Learning Physical Intuition of Block Towers by Example

Adam Lerer ALERER@FB.COM
Facebook AI Research

Sam Gross SGROSS@FB.COM
Facebook AI Research

Rob Fergus ROBFERGUS@FB.COM
Facebook AI Research

Abstract

Wooden blocks are a common toy for infants, allowing them to develop motor skills and gain intuition about the physical behavior of the world. In this paper, we explore the ability of deep feed-forward models to learn such intuitive physics. Using a 3D game engine, we create small towers of wooden blocks whose stability is randomized and render them collapsing (or remaining upright). This data allows us to train large convolutional network models which can accurately predict the outcome, as well as estimating the block trajectories. The models are also able to generalize in two important ways: (i) to new physical scenarios, e.g. towers with an additional block and (ii) to images of real wooden blocks, where it obtains a performance comparable to human subjects.

1. Introduction

Interaction with the world requires a common-sense understanding of how it operates at a physical level. For example, we can quickly assess if we can walk over a surface without falling, or how an object will behave if we push it. Making such judgements does not require us to invoke Newton's laws of mechanics – instead we rely on intuition, built up through interaction with the world.

In this paper, we explore if a deep neural network can capture this type of knowledge. While DNNs have shown remarkable success on perceptual tasks such as visual recognition (Krizhevsky et al., 2012) and speech understanding (Hinton et al., 2012), they have been rarely applied to problems involving higher-level reasoning, particularly those involving physical understanding. However, this is needed to move beyond object classification and detection to a true understanding of the environment, e.g. "What will happen next in this scene?" Indeed, the fact that humans develop such physical intuition at an early age (Carey, 2009), well before most other types of high-level reasoning, suggests its importance in comprehending the world.

To learn this common-sense understanding, a model needs a way to interact with the physical world. A robotic platform is one option that has been explored e.g. (Agrawal et al., 2015), but inherent complexities limit the diversity and quantity of data that can be acquired. Instead, we use Unreal Engine 4 (UE4) (Epic Games, 2015), a platform for modern 3D game development, to provide a realistic environment. We chose UE4 for its realistic physics simulation, modern 3D rendering, and open source license. We integrate the Torch (Collobert et al., 2011) machine learning framework directly into the UE4 game loop, allowing for online interaction with the UE4 world.

One of the first toys encountered by infants, wooden blocks provide a simple setting for the implicit exploration of basic Newtonian concepts such as center-of-mass, stability and momentum. By asking deep models to predict the behavior of the blocks, we hope that they too might internalize such notions. Another reason for selecting this scenario is that real world examples can be constructed, enabling the generalization ability of our models to be probed (see Fig. 1).

Two tasks are explored: (i) will the blocks fall over or not? and (ii) where will the blocks end up? The former is a binary classification problem, based on the stability of the block configuration. For the latter we predict image masks that show the location of each block. In contrast to the first task, this requires the models to capture the dynamics of the system. Both tasks require an effective visual system to analyze the configuration of blocks. We explore

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

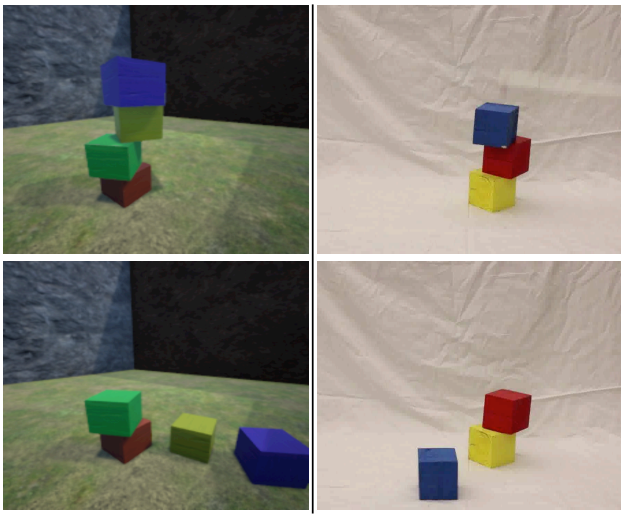

Figure 1. Block tower examples from the synthetic (left) and real (right) datasets. The top and bottom rows show the first and last frames respectively.

models based on contemporary convolutional networks architectures (LeCun et al., 1989), notably Googlenet (Ioffe & Szegedy, 2015), DeepMask (Pinheiro et al., 2015) and ResNets (He et al., 2015). While designed for classification or segmentation, we adapt them to our novel task, using a single end-to-end CNN to both perceive the block arrangement and predict the physics of their movement.

Our paper makes the following contributions:

Convnet-based Prediction of Static Stability: We show that standard convnet models, refined on synthetic data, can accurately predict the stability of stacks of blocks. Crucially, these models successfully generalize to (i) new images of real-world blocks and (ii) new physical scenarios, not encountered during training. These models are purely bottom-up in nature, in contrast to existing approaches which rely on complex top-down graphics engines.

