
Perceiving Physical Equation by Observing Visual Scenarios

Siyu Huang*
Zhejiang University
siyuhuang@zju.edu.cn

Zhi-Qi Cheng*
Southwest Jiaotong University
zhiqicheng@gmail.com

Xi Li
Zhejiang University
xilizju@zju.edu.cn

Xiao Wu
Southwest Jiaotong University
wuxiaohk@gmail.com

Zhongfei (Mark) Zhang
Zhejiang University
zhongfei@zju.edu.cn

Alexander Hauptmann
Carnegie Mellon University
alex@cs.cmu.edu

Abstract

Inferring universal laws of the environment is an important ability of human intelligence as well as a symbol of general AI. In this paper, we take a step toward this goal such that we introduce a new challenging problem of inferring invariant physical equation from visual scenarios. For instance, teaching a machine to automatically derive the gravitational acceleration formula by watching a free-falling object. To tackle this challenge, we present a novel pipeline comprised of an Observer Engine and a Physicist Engine by respectively imitating the actions of an observer and a physicist in the real world. Generally, the Observer Engine watches the visual scenarios and then extracting the physical properties of objects. The Physicist Engine analyses these data and then summarizing the inherent laws of object dynamics. Specifically, the learned laws are expressed by mathematical equations such that they are more interpretable than the results given by common probabilistic models. Experiments on synthetic videos have shown that our pipeline is able to discover physical equations on various physical worlds with different visual appearances.

1 Introduction

Inference is one of the most basic and significant aspects of human intelligence [1] as well as AI [2]. As a high-level aspect of inference, the induction of universal laws from observations of our world is both the core basis and the goal of the scientific research. For example, Sir Isaac Newton saw an apple falling down and then was inspired to discover the law of gravitation. However, for a computing machine, the induction of laws based on visual observations is still a very challenging and open problem, and has been rarely explored by the existing literature until today.

In this paper, we introduce a new problem that we attempt to teach machine to automatically derive mathematical expressions of object dynamics from videos of a physical world. In contrast to the most recent approaches [3–5] which explores to learn object mechanical behaviors by the black box of deep neural networks, we aim at explicitly presenting the symbolic expressions of latent physical laws, leading to a more interpretable model and more visualizable results. A pioneer work [6] learns to derive mathematical equations from the data of physical experiments. While in this work, we propose to learn mathematical expressions directly from complicated videos.

*Equal contributions. This work was done when Siyu Huang and Zhi-Qi Cheng were visiting Carnegie Mellon University.

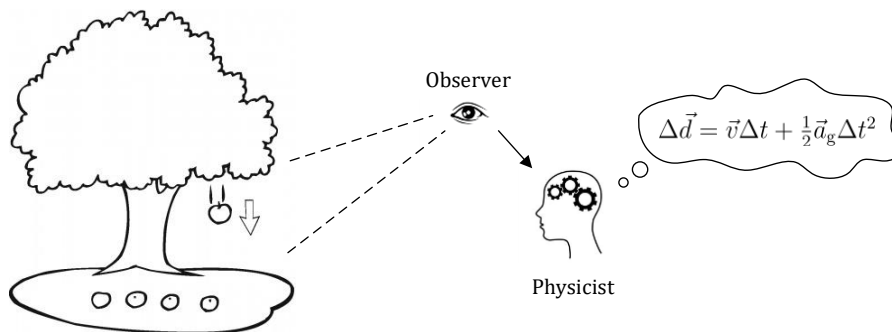


Figure 1: **Observing and thinking:** inferring physical equation from visual scenario. The ability to infer universal law of the environment is one of the significant high-level aspects of human intelligence.

Toward this goal, we propose a novel pipeline comprised of an *Observer Engine* and a *Physicist Engine*. The Observer Engine acts like an observer that watches the videos of a physical scenario and extracts the physical properties of objects in that scenario. Then the Physicist Engine imitates a physicist that summarizes the observed data and finally derives the mathematical equations.

In the experiments, we evaluate our pipeline on synthetic videos of multiple physical scenarios, showing that it is able to learn precise mathematical equations on these physical worlds with diverse visual appearances. We also explore several variants of models for the Observer Engine and the Physicist Engine respectively, so as to quantitatively establish baselines for relevant research in the future.

