detailed motion models and explicitly encode temporal macro behaviors of blocks. Fig. 3 shows a low dimensional representation of the recovered abstractions through clustering in the relevant features space (details in Materials and Methods). As the clusters have no labels, we intuited their semantic meanings from inspection of trajectory traces in force and visual domains. The green cluster denotes traces where the robot is not in contact with any block, in particular at the beginning of the trajectory where measured forces are negligible and blocks do not move. The gray cluster denotes blocks that resist motion and are stuck, exhibiting large resistive forces and little to no translation. The blue cluster denotes blocks that move fairly easily (large displacements) and exhibit negligible resistance. The yellow cluster denotes blocks that move, but offer meaningful resistance to the robot.

Bottom-up learning refers to learning explicit state-transition models, factored by the abstractions, using sensory data. We use a probabilistic model, here a Bayesian Neural Network (BNN), to model the conditional distribution of future states given current states and actions in the joint force and visual domains (see Material and Methods). The BNN is trained on the data collected during exploration.

Fig. 4 depicts two examples of the physics learned by these models. Fig. 4A depicts the analytical friction cone between the fingertip and block overlaid with the predicted normal and tangential forces given their current measured values. This cone is computed assuming Coulomb friction and rigid point contact between the fingertip and block. Under these assumptions, any transferable force between the finger and block must lie on the boundary or in the interior of these cones. The predictions of the models are in good agreement with the cone, implying that it has learned some latent representation of friction. We note that the friction cone is invariant to the abstractions and the predictions of the models reflect this fact well, implying coherency across models and abstractions.

Fig. 4B shows the resistive force of blocks as a function of the tower height. The model is able to capture the intuitive tendency of block to resistance to decrease with tower height (due to the decrease in effective remaining weight on top of each block). The abstractions facilitate this by factoring the state-space between macro-block behaviors and efficiently differentiating between movable vs immobile blocks.

Inference, Controls, and Planning

The abstractions offer coarse but intuitive and interpretable explanations of physics modes. As such, knowledge of the particular mode can be effectively leveraged for controls and planning.

As a first step, the robot must infer the abstraction from noisy sensor measurements. Once learned, our representation provides a joint distribution over the states and abstractions, i.e. a generative probabilistic model of physics. We use Markov Chain Monte Carlo sampling with Hamiltonian dynamics to approximate the posterior distribution over states and abstractions given noisy measurements. Fig. 5 shows two examples of the inferences of block types made by the robot during a push trajectory. The force and displacement plots show the prediction of the state for each abstraction. The abstraction probabilities are directly related to how well they are able to explain the observations.

In the case of a block that does not move, Fig. 5A, as there are no changes initially in force or configuration, the robot assigns the most probability mass to "no block" and some probability to "easy move" as the force measurements match fairly well. As the force climbs but configuration does not change, the probability of "no move" and "hard move" increase. The "hard move" abstraction does not predict the continued climb in force with no displacement and loses likelihood in the next few steps.

Fig. 5B shows a block that moves with very little resistance. During the initial phase, due to the lack of force and displacement, the robot assigns the most probability to "no block". As the push continues, displacements increase while forces do not, shifting the belief to "easy move" with very low probabilities for the other abstractions.

Qualitatively, the inference is mostly driven by force measurements during the initial phase of a push because of the relatively small changes in configuration. Critically, the perception algorithm has the highest uncertainty in the beginning of the push as the mask computed for the block is the least informative and many hypotheses explain the current pose. Configurations play a more decisive role further along the push as they become more certain and discriminative between abstractions. This observation highlights the importance of inference and modeling in the joint domain of visual and tactile modalities.

To use the abstractions in a control or decision-making framework, we first need an association between desirable abstractions and actions. To incorporate this information, we add a cost term to the MPC controller, where actions resulting in increased probability of "no move" are penalized. This approach is very similar to standard reward shaping for the model-free reinforcement learning frameworks. A desired policy is one that leads to minimal damage to a tower. We impose this by assigning a large cost to tower perturbations which is effectively an implicit encoding of our human intuition.

Fig. 6 shows an example trajectory for the block and the controller actions. Here, the robot attempts to push the block to a desired end-configuration using its MPC controller. Using the inferred abstraction and states, the robot forward predicts the costs of sequences of actions over a time-horizon, then executes the first action from the sequence incurring the smallest cost and repeats. The fidelity of the representation in forward prediction coupled with knowledge of the abstraction provides effective fine-grain control of block motions. This in turn significantly mitigates dropping blocks outside the tower or toppling, allowing runs to continue for longer.

Discussion

In this paper, we take a top-down bottom-up approach to manipulation skill learning. This approach to top-down learning is inspired by "concept learning" as studied in cognitive science. A "concept" is a type of abstraction composed of a typical feature-set. We naturally categorize objects and ideas into concepts, for example doors and happiness. Concepts are not measurable, rather they provide useful abstractions for categorization without the need to specify fine-grained details of instances.

When playing Jenga, we intuitively categorize blocks based on their general behaviors. For example, we may refer to blocks that do not move and exhibit large resistance (the typical feature-set) as "stuck". From experience we learn that this type of block does not help us make progress in the game. We infer connections between concepts and our objectives but these abstractions are made independent of the task, rather they are a latent property of the physics. These coarse abstractions organize our reasoning and facilitate decision-making.

From a robotics perspective, there are two important benefits to learning a few general concepts then learning fine-grained physics models. First, by virtue of implicitly factoring the state-space and sequentially learning physics using simpler local models, we can significantly increase the sample-efficiency of learning (Fig. 2B). Second, concepts may capture interpretable modalities in physics that can be used in controls and planning (Fig. 3). In this study, we take a nongoal-directed perspective, first determining these abstractions then learning models. This approach is related to (28) where a goal-directed latent space representation of the task is inferred in the context of learning from demonstration. Certain manipulation tasks, such as meal preparation, may benefit from other abstractions such as soft or hard. In (27), a latent representation of physical attributes of food items is learned that are then used to prepare a salad.

Mixture models are an alternative approach to learning latent states and predictive models. These models are particularly effective in optimizing for predictive ability by simultaneously computing the number of mixtures and model parameters to maximize a likelihood score. A challenge in mixture models is making effective use of the latent variables for model-based controls and planning, in particular when the number of mixtures is large. For example, in the simulation environment the number of mixtures ranges from 2 to 11 (determined using 5-fold cross-validation) while the underlying latent structure does not provide additional insight that we found useful to controls or decision making.

