Curiosity Driven World Models

Atharva Khandait
IIT Bombay
160010020@iitb.ac.in

Naman Aggarwal
IIT Bombay
160010058@iitb.ac.in

Krishna Wadhwani
IIT Bombay
160010031@iitb.ac.in

Abhay Jindal
IIT Bombay
160010060@iitb.ac.in

***Abstract*—Since 1990, Jürgen Schmidhuber has been putting forward the idea for a formal theory of creativity and intrinsic motivation [4]. This idea is based on the concept of maximizing intrinsic reward for the active creation or discovery of novel patterns allowing for improved prediction and data compression. In other words, curious agents build an abstract representation of their environment irrespective of any task. We take inspiration from this idea to extend the work presented in [1], [2], which are essentially derivatives of Schmidhuber's work in slightly different directions.**

***Index Terms*—Reinforcement Learning, World Models, Curiosity, Intrisically Motivated Agents, Deep Learning**

I. Introduction

Humans build a mental model of the world based on what they are able to perceive. They take actions and decisions based on this mental model, which is nothing but an abstract representation of the spatial and temporal aspects of all the vast amount of information that is exposed to our brain. One big problem with reinforcement learning is that it is inefficient in the number of trials. This is a big obstacle in the path to commercialization of reinforcement learning algorithms to practical applications, say driving a car. In practice, if we are learning an agent to drive a car using the current state of the art methods, the agent will drive the car off the cliff tens of thousands of times before learning the correct way (policy) to manoeuvre the car. What are we missing? What is it that allows most humans to learn to drive a car in an insignificant fraction of time?

We humans use our mental model to apprehend the consequences of our actions before those actions are actually taken (at least rational humans do!). We know what actions will lead to crashing of the car while at a turn, because the laws of motion are embedded in our abstract representation of reality.

There is a large body of research in model-based reinforcement learning methods. Several key concepts are highlighted in works such as [5], [6] on combinations of RNN-based World Models and Controllers which share similar ideas of learning a world model and training an agent using this model. Learning a good representative model of the environment is critical to the performance of an RL agent trained on features from the learnt model. This requires efficient exploration of the environment to completely encapsulate it's representation and dynamics in our world model. An exciting research direction is to look at ways to incorporate artificial curiosity or intrinsic motivation and information seeking abilities in an agent to encourage novel exploration of the environment.

In the paper by Ha and Schmidhuber [1], rollouts from a random policy are used to train their world model (details in sub-section A). A random policy might not be able to explore the complete environment, and as a result the model learnt on rollouts from a random policy is not a good representative of the environment. We expect to learn a better model by incorporating curiosity in our agent, as it encourages exploration of the environment by the agent.

In the following section, we discuss briefly the key details of our two primary references for completeness and the motivation for our work, following which, we discuss our work in detail.

II. Background

A. World Models

In this sub-section, we discuss the details of one of our primary references [1]. In their paper, the authors present a simple model inspired by our own cognitive system. In their model, the agent has a visual sensory component (Vision module) that compresses the observation into a compressed representation in the form of latent codes of a standard Variational AutoEncoder. The agent also has a memory component (Motion module), which makes predictions about future observations based on historical information. Finally, the agent has a decision-making component which decides what actions to take based only on the representations provided by the vision and memory components.

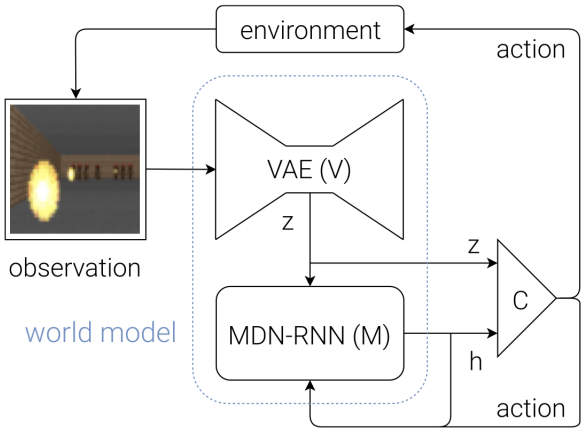

Fig. 1: World Models Architecture from [1]

1) VAE (V) Model: The environment provides the agent with a high-dimensional input observation at each timestep. The input is a 2D image frame that is a part of a video sequence from the gameplay. The role of the Vision module is to learn an abstract, compressed representation of each observed input frame.

