Physics Engines for Model Based Reinforcement Learning Literature Survey

Rohan Saphal

June 14, 2019

Abstract

Physics engines refer to a framework capable of approximately simulating interactions among elements in an environment under the constraint of Newtonian mechanics.Physics engines have garnered interest in recent years in fields of robotics, games, fluid dynamics, physical scene understanding and reasoning etc. With the recent advancement in deep learning, more robust and accurate models have been built for a large range of applications and domains. This survey reviews the various approaches and algorithms used in physics engines and a brief introduction to how it could help improve performance of reinforcement learning algorithms.

1 Introduction

Physics engines (PE) started out originally as an area in cognitive psychology [H+78] aiming to understand how the human mind has mental simulation models of real world physics. Studies were done to know how these mental models help humans to make fast and relatively accurate predictions in real world scenarios. In comparison to machine learning techniques that directly learns from demonstration and has a black box model that is not interpretable, the physics engine models perform simulation from a given input and roll out multiple possibilities, one of which is the ground truth. The use of such models helps us to understand better the trajectories that resulted in the final outcome. This helps in making better decisions at each step of the way. Work in cognitive psychology has shown how humans have priors that are learned from experience and through mental simulation models. Humans leverage these priors to make fast sequential decisions. This is held in comparison to deep reinforcement learning algorithms[MKS+13] that learns the environment absolutely from scratch through very long training periods because it has no prior information of the environment. The bottleneck in the model free reinforcement learning setting is that it requires a large number of training samples, transferability is poor, fails to generalize well and when considering the domain of autonomous driving, safety is not fully guaranteed. Physics engines[GTH98] on the other hand can act as an approximate simulation of the real world and can predict future states in the environment. This can help reinforcement algorithms learn faster and learn from state spaces that it would never have observed through demonstration.

The survey consists of brief summaries of the recent work in the field of physics engines and its applicability for model based reinforcement learning

2 Recent works

2.1 Investing human priors for playing video games [RD18]

The paper aims to study the different inbuilt priors present in humans and the effects of these priors that enable them to generalize and solve tasks faster. This is held in contrast to Deep Reinforcement learning algorithms that have to learn from scratch.The paper concludes that a more structured representation using these priors can help the reinforcement learning community to build efficient and possibly human like agents.

The authors have chosen different game domains to evaluate and compare the performance of human players and deep reinforcement learning algorithms. There are two bottlenecks in this field, first being , the field is still at a very primitive stage and it lacks understanding about the various priors that guide human behavior. Secondly, it is not clear as to how such prior knowledge can be incorporated or how this knowledge can be learned.

The experimental setup involved masking and re-rendering various entities in the game environment such as ladders, enemies, keys and platforms, etc. These were intended to mask prior knowledge. The evaluation measures used at each trial was the time taken to complete the game and the number of deaths before successfully completing the game.

Figure 1: A simple platformer game.

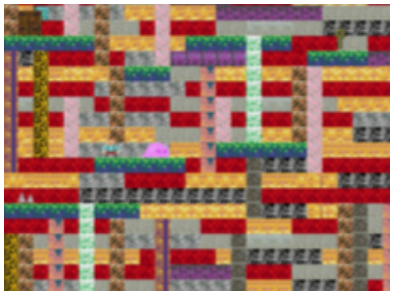

Figure 2: Modified version of the game by re-rendering textures.

The priors tested were:

- Semantics/Semantic information:Although it is not clear how semantics help in advancing in a game, one hypothesis suggests that the presence of semantics allows humans to infer the latent reward structure.This is confirmed in a series of experiments.

- Objects as subgoals for exploration:Studies done on infants show that there is a more general prior knowledge available and that is in the existence of objects and that objects are interesting entities to be explored. Masking of objects is shown to have increased exploration of the environment and longer time to reach the goal.

- Affordance:The knowledge about what to do with certain entities is referred to as affordance of that entity. Instead of having big differences between objects that distinguish it from each other, every category of objects is given a random texture that make it visually distinctive but its difficult to understand the use for that entity.

- Things that look similar behave similar: the notion of similarity was explored as another important prior that was seen to have been used by humans in gameplay. This is the second most important prior after exploration of objects.

- Interaction with objects: Given the various objects in the environment, how does one understand what action should be taken. Reinforcement learning algorithms do not have an intuition about what actions to choose for which object unlike humans. Humans understand that upon seeing a ladder, pressing up leads to climbing up the ladder. This prior understanding of what action has to be taken for an object is explored here.

Another important aspect explored in the paper is when all the physics and motor control was reversed. Human players found it extremely hard to solve the problem.This shows us why it is so difficult for reinforcement learning algorithms in general to solve such problems.

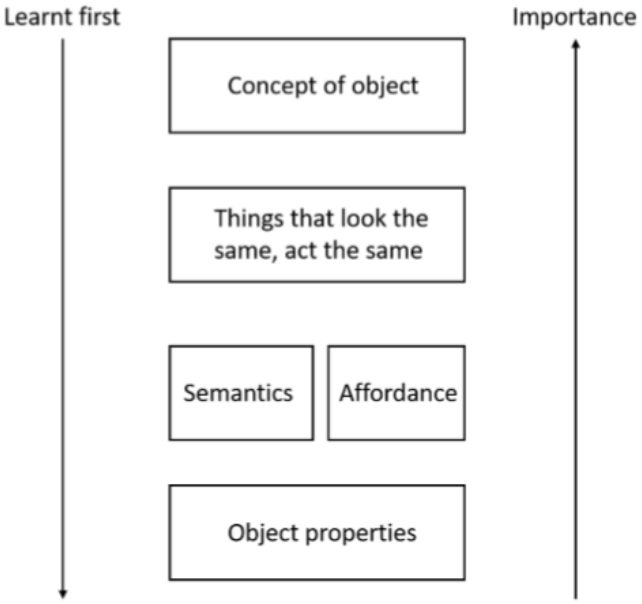

Figure 3: Taxonomy of object priors.

Understanding the effects of various priors and their effects on game-play indicate that the prior that is learned or incorporated at the earliest stages of development contribute more during game play

The paper concludes by stating the importance of such priors for gameplay and how it could help in generalizing across domains. A few cons to having priors for the reinforcement learning setup is that it could lead to suboptimal rewards.

2.2 Simulation as an engine of physical scene understanding [BHT13]

This paper is one of the fundamental works in physics engines and has laid the foundation to the work that has happened recently. This paper proposes a framework to model a physics engine that uses probabilistic simulation to make fast and robust inferences in complex scenes where crucial information about the environment is unobserved.

The work is based on the idea that people's physical intuition when grounded in dynamic perceptual and action contexts, is very much comparable and accurate to Newtonian standards. These intuitions can be modelled as rational probabilistic inferences in a noisy newtonian framework. The goal of the work is to develop and test a computational framework that can be used for everyday scene understanding and reason about a large number of objects interacting in complex ways in an environment that is incompletely observed and make short term approximate predictions. Previous work in relation to physics engines have made progress on building cognition models that help us in understanding high level symbolic reasoning and problem solving but does not explain the quantitative aspects and uncertainties related to objects geometry, its interactions and motions. The Intuitive physics engine(IPE) follows an object based representation of a 3D scene similar to what is seen in CAD programs and also takes into account the physical forces governing a scene's dynamics by looking at how the state of the object changes over time.

The three key design elements related to IPE are:

- IPE is based on simulation and not on symbolic equations that give analytic solutions
- IPE is probabilistic in nature and therefore is robust to uncertainty of the environment.
- IPE is approximate and generates predictions that are accurate enough to plan or understand everyday activities. Thus its a compromise on precision for speed and generality.

Human intuition in general can answer the question, 'what will happen' in scenarios it has not encountered before. For example, in the case of stacked dishes, we can infer which side it might fall to or when a new dish is placed, whether it will fall or not, or if the stacked dishes are placed to the edge of the table will it fall if kids are running around or not and so n and so forth. Interestingly, many of these situations are not encountered before but experienced firsthand and yet humans are able to make reasonably accurate predictions.

Experiments:

The experimental setup consisted of a tower of 10 blocks. The experiment was intended to test whether the IPE model could explain the richness of peoples intuition across a diverse range of tasks. The model takes in visual inputs and has linguistic outputs. However, the model is flexible and can take a variety of different inputs and outputs. The input to the model is a sample from a distribution over scene configurations, object properties and forces based on ground truth. These are modulated by parameters for each input category. Given the input parameters and scene configuration, the IPE model performs simulation over physical dynamics to produce a sample of possible final scene configurations. Depending on the output, an average of the final scene configuration is taken to report whether the model was successfully able to predict. For example, for the question, 'will it fall ?', an average proportion of blocks that fell across the simulation runs was taken as the output.