Prediction of Dynamics: The models are also able to predict with reasonable accuracy the trajectories of the blocks as they fall, showing that they capture notions of acceleration and momentum, again in a purely feed-forward way.

Comparison to Human Subjects: Evaluation of the test data by participants shows that our models match their performance on held-out real data (and are significantly better on synthetic data). Furthermore, the model predictions have a reasonably high correlation with human judgements.

UETorch: We introduce an open-source combination of the Unreal game engine and the Torch deep learning environment, that is simple and efficient to use. UETorch is a viable environment for a variety of machine learning experiments in vision, physical reasoning, and embodied learning.

1.1. Related Work

The most closely related work to ours is (Battaglia et al., 2013) who explore the physics involved with falling blocks. A generative simulation model is used to predict the outcome of a variety of block configurations with varying physical properties, and is found to closely match human judgment. This work complements ours in that it uses a top-down approach, based on a sophisticated graphics engine which incorporates explicit prior knowledge about Newtonian mechanics. In contrast, our model is purely bottom-up, estimating stability directly from image pixels and is learnt from examples.

Concurrent work has also investigated convolutional networks for block tower stability prediction. (Zhang et al., 2016) directly compare convolutional networks with the Intuitive Physics Engine of (Battaglia et al., 2013) for fall prediction, observing certain human judgment biases better captured by IPE. (Li et al., 2016) evaluate convolutional networks for predicting the stability of more complex block towers in a simpler visual environment.

Our pairing of top-down rendering engines for data generation with high capacity feed-forward regressors is similar in spirit to the Kinect body pose estimation work of (Shotton et al., 2013), although the application is quite different. (Wu et al., 2015) recently investigated the learning of simple kinematics, in the context of objects sliding down ramps. Similar to (Battaglia et al., 2013), they also used a top-down 3D physics engine to map from a hypothesis of object mass, shape, friction etc. to image space. Inference relies on MCMC, initialized to the output of convnet-based estimates of the attributes. As in our work, their evaluations are performed on real data and the model predictions correlate reasonably with human judgement.

Prior work in reinforcement learning has used synthetic data from games to train bottom-up models. In particular, (Mnih et al., 2015) and (Lillicrap et al., 2015) trained deep convolutional networks with reinforcement learning directly on simulated images to learn policies for Atari games and the TORCS driving simulator, respectively.

A number of works in cognitive science have explored intuitive physics, for example, in the context of liquid dynamics (Bates et al., 2015), ballistic motion (Smith et al., 2013) and gears and pulleys (Hegarty, 2004). The latter finds that people perform "mental simulation" to answer questions about gears, pulleys, etc., but some form of implicit bottom-up reasoning is involved too. In computer vision, a number of works have used physical reasoning to aid scene understanding (Zheng et al., 2015; Koppula & Saxena, 2016). For example, (Jia et al., 2015) fit cuboids to RGBD data and use their centroids to search for scene interpretations that are statically stable.

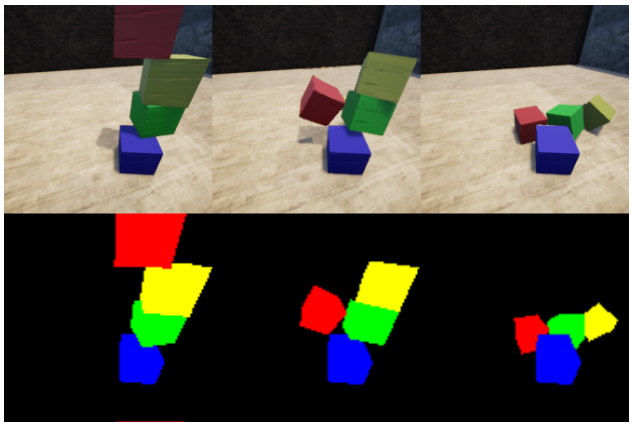

Figure 2. Recorded screenshots and masks at 1-second intervals from the Unreal Engine block simulation.

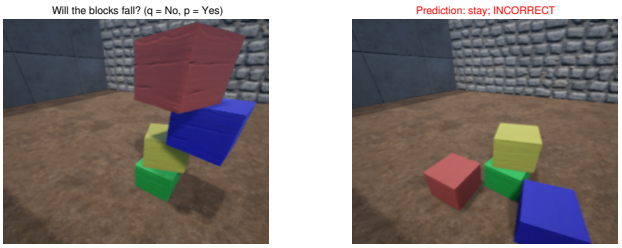

Figure 3. The interface used for human experiments. At each turn, the subject is shown an image on the left and tries to predict if the stack will fall or not. No time limit is imposed. During training phase, the subject receives feedback on their prediction, by showing them the outcome image on the right.