Our contributions are three-fold. First, we introduce a new problem of learning mathematical equations of object dynamics from videos, taking a step toward the automatic induction of universal laws for general AI. Second, we propose a novel pipeline to tackle this challenging problem. Third, empirical studies demonstrate the effectiveness of our approach on several synthetic physical scenarios.

2 Related Work

Physical Reasoning Physical reasoning has drawn much attention of AI researchers in recent years. Previous work on physical reasoning explored to learn the common sense knowledge of physical scenarios [7, 8] and to develop the simulation techniques for inferring the future states of physical systems [9–11]. A typical example is to predict whether a stack of blocks would fall [12–14]. In addition, the simulation and prediction of macroscopic physical phenomena, including weather events [15] and fluid [16, 17], were also studied by researchers. The “NeuroAnimator” [18] was the pioneer work to quantitatively simulate the physical dynamics of articulated bodies with neural networks. Today, the learning of object dynamics [19–22] becomes a research hotspot.

More recently, researchers incorporated the powerful deep neural networks into physical reasoning systems to enable a deeper understanding of the physical properties underlied in visual scenarios. Interaction Network (IN) [3] and Visual Interaction Network (VIN) [4] were successively proposed for modeling the dynamic relationships between physical objects in videos. Wu et al. [5] end-to-end learned a hybrid of graphics engines and physics engines to predict the long-term visual observations of a physical world.

In these approaches, the dynamics of objects and their interactions are generally modeled by the non-linear transformations of neural networks which are black-box models. The explicit symbolic expressions of object kinetic properties are not revealed and interpreted. In this work, we take the first step toward the interpretable physical reasoning model in which we attempt to summarize the object kinetic properties as precise physical equations through observing videos of a physical world.

Equation Regression In this work, we concentrate on inferring a physical equation from the visual scenarios, which is rarely explored in the existing literature. From the perspective of equation regression, there have been many efforts on learning the symbolic relationships from non-structured data [23–25]. In another aspect, several approaches [26, 27] learned to fit the parameters of Newtonian mechanics equations to physical systems depicted by videos. For instance, Wu et al. [28, 29] proposed a deep learning model to infer the physical properties (such as mass, volume, and coefficient friction) of objects from real-world videos. However, the symbolic expression of physical equations themselves are still not learned in these approaches.

A popular method for the learning of mathematical expression is called “symbolic regression” [30, 31], which is adopted in this work for the generation of physical equations. Symbolic regression is a machine learning technique that identifies a mathematical expression to minimize the customized error metric based on genetic programming [32] and evolutionary algorithm [33]. Unlike the traditional linear and non-linear regression methods that fit parameters to an equation of a given form, symbolic regression searches both the parameters and the form of equations simultaneously [6]. More details of our approach are discussed in the following sections.

3 Model

Our model learns to infer the inherent mathematical equation from video frames of a physical system. It consists of an *Observer Engine* and a *Physicist Engine*.

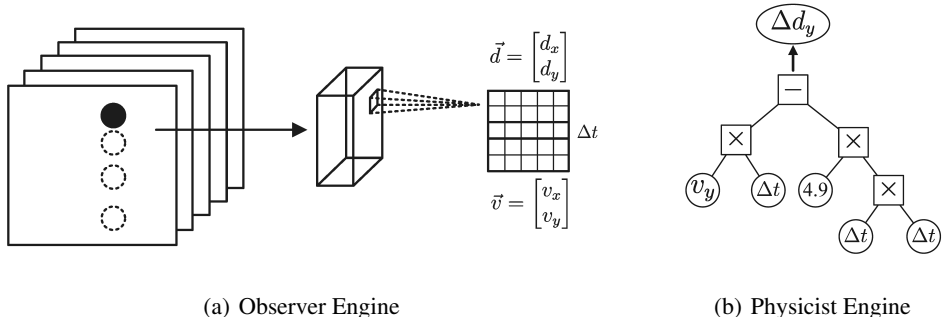


Figure 2: Our model is comprised of (a) the Observer Engine and (b) the Physicist Engine. At left, a video depicts that an object is in free-falling. The Observer Engine uses deep neural networks to extract the physical properties of the object. The Physicist Engine learns a mathematical expression of the object dynamics by evolving a syntax tree based on the property variables.

