CSC-584 Project Final Report

Ian Menezes, Rahul Ravindra, Vinayak Dubey

Introduction

Reinforcement Learning (RL) involves learning whilst interacting with the environment. The RL agent takes actions and the environment responds to them. In traditional Reinforcement Learning, the environment also provides a signal (a reinforcing signal), which the agent interprets as a reward for the action it just took. The agent learns with the objective of maximizing this reward. Reinforcement Learning suffers with two major problems. Firstly, these reward signals are not always existent. Take for example, games such as Super Mario Bros. or Vizdoom. In these games, there is no way a reward signal can exist which will be able to tell the agent that the jump it just made was a good one or not. The objective of these games is fulfilled on passing levels and hence the reward signal is very sparse. The only way to model such a reward function is by explicitly hard-coding the reward value for each possible action at each state. Secondly, the extrinsic nature of this reward puts forth another problem. These rewards are handmade and would need to be explicitly written for the environment. One can imagine the cost of scaling this effort to different games, environments, objectives, etc..

To address these problems *curiosity* was recently introduced in the field of Reinforcement Learning Pathak et al. 2017 in order to tackle scenarios where an extrinsic reward is not present in the environment. The motivation is enormous since extrinsic rewards provided to functions are usually human-crafted and rarely exist. This approach has shown promising results in many games and other applications. However, this approach is known to be affected by a problem now known as *procrastination* (The Noisy TV Problem). **In this effort, we implement the original work and test a hypothesis to tackle procrastination problem**.

In the following sections, we proceed in the following fashion: We first explain the work of Pathak et al. 2017 and how curiosity aids in exploration and generation of a policy for an agent in a game/environment. We then proceed to elaborate on the *procrastination* problem and discuss two previous approaches on solving this problem. Moving on, we put forward our proposed methods to tackle the problem and motivation behind using them. After this, we share our experiment setup. Lastly, we define metrics and share our results.

Curiosity Based Models

Figure 1: Agent imbued with surprise-based curiosity gets stuck when it encounters TV.
Screen grab adopted from a video by Pathak et al. 2017

As discussed above, reward based Reinforcement learning suffers because of two issues:

- The rewards are not always existent or are sparse. One has to hard-code these rewards. Sometimes there is no way to hard-code it either.
- This extrinsic reward is not scalable to many high dimensional environments/games.

This issue has earlier been tackled by modelling an intrinsic reward function which encourages the agent to explore novel states.