Results:

Based on the experiments conducted, the following were the results:

- Feedback driven learning played at most a minimal role in improving prediction
- Hypothesis that people instead of explicitly using physical dynamics to make predictions, were using a learned combination of geometric features (number of blocks, height, position) from initial scene configurations was proven to be less viable and inaccurate
- It is seen that the variation in latent physical property such as mass, friction etc does not affect the predictions and the model is able to make allowances for such variations. This goes onto say that the IPE model can go onto exploit richer aspects of a scene representation and is generalizable to similar domains.
- It is also shown that when the environment is very complex, simulation time would be very large and people tend to fall back on simulation based heuristics

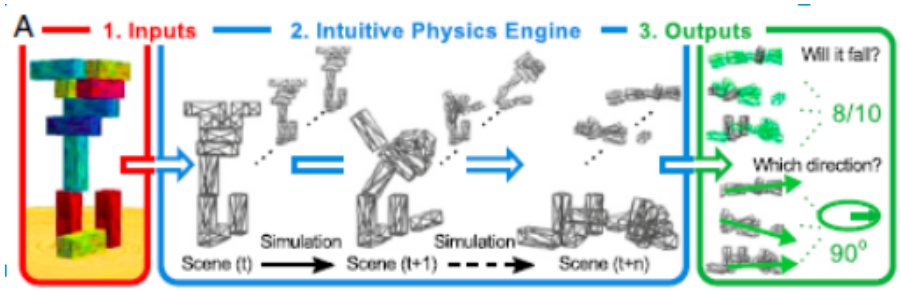

Figure 4: IPE model.

Concluding the discussion, the following points are highlighted:

- The work presented here differs from previous work as it focuses primarily on simulation based results
- Physical scene understanding in humans occurs primarily through model based simulation as it requires humans to handle a wide range of scenes and also learn from a finite amount of experience.
- Model based reasoning is more flexible, general purpose and does not require task specific learning
- Probabilistic simulation offers a way to integrate symbolic reasoning and statistical inference, two competing approaches used for formalizing common sense thought

2.3 Learning visual predictive models of physics for playing billiards[FALM15]

The paper aims to explore the idea of having an internal model of the dynamics of the environment and how this model can be used to plan actions in the future by carrying out simulations.

The internal dynamic model will allow the agent to conduct virtual simulations and help it to decide the optimal action that should be taken at that time step. By learning the internal dynamic model, both the dynamics of the environment and the dynamics of the agent has to be built. This is because, in many cases the environment might change constantly in comparison to the agent and mapping dynamics of both entities helps to generalize better.

The reason existing methods that directly learns policies from visual input don't work is that they fail to model how visual observations evolve in response to an agent's action at a time-step. This makes it difficult to transfer to new domains. Another series of related work [ANA+16] aims to learns physical dynamics directly from experiments and [LGF16] is closely related to what is done here.

Figure 2: **Network architecture.** At each time step t, for each object, the network is provided with the previous four glimpses centered on the object's position, as well as the agent's applied forces $\mathbf{F}_t = (F_t^x, F_t^y)$ and the hidden states of the LSTM units from the previous time step. The output is ball displacements $\mathbf{u}_{t+k} = (\Delta x_{t+k}, \Delta y_{t+k})$ for $k = 1 \cdots h$ in the next h frames.

Figure 5: Billiards model.

The framework consists of visual inputs and forces acting on the billiard balls and the architecture predicts the future velocity of the respective ball. The force magnitude was sampled from a range of 30k newton to 80k newton and the force direction was uniformly sampled. A pre-generated sequence of frames was used as training dataset. Initially the model was trained on a single ball environment. This trained model was used to initialize training of the two ball environment. This in turn was used to initialize the training of a 3 ball environment. Interesting to note, this curriculum based methodology seems to perform better in comparison to when it is trained from scratch.

The proposed framework is object centric model and it is compared with other baselines such as constant velocity and frame centric models. The object centric model seems to perform much better compared to the other two. It also generalizes to domains substantially different in comparison to the original one it was trained on.

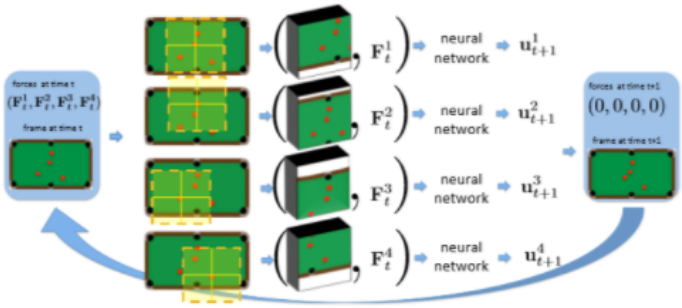

Figure 5: Generating visual imaginations: The visual image of the world at current and past 3 time steps and the applied forces are used to predict the velocity of every ball. The visual image of the world at the next time step is rendered by translating the balls by their predicted velocities. This process is repeated iteratively to generate a sequence of visuals of the future world states.

Figure 6: Billiards model visual imaginations.

One interesting work done by the paper includes planning using these predictive models. The planning is done for basically two tasks, firstly to push a ball to a desired location and secondly to make one ball hit another moving ball. Given the target state, multiple virtual simulations are done to find the optimal force that would result in the desired state or a state that is as close to the desired state. In practice, instead of exhaustively searching for all the optimal forces, the CMA-ES method is used to determine the optimal force.

To conclude, this paper presents an object centric prediction approach that exploits translation invariance in dynamics of physical systems to learn dynamical model of the world.

2.4 Galileo:Perceiving physical object properties by Integrating a physics engine with Deep learning [WYL+15]

The paper proposes a generative model to predict physical events in dynamic scenes and to infer the physical properties of objects from static images in real world scenarios. At the core of it, there is a 3D physics engine which operates on object based representation of physical properties including mass, friction, position and 3D shape. Using Markov chain monte carlo simulation, simulations are run in the physics engine to fit key features of visual observation.

The paper borrows a lot from the idea of intuitive physics engines, which considers that humans run noisy probabilistic simulations of a mental physics engine. However, previous research fails to convey how a mental physics engine takes advantage of previous experience of the agent.

The framework is called Galileo and has 3 parts:

- Physical object based representations: by categorizing the environment into objects, we represent each object by its 3D shape, position and also by its latent properties such as mass and friction. These attributes are estimated from the visual input.
- Fully fledged realistic physics engine: In this paper specifically, the Bullet physics engine is used. The engine which is a third party software, takes in the specification of each object in the scene and simulates it forward in time , generating simulated positions and velocity profiles for each object
- Galileo: The previous two parts are not novel in any nature and have been part of previous research.Galileo, however, adds novelty to the work as it is a likelihood model that compares the results from the simulation to the real world data. A standard tracking algorithm is used to get velocity profiles of the object from real word videos. Galileo does probabilistic inference to recover the underlying physical object properties in the scene

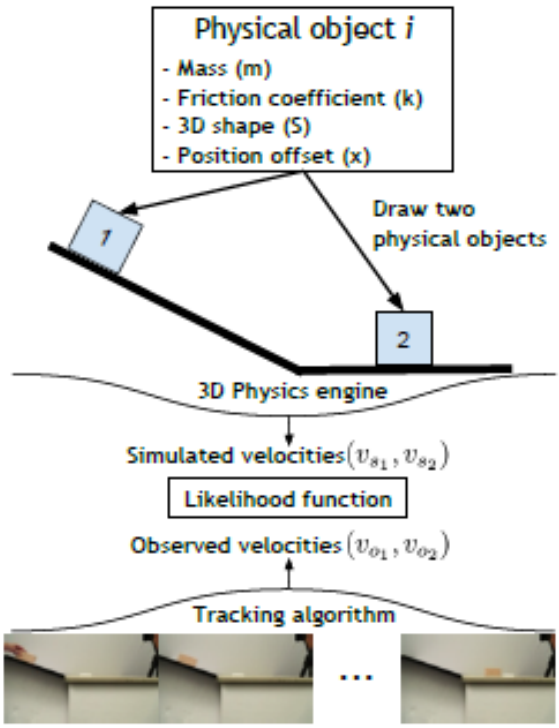

Figure 7: Galileio.

The paper demonstrate the algorithm on a custom video dataset of objects with different physical properties slide down a ramp. Passing this dataset through the framework results in physical values such as velocity and

position which help to describe the scenario. The underlying physical properties such as material, density, mass and friction coefficient are also observed. The focus of the framework is to extract the latent unobserved physical properties of objects through probabilistic inference. The ramp scene is given as an input to the bullet physics engine and it simulates it forward in time, generating simulated velocity values for the objects in the scene.

$$P(T_1, T_2 | v_{o_1}, v_{o_2}, \rho(\cdot)) \propto P(v_{o_1}, v_{o_2} | v_{s_1}, v_{s_2}) \cdot P(v_{s_1}, v_{s_2} | T_1, T_2, \rho(\cdot)) \cdot P(T_1, T_2).$$

Figure 8: Bayesian formulation

Here T represents the latent physical properties of the object in the scene.V_o is the observed values of the velocity of the object from real data obtained using an object tracking algorithm. V_s is the simulated values of the velocity of the object from the physics engine. To alleviate some burden from the inference model, some of the latent variables are obtained from the recognition model such as the shape of the object and its relative distance from a fixed origin.