2. Methods

2.1. UETorch

UETorch is a package that embeds the Lua/Torch machine learning environment directly into the UE4 game loop, allowing for fine-grained scripting and online control of UE4 simulations through Torch. Torch is well-suited for game engine integration because Lua is the dominant scripting language for games, and many games including UE4 support Lua scripting. UETorch adds additional interfaces to capture screenshots, segmentation masks, optical flow data, and control of the game through user input or direct modification of game state. Since Torch runs inside the UE4 process, new capabilities can be easily added through FFI without defining additional interfaces/protocols for interprocess communication. UETorch simulations can be run faster than real time, aiding large-scale training. The `UETorch` package can be downloaded freely at `http://github.com/facebook/UETorch`.

2.2. Data Collection

Synthetic

A simulation was developed in UETorch that generated vertical stacks of 2, 3, or 4 colored blocks in random configurations. The block position and orientation, camera position, background textures, and lighting were randomized at each trial to improve the transferability of learned features. In each simulation, we recorded the outcome (did it fall?) and captured screenshots and segmentation masks at 8 frames/sec. Frames and masks from a representative 4-block simulation are shown in Fig. 2. A total of 180,000 simulations were performed, balanced across number of blocks and stable/unstable configurations. 12,288 examples were reserved for validation. A second held-out test set of 30,000 images was used for the reported results.

Real

Four wooden cubes were fabricated and spray painted red, green, blue and yellow respectively. Manufacturing imperfections added a certain level of randomness to the stability of the real stacked blocks, and we did not attempt to match the physical properties of the real and synthetic blocks. The blocks were manually stacked in configurations 2, 3 and 4 high against a white bedsheet. A tripod mounted DSLR camera was used to film the blocks falling at 60 frames/sec. A white pole was held against the top block in each example, and was then rapidly lifted upwards, allowing unstable stacks to fall (the stick can be see in Fig. 1, blurred due to its rapid motion). Note that this was performed even for stable configurations, to avoid bias. Motion of the blocks was only noticeable by the time the stick was several inches away from top block. 493 examples were captured, balanced between stable/unstable configurations. The totals for 2, 3 and 4 block towers were 115, 139 and 239 examples respectively.

2.3. Human Subject Methodology

To better understand the challenge posed about our datasets, real and synthetic, we asked 10 human subjects to evaluate the images in a controlled experiment. Participants were asked to give a binary prediction regarding the outcome of the blocks (i.e. falling or not). During the training phase, consisting of 50 randomly drawn examples, participants were shown the final frame of each example, along with feedback as to whether their choice was correct or not (see Fig. 3). Subsequently, they were tested using 100 randomly drawn examples (disjoint from the training set). During the test phase, no feedback was provided to the individuals regarding the correctness of their responses.

2.4. Model Architectures

We trained several convolutional network (CNN) architectures on the synthetic blocks dataset. We trained some architectures on the binary fall prediction task only, and oth-

ers on jointly on the fall prediction and mask prediction tasks.

Fall Prediction

We trained the ResNet-34 (He et al., 2015) and Googlenet (Szegedy et al., 2014) networks on the fall prediction task. These models were pre-trained on the Imagenet dataset (Russakovsky et al., 2015). We replaced the final linear layer with a single logistic output and fine-tuned the entire network with SGD on the blocks dataset. Grid search was performed over learning rates.

Fall+Mask Prediction

We used deep mask networks to predict the segmentation trajectory of falling blocks at multiple future times (0s,1s,2s,4s) based on an input image. Each mask pixel is a multi-class classification across a background class and four foreground (block color) classes. A fall prediction is also computed.

DeepMask (Pinheiro et al., 2015) is an existing mask prediction network trained for instance segmentation, and has the appropriate architecture for our purposes. We replaced the binary mask head with a multi-class SoftMax, and replicated this N times for mask prediction at multiple points in time.

We also designed our own mask prediction network, PhysNet, that was suited to mask *prediction* rather than just segmentation. For block masks, we desired (i) spatially local and translation-invariant (i.e. convolutional) upsampling from coarse image features to masks, and (ii) more network depth at the coarsest spatial resolution, so the network could reason about block movement. Therefore, PhysNet take the 7×7 outputs from ResNet-34, and performs alternating upsampling and convolution to arrive at 56×56 masks. The PhysNet architecture is shown in Fig. 4. We use the Resnet-34 trunk in PhysNet for historical reasons, but our experiments show comparable results with a Googlenet trunk.

The training loss for mask networks is the sum of a binary cross-entropy loss for fall prediction and a pixelwise multi-class cross-entropy loss for each mask. A hyperparameter controls the relative weight of these losses.

Baselines As a baseline, we perform logistic regression either directly on image pixels, or on pretrained Googlenet features, to predict fall and masks. To reduce the number of parameters, the pixels-to-mask matrix is factored with an intermediate dimension 128. For fall prediction, we also try k-Nearest-Neighbors ($k = 10$) using Googlenet last-layer image features.

2.5. Evaluation

We compare fall prediction accuracy on synthetic and real images, both between models and also between model and human performance. We also train models with a held-out block tower size and test them on the held out tower size, to evaluate the transfer learning capability of these models models to different block tower sizes.

