
Solving Physics Puzzles by Reasoning about Paths

Augustin Harter
Bielefeld University
aharter@techfak.uni-bielefeld.de

Andrew Melnik
Bielefeld University
andrew.melnik.papers@gmail.com

Gaurav Kumar
Bielefeld University

Dhruv Agarwal
Indian Institute of Information Technology

Animesh Garg
University of Toronto, Vector Institute, Nvidia

Helge Ritter
Bielefeld University

Abstract

We propose a new deep learning model for goal-driven tasks that require intuitive physical reasoning and intervention in the scene to achieve a desired end goal. Its modular structure is motivated by hypothesizing a sequence of intuitive steps that humans apply when trying to solve such a task. The model first predicts the path the target object would follow without intervention and the path the target object should follow in order to solve the task. Next, it predicts the desired path of the action object and generates the placement of the action object. All components of the model are trained jointly in a supervised way; each component receives its own learning signal but learning signals are also backpropagated through the entire architecture. To evaluate the model we use PHYRE - a benchmark test for goal-driven physical reasoning in 2D mechanics puzzles.

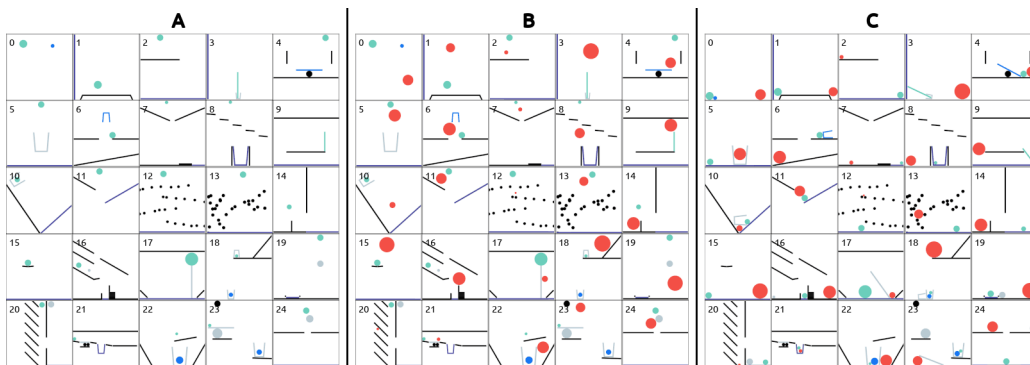


Figure 1: 25 templates of PHYRE-B (BALL) puzzles [1] *A*: initial puzzle scenes. Task: a green target-object has to touch a blue goal-object for 3 seconds. *B*: placing red action-balls that solve the puzzles. *C*: solved puzzle scenes (green target-objects are in firm contact with blue goal-objects). Human players can try to solve these PHYRE tasks here: (<https://player.phyre.ai>).

1 Introduction

Consider the following physics puzzle: select a location for dropping a ball so that it hits a number of other objects in such a manner that a "target" object gets firm contact with a "goal" object (Fig. 1).

PHYRE benchmark¹[1] offers a large set of physics challenges in the described format. The ability of humans to solve even intricate instances of such puzzles reflects a high level of physics cognition that so far is unparalleled in machines.