An additional technical challenge in mixture model learning is parameter estimation. Expectation-maximization is commonly used for estimation but can be sensitive to output variance, initial conditions, and local-minima. Sample-based variational inference approaches do not scale well

with data size and model dimensions, in particular due to the simplex constraint on mixture weights and poor mixing of the samples due to the complex topology of the posterior distribution. In both instances, the potentially large number of mixtures further increases the parameter space, requiring more data to learn and partially explains the results of Fig. 2B.

Mixture model learning is closely related to idea of dynamical mode learning. Dynamic modes are partitions in state-space for which transitions are smooth. Inferring dynamic modes is an important challenge in robotics, however, the level of detail provided by such a description may not be necessary for successful task execution. For example, for a moving block in the Jenga tower there are at least 8 distinct contact formations, 4 of which are unstable, and each has its own dynamics. The subtle differences in the modes renders them difficult to infer and significant expert knowledge is required to render them useful. Learning coarse abstractions can provide sufficiently descriptive representation of the interactions.

An alternative approach to model-based manipulation skill learning is policy learning. Recent trends in reinforcement learning for manipulation have mostly focused on learning in the visual domain, leveraging advances in deep reinforcement learning and computer vision algorithms (19, 20). These algorithms are used to produce autonomously acquired manipulation policies that map from the pixel domain of images to robot actions. Real-world policies learned in the image domain have been shown for planar pushing, bin picking, and insertion (21–25). These approaches work well for tasks in which the visual data-stream provides sufficient information and data collection can be automated effectively.

There are three important challenges in learning policies for manipulation. First, many contact rich manipulation skills are difficult to automate for large-scale data collection. Jenga is a prime example where tower resets are time-intensive; therefore, sample-efficiency is an important concern. Second, Sim-to-Real transfer of policies remains challenging as most simulators use computationally efficient but inaccurate models of frictional interaction (26). In the case of Jenga, the block motions rely on very fine micro-interactions and pressure distribution variations that are not observable or measurable in practice. Third, tactile information is often intermittent, i.e. making and breaking contact over a short duration. Consequently, the effective integration of tactile information together with the persistent visual stream is challenging. These challenges are prevalent in many manipulation tasks such as small parts assembly, warehouse logistics, and disaster response. In such tasks, fine level reasoning for contact is critical and mistakes often incur large costs.

The interplay between tactile and visual information is central manipulation. These two modalities act as complements and provide information in the absence of one-another, for example, visual occlusions are a common part of Jenga, were blocks are occluded during manipulation but the tactile modality continues to provide information. Further, tactile feedback can provide high-resolution local information to complement the global but coarse information from vision. In the experimental setting, our perception system is susceptible to errors in pose estimation during the initial phase of the push as the total visible mask of the block is small and several hypotheses may be equally likely. The force-stream and proprioception, together with their temporal variation increase the likelihood of plausible hypotheses for poses despite the occlusion of the end-effector due to the tower.

Building coarse abstractions in the joint domain of tactile and visual feedback is central to our approach. These abstractions are partially driven by the temporal change in these signals, motivated by time-varying multi-modal mechanics. The effective use of both modalities together with the abstractions leads to significant sample-efficiency, allowing the experimental robot to learn tower physics entirely from real-world experience. Furthermore, the abstractions are effectively used of controls and planning, significantly improving the performance of the system. In particular, for Jenga our controller is able to explicit use the relation between inferred stuck pieces and tower perturbation to mitigate damage.

Materials and Methods

System architecture

In this subsection, we describe the operation of the machine intelligence depicted in Fig. 1B. At a given step, the system records an RGB image using the camera which is processed by the perception algorithm to recover the 6 dof pose of the block. Concurrent with this operation, a low-pass filtered measurement of the force-torque measurement is also recorded. The measured forces and poses are added to a buffer of observations. Next, the inference block uses the observations in the buffer and the learned physics model to estimate the latent states and abstractions.

The estimated states and abstractions are passed to the model predictive controller, where an optimal action (that may involve retraction) is computed using the forward roll-outs of the learned physics model. The computed action is then passed to the robot for execution and the loop is repeated. The game play unit is a hand-coded state-machine that sequences the action primitives and monitors the state of the tower using sensor measurements and the inference block. This loop is repeated until a termination criterion is met. In the following subsections we detail the mechanisms of each block.

Notation

Let $q_t = (x, y, z, \theta_z)_t$, denote the configuration of a block in plane at time t, $f_t = (f_x, f_y)_t$ the force exerted by the robot (we omitted f_z as it does not change and torques as the interaction between the robot and the block is a point contact and cannot transmit torques), p_t a measure of the perturbation of the tower defined as net displacement with respect to an unperturbed tower, and $r_t = (J_x, J_y, \phi, w, v)_t$ the robot pose, push heading, push location on the block in the block frame, and auxiliary action v for retraction. We denote the state as $s_t = (q, f, p, J_x, J_z)_t$ and action as $a_t = (\phi, w, v)_t$.

Baseline models

Mixture of Regression: We formulate the MOR model as the linear mixture model:

$$p(s_{t+1}|s_t, a_t) = \sum_{i=1}^{H} \pi_i(\delta_i, \mathrm{s_t}, a_t) N(s_{t+1}|\, {\alpha_i}^T s_t + {\beta_i}^T a_t, {\sigma_i}^2)$$

Here, the mixture weights are also a function of the input space. We used 5-fold cross validation and expectation-maximization to determine the number of mixture components and train the

mixture model. We found empirically that to maximize the predictive ability of these models, it is best to train a distinct mixture per output variable.

Feed-Forward neural network model: This model also uses (s_t, a_t) to predict s_{t+1}. The hidden layer nodes are fully connected layers with ReLu activations. Empirically, we found that a (11, 10, 9, 17) layout for the hidden layers yielded the best predictive performance over the test-set for the models we tried. In this particular model, the input is also directly passed to the final layer. The network was trained in PyTorch with a learning rate of 0.01 and batch size of 100. Larger batch sizes yielded only marginally better models at the expense of computation time. The resulting models were sensitive to large changes to the learning rate but were consistent in predictive performance for rates close to 0.01.