We evaluate mask predictions with two criteria: mean mask IoU and log likelihood per pixel. We define mean mask IoU as the intersection-over-union of the mask label with the binarized prediction for the $t = 4s$ mask, averaged over each foreground class present in the mask label.

$$MIoU(\mathbf{m}, \mathbf{q}) = \frac{1}{N} \sum_{n=1}^{N} \left[\frac{1}{|C_n|} \sum_{c \in C_n} IoU(m_{nc}, \hat{q}_{nc}) \right] \tag{1}$$

where m_{nc} is the set of pixels of class c in mask n, $C_n = \{c : c \in \{1, 2, 3, 4\} \wedge |m_{nc}| > 0\}$ is the set of foreground classes present in mask n, $\hat{q}_{nc}$ is the set of pixels in model output n for which c is the highest-scoring class, and $IoU(m_1, m_2) = \frac{|m_1 \cap m_2|}{|m_1 \cup m_2|}$.

The mask IoU metric is intuitive but problematic because it uses binarized masks. For example, if the model predicts a mask with 40% probability in a region, the Mask IoU for that block will be 0 whether or not the block fell in that region. The quality of the predicted mask confidences is better captured by log likelihood.

The log likelihood per pixel is defined as the log likelihood of the correct final mask under the predicted (SoftMax) distribution, divided by the number of pixels. This is essentially the negative mask training loss.

Since the real data has a small number of examples ($N = 493$ across all blocks sizes), we report an estimated confidence interval for the model prediction on real examples. We estimate this interval as the standard deviation of a binomial distribution with p approximated by the observed accuracy of the model.

3. Results

3.1. Fall Prediction Results

Table 1 compares the accuracy for fall prediction of several deep networks and baselines described in Section 2.4. Convolutional networks perform well at fall prediction, whether trained in isolation or jointly with mask prediction. The best accuracy on synthetic data is achieved with PhysNet, which is jointly trained on masks and fall prediction. Accuracy on real data for all convnets is within their standard deviation.

As an ablation study, we also measured the performance

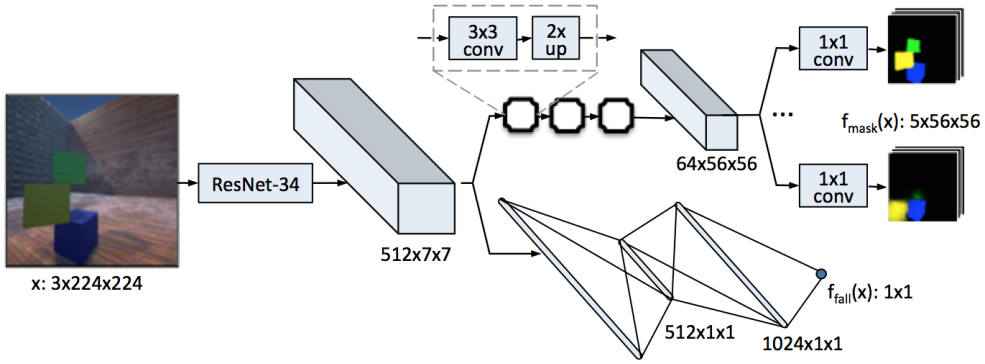

Figure 4. Architecture of the PhysNet network.

Model	Fall Acc. (%) (synthetic)	Fall Acc. (%) (real)
Baselines		
Random	50.0	50.0 ± 2.2
Pixel Log. Reg	50.1	49.3 ± 2.2
Googlenet Log. Reg.	65.5	62.5 ± 2.2
Googlenet kNN	59.6	50.9 ± 2.2
Classification Models		
ResNet-34	86.7	67.1 ± 2.1
Googlenet	86.8	**68.8** ± 2.1
Googlenet (no pretraining)	86.6	65.3 ± 2.2
Mask Prediction Models		
DeepMask	82.6	66.1 ± 2.1
PhysNet	**89.2**	66.7 ± 2.1

Table 1. Fall prediction accuracy of convolutional networks on synthetic and real data. The models substantially outperform baselines, and all have similar performance whether trained singly or jointly with the mask prediction task. Training Googlenet without Imagenet pretraining does not affect performance on synthetic examples, but degrades generalization to real examples. Baselines are described in Section 2.4.

of Googlenet without Imagenet pretraining. Interestingly, while the model performed equally well on synthetic data with and without pretraining, the pretrained model generalized better to real images (Table 1). We speculate that pretraining would be even more important if the real dataset had more complex textures and lighting.

Occlusion Experiments

We performed occlusion experiments to determine which regions of the block images affected the models' fall predictions. A Gaussian patch of gray pixels with standard deviation 20% of the image width was superimposed on the image in a 14×14 sliding window to occlude parts of the image, as shown in Fig. 5A. The PhysNet model was evaluated on each occluded image, and the difference in the fall probability predicted from the baseline and occluded images were used to produce heatmaps, shown in Fig. 5B-D.

