Predicting Stability of Towers using Graph Neural Network

Irmak Guzey
Bogazici University
Computer Engineering
irmak.guzey@boun.edu.tr

Ahmet E. Tekden
Bogazici University
Computer Engineering
tekdenahmet@gmail.com

Emre Ugur
Bogazici University
Computer Engineering
emre.ugur@boun.edu.tr

Abstract

Graph Neural Networks are becoming more popular and widely used in recent years specially as their popularity on environments which has physically observable and learnable multiple objects has rapidly grown. In this paper we talk about an environment that has multiple objects that are put on top of each other so that they could construct a tower and a prediction is made with Graph Neural Network on stability of each object through different trajectories. The network learns how the relations between objects effect each other and predicts their future state (state being the stability).

1 Introduction

Interaction of objects between each other has a complex effect to the system which makes it challenging to make predictions about the future states of the objects. Variety in number of objects and the size of the objects are also features that harden the problem. The main effect of this *interaction between objects* is the chain effect which can be put as the propagation of the motion of an object onto another object. In order to efficiently observe and learn this effect the data that is collected needs to involve easily observable interactions and it should be robust to noise. In this paper an environment with this chain effect is built and the analysis is made with a Graph Neural Network.

In this work we use relations between the objects to predict future states of the objects. We propose a propagation network that is recently proposed by (Li et al., 2019b), that has two multilayer perceptron; one for extracting the effect of each object and one for propagating this effect through relations between object. Propagation Networks are explained in detailed in the Section *Proposed Model*.

Our system is built with rectangle-shaped objects put together in a way that they can construct a tower and their relation is verified with the distance between them. We have two different environments; one is the *Construction Environment* with same sized objects and after the tower construction is done, an extra object is dropped on top of the tower, the second environment is the *Jenga Environment* with different sized objects (same height but different width) and after the tower construction is done one random object is removed from the system in this environment. With both of these environments the computer learns to predict future states of stability of each object, from its' interaction experience in the simulator. This prediction is implemented based on an effect prediction framework that can handle multiple interacting objects.

2 Related Work

The past years has seen quite a progress in modeling physics. Some of these progress and work has been put by using probabilistic approaches. (Battaglia et al., 2013) with a Bayesian model showed that physics of stacked cuboids can be modeled with this and (Deisenroth and Rasmussen, 2011) suggested a probabilistic dynamic model that is capable of predicting the next state. Recently, deep learning methods are also used to model physics. A deep approach for predicting the parameters of a simulation engine is proposed by (Wu et al., 2015). And (Lerer et al., 2016) used raw images of block towers to train a deep network to predict stability of objects.

Motion prediction is also another important topic that use deep learning, convolutional neural networks and/or graph neural networks. And this topic is one of the future applications of this work. Recent years there has been important amount of work that predicts motion from images. (Mottaghi et al., 2016a) trained a CNN on static images

to predict motion. (Mottaghi et al., 2016b) used CNNs on static images to predict the next motion of the objects when an external force is applied on the objects. Also convolutional neural networks are used in extracting action-effect prediction from videos. (Finn et al., 2016) trained a recurrent CNN to predict the future image frames using only the current image frame and actions of the robot.

Graph neural networks also allow next state / motion prediction in complex systems by learning object representations and relations among them. Hence it has been widely used in physics prediction. Physics engines based on GNNs are firstly proposed by (Battaglia et al., 2016) and (Chang et al., 2017). These models do split object oriented and relation oriented perspective. The missing part of these proposals were that it worked with flaw when there were propagation of effects between objects (chain effect). Network models with an extra information on nodes was proposed by (Li et al., 2019b), (Mrowca et al., 2018), (Li et al., 2019a), (Watters et al., 2017) and hybrid network models which encode object information directly from images via CNNs and which predict the next states of the objects with the use of GNNs were proposed by (van Steenkiste et al., 2018).