2 Methods

2.1 Action generation model

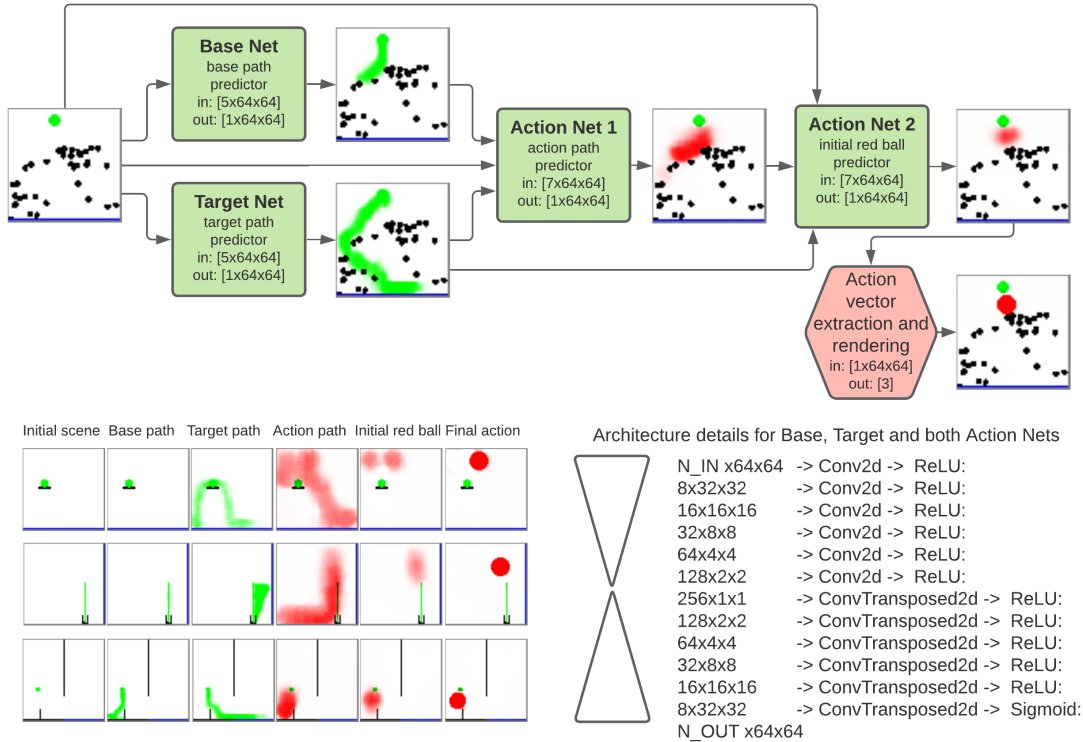


Figure 2: *Top*: Action generation pipeline. NNs modules are highlighted with green rectangles. The task’s initial scene is presented to the agent as five bitmap channels; one channel for each object class: Green target-object, blue dynamic goal-object, blue static goal-object, dynamic grey objects, static black objects. *Bottom left*: Model prediction examples. All examples of the generated *final action* in the figure solve the corresponding tasks. *Bottom right*: Model architecture details: Every Conv2d and ConvTransposed2d Layer has a kernel size of 4x4, stride of 2 and padding of 1.

Here we describe the building blocks of our NN model²outlined in Figure 2, and compare them to the introspection of players trying to solve such a puzzle:

1. *Base Net*. Humans can get useful information by imagining how the scene will develop without the solving intervention. Therefore, we train a *Base Net* neural network (NN) using the PHYRE simulator to predict the path (*Base Path*) of the green target-object (Fig. 1A) when there is no interaction with the red action-ball. The *Base Net* input consists of the five channel bitmap of the initial scene. Target output is the probability density of 2D-points predicted to be traversed by the *Base Path*. The training procedure is described in detail in the section 2.3 *Training*.
2. *Target Net*. Human players can imagine the path the green target-object should follow in order to solve the puzzle on a set of acquired intuitions: Getting closer to the goal is better as well as following the direction of gravity and obeying interaction constraints. Therefore,

¹<https://player.phyre.ai>

²<https://github.com/ndrwmlnk/PHYRE-Reasoning-about-Paths>

we train a *Target Net* (Figure 2) to generate possible solution paths (*Target Path*) of the green target-object without information about the red action-ball. This gives the *Target Net* the freedom to "dream up" *Target Paths* that appear "interaction incomplete" since they become physically valid only through some, so far unknown, interaction with the red action-ball. Input to the *Target Net* is the initial scene 5-channel bitmap. Target output now is the probability map (2D-density of traversed points) of the *Target Path*.