2) MDN-RNN (M) Model: The Motion (M) module learns temporal dependencies between the frames. In essence, it learns the dynamics of the environment and acts as a predictive model for the next frame representation z_{t+1}, given z_t and, action taken at current timestep (from the random policy used to generate the video sequence of the gameplay) a_t. It outputs a probability distribution over z_t, and models $P(z_{t+1}|z_t, a_t, h_t)$ where h_t is the hidden state of the RNN at time t.

3) Controller (C) Model: The Controller (C) module is a simple single layer linear model which takes as input the hidden state of the RNN, h_t and current state representation, z_t, and outputs the action, a_t to be taken at the current time step. This architecture enables a minimal design for the controller module as almost all the complexity of the model is captured in the Vision and Motion modules.

The entire model is not trained end-to-end (details in [1]), which seems a bit counter-intuitive with respect to how humans learn. A model trained end-to-end seemed a more natural substitute for human intelligence as we learn about vision, motion and decision making in a joint manner. Also, the V and M modules as presented in [1], are not updated while running online experiments. Humans constantly update their mental model while undergoing any experience, and hence, there is a need for an architecture that keeps learning even during online gameplay. This observation motivated us to come up with our own architecture, details of which are presented in the following section.

B. Curiosity-Driven Learning

The basic idea of Curiosity-Driven Learning is to provide provide intrinsic motivation to the agent in the form of a reward, that is unique to the agent and is generated by the agent itself. Curiosity as a concept has been in the Reinforcement Learning community since the 1990s [4]. This concept was reintroduced in [2] with impressive results in several games where the agent was able to complete games such as Level 1 of Super Mario Bros without even being provided an external reward [3]. Essentially, curiosity is incorporated in a learning agent to tackle the two major problems of sparse rewards and inefficient exploration of the environment.

In the paper [2], the authors introduce a general framework of Intrinsic Curiosity Module (ICM) which can be combined with any learning algorithm to provide an intrinsic reward to the agent. They define curiosity as "the error in an agent's ability to predict the consequence of its own actions". So essentially in ICM, they learn a compressed representation

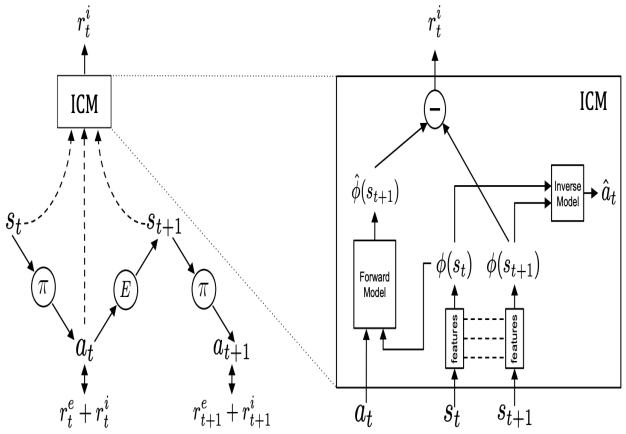

Fig. 2: Architecture of Curiosity driven Learning from [2]

of the environment and define curiosity as the error between the representation of the next state predicted by the agent given the current state and action and the representation of the actual next state which is obtained when the agent takes the particular action (See Figure 2). For such a definition of curiosity, learning a compressed representation is important as not all things in the image (state) of the environment is important in learning an optimum policy and even minor and irrelevant changes in the background of the environment might cause the prediction error to be high. In order to learn a representation of the state, they train a network that predicts the action given a state and the state at the next timestep. By doing so, the representation learnt will only capture those things in the environment which are influenced by the agent's action and affect the agent. The representation will ignore the aspects of the environment that do not affect the agent and are out of its control.