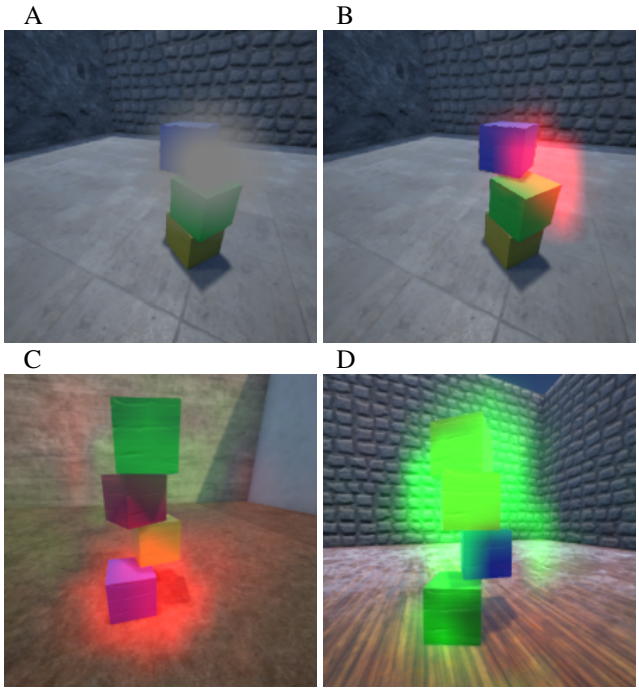

Figure 5. **A**: Example of Gaussian occlusion mask, applied in a sliding window to generate fall prediction heatmaps. **B–D**: Heatmaps of predictions from occluded images. A *green* overlay means that an occlusion in this region *increases* the predicted probability of falling, while a *red* overlay means the occlusion *decreases* the predicted probability of falling. The model focuses on unstable interfaces (**B,C**), or stabilizing blocks that prevent the tower from falling (**D**).

These figures suggest that the model makes its prediction based on relevant local image features rather than memorizing the particular scene. For example, in Fig. 5B, the model prediction is only affected by the unstable interface between the middle and top blocks.

Model vs. Human Performance

Fig. 6 compares PhysNet to 10 human subjects on the same set of synthetic and real test images. ROC curves compar-

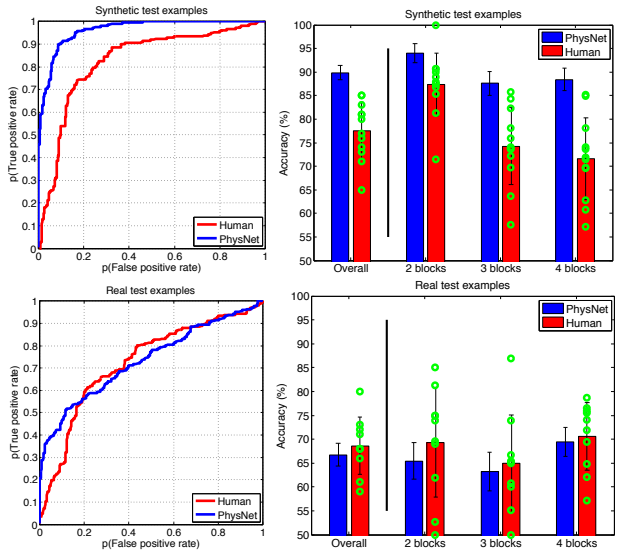

Figure 6. Plots comparing PhysNet accuracy to human performance on real (Top) and synthetic (Bottom) test examples. Left: ROC plot comparing human and model predictions. Right: a breakdown of the performance for differing numbers of blocks. For humans, the mean performance is shown, along with the performance of individual subjects (green circles). Overall, the PhysNet model is better than even the best performing of the human subjects on synthetic data. On real data, PhysNet performs similarly to humans.

ing human and model performance are generated by using the fraction of test subjects predicting a fall as a proxy for confidence, and comparing this to model confidences.

Overall, the model convincingly outperforms the human subjects on synthetic data, and is comparable on real data. Interestingly, the correlation between human and model confidences on both real and synthetic data ($\rho = (0.69, 0.45)$) is higher than between human confidence and ground truth ($\rho = (0.60, 0.41)$), showing that our model agrees quite closely with human judgement. However, we observe that for real data the model's fall prediction confidence is nearly always close to 0% or 100%, inconsistent with human judgment. This is likely because the real image statistics are outside the domain of the training data, leading to high-magnitude ReLU features saturating the classifier.

3.2. Mask Prediction Results

Table 2 compares mask prediction accuracy of the DeepMask and PhysNet networks described in Section 2.4. PhysNet achieves the best performance on both Mean Mask IoU and Log Likelihood per pixel (see Section 2.5), substantially outperforming DeepMask and baselines. Predicting the mask as equal to the initial ($t = 0$) mask has a high Mask IoU due to the deficiencies in that metric described in Section 2.5.