Reinforcement Learning: We used the PPO implementation from the OpenAI Gym (12). We pass the state vector s_t to the algorithm and apply the computed a_t to the system. We then evaluate s_{t+1} and the reward function and repeat they cycle. The reward is composed of an accumulating term for positive displacement, a terminal award for a grasped block, an accumulating negative reward for moderate perturbations and a large negative reward for large perturbations. Please see the Supplementary Materials for the details of the reward function and hyper-parameters.

Concepts Learning via Clustering

We used a Gaussian Mixture Model (GMM) with a Dirichlet Process (DP) prior over the number of clusters to learn the abstractions. We used the feature set

$$x_t = \{q, \Delta q, f, \Delta f, J, a\}_t,$$

where Δ denotes temporal difference and the variables maintain the definitions from the previous subsection. To generate a dataset of features, we randomly sampled 80 samples from random push actions conducted during the exploration phase. We found empirically that very similar clusters are recovered with as few as 40 samples though occasionally fewer clusters were discovered due to random selection of samples. We set the α prior for the DP to 0.7 and let the variational inference algorithm iterate for a maximum of 30 times. Values lower than 0.7 consistently yielded fewer clusters, in particular, the no move, move, and no blocks occur often. On some occasions, the no block and move meld into one. Values higher than 0.7 yield a larger number of clusters, approaching numbers determined by the mixture models. To check cluster consistency, we repeated the dataset construction and variational inference steps 10 times.

Once the variational inference algorithm concludes, we evaluated the number of clusters by checking the posterior distribution over the mixture weights. We found that the posterior weights for 4 of the clusters were meaningful and used the mean and covariances of these 4 clusters as the abstractions in the model.

Bayesian Neural Network Motion Model

We used a Bayesian neural network (BNN) to model the state transitions (configurations and forces) in the physics model of the Jenga tower. These models use the same input features as the baseline models with an additional one-hot encoding of the latent variable denoted with c_t and

predicts the same set of variables and computes a probability distribution. We use a probabilistic model in order to build a generative model of physics to be used with probabilistic inference.

For the network architecture, we used an arrangement of (10, 7, 5) for the hidden nodes. This form of network was chosen to emulate an encoder network, as the various inputs to the network are correlated and potentially span a submanifold of the space. For details, please see Supplementary Materials.

Probabilistic Inference and MPC Control

We used MCMC to perform probabilistic inference over the learned representation of physics. To sample from the Bayesian hierarchical model, we employed a prior over the latent variables using a Dirichlet Process (DP). As the variables are one-hot encoded in the motion models, we chose our prior hyper-parameters such that the DP, when sampled, returns a value close to a class one-hot encoding. For the observations, we assumed a zero mean normal distribution with a variance equal to the variance measured from the sensors. We used PyMC3 to encode the model, and ran MCMC given measurements accumulated by the robot at run-time. We ran 4 parallel MCMC chains on 4 CPUs at run time, each with 3000 samples and a burn in of 500. One call to the inference algorithm takes approximately 0.5 seconds. We found that the benefit of running inference for longer periods to yield better estimates of the latent variables was not worth the incurred computational expense.

To control the block motions, we used a sampling-based, greedy model predictive controller. In particular, the MPC controller assigns a quadratic cost to the distance from goal, change in action, and perturbation of the tower. The controller looks ahead 5 steps using a learned model and selects the action sequence with lowest cost. It executes this action and repeats. For details of implementation, please see Supplementary Materials.

Perception

Our visual perception system maps a single-color image of the Jenga tower to a collection of blocks, each with its own position and rotation. The system operates in two steps: it first obtains the segment of each block using deep convolutional networks; it then applies template matching to recover the 6DOF of the block. Leveraging state-of-the-art techniques in visual perception and traditional model-fitting algorithms, the system is fast, accurate, and robust to noise.

The vision module first segments out each block from the image. To do this, we employed the state-of-the-art convolutional networks for this purpose, Mask R-CNN (29), with a backbone structure of ResNet-50 (30). The camera output has a resolution of 640×480, and we only use a 280×280 patch with the tower at the center. During inference, we set the non-maximum suppression threshold to 0.4 to avoid overlapping segments. We follow the original Mask R-CNN paper for other implementation details (29). The segmenting network is learned using a combination of 3,000 synthetic images of the Jenga tower rendered in Blender and 150 images of the Jenga tower in the experimental setup. See Supplementary Materials for further details.

After obtaining the segments, the visual module maps each segment to a 3D block with its position and pose via a hidden Markov model (HMM). The main idea is to find the sequence of

block positions and poses that explain visual observations well, while maintaining temporal smoothness. Specifically, given segments $M = \{m_i\}$ as our observations, we want to find the underlying states $S = \{s_i\}$ that maximize $\Pi_i\, P(m_i|s_i)P(s_{i+1}|s_i)$. A state s is parametrized as a 5-tuple (p, q, x, y, θ), where p represents the block's level of height, q represents its order (left, middle, right) among the blocks at the same level, x and y represent its horizontal and vertical translation with a granularity of 0.002 m, and θ represents its in-plane rotation with a granularity of 3 degrees.

We used the classic Viterbi algorithm for inference. We compute the probability of observation m_i given state s_i using three criteria: the intersection over union between the segment provided by Mask R-CNN and the segment of the block from the template (IOU_s), the intersection between the bounding box of the Mask R-CNN segment and the template segment (IOU_b), and the chamfer distances for the two sets of pixels within the two segments. For details of the transition probability and related cost-function, please see Supplementary Materials.

Acknowledgments: We thank Professors Kaelbling and Lozano-Perez for the insightful discussions related to the project. **Funding:** The research is supported by NSF grants 1231216 and 1637753. **Author contributions:** N.F. main ideas of the paper, experimental and simulation implementations, writing, and hierarchical learning, M.O. lion's share of the coding and design in experimental and simulation setup, J.W. and J.W. the perception algorithm and insights into model learning, J.T. writing, provided deep insights and guidance on hierarchical learning and implementation, and A.R. writing, provided deep insights into manipulation, system architecture, and physics models. **Data availability:** The code-base and data needed to evaluate the conclusions of this paper are available at https://github.com/mcubelab/mo_jenga_public.