3. *Action Net 1*. Humans are able to reason and imagine which trajectories the red action-ball has to take to "add" the missing interaction to turn a *Base Path* into a potential *Target Path*. Here humans might heuristically start from the point where *Base Path* and *Target Path* diverge. *Action Net 1* generates possible *Action-Ball Paths*. NN input: Initial scene 5-channel bitmap and 2-channel probability maps of the *Base Path* from step 1 and the *Target Path* from step 2. NN output: Probability map of the *Action-Ball Path*.
4. *Action Net 2* generates a probability map of the initial red action-ball position (Fig. 1B). NN input: Initial scene 5-channel bitmap and probability maps of the *Target Path* from step 2 and the *Action Path* from step 3.
5. Convert the red action-ball probability maps from step 4 to a 3-dim action vector, comprised of x, y and radius values. This is done with a non-learning algorithm, which randomly selects initial radius and initial position proposals (x, y) from pixels that are over a certain threshold value. Then it iteratively tries improve the overlap of the red action-ball rendered from a new action vector, and the probability map by selecting position and radius values within a close neighborhood. We sample 5 different initial positions and radius value and update each of them 5 times. We take these as the first 5 actions for solving the task. If all fail, we randomly sample from them and add Gaussian noise until the task is solved or the limit of 100 tries is reached.

2.2 Encoder-Decoder Hourglass-like model

The artificial agent needs knowledge about the whole scene for reasoning about paths: Object interactions might be understood with local information but to propose a path which connects the green target-object and blue goal-object the whole scene needs to be considered. This motivated the following architecture for *Base Net*, *Target Net*, *Action Net 1*, and *Action Net 2*, illustrated in Figure 2. A stack of convolutional layers 'folds' the input channels into a 256-dimensional encoding of the complete scene. Then a similar stack of transposed convolutional layers 'unfolds' this global encoding into the desired number of output channels, each having the same dimensionality as the input channels. We use *ReLU* as the activation function for hidden layers and the *sigmoid* function for the output layer to allow smooth prediction.

2.3 Training

All 4 NNs of the model are trained jointly in a supervised way; each NN receives its own learning signal but learning signals are also backpropagated through the entire architecture. We collect *BasePath*, *TargetPath*, *ActionPath* and the initial red action-ball bitmap channels from rollouts in the PHYRE simulator and use them to impose a cross-entropy loss between every pixel of the NNs output and the corresponding ground truth bitmaps. *Action Net 1* and *Action Net 2* receive predicted *BasePath* and *TargetPath* channels from *BaseNet* and *TargetNet* NNs. For training we use a data set that contains 10 solving rollouts per task, 80 tasks per template, and 25 PHYRE-B templates. We randomly shuffle the samples and split them into 625 batches with a batch size of 32 and train the NNs for 10 epochs.

3 Results

3.1 Task Solving Performance

PHYRE uses the *success* metric to score performance: Agents can try up to 100 attempts per task and the area under the logarithmically scaled *percentage of tasks solved* curve is the *success* value. See [1] for more details. As described in 2.1, we generate 5 action proposals for each task based on the action ball prediction from *Action Net 2*. If the proposed actions led to unsuccessful attempts,

we sample further action vectors from a multivariate normal distribution centered at the original action proposals. We slowly increase its standard deviation during the 100 tasks (starting from 0.02), leading to increased deviation from the original action for later attempts. There are 25 templates in the PHYRE-B (BALL) problem [1]. In the within-template setting the model is evaluated on tasks that share the same template with training tasks. Such tasks differ in the placement and size of scene objects, but not in their otherwise structure. In the cross-template setting, the model is evaluated on tasks from templates that were not shown during training. Bakhtin et al. [1] use a "process that deterministically splits the tasks into 10 folds containing a training, validation, and test set" allowing fair comparison between agents and studies. Table 1 shows success values of our trained NN model which are collected individually for each template in each fold and then averaged over all 10 folds.