Once the values have been estimated through probabilistic inference, the video data and the inferred data is used for a supervised learning scenario. In this case, the input is the video feed and the output is the latent physical properties of the objects in the scene. This is done by using LeNet.

Experiments compare the performance of Galileo with humans in a number of tasks such as destination prediction after collision, mass prediction and 'will it move' prediction. It is observed that Galileo performs at par with humans and sometimes even better. However, interestingly, humans and the model make similar mistakes.

The paper concludes on the note that generative physical knowledge along with various recognition pipelines can help in efficient inference of a static physical scene

2.5 A Comparative Evaluation of Approximate Probabilistic Simulation and Deep Neural Networks as Accounts of Human Physical Scene Understanding[ZWZ+16]

The paper is a comparison between simulation based models and conventional CNN based models for predicting physical outcomes. A set of experiments are conducted that compare the performance of the two methods and also compare it to how humans would predict its behaviour.

These simulation engines approximate object dynamics interacting under Newtonian mechanics over short times scales and is efficient enough to run in real time complex scenes.

The comparison between these two methods is taking place for the following reasons:

- Other authors have suggested that simulation based models might be expensive to run in the human mind and a possible alternative to that could be using memory. Using stored experiences and their outcomes along with deep learning architectures could allow the brain to access the correct memory to make the correct prediction in a new scene.

- Researchers at Facebook tried the same with CNNs to make predictions from direct visual scenes and the network is known as PhysNet.It showed good accuracy(89) and generalizability(67).

- Although CNN's gave good results on the prediction task and are similar to models of the visual cortex, they also have features that are less human like. They require huge amounts of data to train which a human does not do. On the other hand IPE model is able to generalize well to scenes it has never encountered before.

One interesting fact to note about physics models and its similarity with humans is the asymmetric pattern of errors it makes. An unstable structure will be predicted to fall by both humans and the IPE model but if the structure is carefully balanced but looks as though it might fall, the IPE model predicts it to fall. This emphasizes that only a small amount of uncertainty is required to make a prediction go from 'stable' to 'unstable'.

Four experiments are conducted are as follows:

- Exp 1: Evaluates performance of the IPE model in comparison to neural network models in predicting stability of a block.
  - Results :
    - Accuracy decreases as noise increases for the IPE model.

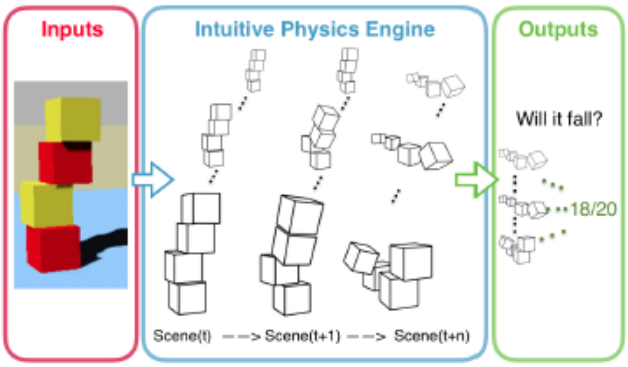

Figure 9: IPE model.

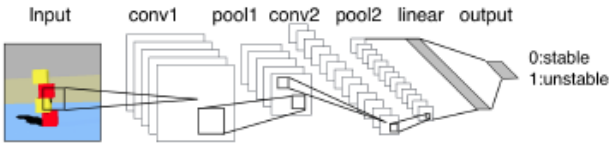

Figure 10: LeNet model.

  - ∗ Human predictions and the IPE model have asymmetric pattern while this is not seen with neural networks.

- Exp 2: Evaluates performance of CNN with low amount of training data to see if it behaves more human like.
  - Results :
    - ∗ IPE model is shown to have stronger correlation with humans compared to LeNet. LeNet however shows human like predictions when trained on limited data.

- Exp 3:Evaluates the model for asymmetries in the stability as mentioned before.
  - Results :
    - ∗ Both the IPE model and human predictions are highly correlated in this case and predicts that blocks will fall in cases of visual instability.The neural network on the other hand is much more accurate and robust to visual instability cases.

- Exp 4:Tests the generalizability of IPE and CNN model to situations slightly different from what the CNN was trained on.
  - Results :
    - ∗ The performance of the IPE model and humans decreases slightly with increase in the number of blocks and also in the complexity of the scene.
    - ∗ It is concluded that knowledge gained by neural networks cannot be transferred in a straightforward manner.

To finally conclude, CNN's do not capture a good model of people's physical intuition. IPE model however is quite consistent with the predictions that human make. This comparison helps to show something fundamental about human cognition and that is, existence of a mental model of the worlds processes. One possible combination of both models mentioned is the case where, a CNN model can be trained to output the physical properties necessary for conducting simulations in the IPE model. This would allow more sophisticated simulation, reasoning and prediction by the IPE.

2.6 Learning to See Physics via Visual De-animation[WLK+17]

The paper introduces a method for understanding physical scenes. It involves converting the visual input consisting of images and extracting from it the physical attributes related to the particular scene. These inputs are then fed into a graphics and physics engine to simulate what will happen next. The outputs are then inverted through a graphical engine to understand it through a visual medium. Scene understanding consists of two main features that define it especially for humans. First, not only do humans have the ability to interpret the scene but also predict as to what might happen from visualizing the given scene. Secondly, this happens very fast in terms of computation.

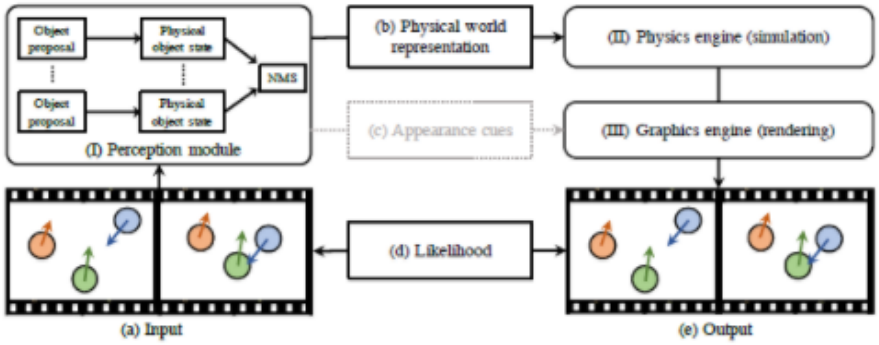

Figure 11: Visual De-Animation framework.

A deep learning model is used to compute the physical features of the objects in the scene from the visual input. These deep learning models are trained separately to output the physical states of the objects in the scene and are not part of the end to end training scene. These features obtained are then passed onto the physics engine that perform simulations to obtain comprehensible scene representation. A graphics engine is also put in place to obtain interpretable representation of the scene once the simulation through the physics engine is completed.

The paper also compares the two most dominant approaches to physical scene understanding. The first one emphasizes on learning from pixels. The other emphasizes on learning a flexible model of the dynamics of object interactions by decomposing the scene into objects and their physical properties. They claim to combine both the approaches in their work by having a visual pipeline and an object based simulation physics engine.

Although the paper claims that there is no need for a supervised learning setup, both the visual pipeline and the physics engine is pre-trained and works well only if it is pre-trained. The pipeline is then fine tuned in the end to end training setting.

The model is tested on two datasets, the Billiard Tables and the Block world. The billiard table dataset is used to emphasize the models applicability to videos while the block world is for static images.

The authors show generalization of the algorithm by testing it out on the same environments but with different number of objects in the scene.

2.7 A compositional object-based approach to learning physical dynamics [CUTT16]

The paper proposes the factorization of a physical scene into object based representations and also a Neural network architecture that factorizes the object dynamics into pairwise interactions. By introducing a physical reasoning system for the environment, we are basically providing the algorithm with priors about the environment , which it can leverage to get a better understanding of the surroundings. The paper explores the idea of building priors into the program for the agent to understand.

The objective of the paper is to have a generalized framework that can be compatible for variable number of objects and also for different scene configurations and which can be transferred to other tasks.

Humans naturally have a prior of physics that they leverage. This paper explores this aspect of building this prior into a program. This can be viewed as a simulator that takes input as physical scenes and past state of objects and outputs future states and physical properties of relevant objects

There are two approaches for designing such an engine. The first one is a top down approach that formulates the problem as an inference from a proper physics engine. The bottom up approach involves directly learning from physical scenes to the physical outcomes. However the bottom up approach does not generalize well and requires training from scratch. The top-down approach is much more generalizable but it cannot perform for

environments that is not part of the simulation engine. These environments will have to be created for the top-down approach to work.