Model	Mask IoU (%) (synthetic)	Log Likelihood/px (synthetic)
DeepMask	42.4	-0.299
PhysNet	**75.4**	**-0.107**
Baseline		
Pixel Log. Reg.	29.6	-0.562
Googlenet Log. Reg.	23.8	-0.492
Mask @ $t = 0$	72.0	$-\infty$
Class-Constant	0	-0.490

Table 2. Mask prediction accuracy of DeepMask and our PhysNet network. The metrics used are described in Section 2.5; baselines are described in Section 2.4. As an additional IoU baseline we evaluate the $t = 0$ mask as a prediction of the final mask, and as a log likelihood baseline we predict each pixel as the average likelihood of that class in the data. The PhysNet network provides the highest accuracy in both metrics. Mask examples are shown in Fig. 7.

Examples of PhysNet mask outputs on synthetic and real data are shown in Fig. 7. We only show masks for examples that are predicted to fall, because predicting masks for stable towers is easy and the outputs are typically perfect. The mask outputs from PhysNet are typically quite reasonable for falling 2- and 3-block synthetic towers, but have more errors and uncertainty on 4-block synthetic towers and most real examples. In these cases, the masks are often highly diffuse, showing high uncertainty about the trajectory. On real examples, model predictions and masks are also skewed overstable, likely because of different physical properties of the real and simulated blocks.

3.3. Evaluation on Held-Out Number of Blocks

Table 3 compares the performance of networks that had either 3- or 4-block configurations excluded from the training set. While the accuracy of these networks is lower on the untrained class relative to a fully-trained model, it's still relatively high – comparable to human performance. The predicted masks on the untrained number of blocks also continue to capture the fall dynamics with reasonably accuracy. Some examples are shown in Fig. 8.

4. Discussion

Our results indicate that bottom-up deep CNN models can attain human-level performance at predicting how towers of blocks will fall. We also find that these models' performance generalizes well to real images if the models are pretrained on real data (Table 1).

Several experiments provide evidence that the deep models we train are gaining knowledge about the dynamics of the block towers, rather than simply memorizing a mapping from configurations to outcomes. Most convincingly, the