Observer Engine The Observer Engine acts like an observer that watches the videos of a physical world, and at the same time records the physical-property variables. As illustrated in Fig. 2(a), it captures the physical properties of the kinetic objects and the environment in videos. In this work, we use the Faster-RCNN [34] model to detect an object and localize its position \vec{d} according to coordinates of the bounding-boxes. In order to get a more precise object position, we employ a two-stage approach to refine the position on coarse-to-fine spatial scales. Specifically, a Faster-RCNN detector is applied on an image to get a coarse window of an object, then another Faster-RCNN detector is applied on the window to get a fine bounding-box. The two-stage approach ensures a precise object localization and a speed up of the detection procedure. The velocity \vec{v} of an object is computed by $\vec{v} = \Delta\vec{d}/\Delta t$, where Δt is the time interval between two video frames. Observation data \vec{d} , \vec{v} , and Δt are fed to the Physicist Engine serving as the independent variables.

Physicist Engine The Physicist Engine acts like a physicist that infers the equation based on the observations given by the Observer Engine. It takes a set of objects’ physical properties (output from the visual engine applied to a series of videos) as input. It outputs the equation between displacement $\Delta\vec{d}$ and the independent variables. In this work, we adopt symbolic regression with

genetic programming (GP) [30, 31] for the inference of mathematical equation, implemented based on Gplearn Toolkit².

As illustrated in Fig. 2(b), the formula is represented as a syntax tree. The variables, denoted as the round nodes, are leaves of the tree. The mathematical operations, denoted as the square nodes, connect the independent variables. Our goal is to find the best formula consisting of arbitrary independent variables and mathematical operations to minimize the mean absolute error (MAE) corresponding to the given target. At the very beginning, a population of formulas is randomly initialized. In an evolutionary manner, GP evolves the fittest ones of every generation until convergence. More details of GP are discussed in Section 4.2.

4 Experiments

4.1 Physical Scenarios

We conduct experiments on five types of physical scenarios. In each scenario, there is an object obeying the basic dynamic equation as

$$\Delta \vec{d} = \vec{v} \Delta t + \frac{1}{2} \vec{a} \Delta t^2 \quad (1)$$

$\Delta \vec{d}$ is the displacement vector and \vec{v} is the velocity vector. \vec{a} is the accelerated velocity vector corresponding to the specific object dynamics of each physical scenario, including

- **Drift** There is no external force applied on the object. The object drifts with its initial velocity. The accelerated velocity \vec{a}_{drift} is

$$\vec{a}_{\text{drift}} = 0 \quad (2)$$

- **Free-falling** The object goes into free-falling under gravity. The accelerated velocity \vec{a}_g is

$$\vec{a}_g = \begin{bmatrix} 0 \\ -g \end{bmatrix} \quad (3)$$

g is the gravitational acceleration constant.

- **Parabola** The object moves along a parabola under gravity. The accelerated velocity is the same as \vec{a}_g defined in Eq. 3, while the object has a random initial horizontal velocity v_x .
- **Slope** The object slides downhill on a smooth slope. The accelerated velocity \vec{a}_{slope} is

$$\vec{a}_{\text{slope}} = \begin{bmatrix} g \sin \theta \cos \theta \\ -g \sin^2 \theta \end{bmatrix} \quad (4)$$

where θ is the slope gradient.

- **Spring** The object is connected to a horizontal wall with a visible spring obeying Hooke’s law. The accelerated velocity \vec{a}_{spring} is

$$\vec{a}_{\text{spring}} = \begin{bmatrix} 0 \\ -k \cdot (d_y - D - X) / m \end{bmatrix} \quad (5)$$

The Hooke’s constant k , attachment point y-coordinate D , and equilibrium distance X are constants in an experiment.