The bottom up approach is also not preferred for two reasons. One, it couples the vision and dynamics of the physical scene which leads to lesser generalizability. It seems necessary to separate the vision and the dynamics part so that scenes having different visual appearance but having the same dynamics can be taken into consideration. Two, it is indeed possible to do what is mentioned above. The visual scene can be mapped to a intermediate latent space and then the dynamics model can infer from that and predict future states.

The proposed framework takes a step towards bridging a gap between the above two frameworks by having the symbolic structure of a simulator along with the gradient based learning.By symbolic structure, it captures the object properties and the object-object interaction. By using gradient based learning, we can focus on the symbolic aspects and train the model to capture what happens at the ground truth.

The framework works on a few ideas as follows:

- Object based representation. All entities in the environment is represented by objects and object based representation is the primary idea.
- Given an object of focus, how do we select the other important objects that has potential chance of interacting with the main object. Devising heuristics for the same leads to better understanding of how objects interact.
- Factorization: which grounds the idea of interaction in the environment as the interaction of pairwise objects.Thus all the environment dynamics is a result of pairwise interaction.
- Compositionality: the future state of an object is a function composition of the pairwise interactions between that object and other objects around it.

Approach:

Object based representation: This kind of representation consist of spatially local computation (predicting the future location of the object) and temporally local computation (physics is Markovian and hence we need only predict for the next time step). These two ideas lead to having the environment being represented by objects. A state representation for the object consists of its extrinsic properties, intrinsic properties and global properties

Pairwise factorization: The framework selects a particular object as the object of focus and all other objects as context objects that tend to interact with the focus object. The framework models the velocity of the focus object as a composition of the pairwise interaction with the context objects. The pairwise interaction is encoded and then concatenated with the focus objects past state. This is fed as input to the the decoder which predicts the velocity of the focus object.

Context selection: this part of the framework decides the condition for two objects to be considered as a pair that could be interacting with each other. This is taken care of by a masking function called as broad phase. The part of the framework actually looks into the collision of the objects and basically detects collisions is called narrow phase. The algorithm uses broad phase for the selection process but uses NPE for the collision detection process.

Function composition: The output of the encoder can be interpreted as the effect of the context objects on the focus object. These forces are also additive in nature.

Baselines

No-Pairwise :This is similar to the NPE but it does not take into consideration pairwise interaction. It also has the similar architecture to that of NPE, i.e having an encoder and decoder. One possible mechanism to predict the future velocity of the focus object would be to have an abstract representation for all objects and a force field of that object. This is passed to the encoder. The decoder can then sum up all the force fields on the abstract representation of the focus object to predict the velocity of the focus object.

LSTM: previous work has shown that LSTm tends to sequentially attend to objects. This idea is tested to know whether LSTm could model object interactions. The past states of the context objects is passed to the LSTM and at the last step takes in the focus objects state and predicts the velocity.

Experiments:

Neighbor hood mask, the parameter that is used to determine whether a context object is detected or not is a hyper parameter that needs to be tuned for different domains and object geometries. It is shown that, it is more important to understand the effects of focus object with each context to determine an intrinsic property than without taking into consideration these interaction.

One of the key future works could be to use NPE in agents in model based RL or planning. This way, NPE will act as a prior on the environment and guide learning and reasoning.

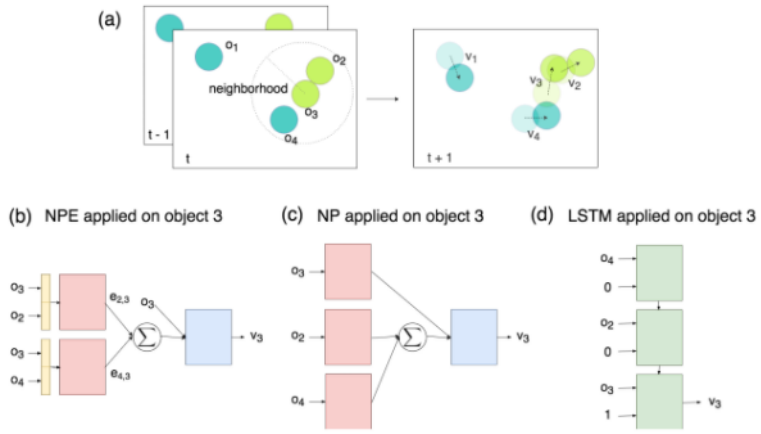

Figure 12: Neural Physics engine framework and baseline frameworks

2.8 Interaction networks for learning about objects, relations and physics[BPL+16]

It is a framework that tries to perform an analogous form of reasoning about objects and relations in complex systems. The paper claims to be the first general purpose learnable physics engine and a general framework for reasoning about objects and relations in a wide variety of complex domains.

In comparison to model free learning methods or simply end to end learning models, simulation is an effective method for approximating dynamical systems and predicting how the dynamics of the system is influenced by the mutual interaction between elements.

The framework exploits three approaches:

- Structured representation: exploiting the knowledge of relations between objects
- Simulation:predicting how the elements of the system will interact with each other and approximating the dynamics of the system
- Deep learning: it allows scalable learning in challenging world settings.

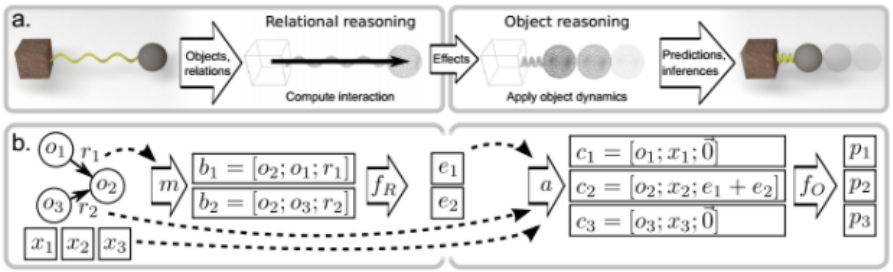

Figure 1: Schematic of an interaction network. *a.* For physical reasoning, the model takes objects and relations as input, reasons about their interactions, and applies the effects and physical dynamics to predict new states. *b.* For more complex systems, the model takes as input a graph that represents a system of objects, o_j, and relations, $\langle i, j, r_k \rangle_k$, instantiates the pairwise interaction terms, b_k, and computes their effects, e_k, via a relational model, $f_R(\cdot)$. The e_k are then aggregated and combined with the o_j and external effects, x_j, to generate input (as c_j), for an object model, $f_O(\cdot)$, which predicts how the interactions and dynamics influence the objects, p.

Figure 13: Interaction network schematic

One notable part of their framework is that, the relations between objects have to be explicitly given as input. This reduces the complexity of the problem as the network doesn't have to consider all possible interactions between objects but instead the potential ones specified as input.

The model works on an object based representation and the relations between these objects/interaction between these objects represented in a graphical form. There is an object centric function Fo that takes as input the state of the object at time t along with any interaction it has with any other object ,et+1, and outputs the object state at t+1. The interaction et+1 is modeled through a function Fr, that takes as input the states of the objects and the basis of their interaction/relation.

$$\text{IN}(G) = \phi_O(a(G,\ X,\ \phi_R(\ m(G)\ )\ ))$$

$$\begin{aligned} m(G) &= B = \{b_k\}_{k=1\ldots N_R} & a(G,X,E) &= C = \{c_j\}_{j=1\ldots N_O} \\ f_R(b_k) &= e_k & f_O(c_j) &= p_j \\ \phi_R(B) &= E = \{e_k\}_{k=1\ldots N_R} & \phi_O(C) &= P = \{p_j\}_{j=1\ldots N_O} \end{aligned}$$

Figure 14: Mathematical framework of IPE

The function m(G) basically orders all the relations and objects into a tuple of two objects and a relation between them. This tuple, b_k , is then passed through f_r to obtain the approximate effect of this interaction through the entity, e_k. All the effects that a particular object is subjected to is the aggregated along with the external stimuli,X, and the objects original state, O to produce model inputs C. This is then fed into the object centric function to obtain the final output state of the object.

The architecture mainly involves the use of matrices and MLP. MLP is used to learn the mapping for the functions ϕ_R and ϕ_O. The functions m and a are basically matrix operations.

In comparison to analytic physics models that are not differentiable, the interaction network serves as a powerful tool that allows gradient based planning. Another important point to note is that, having a RNN as part of the model did not improve the performance to a very great extent. However, previous work on physical scene understanding[FALM15], the use of LSTM showed improved performance.

2.9 Visual Interaction Networks: Learning a Physics Simulator from Video[WTW+17]

This paper is a follow-up on its previous work called Interaction networks. On a fundamental level, the paper proposes a framework that learns the dynamics of physical systems directly from raw visual inputs.

This work opens up possibilities for model based decision making and planning from raw sensory observations in complex environments. The framework is a general purpose model predicting future states from video data. It can be trained from supervised data sequences consisting of input image frames and target object state values.