In such a setting, the agent learns a policy to optimize the sum of extrinsic and intrinsic rewards. The authors combine this ICM module with Asynchronous Advantage Actor-Critic learning algorithm to train the agent in Super Mario Brothers and VizDoom environments.

III. Our Work

Considering Schmidhuber's theory as presented in [4] as our conceptual goal, we made an attempt to come up with an architecture that is closer to the theory than both [1] and [2]. The motivation for our model can be summed up in the following points:

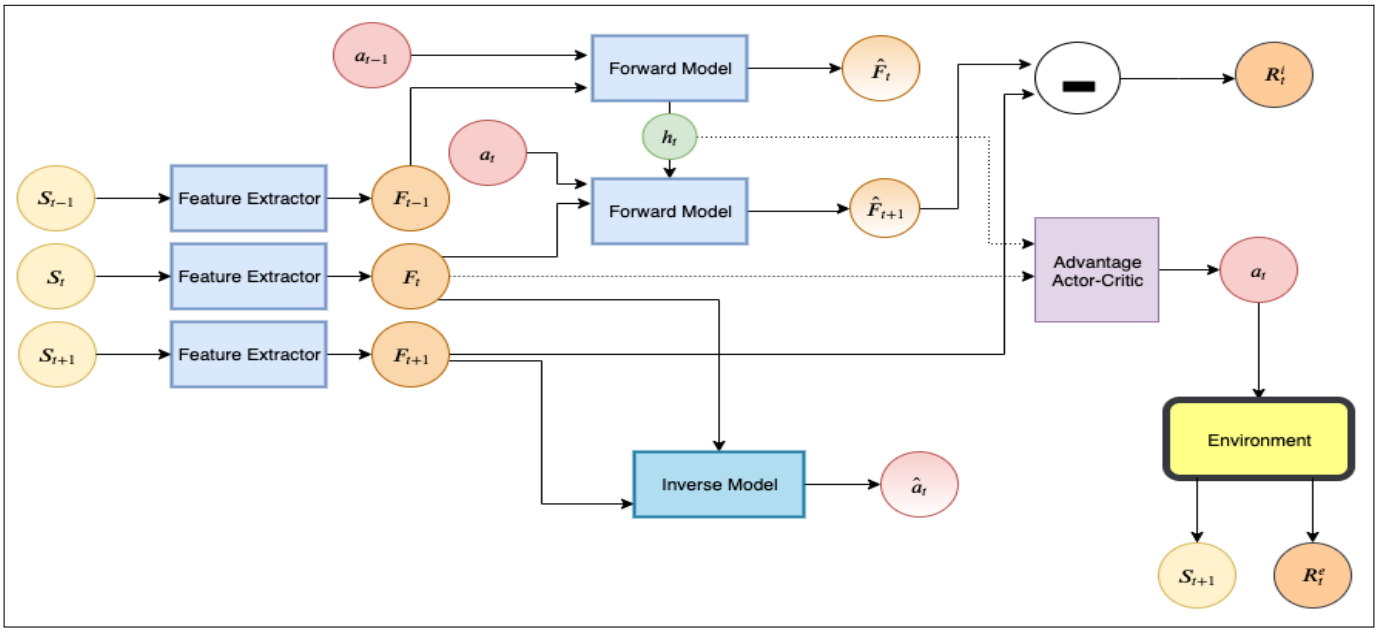

Fig. 3: Flow diagram of our architecture. The agent in state S_t interacts with the environment by taking action a_t, sampled from its current policy π and ends up in state S_{t+1}. This part is the same as in [2], except in our model, we have a very compact policy (like in [1]) which receives the output of the Feature Extractor (modelled as a CNN) F_t. Now, this compact policy network can be trained to optimize both the extrinsic and intrinsic rewards ($R_t^e + R_t^i$). Further details of our experiments are given in the next section. In our model, the intrinsic reward is generated by a modified version of ICM, where we use a Mixture Density Network-LSTM (MDN-LSTM) to predict $\hat{F}_{t+1}$, given F_t and a_t, instead of a feedforward network as used in [2]. We use Gaussian Mixture Modelling (GMM) loss between the outputs of the MDN ($\hat{F}_{t+1}$) and the output of the feature network (F_{t+1}). This loss is used to train the MDN-LSTM and is used as the intrinsic reward R_t^i (error in predicting the next state representation). We haven't made any changes in the feature network of the ICM, it encodes the states S_t and S_{t+1} into their features F_t and F_{t+1} that are trained to predict a_t (i.e. inverse dynamics model). Please note that dotted arrow in the above figure represents that gradient flow does not take place through that edge. This was done to stabilize the training process or else it might create a loop in the graph in which the network can simply collapse to predict zero for every thing in order to minimize the defined loss functions.

1) The World Model (V and M modules) in [1] is pre-trained on rollouts collected using a random policy and is not updated as the controller is trained. The learnt model is thus only useful in the parts of the environment which the random policy manages to reach. It does not have any insight about parts of the environment that the random policy couldn't reach but the agent might encounter during online gameplay. This might work for simple environments where the overall structure of the environment remains the same as the agent progresses. However, in environments where the structure changes over time, the pre-trained world model would have little understanding and wouldn't have the required information that the controller needs to take optimal actions.