3 Proposed Model

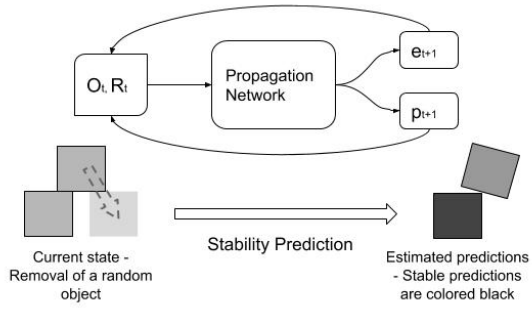

Figure 1: Visualization of the network used

In this section, we introduce our framework. Our framework consists of a propagation network that takes definitions of object states and relations and predict the future states of the manipulated objects. In the following we give technical details of these models.

Preliminaries: If we assume that the number of objects in the environment is O, then we can express the scene with a graph structure $G = \langle O, R \rangle$ where the nodes $O = \{o_i\}_{i=1:N^o}$ represent the set of objects, each node $o_i = \langle x_i \rangle$ storing object related information $x_i = \langle q_i, y_i, w_i \rangle$ where q_i being the position, y_i being the altitude and w_i being the width of the object and the edges $R = \{r_k\}_{k=1:N^r}$ represent the set of relations between the objects. Each edge $r_k = \langle d_k \rangle$ represents the relation between objects i and j by holding their distance towards each other as $d_k = q_i - q_j$. And also since the relations between objects only consist of them touching each other, the relation also doesn't have an individual physical attribute.

Propagation networks use the states of the objects and their relations on two different multilayer perceptron. And the data received becomes encoded. This encoding is carried out by two encoders, one for the relations denoted by f_R^{enc} and one for the objects denoted by f_O^{enc}, defined as follows:

$$c_{k,t}^r = f_R^{enc}(r_{k,t}), \quad k = 1...N^r \tag{1}$$

$$c_{i,t}^o = f_O^{enc}(o_{i,t}), \quad i = 1...N^o \tag{2}$$

where $o_{i,t}$ and $r_{k,t}$ represent the object i and the relation k at time t, respectively.

To predict the next state of the system, these encoders are used in the subsequent propagation steps within two different propagator functions, f_R^l for relations and f_O^l for objects, at the propagation step l as follows:

$$e_{k,t}^l = f_R^l(c_{k,t}^r, p_{i,t}^{l-1}, p_{j,t}^{l-1}), k = 1...N^r \tag{3}$$

$$p_{i,t}^l = f_O^t\left(c_{i,t}^o, p_{i,t}^{l-1}, \sum_{k \in N_i} e_{k,t}^{l-1}\right), i = 1...N^o \tag{4}$$

where N_i denotes the set of relations where object i is begin a part of, and while $e_{k,t}^l$ represents the propagating effects from relation k at propagation step l and time t, $p_{i,t}^l$ represents the propagating effects from object i at propagation step l at time t, respectively.

4 Experimental Setup

We evaluate our model in simulation through a set of experiments. In the following, we give the details of the experimental setups designed to assess the generalization performance to the changing number of objects and time steps and transferability of our model to different object-relation distributions.

4.1 Simulated Setup

Our environment is built by using physics motor Pymunk and the visualization is made with Pyglet libraries in Python. Base of the environment is a tower construction with same sized rectangle-shaped objects. There are two different model built on top of this basic environment and separate predictions are made with these two different models. Both of these models are built on top of on existing tower.

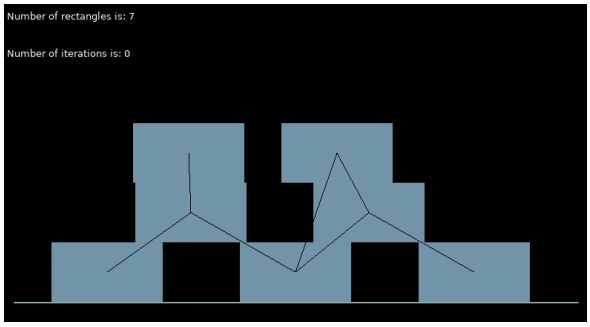

Figure 2: Example Tower with 7 Objects

After collecting different data for each of the model, prediction after one of the actions (drop or remove one object) is made.