The model consists of two main components: a visual encoder based on CNN and a recurrent neural network with an interaction network at its core for making physical predictions. The approach followed by this framework for physical reasoning is to train models to make pixel to state predictions unlike the other branches that learned to predict summary physical judgments and produce simple action from images or make state to state predictions.

Model: It consists of a visual encoder which takes as input 3 image frames from the video and outputs a state vector. These state vectors are a representation of the position and velocity of objects in the scene. The 3 input frames are paired up in a consecutive fashion and a constant x,y channel is also introduced to help incorporate position. These state vectors are fed into a predictor framework with Interaction network at its core. It predicts the future states of the object in the environment in the same format as the encoder, in the form of state vectors. These state vectors obtained are then fed into a state decoder that transforms the state vector to the object's position and velocity state.

The interaction network used here in the dynamic predictor framework is slightly different from the vanilla IN network. The difference being that the new IN aggregates over multiple temporal offsets.

The framework was experimented on the following physical systems:

- Objects connected by a spring
- Large objects that obey Newton's law of gravity
- Billiards
- Magnetic billiards : instead of bouncing off each other, they repel off each other as all balls are positively charged.
- Objects drifting with initial velocities

The model is seen to outperform all baselines and also accurately predict the future states of the objects in the scene. It even outperforms the state of the art, state to state predictor models. One of the reasons for the better performance is the noisy input to the dynamic predictor from the visual encoder which helps it to

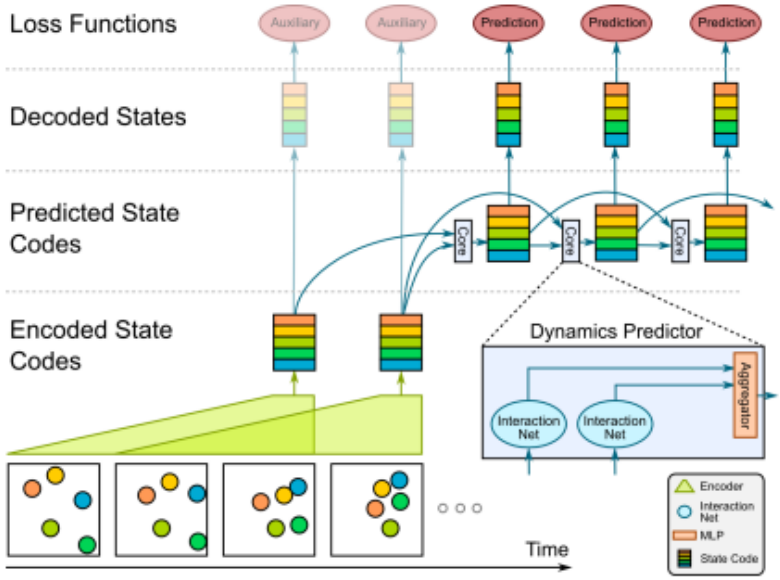

Figure 1: **Visual Interaction Network:** The general architecture is depicted here (see legend on the bottom right). The visual encoder takes triplets of consecutive frames and produces a state code for the third frame in each triplet. The visual encoder is applied in a sliding window over the input sequence to produce a sequence of state codes. Auxiliary losses applied to the decoded output of the encoder help in training. The state code sequence is then fed into the dynamics predictor which has several Interaction Net cores (2 in this example) working on different temporal offsets. The outputs of these Interaction Nets are then fed into an aggregator to produce the prediction for the next time step. The core is applied in a sliding window manner as depicted in the figure. The predicted state codes are linearly decoded and are used in the prediction loss when training.

Figure 15: Visual Interaction network framework

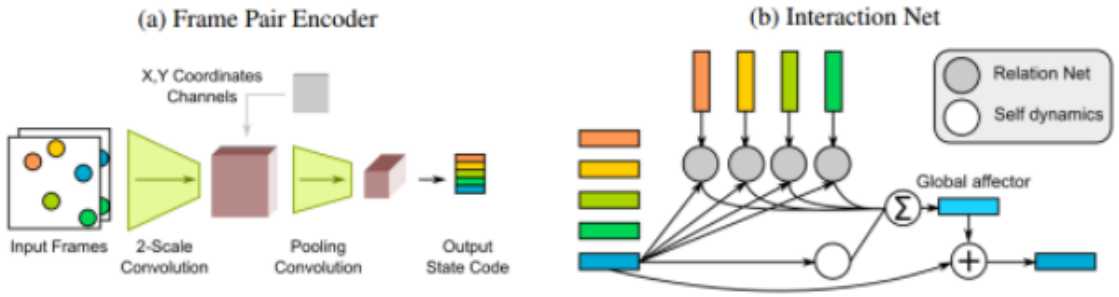

Figure 2: **Frame Pair Encoder and Interaction Net. (a)** The frame pair encoder is a CNN which transforms two consecutive input frame into a state code. Important features are the concatenation of coordinate channels before pooling to unit width and height. The pooled output is reshaped into a state code. **(b)** An Interaction Net (IN) is used for each temporal offset by the dynamics predictor. For each slot, a relation net is applied to the slot's concatenation with each other slot. A self-dynamics net is applied to the slot itself. Both of these results are summed and post-processed by the affector to produce the predicted slot.

Figure 16: Encoder and Dynamic predictor frameworks

generalize better. The reason being, the model learns to overcome temporally compounding errors generated by inaccurate predictions.

A lot of the future work is aimed at improving the generalization of the framework by improving the encoder framework and modifying the loss function to be order-agnostic loss function

2.10 Relational Neural Expectation Maximization: Unsupervised Discovery Of Objects and their Interactions

The paper is trying to address a fundamental problem in learning object representations from visual images, i.e discovery of objects and their relations. More importantly, the discovery is done in an unsupervised fashion. The motivation, similar to previous works, stems from the notion that humans perceive and predict physical scenes by decomposing the environment int objects and their relations. This notion is tied strongly with the idea of common sense physical reasoning.

At its core, there is the Neural expectation maximization(NEM) module that discovers object representation from raw visual images but fails to realize the relations among different objects. To handle this, the authors use the Relational net (R) module to augment the performance of NEM.

Learning representations using conventional methods such as VAEs or using GANs leads to the input being superimposed into the representation without any regard for the structure of the environment.NEM on the other had learns seperate distributed representations for all the objects in the environment through an iterative process of perceptual grouping.

The goal of NEM is to group together pixels that belong to the same object and learn a representation θ_k for that object. The problem in this methodology is that the number of representations again have to be defined before. By using Expectation Maximization to compute a Maximum Likelihood Estimate (MLE) of the parameters of this mixture $\theta_1 ... \theta_k$, a grouping (clustering) of the pixels to each object (component) and their corresponding representation is obtained. However, the true distribution is unavailable and is represented by a neural network and learned through back-propagation. The distribution is learned by a latent variable, $Z_{i,k}$ that is assigned a value of 1 if the pixel was generated by parameters k. The distribution of images given the latent representations of the objects is then formulated as follows:

$$P(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{D} \sum_{\boldsymbol{z}_i} P(x_i, \boldsymbol{z}_i|\boldsymbol{\psi}_i) = \prod_{i=1}^{D} \sum_{k=1}^{K} P(z_{i,k}=1) P(x_i|\psi_{i,k}, z_{i,k}=1)$$

The object representations for a certain image can be found by maximizing $P(x|\theta)$.Marginalization over z complicates this process, thus generalized EM is used to maximize the following lowerbound instead:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{z}} P(\mathbf{z}|\boldsymbol{x}, \boldsymbol{\psi}^{\text{old}}) \log P(\boldsymbol{x}, \mathbf{z}|\boldsymbol{\psi})$$

Each iteration has two steps, the expectation process that calculates the posterior probability and the maximization step that updates the representations parameters for each object through back propagation

It is clearly understood that NEM can only discover objects but not discover the relations among the objects. A RNN variant of NEM was also used but it also cannot capture the inter-object dependence but can only capture the object dependence on its past information. In order to capture the inter object dependence, an interaction module is used that is similar to Interaction network.

$$\boldsymbol{\theta}_k^{(t)} = \text{RNN}(\tilde{\boldsymbol{x}}^{(t)}, \Upsilon_k(\boldsymbol{\theta}^{(t-1)})) := \sigma(\boldsymbol{W} \cdot \tilde{\boldsymbol{x}}^{(t)} + \boldsymbol{R} \cdot \Upsilon_k(\boldsymbol{\theta}^{(t-1)}))$$

$$\Upsilon_k^{\text{R-NEM}}(\boldsymbol{\theta}) = [\hat{\boldsymbol{\theta}}_k; \boldsymbol{E}_k] \;\text{ with }\; \hat{\boldsymbol{\theta}}_k = \text{MLP}^{enc}(\boldsymbol{\theta}_k) \;,\; \boldsymbol{E}_k = \sum_{i \neq k} \alpha_{k,i} \cdot \boldsymbol{e}_{k,i}$$

$$\alpha_{k,i} = \text{MLP}^{att}(\boldsymbol{\xi}_{k,i}) \;,\; \boldsymbol{e}_{k,i} = \text{MLP}^{eff}(\boldsymbol{\xi}_{k,i}) \;,\; \boldsymbol{\xi}_{k,i} = \text{MLP}^{emb}([\hat{\boldsymbol{\theta}}_k; \hat{\boldsymbol{\theta}}_i])$$

An interesting thing to note is that there is an attention layer that looks at which effects between objects to consider in the calculation. The attention actually acts like a pruning mechanism enabling you to define the relations between the objects with a little bit more clarity.