References and Notes

[1] E. S. Spelke and K. D. Kinzler, Core knowledge. *Dev. Sci.*, vol. 10, no. 1, pp. 89–96, 2007.

[2] M. T. Mason, Towards Robotic Manipulation. *Annu. Rev. Control. Robot. Auton. Syst.*, vol. 1, pp. 1–28, 2018.

[3] G. Erdogan, I. Yildrim, and R. A. Jacobs, Transfer of object shape knowledge across visual and haptic modalities, in *Proceedings of the Annual Meeting of the Cognitive Science Society* (2014), vol. 36, no. 36.

[4] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, Simulation as an engine of physical scene understanding. *Proc. Natl. Acad. Sci.*, vol. 110, no. 45, pp. 18327–18332, 2013.

[5] J. Fischer, J. G. Mikhael, J. B. Tenenbaum, and N. Kanwisher, Functional neuroanatomy of intuitive physical inference. *Proc. Natl. Acad. Sci.*, vol. 113, no. 34, pp. E5072–E5081, 2016.

[6] T. D. Ullman, E. Spelke, P. Battaglia, and J. B. Tenenbaum, Mind games: Game engines as an architecture for intuitive physics. *Trends Cognit. Sci.*, vol. 21, no. 9, pp. 649–665, 2017.

[7] R. S. Johansson and J. R. Flanagan, Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nat. Rev. Neurosci.*, vol. 10, no. 5, p. 2345, 2009.

[8] A. Petrovskaya, O. Khatib, S. Thrun, and A. Y. Ng, Touch based perception for object manipulation, in *Robotics Science and Systems, Robot Manipulation Workshop* (2007), pp. 2–7.

[9] M. O. Ernst and M. S. Banks, Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, vol. 415, no. 6870, pp. 429–433, Jan. 2002.

[10] K. P. Körding and D. M. Wolpert, Bayesian integration in sensorimotor learning. *Nature*, vol. 427, no. 6971, pp. 244–247, Jan. 2004.

[11] E. Todorov and T. Erez, and Y. Tassa, Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033, 2012.

[12] J. Schulman and J. Wolski and F. Dhariwal and P.A. Radford, and O. Klimov, Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] R. M. Sanner and J.-J. E. Slotine, Gaussian networks for direct adaptive control. *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 837–863, 1992.

[14] H. Asada, Teaching and learning of compliance using neural nets: representation and generation of nonlinear compliance, in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1237–1244.

[15] S. Lee and M. H. Kim, Learning expert systems for robot fine motion control, in *Proceedings of IEEE International Symposium on Intelligent Control* (1988), pp. 534–544.

[16] J. A. Franklin, Refinement of robot motor skills through reinforcement learning, in *Proceedings of the 27th IEEE Conference on Decision and Control*, pp. 1096–1101.

[17] M. Erdmann, Using backprojections for fine motion planning with uncertainty, in *Proceedings of IEEE International Conference on Robotics and Automation* (1985), vol. 2, pp. 549–554.

[18] B. Donald, Robot motion planning with uncertainty in the geometric models of the robot and environment: A formal framework for error detection and recovery, in *Proceedings of IEEE International Conference on Robotics and Automation* (1986), vol. 3, pp. 1588–1593.

[19] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.

[20] P. Agrawal, A. V Nair, P. Abbeel, J. Malik, S. Levine, Learning to poke by poking: Experiential learning of intuitive physics, in *Proceedings of Advances in Neural Information Processing Systems* (2016), pp. 5074–5082.

[21] Y.Gao, L. A. Hendricks, K. J. Kuchenbecker, and T. Darrell, Deep learning for tactile understanding from visual and haptic data. *In 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 536-543. 2016.

[22] A. Kloss, S. Schaal, J. Bohg, Combining learned and analytical models for predicting action effects. *CoRR*, vol. abs/1710.0, 2017.

[23] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, D. Quillen, Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Rob. Res.*, vol. 37, no. 4–5, pp. 421–436, 2018.

[24] J. Mahler, K. Goldberg, Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences, in *Proceedings of the 1st Conference on Robotic Learning* (2017), pp. 515–524.

[25] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[26] N. Fazeli, S. Zapolsky, E. Drumwright, and A. Rodriguez, Fundamental Limitations in Performance and Interpretability of Common Planar Rigid-Body Contact Models. *CoRR*, vol. abs/1710.0, 2017.

[27] M. C. Gemici, and A. Saxena, Learning haptic representation for manipulating deformable food objects. *In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on* pp. 638-645, 2014.

[28] S. Calinon, F. Guenter, and A. Billard, On learning, representing, and generalizing a task in a humanoid robot, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* Vol. 37, Issue: 2, 2007.

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick, Mask R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2980–2988.

[30] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

Figures and Notes:

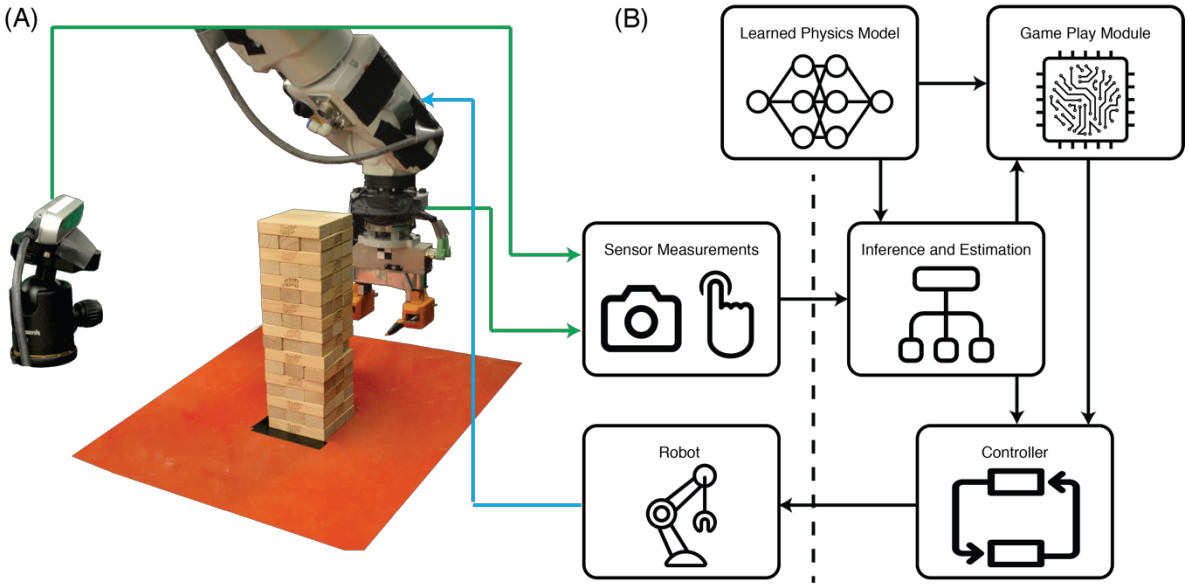

Fig. 1: Robot Setup. (**A**) Physical setup consisting of the robot, Jenga tower, Intel RealSense D415 camera, and ATI Gamma Force/Torque sensor (mounted at the wrist). (**B**) The machine intelligence architecture with the learned physics model.

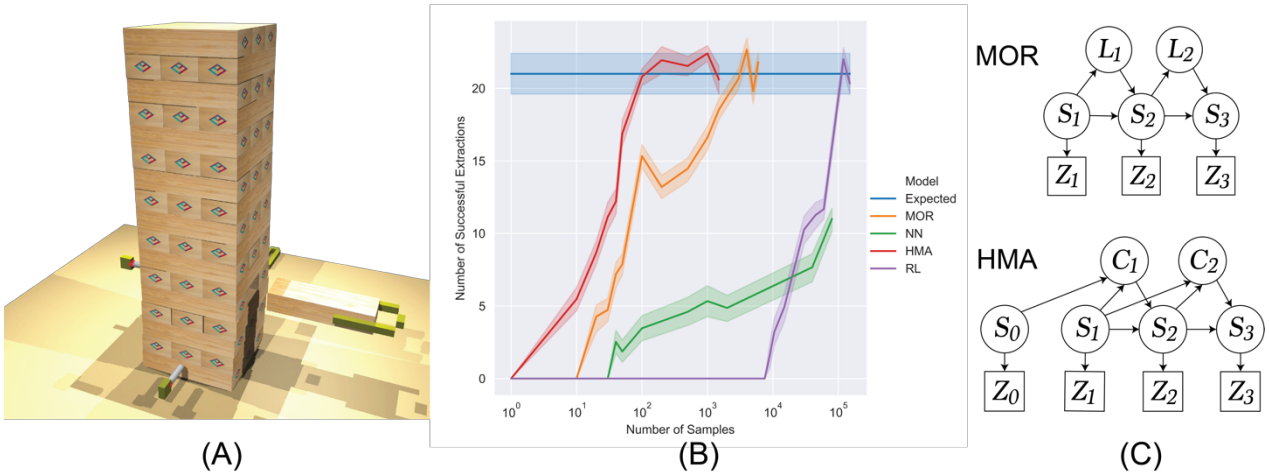

Fig. 2: Jenga setup in simulation and the baseline comparisons. A) The simulation setup is designed to emulate the real-world implementation. B) The learning curve of the different approaches with confidence intervals evaluated over 10 attempts. C) A visual depiction of the structure of the mixture of regressions (MOR) and the proposed approach (HMA).

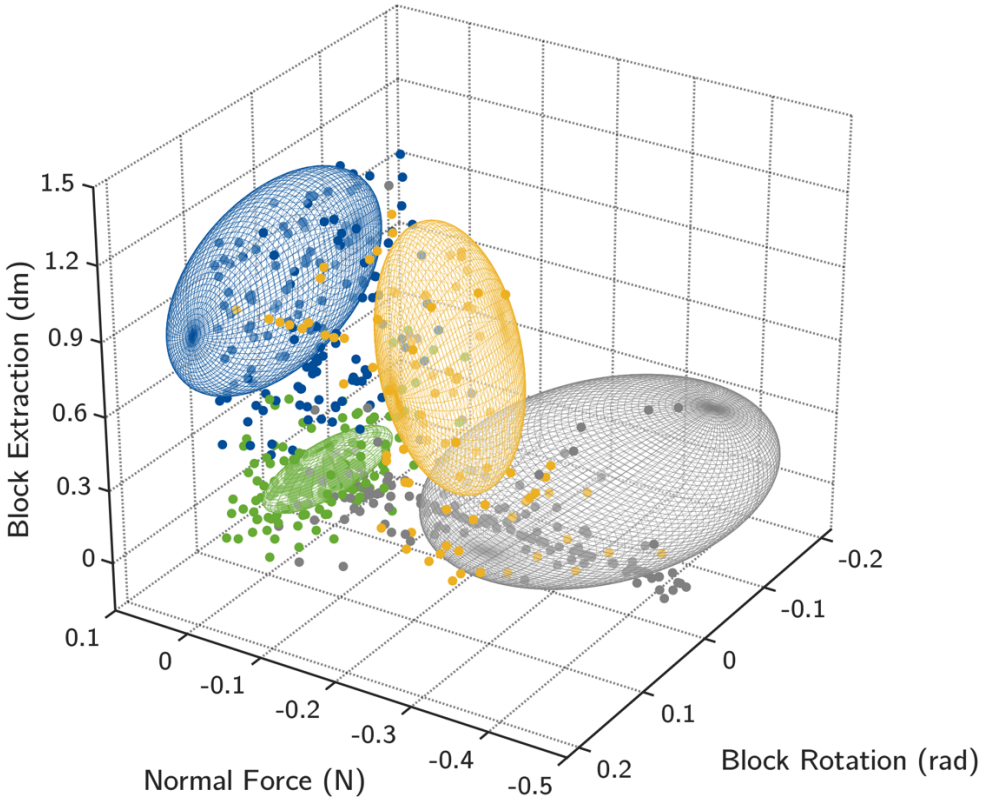

Fig. 3: Concepts learned from exploration data. Means and covariances of the four clusters are projected to the space of 'normal force (N)', 'block rotation (rad)', and 'block extraction/depth (dm)'. The four clusters carry intuitive semantic meanings and we refer to them as: Green: "no block"; gray: "no move"; blue: "small resistance"; and yellow: "hard move"