2) As presented in [2], the ICM (Inverse Curiosity Module) does in some sense build a model of the environment. It learns a feature network for the spatial aspects of the environment, and a forward network as a predictive model for the next state. It immediately follows that the ICM has a very promising understanding of the spatial and temporal aspects of the environment. However, the agent uses the learnt features in the ICM to only provide intrinsic reward to the controller, and the model learnt is not used by the controller to make decisions. The controller learns a completely separate world model for the environment. We hypothesize that a lot of the features in the ICM and the world model used by the controller might be of similar nature. So, we re-use the features learnt by the ICM and pass them to our compact controller, as done in [1].

3) Another aspect of [2] which might be limiting is the usage of a feedforward network to model $P(\hat{F}(S_{t+1}) | F(S_t), a_t)$. The World Models paper [1] uses an RNN (Recurrent Neural Network) to model the dynamics of the environment which seems like a natural choice. In order to model temporal dependencies in [2], the state representation (S_t) of the environment is constructed by concatenating the current frame with the three previous frames. As done in [1], we use

an MDN-RNN (LSTM) in our architecture to model $P(\hat{F}(S_{t+1})|F(S_t), a_t, h_t)$, where h_t is the hidden state of the LSTM at time t. We use only the current frame as S_t which as a byproduct, reduces the training time.

A. Experiments

We performed our experiments on the *Super Mario Bros* environment, because it provides continually changing visuals and newer and newer aspects to the environment as the agent progresses. A static world model, like in [1], will struggle to perform in such a setting due to the continuously changing nature of the environment. We compared and analysed results of our model with the Curiosity driven learning model from [2].

Fig. 4: The environment rewards agents for moving towards the right as quickly as possible. The environment provides negative reward whenever the agent goes to the left.

The first part of our experimentation was to train the original architecture from [2] to establish the baseline for comparison. Following the experiments in the paper, no extrinsic rewards were provided and the model was trained entirely based on intrinsic rewards given by the ICM. In order to make comparisons, we record the extrinsic rewards and compare the models based on it. By the very definition of it, the curiosity module would encourage the agent to explore more, so it is fair to assume that a higher extrinsic reward implies a more curious agent. The training details have been provided in the following section.