The paper falls short in scaling up with the number of objects in the environment. It is shown to fail and becomes unstable during training. This is mainly because of the number of agents in the environment and the large number of relations that are formed by pairing up objects. Also, the relation between objects may not be optimal and may change depending on the task and therefore it is important to have a dynamic graph instead of one that is learned in an unsupervised manner. One severe limitation of the paper is the predefined number of

object parameters that are considered. If a new and unseen object comes into the environment, the framework will fail to detect it or could fail catastrophically.

2.11 Neural Relational Inference for Interacting Systems

The paper aims to learn the implicit interactions between objects in the environment while simultaneously also learning the dynamical model of the interacting system. The neural relational inference (NRI) framework is built fundamentally on graph neural networks(GNN) and uses it as a latent graph structure for learning.

$$v \to e: \quad \mathbf{h}_{(i,j)}^{l} = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$$
$$e \to v: \quad \mathbf{h}_j^{l+1} = f_v^l([\textstyle\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$$

The first equation predicts the interaction between the nodes i and j of the graph using its node information and latent hidden representation. The second equation is used to update the value of node j leveraging the information from all the nodes its connected to by the edges.

The interaction and the dynamics model is learned in an unsupervised manner using a VAE framework. The unique features of this framework is that, instead of predicting only for one time step similar to the original VAE framework, the framework predicts for multiple time-steps. This is necessary since interactions have an effect over a long time period and cannot be effectively captured in a single time-step. Also, the latent distribution is discrete and hence a continuous relaxation is used in order to use the reparameterization trick

- Encoder Framework
  - The main use of the encoder framework is to infer the pairwise relations from observed trajectories.Since the graph structure is not known, we assume a fully connected graph with GNN.

$$\mathbf{h}_j^1 = f_{\text{emb}}(\mathbf{x}_j) \tag{5}$$
$$v \to e: \quad \mathbf{h}_{(i,j)}^1 = f_e^1([\mathbf{h}_i^1, \mathbf{h}_j^1]) \tag{6}$$
$$e \to v: \quad \mathbf{h}_j^2 = f_v^1(\textstyle\sum_{i \neq j} \mathbf{h}_{(i,j)}^1) \tag{7}$$
$$v \to e: \quad \mathbf{h}_{(i,j)}^2 = f_e^2([\mathbf{h}_i^2, \mathbf{h}_j^2]) \tag{8}$$

    The use of multiple passes, two in the above model, helps to disentangle multiple interactions while still using only binary terms. The embedding in equation 5, is dependent only on nodes i and j. However, in the second iteration , in equation 8, the embedding is dependent on the entire graph. The latent variable that the encoder learns has to be discrete and hence back-propagation using the reparameterization trick does not work. The solution adopted is to use a sample from a continuous approximation of the discrete distribution and use the repramatrization trick to get (biased) gradients from this approximation

$$\mathbf{z}_{ij} = \text{softmax}((\mathbf{h}_{(i,j)}^2 + \mathbf{g})/\tau)$$

- Decoder Framework
  - The main use of the decoder framework is to predict the future dynamics of the system conditioned on the graph z that has been obtained from the encoder.

$$v \to e: \quad \tilde{\mathbf{h}}_{(i,j)}^t = \sum_k z_{ij,k} \tilde{f}_e^k([\mathbf{x}_i^t, \mathbf{x}_j^t]) \tag{10}$$
$$e \to v: \quad \boldsymbol{\mu}_j^{t+1} = \mathbf{x}_j^t + \tilde{f}_v(\textstyle\sum_{i \neq j} \tilde{\mathbf{h}}_{(i,j)}^t) \tag{11}$$
$$p(\mathbf{x}_j^{t+1} | \mathbf{x}^t, \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_j^{t+1}, \sigma^2 \mathbf{I}) \tag{12}$$

Predicting for only one time-step can lead to degenerate decoders and hence multiple step decoder system is used. This is done in an interesting manner. The ground truth is used initially for the first step but the next iteration uses the prediction for the previous step. This is done for some number of iterations before the ground truth is again used. This leads to a better decoder structure being learned.

$$\begin{aligned}
\boldsymbol{\mu}_j^2 &= f_{\text{dec}}(\mathbf{x}_j^1) \\
\boldsymbol{\mu}_j^{t+1} &= f_{\text{dec}}(\boldsymbol{\mu}_j^t) \qquad\qquad t = 2, \ldots, M \\
\boldsymbol{\mu}_j^{M+2} &= f_{\text{dec}}(\mathbf{x}_j^{M+1}) \\
\boldsymbol{\mu}_j^{t+1} &= f_{\text{dec}}(\boldsymbol{\mu}_j^t) \qquad\qquad t = M+2, \ldots, 2M \\
&\ldots
\end{aligned}$$

Although a lot of physical systems are assumed to be markovian in nature, it is better to capture the temporal information of the system to be able reason about the dynamics better.Hence, a recurrent decoder formulation is also used.

One of the shortcomings of this formulation again is the use of unsupervised learning to estimate the interactions. It should however be learned in a supervised manner as the graph structure can change depending on the task at hand. This method aims to identify the interactions between various objects and hence does not require the attention mechanism that has been used across a lot of previous works and is one of the key novelty in this work.

2.12 Graph Neural Networks: A Review of Methods and Applications

The first motivation of GNNs roots in convolutional neural networks (CNNs) As we are going deeper into CNNs and graphs, we found the keys of CNNs: local connection, shared weights and the use of multilayer [49]. These are also of great importance in solving problems of graph domain, because 1) graphs are the most typical locally connected structure. 2) shared weights reduce the computational cost compared with traditional spectral graph theory [20]. 3) multi-layer structure is the key to deal with hierarchical patterns, which captures features of various sizes. Therefore, it is straightforward to think of finding generalization of CNNs to graphs.

In the following part, we explain the fundamental reasons why graph neural networks are worth investigating. Firstly, the standard neural networks like CNNs and RNNs cannot handle the graph input properly in that they stack the feature of nodes by a specific order. However, there isn't a natural order of nodes in the graph. To present a graph completely, we should traverse all the possible orders as the input of the model like CNNs and RNNs, which is very redundant when computing. To solve this problem, GNNs propagate on each node respectively, ignoring the input order of nodes. In other words, the output of GNNs is invariant for the input order of nodes.

Secondly, an edge in a graph represents the information of dependency between two nodes. In the standard neural networks, the dependency information is just regarded as the feature of nodes. However, GNNs can do propagation guided by the graph structure instead of using it as part of features. Generally, GNNs update the hidden state of nodes by a weighted sum of the states of their neighborhood. Thirdly, reasoning is a very important research topic for high-level artificial intelligence and the reasoning process in human brain is almost based on the graph which is extracted from daily experience.

we also investigate other mechanisms used in graph neural networks such as gate mechanism, attention mechanism and skip connection.

Types of Graph:

- Directed Graphs
- Heterogeneous Graphs
- Graphs with Edge Information
- Graph Neural Networks:
  - Limitations Though experimental results showed that GNN is a powerful architecture for modeling structural data, there are still several limitations of the original GNN. Firstly, it is inefficient to update the hidden states of nodes iteratively for the fixed point. If relaxing the assumption of the fixed point, we can design a multi-layer GNN to get a stable representation of node and its

neighborhood. Secondly, GNN uses the same parameters in the iteration while most popular neural networks use different parameters in different layers, which serve as a hierarchical feature extraction method. Moreover, the update of node hidden states is a sequential process which can benefit from the RNN kernel like GRU ans LSTM. Thirdly, there are also some informative features on the edges which cannot be modelled in the original GNN. For example, the edges in the knowledge graph have the type of relations and the message propagation through different edges should be different according to their types. Besides, how to learn the hidden states of edges is also an important problem. Lastly, it is unsuitable to use the fixed points if we focus on the representation of nodes instead of graphs because the distribution of representation in the fixed point will be much smooth in value and less informative for distinguishing each node.

2.13 Graph Networks as learnable physics engines for inference and control

2.14 Relational Deep Reinforcement Learning

The paper introduces an approach for deep reinforcement learning (RL) that improves upon the efficiency, generalization capacity, and interpretability of conventional approaches through structured perception and relational reasoning. It uses self-attention to iteratively reason about the relations between entities in a scene and to guide a model-free policy.