After having the predictions, actions such as *drop to demolish* (for the construction model) and *remove to demolish* (for the jenga model) can be made through simulation. These actions try to demolish the tower by trying out different positions to drop the object for construction model or different objects to remove for the jenga model and predicting the total stability of the system and then by doing the action that causes the least stability. Results of each of the action is explained in detail in *Results* section.

Models are explained in detailed in the next two subsections.

4.1.1 Construction Environment

This model drops one more object on top of the existing tower and saves the trajectories of each object. The range of the object to be dropped varies between left edge and the right edge of the top layer.

Trajectories are taken after the drop of the object and the trajectories and statement whether each object stays stable or not is afterwards given to a graph neural network as training and validation data set.

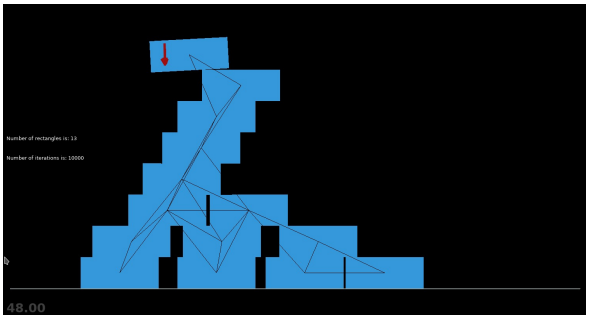

Figure 3: Example Tower After Dropping One Object on Top of a Tower with 13 Objects

4.1.2 Jenga Environment

This model removes one random object from the existing tower and saves the trajectories of each object. Only one object is removed at each trajectory and the object to be removed is selected random and can be any of the existing objects.

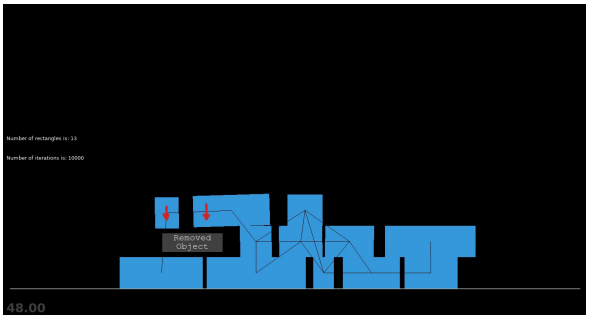

Figure 4: Example Tower After Removing One Object

At Figure 4 the third object from the top of the right group of the tower is removed. And the trajectory and stability (the two objects' on top of the one that is removed being 0 and the others' being 1) of each object are calculated and saved.

4.2 Implementation Details

Our propagation network takes object position as object features and relation type as relation features. Specifically, object encoder takes y position of the object and is implemented with a MLP with 3 hidden layers of 150 neurons. The relation encoder takes position and velocity differences between objects at the receiver and sender end of the edges and is implemented with a MLP with 1 hidden layer of 100 neurons. While our relation propagator is a MLP with 2 hidden layers of 150 neurons, out object propagator is a MLP with one hidden layer of 100 neurons. During training, at each epoch, we validated our physics prediction module on the validation set containing instance from

the sparse configuration, and selected the model that has the highest accuracy (MSE) over trajectory roll-outs.

5 Results

In this section results of this application is explained in detailed. For every training used in the tests, the data with 10000 trajectories is used (even though the model had different amount of inputs) and the model is trained with 100 epochs.

5.1 Training Results

Training accuracies of the models for different number of objects are as follows;

7	**9**	**11**	**13**
83%	84%	83%	80%

Table 1: Training accuracy of the Construction Environment

7	**9**	**11**	**13**
91%	88%	88%	88%

Table 2: Training accuracy of the Jenga Environment

Top row numbers represents the number of objects when the model is trained.

5.2 Prediction Results

Binary prediction accuracy of the Construction Model is given at Table 3 and the binary prediction accuracy of the Jenga Model is given at Table 4.

Predictions are made after the drop of the object (at the Construction Model) or after the removal of the object (at the Jenga Model).