We ran experiments for both the architectures (proposed and baseline) three times each and compared the two network's best performance in the three runs. As no extrinsic reward is actually provided during training, in both the cases, the agent tended to spend considerable time exploring difficult parts of the environment on the left and did not move towards the right (which would maximize the extrinsic reward). It is important to note that the environment also provides rewards for speed. Our agent received significantly less extrinsic reward in comparison to the baseline, which we reckon accounts for the inferior performance of our model. We have tried to summarise possible reasons for the failure of our model in a separate section. However, there were some encouraging results and insights from our model. The maximum position it reached to the right (maximum x position) was not as less as the extrinsic reward might have us believe (Table 1). This might be because

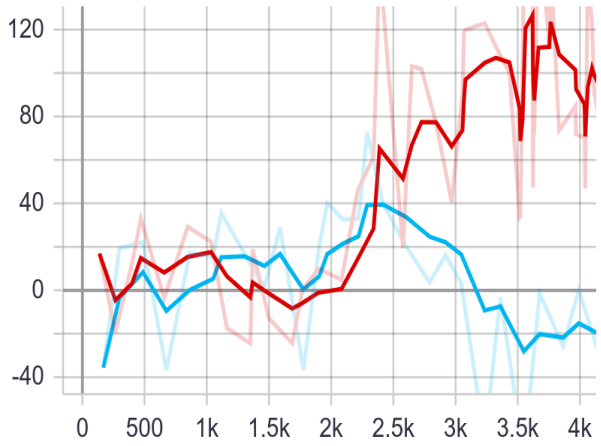

Fig. 5: **y-axis**: Extrinsic reward;
x-axis: Number of rollouts(each rollout is 2048 time steps);
Red: Original model; **Blue**: Our model;
We have shown smoothed out graphs to even out the outliers

the environment also rewards speed and negatively rewards the agent for any movement to the left.

Original	Our Model
1488	1105

TABLE I: Maximum x position

We also plotted the intrinsic rewards that the ICM gave as the training progressed (Fig. 6). We can see that the original

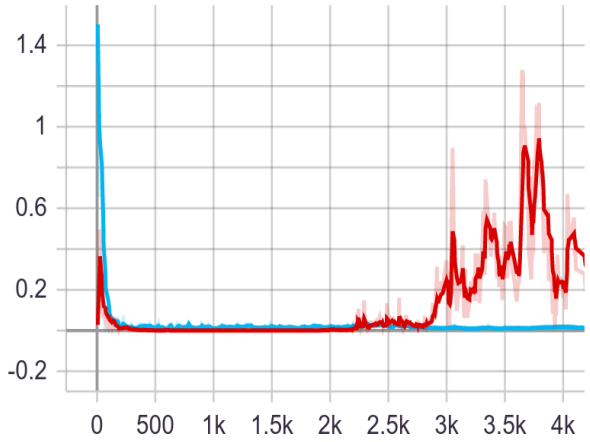

Fig. 6: **y-axis**: Intrinsic reward;
x-axis: Number of rollouts(each rollout is 2048 time steps);
Red: Original model; **Blue**: Our model

model's intrinsic reward peaked when it's extrinsic reward peaked, thus validating our earlier assumption that an higher extrinsic reward means a more curious agent. As the agent moves towards the right, it encounters newer visual aspects and different kinds of objects, which accounts for the higher

intrinsic reward, as the ICM cannot predict the next state well and the error is high. However for out model, the intrinsic reward in our model failed to pick up even when our model peaked in terms of the extrinsic performance. It never crossed a certain threshold, we try to explore the reasons in a following sections.

We ran another experiment with our model where we provided it the extrinsic reward given by the environment along with our intrinsic reward. Note that the environment has a dense reward setting.