Table 1: Success (*auc.*) and percentage (*perc.*) of solved tasks after 10 attempts in PHYRE-B (BALL) templates (Fig. 1)

template	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	mean
within <i>auc.</i>	80	89	87	95	79	82	86	62	69	77	31	53	52	65	84	78	39	41	69	45	46	29	30	45	42	62
within <i>perc.</i>	86	98	90	100	86	90	94	62	72	81	27	57	55	70	93	83	37	44	82	51	48	23	27	46	44	66
cross <i>auc.</i>	51	54	67	67	57	63	-	30	24	14	14	38	48	51	15	25	14	5	37	17	24	3	9	6	6	31
cross <i>perc.</i>	50	51	72	70	58	67	-	31	22	13	11	34	50	53	11	21	12	6	35	13	22	0	7	4	6	31

4 Discussion and related works

Our action generation model (Table 1) falls behind the baseline DQN’s performance [1] of 77 within template *success*, 81 within template *percentage* of solved tasks after 10 attempts, 37 cross template *success*, and 35 cross template *percentage* of solved tasks after 10 attempts. But one important difference is to be noted: The baseline DQN *only* uses a deep reinforcement learning setup to guess the success value of a batch of 10000 possible combinations of action parameters (x , y , radius) with a certain level of grid discretization, while our model performs a basic reasoning about developments in the scene and interactions about objects, thus capable of generating an informed action proposal in an explainable fashion. Our model can also be thought of as a kind of a jointly trained autoencoder for multi-modal fusion [2].

Forward prediction for physical reasoning [3], an error-based dynamic model learning [4], or continues-action-space policy gradient algorithms [5] are possible ways to improve performance and generalize learning. Allen et al. [6] introduced the “Virtual Tools” game to measure the capacity of human beings for flexible, creative tool use. The study proposes that the flexibility of human physical problem solving rests on an ability to imagine the effects of hypothesized actions. Our deep learning architecture (Fig. 2) can serve as a mechanism to imagine the effects of such hypothesized actions. Kurutach et al. [7] demonstrated on a rope manipulation example that a plausible sequence of observations evolving from its current configuration to a desired goal state can be used as a reference trajectory for control. In contrast, the PHYRE type problems have an additional constraint of one control-action per episode, and therefore requires a different NN architecture.

Melnik et al. [8] described a set of functional modules for specific types of interaction primitives, which are common to a broad range of arcade ball-game environments. Results of this case study in different Atari ball-games suggest that human-level performance can be achieved by a learning agent within a human amount of game experience (10-15 minutes game time) when a proper decomposition of an environment or a task is provided. However, automatization of such decomposition remains a challenging problem.

5 The broader impact statement

Human beings start to understand and reason about the goal-driven interaction of physical objects in the environment in early childhood, and therefore this skill also contributes to learning of other skills, e.g. language, logic, etc. Solving this problem in simpler 2D environments is likely to be an important step in tackling the difficult general problem in real-world 3D environments. In this work, we developed such a goal-driven intuitive-physics-reasoning NN model with strong generalization properties mirroring those of humans.

References

- [1] Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. In *Advances in Neural Information Processing Systems*, pages 5082–5093, 2019.
- [2] Timo Korthals, Marc Hesse, Jürgen Leitner, Andrew Melnik, and Ulrich Rückert. Jointly trained variational autoencoder for multi-modal sensor fusion. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2019.
- [3] Rohit Girdhar, Laura Gustafson, Aaron Adcock, and Laurens van der Maaten. Forward prediction for physical reasoning. *arXiv preprint arXiv:2006.10734*, 2020.
- [4] Nicolas Bach, Andrew Melnik, Federico Rosetto, and Helge Ritter. An error-based addressing architecture for dynamic model learning. In *6th International Conference, LOD 2020, Siena, Italy, Proceedings*, 2020.
- [5] Nicolas Bach, Andrew Melnik, Malte Schilling, Timo Korthals, and Helge Ritter. Learn to move through a combination of policy gradient algorithms: Ddpg, d4pg, and td3. In *6th International Conference, LOD 2020, Siena, Italy, Proceedings*, 2020.
- [6] Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. The tools challenge: Rapid trial-and-error learning in physical problem solving. *arXiv preprint arXiv:1907.09620*, 2019.
- [7] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In *Advances in Neural Information Processing Systems*, pages 8733–8744, 2018.
- [8] Andrew Melnik, Sascha Fleer, Malte Schilling, and Helge Ritter. Modularization of end-to-end learning: Case study in arcade games. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Workshop on Causal Learning*, 2018.