Deep RL models still face important limitations, namely, low sample efficiency and a propensity not to generalize to seemingly minor changes in the task.These limitations suggest that large capacity deep RL models tend to overfit to the abundant data on which they are trained, and hence fail to learn an abstract, interpretable, and generalizable understanding of the problem they are trying to solve.

Relational reinforcement learning (RRL) advocates the use of relational state (and action) space and policy representations, blending the generalization power of relational learning (or inductive logic programming) with reinforcement learning.

The paper's contributions are as follows: (1)it creates and analyzes an RL task called Box-World that explicitly targets relational reasoning, and demonstrate that agents with a capacity to produce relational representations using a non-local computation based on attention exhibit interesting generalization behaviors compared to those that do not, and (2) it applies the agent to a difficult problem – the StarCraft II mini-games – and achieve state-of-the-art performance on six minigames.

Moving from a propositional to a relational representation facilitates generalization over goals, states, and actions, exploiting knowledge learnt during an earlier learning phase. Additionally, a relational language also facilitates the use of background knowledge. Background knowledge can be provided by logical facts and rules relevant to the learning problem.

Neural nets have traditionally been associated with the attribute-value, or propositional, RL approaches. The paper translates ideas from RRL into architecturally specified inductive biases within a deep RL agent, using neural network models that operate on structured representations of a scene – sets of entities – and perform relational reasoning via iterated, message-passing-like modes of processing.

The framework equips a deep RL agent with architectural inductive biases that may be better suited for learning (and computing) relations, rather than specifying them as background knowledge as in RRL. The contribution is founded on two guiding principles: non-local computations using a shared function and iterative computation.

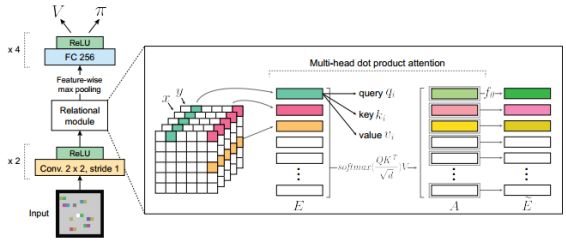

Figure 2: Box-World agent architecture and multi-head dot-product attention. E is a matrix that compiles the entities produced by the visual front-end; f_θ is a multilayer perceptron applied in parallel to each row of the output of an MHDPA step, A, and producing updated entities, $\bar{E}$.

$$A = \underbrace{softmax\left(\frac{QK^T}{\sqrt{d}}\right)}_{\text{attention weights}} V$$

Figure 17: A, Q, K, and V compile the cumulative interactions, queries, keys, and values into matrices, and d is the dimensionality of the key vectors used as a scaling factor

The framework starts by assuming that we already have a set of entities for which interactions must be computed. It consider multi-head dot-product attention (MHDPA), or self-attention, as the operation that computes interactions between these entities. The attention for any particular entity is calculated by taking into consideration all other entities and weighting them individually. The θ_k^h vectors, where h indexes the head, are concatenated together, passed to a multi-layer perceptron (2-layer MLP with ReLU non-linearities) with the same layers sizes as e_i , summed with e_i (i.e., a residual connection), and transformed via layer normalization, to produce an output

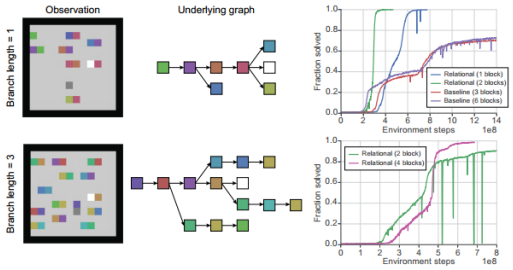

Figure 3: Box-World task: example observations (left), underlying graph structure that determines the proper path to the goal and any distractor branches (middle) and training curves (right).

	Mini-game						
Agent	①	②	③	④	⑤	⑥	⑦
DeepMind Human Player [15]	26	133	46	41	729	6880	138
StarCraft Grandmaster [15]	28	177	61	215	727	7566	133
Random Policy [15]	1	17	4	1	23	12	< 1
FullyConv LSTM [15]	26	104	44	98	96	3351	6
PBT-A3C [33]	–	101	50	132	125	3345	0
Relational agent	**27**	**196** ↑	**62** ↑	**303** ↑	**736** ↑	4906	**123**
Control agent	**27**	187 ↑	61	295 ↑	602	**5055**	120

Table 1: Mean scores achieved in the StarCraft II mini-games using full action set. ↑ denotes a score that is higher than a StarCraft Grandmaster. Mini-games: (1) Move To Beacon, (2) Collect Mineral Shards, (3) Find And Defeat Zerglings, (4) Defeat Roaches, (5) Defeat Zerglings And Banelings, (6) Collect Minerals And Gas, (7) Build Marines.

It is observed that – at least for medium size networks – there may be some interesting generalization capabilities, with the best seed of the relational agent achieving better generalization scores in the test scenario.High variability in these results is also noticed, with the effect diminishing when using larger models (which may be more prone to overfitting on the training set).

Given the combinatoric richness of the full-game, an agent is frequently exposed to situations on which it was not trained. Thus, an improved capacity to generalize to new situations caused by a better understanding of underlying, abstract relations is important.

2.15 End-to-End Differentiable Physics for Learning and Control

We present a differentiable physics engine that can be integrated as a module in deep neural networks for end-to-end learning. As a result, structured physics knowledge can be embedded into larger systems, allowing them, for example, to match observations by performing precise simulations, while achieves high sample efficiency. Specifically, in this paper we demonstrate how to perform backpropagation analytically through a physical simulator defined via a linear complementarity problem

Physical simulation environments, such as MuJoCo [Todorov et al., 2012], Bullet [Coumans et al., 2013], and others, have played a fundamental role in developing intelligent reinforcement learning agents. However, despite their ubiquity, these simulation environments are in some sense poorly suited for deep learning settings: the environments are not natively differentiable, and so gradients (e.g., policy gradients for control tasks, physical

property gradients for modeling fitting, or dynamics Jacobian for model-based control) must all be evaluated via finite differencing, with some attendant issues of speed and numerical stability.

In this paper, we propose and present a differentiable two-dimensional physics simulator that addresses the main limitations of past work. Specifically, like many past simulation engines, our system simulates rigid body dynamics via a linear complementarity problem (LCP) which computes the equations of motion subject to contact and friction constraints.

Model-based (deep) RL typically focuses on one of two settings: either a general neural network is used to simulate the dynamics (e.g. Werbos [1989], Nagabandi et al. [2017]), often with a specific loss function or network structured aimed at predicting on the relevant time scales [Abbeel and Ng, 2005, Mishra et al., 2017], or the model used is a more "pure" physics model, where learning essentially corresponds to traditional system identification [Ljung, 1999]. Our approach lies somewhere in between these two extremes (though closer to the system identification approach), where we can use the differentiability of the simulation system to identify model parameters and use the system within a model-based control method, but where the generic formulation is substantially more general than traditional system identification, and e.g. the number of objects or joints can even be dictated by another portion of the network.

2.16 DYNAMIC GRAPH REPRESENTATION LEARNING VIA SELF-ATTENTION NETWORKS

Learning latent representations of nodes in graphs is an important and ubiquitous task with widespread applications such as link prediction, node classification, and graph visualization. Previous methods on graph representation learning mainly focus on static graphs, however, many real-world graphs are dynamic and evolve over time. This is of particular interest in the case of physical systems where relationas between objects can constantly change with time or as a function of distance.

Learning dynamic node representations is challenging, compared to static settings, due to the complex time-varying graph structures: nodes can emerge and leave, links can appear and disappear, and communities can merge and split. This requires the learned embeddings not only to preserve structural proximity of nodes, but also to jointly capture the temporal dependencies over time.

This framework employs self-attention mechanisms to compute a dynamic node representation by attending over its neighbors and previous historical representations.

DySAT consists of a structural block followed by a temporal block, where each block may contain multiple stacked layers of the corresponding layer type. The structural block extracts features from the local neighborhood through self-attentional aggregation, to compute intermediate node representations for each snapshot. These representations feed as input to the temporal block, which attends over multiple time steps, capturing temporal variations in the graph.