For each physical scenario, we generate 300 videos for training the Observer Engine, and 100 videos for testing our pipeline, where each video has 100 frames. To simulate the real-world scenarios, by following [4] we use a random Cifar-10 [35] natural image as the background of each synthetic video. There is no overlap of background images between training set and testing set. The image size of a video frame is set as 38K×38K because a larger image size enables a smaller relative error when estimating the object position. The sizes of objects in videos are the same. As for the constant independent variables, we fix $g = 9.8$, $k = 2$, $D = -15\,000$, and $X = 5\,000$ in the experiments. We do not fix the slope gradient θ as it is an observable variable. The object initial position, the object initial velocity, object mass, and the slope gradient are random in every video.

²<http://gplearn.readthedocs.io/en/stable/index.html>

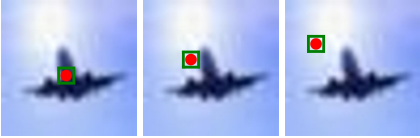
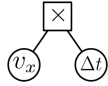
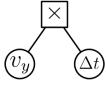


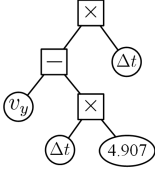
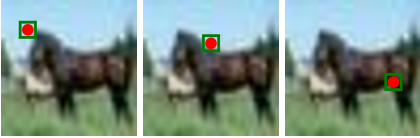
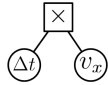
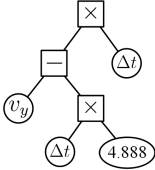
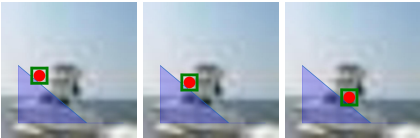
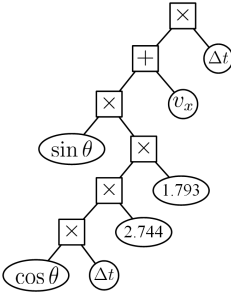
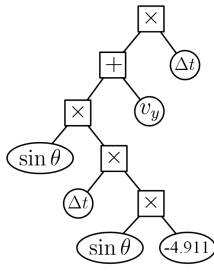


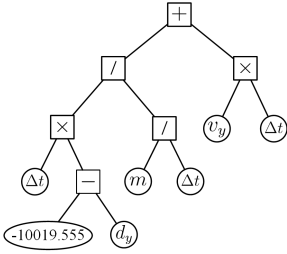
Physical Scenario	Learned Δd_x	Learned Δd_y
 Dirft (http://bit.ly/2L6JR8W)	 $v_x \Delta t$	 $v_y \Delta t$
 Free-falling (http://bit.ly/2k3HIyu)	 0	 $v_y \Delta t - 4.907 \Delta t^2$
 Parabola (http://bit.ly/2It4Kcq)	 $v_x \Delta t$	 $v_y \Delta t - 4.888 \Delta t^2$
 Slope (http://bit.ly/2L7CIF1)	 $v_x \Delta t + 4.920 \sin \theta \cos \theta \Delta t^2$	 $v_y \Delta t - 4.911 \sin^2 \theta \Delta t^2$
 Spring (http://bit.ly/2KuAfnj)	 0	 $v_y \Delta t - (d_y + 10019.555) / m \cdot \Delta t^2$

Figure 3: **Physical scenarios and our learned equations.** In each scenario, the object moves under particular dynamic equations. Please click the URLs to watch the synthetic videos if interested. Results show that our method is able to learn correct mathematical equations with relatively accurate physical constants in all of the scenarios. The syntax trees are shown together with the equations. The images are resized to 600×600 for visual clarity.

4.2 Learning Details

In the Observer Engine, we use a two-stage Faster-RCNN object detector whose backbone network is the pretrained ResNet-101 [36] model. In each stage, 2,000 and 1,000 images are randomly sampled as training set and validation set respectively. In the first stage, we train an object detector to detect objects on the original video frames ($38K \times 38K$ pixels). The object detector is trained by the SGD optimizer with a learning rate of 0.005, a batch size of 4, and a learning rate decay of 8. After 4 epochs of training, the detection model gets converged and obtains a 95.5% MAP on validation set. In the second stage, we crop a $4K \times 4K$ part from the original image with the bounding box output by the first stage. Then we train another object detector to refine to a more precise object position. The detector is trained by the SGD optimizer with a learning rate of 0.001, a batch size of 1, and a learning rate decay of 4. The detection model gets converged after 6 epochs and obtains a 97.7% MAP on validation set. The object position is estimated as the center point of the bounding box output by the second detector. Table 1 shows that the euclidean distance error of our estimated position is less than 1 pixel in average.