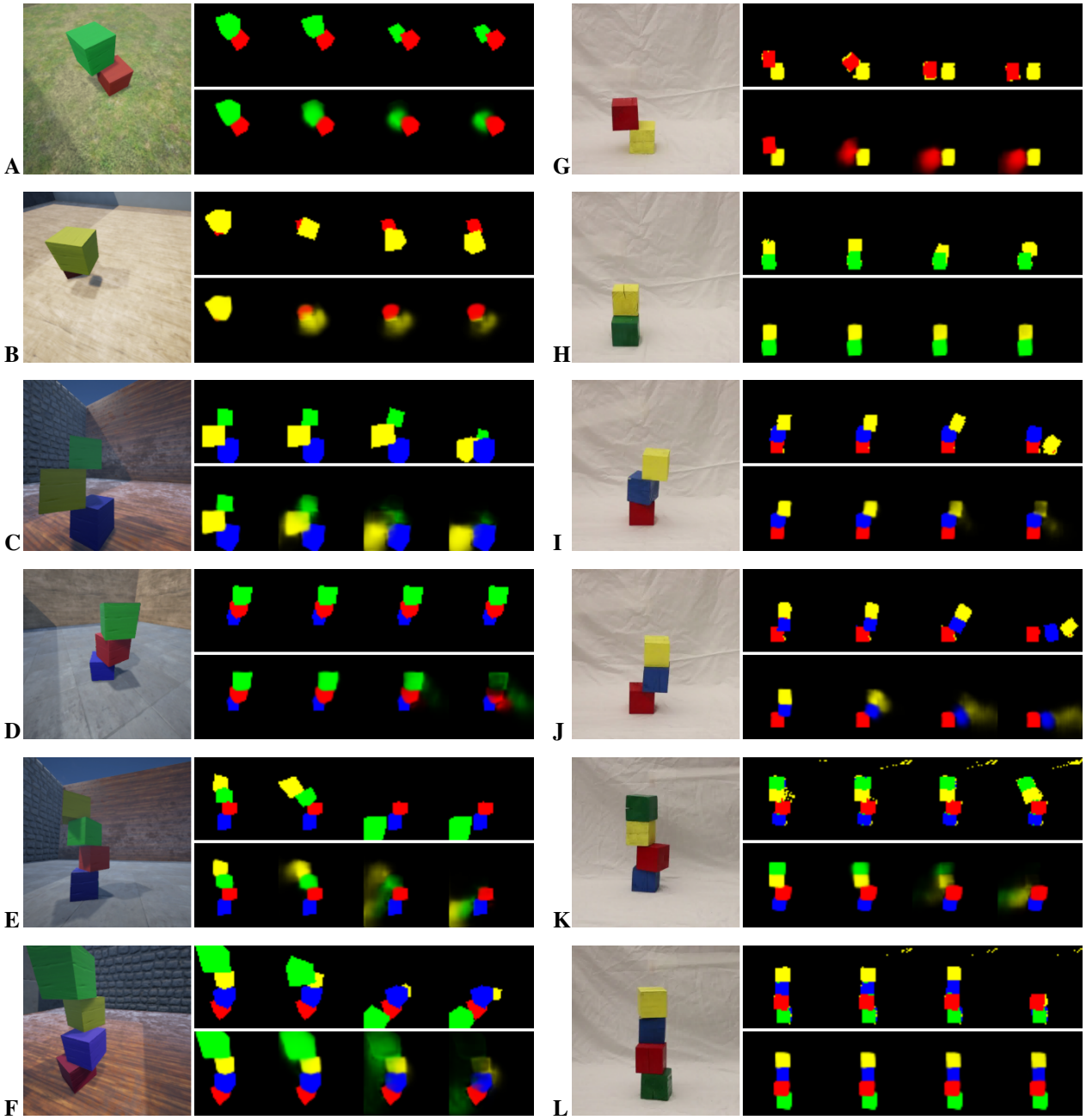

Figure 7. PhysNet mask predictions for synthetic (**A–F**) and real (**G–L**) towers of 2, 3, and 4 blocks. The image at the left of each example is the initial frame shown to the model. The top row of masks are the ground truth masks from simulation, at 0, 1, 2, and 4 seconds. The bottom row are the model predictions, with the color intensity representing the predicted probability. PhysNet correctly predicts fall direction and occlusion patterns for most synthetic examples, while on real examples, PhysNet overestimates stability (**H,L**). In difficult cases, Physnet produces diffuse masks due to uncertainty (**D–F,I**). **B** is particularly notable, as PhysNet predicts the red block location from the small patch visible in the initial image.

Model	# Blocks Training	Accuracy (%) (synth.) 2	3	4	Accuracy (%) (real) 2	3	4	Mask Log Likelihood/px (synth.) 2	3	4
Googlenet	2,3,4	93.1	86.3	80.8	69.6 ± 4.3	69.8 ± 3.9	69.9 ± 3.0			
Googlenet	2,3	93.5	84.8	75.1	65.2 ± 4.4	66.9 ± 4.0	69.0 ± 3.0			
Googlenet	2,4	93.0	82.1	78.8	69.6 ± 4.3	66.9 ± 4.0	70.7 ± 2.9			
PhysNet	2,3,4	94.7	89.0	83.8	66.1 ± 4.4	65.5 ± 4.0	73.2 ± 2.9	-0.035	-0.096	-0.190
PhysNet	2,3	94.4	87.7	75.5	60.0 ± 4.6	64.0 ± 4.1	70.1 ± 2.9	-0.042	-0.125	-0.362
PhysNet	2,4	94.0	86.2	82.3	55.7 ± 4.6	67.6 ± 4.0	69.9 ± 3.0	-0.040	-0.154	-0.268

Table 3. Fall prediction accuracy for Googlenet and PhysNet trained on subsets of the block tower sizes, and tested on the held-out block tower size (blue cells). Prediction accuracy on the held-out class is reduced, but is still comparable to human performance (see Fig. 6). On real block data, performance on the held out class is equivalent to the fully-trained model, to within standard deviation. PhysNet mask predictions for held-out classes are only moderately degraded, and log likelihood scores are still superior to DeepMask predictions (Table 1). Physnet masks for the held-out class are shown in Fig. 8.

relatively small degradation in performance of the models on a tower size that is not shown during training (Table 3 & Fig. 8) demonstrates that the model must be making its prediction based on local features rather than memorized exact block configurations. The occlusion experiments in Fig. 5 also suggest that models focus on particular regions that confer stability or instability to a block configuration. Finally, the poor performance of k-nearest-neighbors on Googlenet features in Table 1 suggests that nearby configurations in Googlenet's pretrained feature space are not predictive of the stability of a given configuration.

Compared to top-down, simulation-based models such as (Battaglia et al., 2013), deep models require far more training data – many thousands of examples – to achieve a high level of performance. Deep models also have difficulty generalizing to examples far from their training data. These difficulties arise because deep models must learn physics from scratch, whereas simulation-based models start with strong priors encoded in the physics simulation engine. Bottom-up and top-down approaches each have their advantages, and the precise combination of these systems in human reasoning is the subject of debate (e.g. (Davis & Marcus, 2016) and (Goodman et al., 2015)). Our results suggest that deep models show promise for directly capturing common-sense physical intuitions about the world that could lead to more powerful visual reasoning systems.

We believe that synthetic data from realistic physical simulations in UETorch are useful for other machine learning experiments in vision, physics, and agent learning. The combination of synthetic data and mask prediction constitutes a general framework for learning concepts such as object permanence, 3D extent, occlusion, containment, solidity, gravity, and collisions, that may be explored in the future.