	7		**9**		**11**	
7	66.5	10.75	70.1	11.6	70.0	0.0
	21.5	1.25	17.4	0.9	30.0	0.0
9	20.0	40.0	60	2.6	57.5	8.3
	30.0	10.0	37.3	0.1	25.0	9.2
11	57.7	3.41	64.0	0.0	59.3	8.7
	37.4	1.41	36.0	0.0	29.3	2.6

Table 3: Prediction accuracy of the Construction Environment

Table 4 gives the true positive, true negative, false positive and false negative (from left most top value to right most bottom value) prediction percentages when data is trained with number of top row objects and tested with leftmost column objects. (e.g. When the model is trained with 7 objects and tested with 9 objects 48.25% of true positive prediction is made)

	7		**9**		**11**	
7	90	3.49	68	15.6	71.5	13
	5.8	0.16	13	3.4	14	1.5
9	48.25	27.6	88	0	60.5	20.7
	11.1	13	12	0	8.8	9.8
11	44.5	31.5	60.5	20.7	85	0
	7.1	16.9	8.8	1.2.6.	15	0

Table 4: Prediction accuracy of the Jenga Environment

5.3 Actions Taken

After receiving the predictions these predictions are used (in the case of the Construction Model) to drop an object to demolish the tower and/or (in the case of the Jenga Model) to remove an object to demolish the tower.

For the **Construction Model**: 100 different positions are randomly produced to drop to the existing tower and each objects' predicted stability is summed up and saved, with the produced positions taken as the position of the object to drop. The position with the least predicted stability of the system is chosen and the object is dropped to that position.

	7	**9**	**11**
7	20.3	26.0	20.6
9	34.2	23.12	25.8
11	28.3	30.6	32.7

Table 5: Demolish action accuracy of the Construction Environment

For the **Jenga Model**: Each of the object is chosen as the object to remove and the positions of the objects that are not chosen as the remove object are used to predict the stability of the system and the outcome stability is saved. The object that causes the minimum stability is finally chosen and removed from the system.

6 Conclusion

In this paper we represented a different approach to stability prediction of towers which is the us-

	7	**9**	**11**
7	90	15.6	71.5
9	48.25	27.6	88
11	44.5	31.5	60.5

Table 6: Demolish action accuracy of the Jenga Environment

age of graph neural networks. We introduced the propagation network that we use, how propagation networks are structured to construct a graph neural network and we showed the features of our graph neural network.

We also used the predictions that we received to drop or remove an object to demolish the tower. The same approach can be used to construct a tower that is hard and/or easy to demolish.

In the future we aim to work on 3D simulated/real environment and with differently shaped objects. Right now the nodes of the graph neural network does not have separate physical attributes but with differently shaped objects that would change which would complicate the learning.

References

P. Battaglia, R.Pascanu, M. Lai, and D. J. Rezende. 2016. Interaction networks for learning about objects, relations and physics.

P. W. Battaglia, J. B. Hamrick, , and J. B. Tenenbaum. 2013. *Simulation as an engine of physical scene understanding*, volume 110.

M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. 2017. A compositional object-based approach to learning physical dynamicsıky.

M. Deisenroth and C. E. Rasmussen. 2011. *Pilco: A model-based and dataefficient approach to policy search*.

C. Finn, I. Goodfellow, and S. Levine. 2016. *Unsupervised learning for physical interaction through video prediction*.

A. Lerer, S. Gross, and R. Fergus. 2016. Learning physical intuition of block towers by example. *International Conference on Machine Learning*.

Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. 2019a. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *International Conference on Learning Representations*.

Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake. 2019b. Propagation networks for model-based control under partial observation. *International Conference on Robotics and Automation*.

R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi. 2016a. *Newtonian scene understanding: Unfolding the dynamics of objects in static images*.

R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi. 2016b. *what happens if... learning to predict the effect of forces in images*.

D. Mrowca, C. Zhuang, E. Wang, N. Haber, L. F. Fei-Fei, J. Tenenbaum, and D. L. Yamins. 2018. *Flexible neural representation for physics prediction*.

S. van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber. 2018. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *International Conference on Learning Representations*.

N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. 2017. *Visual interaction networks: Learning a physics simulator from video*.

J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. 2015. *Galileo: Perceiving physical object properties by integrating a physics engine with deep learning*.