In the Physicist Engine, we use genetic programming to evolve the syntax tree which represents a mathematical equation. The independent variables include position d_x, d_y , velocity v_x, v_y , mass m , and time interval Δt . The mass m of object is set as known by the Physicist Engine as it could be easily estimated in the real world. In scenario of Slope, the independent variables also include $\sin \theta$ and $\cos \theta$. We do not use θ as independent variable, because the difference between θ and $\sin \theta$ is numerically trivial for regression under small θ . The arithmetic operations, including addition (+), subtraction (-), multiplication (\times), and division ($/$), are used for every scenario. Genetic operations including crossover ($p = 0.5$), subtree mutation ($p = 0.15$), hoist mutation ($p = 0.15$), and point mutation ($p = 0.15$) are employed in evolution.

5 Results

5.1 Perceiving Mathematical Equation

We show that our pipeline is able to perceive mathematical equations on a variety of physical scenarios with diverse visual appearances in Fig. 3. Five different physical scenarios are shown in the first column, where the objects are moving under corresponding dynamic equations. Our Observer Engine detects the bounding boxes (green) of objects, providing precise object positions to the Physicist Engine. At the right part of Fig. 3, we show the mathematical equations and syntax trees learned by the Physicist Engine, where x -component Δd_x and y -component Δd_y of displacement $\Delta \vec{d}$ are respectively shown in the second column and the third column.

Fig. 3 demonstrates that the Physicist Engine can learn dynamic equations of all the physical scenarios, even though the dynamic equations of some scenarios (Slope and Spring) are complex. Not only the symbolic relationships are correctly learned, the physical constants in mathematical equations are also accurately estimated by our method (e.g., the ground truth $g = 4.9$ in scenarios of Free-falling, Parabola, and Slope; the ground truth $D + X = -10\,000$ in scenario of Spring).

Please note that every equation in Fig. 3 is generated based on the same independent variables (except particular arguments of environment) and the same arithmetic operations across all the scenarios. It reveals that our method is scalable to many other physical systems which are not included in the experiments of this work. In addition, our Observer Engine is effective in complex real-world background images with diverse visual appearances, indicating that our method is probably able to be applied to real-world videos in a future study.

5.2 Baselines

We have explored several variants of models for the Observer Engine and the Physicist Engine respectively, so as to quantitatively establish baselines for our newly proposed problem.

For the Observer Engine, we study two baseline methods for a quantitative comparison with our used Two-stage Detector:

Table 1: Baselines of the Observer Engine

Method	MED (pixels)
Single Detector	39.76
Detection + Segmentation	5.26
Two-stage Detector	0.55

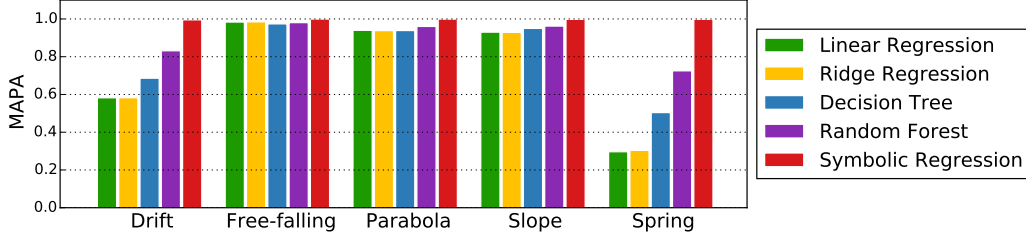


Figure 4: Baselines of the Physicist Engine.

- **Single Detector:** It is the basic Faster R-CNN [34] model, where the Region Proposal Network (RPN) is used for estimating the bound of object. The bound is used for computing the position of object.
- **Detection + Segmentation:** First, a Single Detector is used for getting an object bound. Then, a fully convolutional network (FCN) is used for segmenting the object in the pre-detected bound to localize a more accurate object position.
- **Two-stage Detector:** The method adopted by this work. Two Single Detectors are stacked to detect the object in a coarse-to-fine strategy.