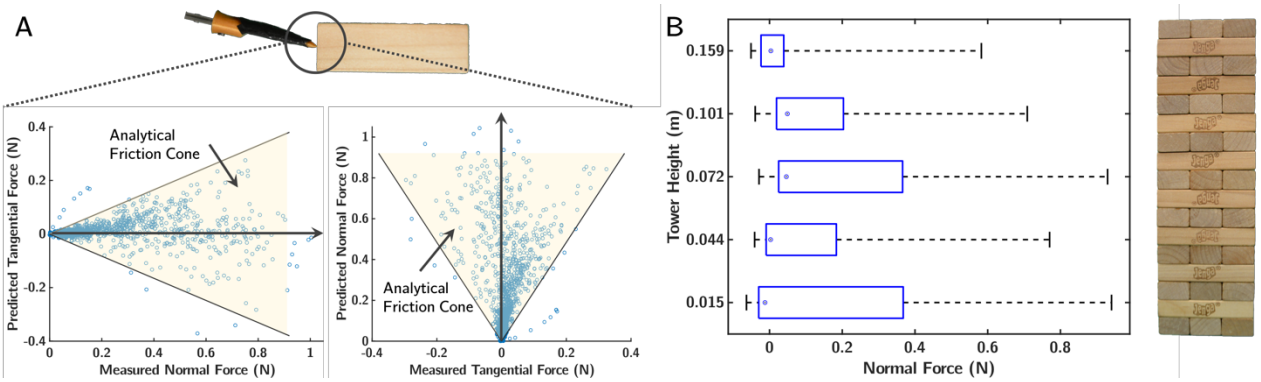

Fig. 4: Learned Intuitive Physics. (**A**) Overlay of the analytical friction cone and predicted forces given the current measurements. The friction coefficient between the finger material (PLA) and wood is between 0.35 and 0.5, here we use 0.42 as an approximation. (**B**) The normal force applied to the tower as a function of the height of the tower.

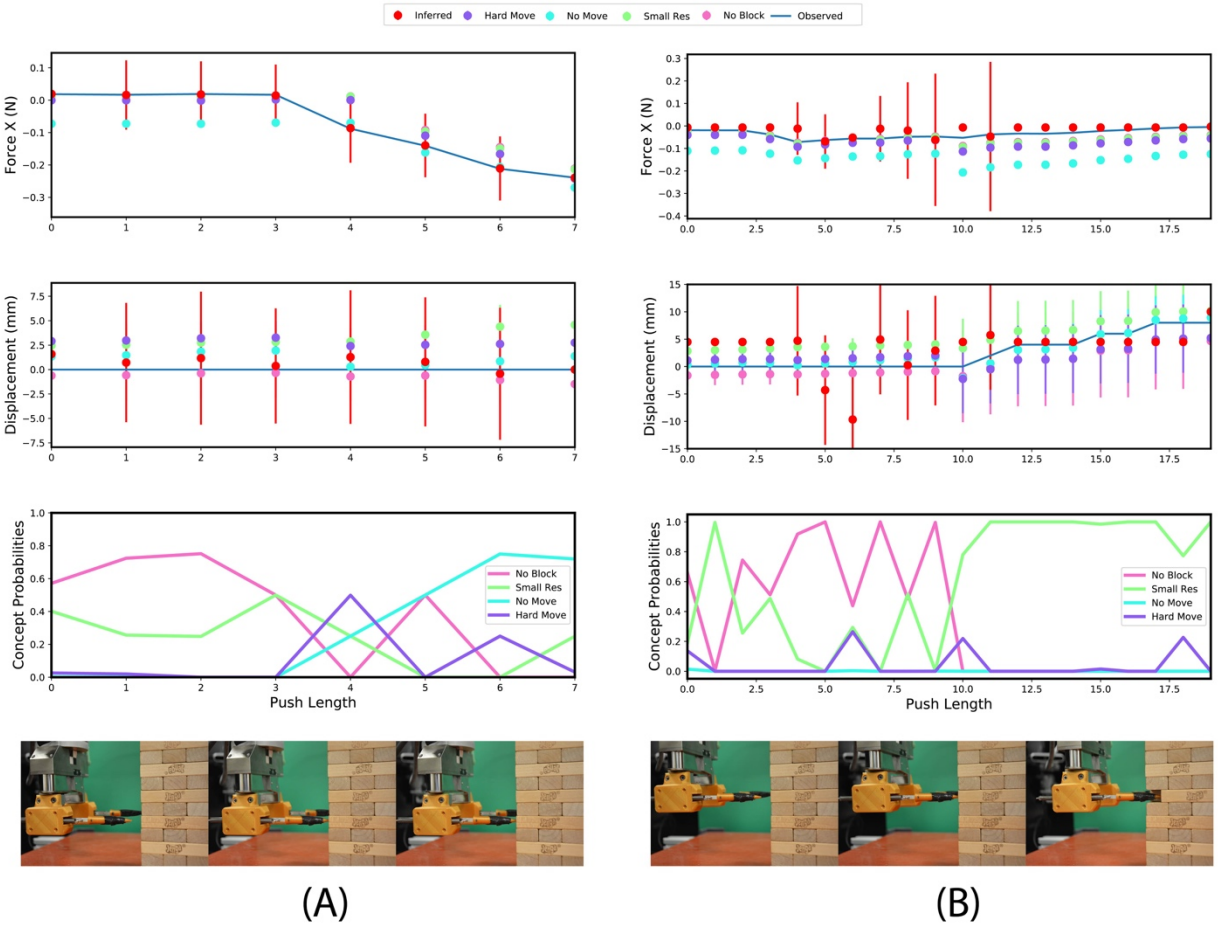

Fig. 5: Inference using the learned representation. Evolution of the beliefs of the robot as it interacts with the tower. (**A**) For a block that is stuck. (**B**) For the block that moves easily.