STRUCTURAL SELF-ATTENTION:
The structural self-attention layer attends over the immediate neighbors of a node v (in snapshot of the graph, G), by computing attention weights as a function of their input node embeddings. Note that A_{uv} is the weight of link (u, v) in the current snapshot G.The set of learned coefficientsα_u^v , obtained by a softmax over the neighbors of each node, indicate the importance or contribution of node u to node v at the current snapshot.

$$\boldsymbol{z}_v = \sigma\Big(\sum_{u \in \mathcal{N}_v} \alpha_{uv} \boldsymbol{W}^s \boldsymbol{x}_u\Big), \quad \alpha_{uv} = \frac{\exp\Big(\sigma\Big(A_{uv} \cdot \boldsymbol{a}^T [\boldsymbol{W}^s \boldsymbol{x}_u || \boldsymbol{W}^s \boldsymbol{x}_v]\Big)\Big)}{\sum_{w \in \mathcal{N}_v} \exp\Big(\sigma\Big(A_{wv} \cdot \boldsymbol{a}^T [\boldsymbol{W}^s \boldsymbol{x}_w || \boldsymbol{W}^s \boldsymbol{x}_v]\Big)\Big)}$$

TEMPORAL SELF-ATTENTION:
The input of this layer is a sequence of representations of a particular node v at different time steps. The key objective of the temporal self-attentional layer is to capture the temporal variations in graph structure over multiple time steps. The input representation of node v at time-step t, x_v^t , constitutes an encoding of the current local structure around v. We use x_v^t as the query to attend over its historical representations ($<$ t), tracing the evolution of the local neighborhood around v. Thus, temporal self-attention facilitates learning of dependencies between various representations of a node across different time steps.

$$\boldsymbol{Z}_v = \boldsymbol{\beta}_v (\boldsymbol{X}_v \boldsymbol{W}_v), \quad \beta_v^{ij} = \frac{\exp(e_v^{ij})}{\sum_{k=1}^{T} \exp(e_v^{ik})}, \quad e_v^{ij} = \Big(\frac{((\boldsymbol{X}_v \boldsymbol{W}_q)(\boldsymbol{X}_v \boldsymbol{W}_k)^T)_{ij}}{\sqrt{F'}} + M_{ij}\Big)$$

$$M_{ij} = \begin{cases} 0, & i \leq j \\ -\infty, & \text{otherwise} \end{cases}$$

To conclude, DySAT computes dynamic node representations using self-attention over the (1) structural neighborhood and (2) historical node representations, thus effectively captures the temporal evolutionary patterns of graph structures

2.17 Graph Attention Networks

The paper presents a novel neural network architectures that operate on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations. By stacking layers in which nodes are able to attend over their neighborhoods' features, it enables (implicitly) specifying different weights to different nodes in a neighborhood, without requiring any kind of computationally intensive matrix operation (such as inversion) or depending on knowing the graph structure upfront.

When an attention mechanism is used to compute a representation of a single sequence,it is commonly referred to as self-attention or intra-attention

The framework introduces an attention-based architecture to perform node classification of graph-structured data. The idea is to compute the hidden representations of each node in the graph, by attending over its neighbors, following a self-attention strategy. The attention architecture has several interesting properties: (1) the operation is efficient, since it is parallelizable across node neighbor pairs; (2) it can be applied to graph nodes having different degrees by specifying arbitrary weights to the neighbors; and (3) the model is directly applicable to inductive learning problems, including tasks where the model has to generalize to completely unseen graphs.

The input to our layer is a set of node features, where N is the number of nodes, and F is the number of features in each node. The layer produces a new set of node features (of potentially different cardinality), as its output. In order to obtain sufficient expressive power to transform the input features into higher-level features, at least one learnable linear transformation is required. To that end, as an initial step, a shared linear transformation, parameterized by a weight matrix,is applied to every node.Self-attention is then performed on the nodes—a shared attentional mechanism-computes attention coefficients that indicate the importance of node j's features to node i.

In its most general formulation, the model allows every node to attend on every other node, dropping all structural information.

Once obtained, the normalized attention coefficients are used to compute a linear combination of the features corresponding to them, to serve as the final output features for every node (after potentially applying a nonlinearity,):

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right)$$

To stabilize the learning process of self-attention, the framework employs multi-head attention. Specifically, K independent attention mechanisms execute the transformation, and then their features are concatenated, resulting in the following output feature representation:

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

Specially, if multi-head attention is performed on the final (prediction) layer of the network, concatenation is no longer sensible—instead, we employ averaging, and delay applying the final nonlinearity (usually a softmax or logistic sigmoid for classification problems) until then:

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

The research presents graph attention networks (GATs), novel convolution-style neural networks that operate on graph-structured data, leveraging masked self-attentional layers. The graph attentional layer utilized throughout these networks is computationally efficient (does not require computationally intensive matrix operations, and is parallelizable across all nodes in the graph), allows for (implicitly) assigning different importances to different nodes within a neighborhood while dealing with different sized neighborhoods, and does not depend on knowing the entire graph structure upfront— thus addressing many of the theoretical issues with previous spectral-based approaches.

3 Conclusion and future work

The objective of physics engines is to provide fast approximate inferences about the real world through simulations.In recent years multiple techniques have been tried to build robust physical engines and with rise of deep learning, the scalability has also increased. Ideas like structured representation, object based representation, learning dynamics through learning interactions, etc have made the algorithms more generalized and also transferable to a variety of tasks. Future work aims to exploit these ideas and use them for model based planning and planning in unobserved environments. Physics engines could be used for active learning to improve safety and reliability in domains such as autonomous driving.

4 Application to Autonomous Driving

Over the years, starting with ALVINN[Pom89], autonomous driving has advanced considerably owing to the rise of deep learning[KSH12]. Some of the state of the art algorithms employed for autonomous driving come from behavioral cloning and imitation learning.These algorithms work well and good and with considerable reliability. However, the limitation of these algorithms is that they require a lot of data to learn and fail to respond 'safely' to unseen scenarios.

Let us consider the most optimal way for programming an autonomous vehicle. The most optimal way is to hard-code every possible scenario that is out there. This is the most 'optimal' way. Eg: Hard-coding when the distance to another vehicle is X meter, brakes should be applied. If we analyze the way this program has been written, we can understand that the position of the other vehicle is known relative to ours and based on that, we calculate the stopping distance using Newtonian mechanics and at that value of distance, the brakes are applied. There is considerable physics applied here. Let us consider the behavior cloning (BC) approach (a sub-optimal approach), we have expert driver data on the road. Considering the same scenario, we see that as soon as the driver approaches an approximate threshold distance to the other vehicle, the brakes are applied.The earlier physics is approximately computed by the driver while he made this decision.

What if we could have a model, which could learn the dynamics of the interaction between the two cars from the given data?. The behavioral cloning algorithm is only learning a mapping of state to action and not the latent physics that is related to the scene. The advantage?. In any similar scenario, irrespective of location, irrespective of the model of the nearby car, the physics model will kick in and perform while there is uncertainty related to behavioral cloning.

While using simulators (with analytic physics engine present at the back-end) for training , the BC model is not transferable to real domains. Why?. The state-action mapping of the vehicle that is learnt by BC has visual information and physical dynamics of the scene coupled together. While transferring to real domains, the visual information is considerably different even though the physical dynamics is very similar.The physics engine on the other hand, decouples the visual information from the physical dynamics. The physical dynamics that is learnt can be transferred directly or with a little bit of fine-tuning to the real world scenario. This further allows us to train the model in every scenario in the simulator such as crashes, accidents with pedestrians, etc. With reinforcement learning, we can train our agent with appropriate rewards to respond accordingly to such dangerous situations. This way, physics engines coupled with model based reinforcement learning is a 'better' solution than existing BC/IL algorithms

4.1 Few challenges in the approach

The physics engine learns the interaction dynamics between objects in the environment. There are a lot of interactions between car and its surrounding objects while driving. Decomposing the scene to capture all the objects interacting with the car is challenging. The interactions in this environment are not just pairwise but scaled up and complex. Capturing these interactions accurately is a challenge.

References

[ANA+16] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419, 2016.

[BHT13] 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.

[BPL+16] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016.

[CUTT16] Michael B. Chang, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *CoRR*, abs/1612.00341, 2016.

[FALM15] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *CoRR*, abs/1511.07404, 2015.

[GTH98] 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*, SIGGRAPH '98, pages 9–20, New York, NY, USA, 1998. ACM.

[H+78] Patrick J Hayes et al. The naive physics manifesto. 1978.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LGF16] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. *CoRR*, abs/1603.01312, 2016.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[Pom89] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 305–313. Morgan-Kaufmann, 1989.

[RD18] Deepak Pathak Alyosha Efros Thomas L. Griffiths Rachit Dubey, Pulkit Agarwal. Investigating human priors for playing video games, 2018.

[WLK+17] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 152–163. Curran Associates, Inc., 2017.

[WTW+17] Nicholas Watters, Andrea Tacchetti, Theophane Weber, Razvan Pascanu, Peter Battaglia, and Daniel Zoran. Visual interaction networks. *CoRR*, abs/1706.01433, 2017.

[WYL+15] 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 C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 127–135. Curran Associates, Inc., 2015.

[ZWZ+16] Renqiao Zhang, Jiajun Wu, Chengkai Zhang, William T. Freeman, and Joshua B. Tenenbaum. A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding. *CoRR*, abs/1605.01138, 2016.