Table 1 shows the baseline performances of the Observer Engine, under the metric of mean Euclidean distance (MED) between the estimated position and the ground-truth position. Comparing Detection + Segmentation to Single Detector, the segmentation operation is able to refine the output of single RCNN model by about 8X. Comparing Two-stage Detector to Single Detector, the second detector successfully reduces the error by about 72X based on output of the first detector. The Two-stage Detector used in this work shows a surprising performance such that the mean error is 0.55 pixel under the 38K×38K coordinate system, indicating that it can be extended to various visual scenarios and real-world applications.

For the Physicist Engine, we also study a series of common regression methods for a comparison with the symbolic regression algorithm used in this work. The baselines include (1) linear regression, (2) ridge regression, (3) decision tree, and (4) random forest. These models are implemented based on the scikit-learn toolbox [37]. Fig. 4 shows the baseline performances of the Physicist Engine, under the metric of mean absolute percentage accuracy (MAPA) between ground-truth displacement Δd and estimated displacement $\Delta \hat{d}$ as

$$\text{MAPA} = 1 - \frac{1}{N} \sum_{i=1}^N \left| \frac{\Delta \hat{d} - \Delta d}{\Delta d} \right| \quad (6)$$

MAPA denotes the relative estimation accuracy. In Fig. 4, baseline methods perform well on scenarios of Free-falling, Parabola, and Slope. While on scenarios of Drift and Spring, methods show distinct difference such that decision tree and random forest perform significantly better than linear regression and ridge regression. The main reason is that decision tree and random forest are non-linear models thus having much better non-linear representation capabilities than linear/ridge regression. Our symbolic regression algorithm performs the best such that its accuracy is almost 1.0 on every scene. Apparently the syntax tree of symbolic regression can perfectly represent the relationships (such as multiplication and division) between independent variables, such that symbolic regression is naturally suited for learning mathematical equations.

Table 2: Ablation study of our pipeline (R^2 score). Baseline methods of the Observer Engine and the Physicist Engine are row-wise listed and column-wise listed respectively. LR: linear regression; RR: ridge regression; DT: decision tree; RF: random forest; SR: symbolic regression; GT: ground-truth equation.

	LR	RR	DT	RF	SR	GT
Single Detector	0.917	0.908	0.812	0.932	0.926	0.904
Detection + Segmentation	0.958	0.958	0.832	0.922	0.954	0.954
Two-stage Detector	0.945	0.944	0.960	0.983	1.000	1.000
Ground-truth Position	0.945	0.944	0.970	0.984	1.000	1.000

Table 2 shows a more comprehensive ablation study of our pipeline, where the baseline methods of two engines are pairwise combined to be evaluated in all the physical scenarios. We use R^2 coefficient score as the metric to evaluate the fitting goodness in this study. It is interesting that when working with Single Detector or Detection + Segmentation, sometimes the methods of Physicist Engine perform better than the ground-truth equation. It is mainly because these methods eliminate some position errors in fitting. We observe that our pipeline (a combination of Two-stage Detector and SR) gets an 1.000 R^2 score, as it successfully identifies all of the dynamic equations as well as accurately estimates the constants, as shown in Fig. 3. Comparing Two-stage Detector with Ground-truth Position and comparing SR with GT, both methods show performances close to the ground-truth, indicating that they have good compatibilities with different methods of the other engine.

6 Discussion

We have introduced a new problem of deriving mathematical equations from physical scenarios, taking a step toward the goal of reasoning about universal laws from a complex environment. We have presented a pipeline including an Observer Engine and a Physicist Engine to tackle this problem for the first time. In the experiments, we have shown that our pipeline is able to perceive dynamic equations on various physical scenarios whose visual appearances are quite different. Ablation studies conducted on combinations of baselines further demonstrate the effectiveness of our pipeline. In general, our pipeline is an effective template for reasoning about the physical and dynamic systems. By combining deep learning, symbolic learning, and evolutionary algorithm, we show the potential of a hybrid machine learning system for AI reasoning. We hope this work may inspire future study on inference, induction, and conceptual understanding of general AI.