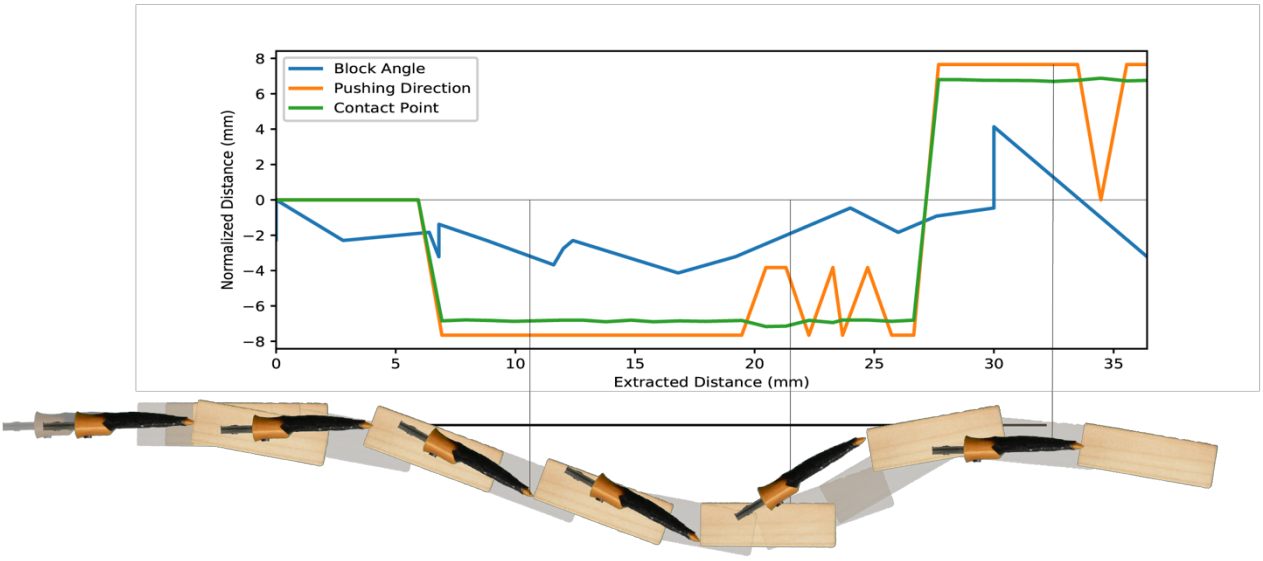

Fig. 6: Controlled block pushing. The robot selects the point on the block and the appropriate angle to push with such that it realigns the block with the goal configuration. Here, the block is beginning to rotate counter clockwise and is starting to move out of the tower. The robot selects a point close to the edge of the block and pushes it back in towards to tower center. We convert angles to normalized distances by scaling with the radius of gyration of the block, 0.023 meters. We have exaggerated the block translation to illustrate the fine grain details of motion.

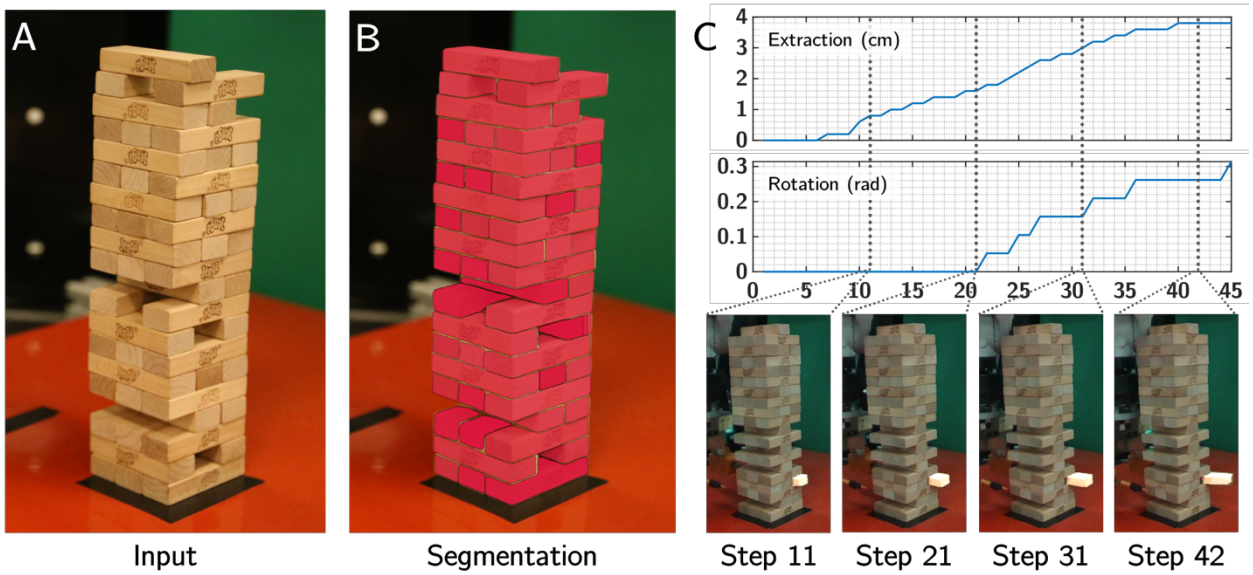

Fig. 7: The vision system. (A) We use a 280×280 patch with the tower at the center. For clarity, in this figure we crop the irrelevant regions on both sides. (B) The segmented blocks. (C) For a single push, we identify the block being pushed, and based on its associated segment, infer the underlying pose and position of the block using an HMM.

Table 1: Summary statistics for Exploration and Learned Physics. Table comparing the performance of the robot using the exploration strategy and the learned model.

Block Position	Action	Exploration		Learned	
		Attempts	Successes	Attempts	Successes
All	Push	403	172 (42.7%)	203	93 (45.8%)
	Extract	172	97 (56.4%)	93	82 (88.2%)
	Place	97	85 (87.6%)	82	72 (87.8%)
Side	Push	288	122 (42.4%)	133	69 (51.9%)
	Extract	122	52 (42.6%)	69	54 (78.3%)
	Place	52	44 (84.6%)	54	49 (90.7%)
Middle	Push	115	50 (43.5%)	70	33 (47.1%)
	Extract	50	45 (90.0%)	33	28 (84.8%)
	Place	45	41 (91.1%)	28	23 (82.1%)

Supplementary Materials for

See, Feel, Act: Hierarchical Learning for Complex Manipulation Skills with Multi-sensory Fusion

N. Fazeli, M. Oller, J. Wu, Z. Wu, J. B. Tenenbaum, A. Rodriguez

*Corresponding author. Email: nfazeli@mit.edu

The PDF file includes:

Details of implementation from the Methods and Materials section.