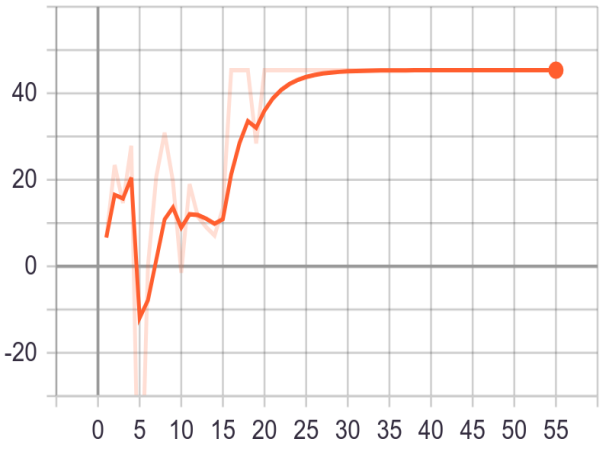

Fig. 7: **y-axis**: Extrinsic reward;
x-axis: Number of episodes (an episode ends when mario looses all three lives)

In this case (Fig. 7), the agent gets stuck in a local optima which is manifested as the player going straight towards the first enemy and dying. As the environment rewards speed, even at a maximum x position of just 201 (compared to Table 1), it gets a reward above 40 and then refuses to learn anything new.

B. Training details

All the agents were trained on image frames from the environment. We pre-processed the RGB images to grey-scale and resized to 84 x 84. We used Advantage Actor-Critic(A2C) method to train the controller.

A2C architecture: We used an actor and a critic network consisting of two linear layers each. Both take as input the concatenated $[h_t, F_t]$ having dimensions 256 + 256 = 512.

ICM architecture: The feature network consisted of 3 convolutional layers followed by a linear layer that outputs F_t. The inverse network (Inverse dynamics model) consists of 2 linear layers. For the forward network, we used an MDN-LSTM (5 Gaussian MDN) with a single hidden layer.

We use a learning rate of 0.0001 and Adam Optimizer for both A2C and ICM.

C. Possible reasons for the inferior performance of the model

As can be seen from the results, our model did not outperform the baseline. Some of the possible reasons for this might be as follows:

1) The first possibility is that there might some bug in the implementation of our model and the conceptual design of the architecture. Although, we have reviewed our code and architectural design thoroughly, but given the time crunch we cannot rule out this possibility with certainty.
2) It is also possible that there is a fundamental problem with our model. We attempted to use the features from ICM in the controller. It might be the case that the information/features that the controller needs to take optimal action that generate novel data for the ICM is different from the features that the ICM actually learns. However, as the ICM learns a very general abstract representation of the environment, we believe that this should not be the case. Moreover as the features learnt by the feature extractor network are directly used by the controller to predict the action, the learnt features should correlate to the optimum policy that is learned.

IV. Conclusion and Future Work

The concepts of Curiosity and World Models have seen a lot of interest in recent times and we feel like we have barely scratched the surface. The first direction for our future work would obviously be to get good performance out of our model either via training enhancements or modifications to the architecture while retaining our conceptual aim. Even after matching the baseline, there is still a lot of scope for further improvements. The agent in the original model manages to complete 38% of level 1 in Mario without any extrinsic reward. We can keep a goal of completing the level without any extrinsic reward, that would be a good step towards a truly curious agent. Also, we can try to generalize these models built by curious agents over multiple environments. This will be be especially significant as we believe that generalization seems to be the next step after the mental model abstraction and curiosity in mimicking the human cognitive intelligence.

References

[1] Ha and Schmidhuber, "Recurrent World Models Facilitate Policy Evolution", 2018.
[2] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros and Trevor Darrell. "Curiosity-driven Exploration by Self-supervised Prediction". In ICML 2017.
[3] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell and Alexei A. Efros. "Large-Scale Study of Curiosity-Driven Learning". In ICLR 2019.
[4] Schmidhuber, J. "Formal Theory of Creativity, Fun, and Intrinsic Motivation". IEEE Trans. on Auton. Ment. Dev. September 2010
[5] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath. "Deep Reinforcement Learning: A Brief Survey". IEEE Signal Processing Magazine, Vol 34(6), pp. 26-38. 2017.
[6] J. Schmidhuber. Deep Learning in Neural Networks: An Overview. Neural Networks, Vol 61, pp. 85-117. 2015.