In the future, an important work is to demonstrate the proposed pipeline in real-world scenarios which may have more unknown noise than the synthetic data. It will also be important to develop techniques to handle the multi-object physical system [3–5], in which there are interactions between objects other than the dynamics of a single object. It is a challenging and meaningful task to learn equations of a composite set of dynamic laws. In addition, our pipeline is probably able to be extended to some practical applications, e.g., helping physicists to summarize and analyse the experimental data in complex visual scenarios.

References

- [1] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011.
- [2] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.
- [3] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *NIPS*, pages 4502–4510, 2016.
- [4] Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *NIPS*, pages 4542–4550, 2017.
- [5] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. In *NIPS*, pages 152–163, 2017.
- [6] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.
- [7] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.
- [8] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The” something something” video database for learning and evaluating visual common sense. In *ICCV*, 2017.
- [9] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. 2016.
- [10] Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. “what happens if...” learning to predict the effect of forces in images. In *ECCV*, pages 269–285. Springer, 2016.
- [11] Zhihua Wang, Stefano Rosa, Bo Yang, Sen Wang, Trigoni Niki, and Andrew Markham. 3d-physicsnet: Learning the intuitive physics of non-rigid object deformations. In *IJCAI*, 2018.
- [12] Abhinav Gupta, Alexei A Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, pages 482–496. Springer, 2010.
- [13] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *ICML*, pages 430–438, 2016.
- [14] Wenbin Li, Seyedmajid Azimi, Aleš Leonardis, and Mario Fritz. To fall or not to fall: A visual approach to physical stability prediction. *arXiv preprint arXiv:1604.00066*, 2016.
- [15] Evan Racah, Christopher Beckham, Tegan Maharaj, Samira Ebrahimi Kahou, Mr Prabhat, and Chris Pal. Extremeweather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. In *NIPS*, pages 3402–3413, 2017.
- [16] SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, Markus Gross, et al. Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics*, 34(6):199, 2015.
- [17] Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. *ICLR*, 2018.
- [18] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 9–20, 1998.
- [19] Roozbeh Mottaghi, Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *CVPR*, pages 3521–3529, 2016.
- [20] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *ICRA*, pages 173–180, 2017.
- [21] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *ICLR*, 2017.
- [22] Sebastien Ehrhardt, Aron Monszpart, Niloy J Mitra, and Andrea Vedaldi. Learning a physical long-term predictor. *arXiv preprint arXiv:1703.00247*, 2017.
- [23] Liliana Teodorescu and Daniel Sherwood. High energy physics event selection with gene expression programming. *Computer Physics Communications*, 178(6):409–419, 2008.

- [24] Ilya Sutskever and Geoffrey E Hinton. Using matrices to model symbolic relationship. In *NIPS*, pages 1593–1600, 2009.
- [25] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O’Neill, Robert I McKay, and Edgar Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.
- [26] Kiran S Bhat, Steven M Seitz, Jovan Popović, and Pradeep K Khosla. Computing the physical parameters of rigid-body motion from video. In *ECCV*, pages 551–565. Springer, 2002.
- [27] Marcus A Brubaker, Leonid Sigal, and David J Fleet. Estimating contact dynamics. In *ICCV*, pages 2389–2396, 2009.
- [28] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NIPS*, pages 127–135, 2015.
- [29] Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, volume 2, page 7, 2016.
- [30] Douglas Adriano Augusto and Helio JC Barbosa. Symbolic regression via genetic programming. In *Proceedings. Sixth Brazilian Symposium on Neural Networks*, pages 173–178, 2000.
- [31] Orazio Giustolisi and Dragan A Savic. A symbolic data-driven technique based on evolutionary polynomial regression. *Journal of Hydroinformatics*, 8(3):207–222, 2006.
- [32] John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 1994.
- [33] Ben McKay, Mark J Willis, and Geoffrey W Barton. Using a tree structured genetic algorithm to perform symbolic regression. In *International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 487–492, 1995.
- [34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.
- [35] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.
- [36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.
- [37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.