Supplementary Materials

Reinforcement Learning – Reward and Hyper-parameters

We used the following reward for training:

$$R(o_{1\dots t_{end}}, a_{1\dots t_{end}}) = \sum_{i=1}^{t_{end}} (b_1 dx_i - b_2 D_1(q_i, p_i) - b_3 D_2(q_i, p_i)) + b_4 D_3(q_{t_{end}})$$

where $b = (0.1,\ 0.2, 100, 4)$ is a constant vector of gains for the components of the reward function. These values were tuned to the setup through a trial and error procedure to yield converging policies. $b_1 dx_i$ is an accumulating term for positive displacement along the major axis of the block to incentives pushing. In this implementation, we use a linear form for $D_1 = p_i$ over the range of $p \in (0.08, 0.4)$ and zero elsewhere, a fixed value for $D_2 = 100$ for $p > 0.4$, and a fixed value for $D_3 = 1$ if the block is securely grasped by the gripper. The values where found empirically through tuning to yield converging policies. To provide intuition, the 0.4 value of perturbation refers to rotations or displacements exceeding 30 degrees and 20 mm for the entire tower. These large negative costs were necessary to guide the robot away from excessive perturbations due to rewards from positive block motion, else the policy greedily pushes the tower around until collapse, falling short of larger rewards available for a longer game with an intact tower.

The final term denotes a reward over the successful extraction of a block using the gripper. This reward is evaluated per run and summed over the blocks attempted, and each tower in a run is generated at random. A run is ended after a tower topple, if the robot chooses to stop premature to exploring blocks, or all blocks are explored. Empirically we found that policies trained at every 150 samples worked well. Shorter training cycles did not yield convergent policies. Interestingly longer cycles would occasionally yield divergent policies too. This may be attribute to the blame assignment problem.

Bayesian Neural Network Details

The particular numbers where found through a coarse cross-validation across several model structures, including deeper and shallower networks. We found that using tangent hyperbolic nonlinearities in only the first layer; with fully connected remaining layers yielded the best predictive performance. For the inference of weights, we used normal distributions with priors over the means and standard deviations. For the prior over means, we used a zero mean normal distribution with standard deviation of 1, and for the variances we used a half normal distribution with variance of 1. We found empirically that the network performance did not change significantly for variances larger than 1, rather the training time was longer; however, smaller variances would occasionally produce poor predictive models. We hypothesize that this is due to poor mixing at the tails of the narrow normal distributions. To train the models, we used a total of 200 trajectories (full push sequences) with the Automatic Differentiation Variational Inference implementation of PyMC3.

Model Predictive Controller Details

At each time-step, the controller queries the physics model using samples from the robot action space and computes a cost per action using

$$J_t = \bar{q}_t{}^T Q \bar{q}_t + \bar{a}_t^T R \bar{a}_t + p_t^T T p_t,$$

where $\bar{q}_t$ denotes the current distance of the block to its goal configuration, $Q = diag\{0.3, 0.3, 0.3\}$ denotes the weight matrix penalizing the error in configuration, $\bar{a}_t$ denotes the change in robot action, and $R = diag\{0.1, 0.1\}$ denotes the cost of changing actions. $P = 0.3$ and denotes the cost on perturbing the tower using the proxy value p_t. The first term guides the block to a goal configuration to be grasped. The second term prevents excessive jittering motion of the robot. The last term penalizes the robot for excessive perturbations of the tower. The hyper-parameters were found through extensive empirical analysis. The HMA model has an additional term that penalizes actions that lead to "no move" regimes but does not contain the quadratic penalization term on perturbation.

As a means of controlling the behavior of the robot, we coded a game play module that uses the results of inference and vision system and a finite state-machine to decide on whether to push, extract, place or stop execution. The control loop, together with perception, inference, and planning runs at approximately 0.5 Hz.

Perception – Segmenting Neural Networks

The neural nets were first trained on synthetic renderings of block towers, and then fine-tuned on real block towers with segment annotations. For synthetic data, we rendered 3,000 synthetic images of Jenga tower using Blender. Specifically, each block is a cuboid of size (24.8mm, 75.2mm, 14.3mm). We crawled five images of wood texture from Google Images search and randomly applied them to the blocks. The camera position is a Gaussian distribution in the world coordinate with mean (0.436m, 0.408m, 0.234m) and standard deviation (0.03m, 0.03m, 0.01m). The camera rotation in Euler angles (radian) is also a Gaussian distribution with mean (1.3079, 0.0510, 2.2513) and standard deviation (0.002, 0.002, 0.002). We used a point light source co-located with the camera and the Cycles renderer. For each rendering, its background is randomly chosen from 156 HDR images of outdoor scenes. We emulate the process of a Jenga game by randomly choosing a block from the tower and place it on the top. This process is repeated up to 9 times before we reset the tower. For real data, we collected 150 images of the Jenga tower in different configurations and labeled the mask of each block in each image. We also augmented the data by adding randomly cropped patches of the collected images into the training set; each patch keeps at least 90% of the original image.

Training on synthetic data was conducted on 2 Titan Xp GPUs for 60,000 iterations using stochastic gradient descent (SGD). We used a learning rate of 0.0025, momentum of 0.9, batch size of 2, and weight decay of 1×10^{-4}. We fine-tuned the model on real images for 6,000 iterations using SGD with a learning rate of 2×10^{-5}. All other parameters are the same as those for pre-training.

Perception – Temporal Smoothing

We denote the chamfer distance with (C). In practice, we have $log\ P(m_i|s_i) = 2 \times IoU_s + IoU_b - 5 \times C$. We compute the transition probability as

$$-log\ P(s_{i+1}|s_i) = [2 \times (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + \beta(\theta_{i+1} - \theta_i)^2]^{1/2},$$

where $\beta = \max_i \left(1.5, 10 - 0.5 \times \max_i(0, i - 20)\right)$.

Here we represent rotation θ in radian. To speed up inference, we only considered the configurations that are physically plausible, i.e. blocks not penetrating each other. For each time step i, we only consider the 50 candidate states (tuples) that give the highest probability for the observation m_i.