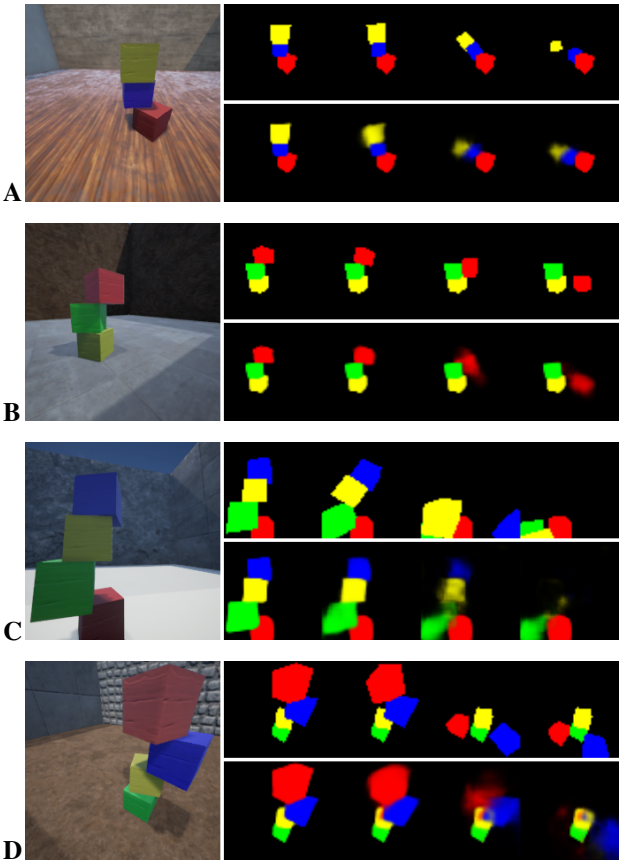

Figure 8. PhysNet mask predictions on a tower size (3 or 4 blocks) that the network was *not* trained on. Mask predictions for 3 blocks (**A–B**) still capture the dynamics well even though the network never saw towers of 3 blocks. Mask predictions for 4 blocks capture some of the dynamics but show some degradation.

Acknowledgements: The authors would like to thank: Soumith Chintala and Arthur Szlam for early feedback on experimental design; Sainbayar Sukhbaatar for assistance collecting the real-world block examples; Y-Lan Boureau for useful advice regarding the human subject experiments; and Piotr Dollar for feedback on the manuscript.

References

Agrawal, Pulkit, Carreira, Joao, and Malik, Jitendra. Learning to see by moving. In *The IEEE International Conference on Computer Vision (ICCV)*, June 2015.

Bates, C.J., Yildirim, I., Tenenbaum, J. B., and Battaglia, P. W. Humans predict liquid dynamics using probabilistic simulation. In *In Proc. Conf. Cognitive Science Society*, 2015.

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

Carey, Susan. *The origin of concepts*. Oxford University Press, 2009.

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

Davis, Ernest and Marcus, Gary. The scope and limits of simulation in automated reasoning. *Artificial Intelligence*, 233:60–72, 2016.

Epic Games. Unreal engine 4. https://www.unrealengine.com, 2015.

Goodman, Noah D, Frank, Michael C, Griffiths, Thomas L, Tenenbaum, Joshua B, Battaglia, Peter W, and Hamrick, Jessica B. Relevant and robust a response to marcus and davis (2013). *Psychological science*, 26(4):539–541, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. *ArXiv 1512.03385*, December 2015.

Hegarty, Mary. Mechanical reasoning by mental simulation. *Trends in cognitive sciences*, 8(6):280–285, 2004.

Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George E, Mohamed, Abdel-rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara N, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

Jia, Zhaoyin, Gallagher, A.C., Saxena, A., and Chen, Tsuhan. 3d reasoning from blocks to stability. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(5):905–918, 2015.

Koppula, Hema S and Saxena, Ashutosh. Anticipating human activities using object affordances for reactive robotic response. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 38(1):14–29, 2016.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C.J.C., Bottou, L., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4): 541–551, Winter 1989.

Li, Wenbin, Azimi, Seyedmajid, Leonardis, Aleš, and Fritz, Mario. To fall or not to fall: A visual approach to physical stability prediction. *arXiv 1605.01138*, 2016.

Lillicrap, Timothy P., Hunt, Jonathan J., Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K., Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

Pinheiro, Pedro O, Collobert, Ronan, and Dollar, Piotr. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pp. 1981–1989, 2015.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Shotton, Jamie, Sharp, Toby, Kipman, Alex, Fitzgibbon, Andrew, Finocchio, Mark, Blake, Andrew, Cook, Mat, and Moore, Richard. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

Smith, K. A., Battaglia, P. W., and Vul, E. Consistent physics underlying ballistic motion prediction. In *Proc. 35th Ann. Conf. Cognitive Science Society*, 2013.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

Wu, Jiajun, Yildirim, Ilker, Lim, Joseph J, Freeman, Bill, and Tenenbaum, Josh. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 127–135. Curran Associates, Inc., 2015.

Zhang, Renqiao, Wu, Jiajun, Zhang, Chengkai, Freeman, William T., and Tenenbaum, Joshua T. A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding. In *CogSci*, 2016.

Zheng, Bo, Zhao, Yibiao, Yu, Joey, Ikeuchi, Katsushi, and Zhu, Song-Chun. Scene understanding by reasoning stability and safety. *International Journal of Computer Vision*, 112(2):221–238, 2015.