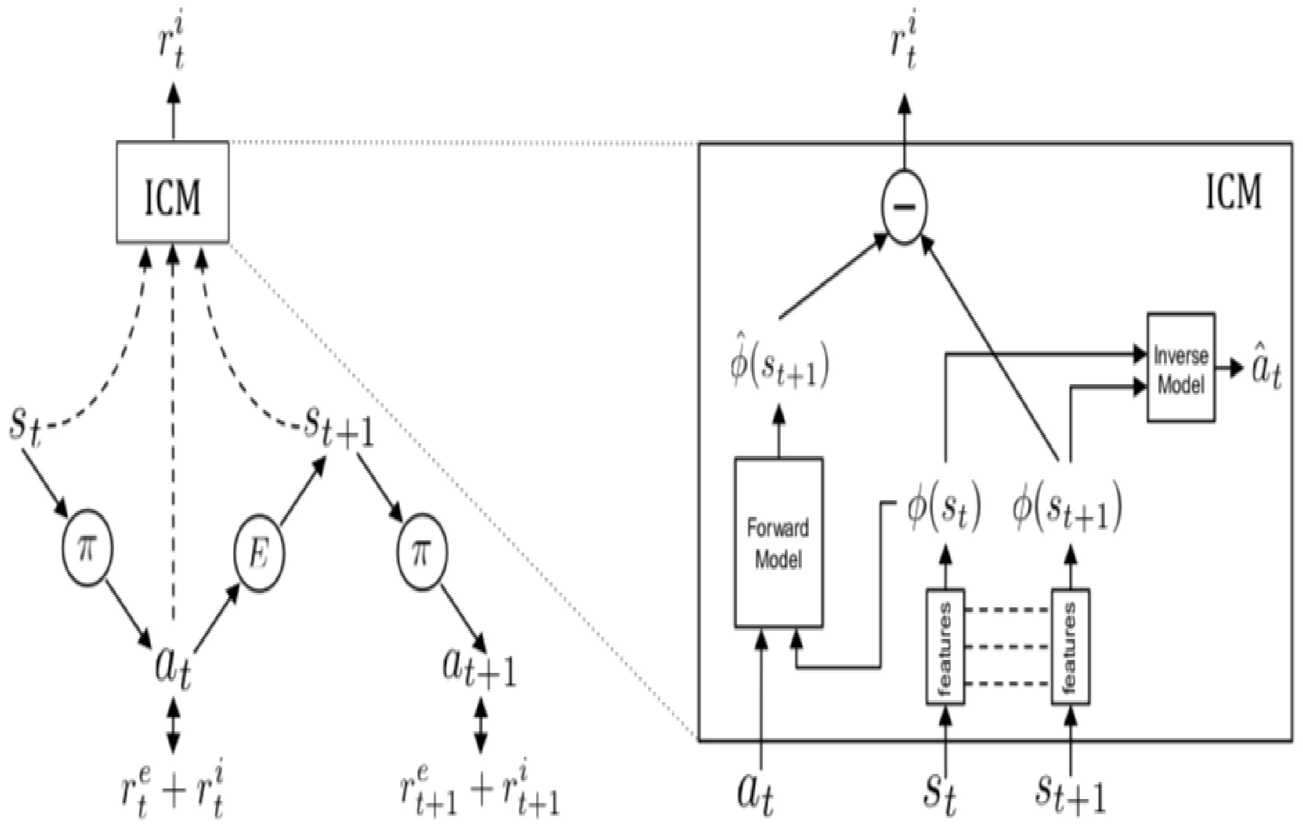

Figure 2: The model developed as part of Pathak et al. 2017, source: Pathak et al. 2017

Pathak et al. 2017 addressed this by bringing in the concept of curiosity into Reinforcement Learning. They modelled an intrinsic reward for the agent when no extrinsic one is present. The value of this intrinsic reward defined as the agent's uncertainty in knowing the consequences of its own actions. As the authors state: "curiosity is a way of learning new skills which might come handy for pursuing rewards in the future". In state machine terms, it is the prediction error of the agent in trying to predict the next state given as input the executed action and the current state.

This prediction error serves as an intrinsic reward. Now, the decoupled reward function becomes: $r_t = r_e + r_i$, where the extrinsic reward function is either sparse or non-existent for most states.

This approach has been shown to work exceptionally well when playing games such as Super Mario Bros. and VizDoom. However, there are more technical details to this to be taken care of when using this technique. Those are explained in the following subsection.

State Representation

The authors notice that predicting the next state in the raw pixel space is problematic. To understand why, let us consider an example from the original paper: In Super Mario Bros., it does not make sense to predict the next state in raw pixel space. This is because there are pixels which are not affected by the executed action such as a breeze and the consequent shift of tree leaves. Hence, the corresponding error when predicting the next state in raw pixel space provides a confused intrinsic reward signal.

To take care of this, the authors introduced a new module called the Intrinsic Curiosity Module which uses a self-supervised inverse dynamics model. The objective is to find a representation for the state which just contains the part of the state that is affected by the executed action. This module, tries to predict the action which was executed, given the current and next state as input. Once trained, the feature space of this model provides a representation of states which are relevant to the action taken. This feature space is then used to predict the next state.

Their Experimental Setup

The authors tested their idea on two environments: Vizdoom and Super Mario Bros. In Vizdoom, it was a navigation task where the action space is moving in the standard four directions. The map consists of nine rooms and the only reward provided is at the end when the agent finds the vest.

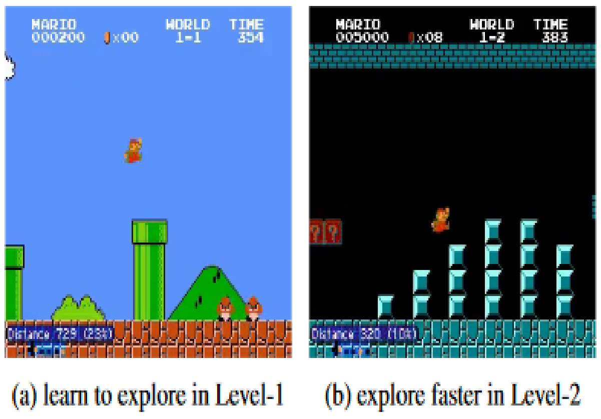
(a) learn to explore in Level-1
(b) explore faster in Level-2

Figure 3: Discovering how to play Super Mario Bros Pathak et al. 2017, Pre-training on the first level, ensures generalization on subsequent levels, source: Pathak et al. 2017

In the Super Mario Bros. experiment, there are 4 levels and the regular Mario actions. In this setup, there is no extrinsic reward function at all. The authors pre-train on just the first level and are able to demonstrate generalization on the subsequent levels.

The authors used A3C architecture (Asynchronous Advantage Actor Critic Mnih, Badia, et al. 2016) as their Reinforcement Learning agent and were able to demonstrate better performance than state-of-the-art methods.

The Noisy TV Problem

The authors in a subsequent paper Burda, Edwards, Pathak, et al. 2018, show that these curiosity based models are affected by an issue unsurprisingly known as Procrastination.

Since the basic focus of the curious agent is to optimize the intrinsic reward (which is the prediction error stated above), it essentially means that the agent keeps on trying to predict the next state of the game, given its current state and the action the agent performed.

Clearly, this approach will fail in a scenario when the agent is trying to predict the next state which is random. This is the gist of the Noisy TV Problem. For example, imagine a curious Mario going through Level-1, and encounters a TV. The existence of the TV allows Mario a new action, which is pressing the TV remote. However, the TV always changes to a random channel. Recall that the major focus of the curiosity based agent is to correctly predict the next state. However, in this scenario it will never be able to. This problem has been shown in Burda, Edwards, Storkey, et al. 2018 to cause the curiosity based agents to get stuck on the issue. In other words, the agent procrastinates instead of trying to reach the final/goal state.

The reason behind this is the fact that the prediction error will always be high when the noisy TV problem is encountered. Hence, the intrinsic reward is high. The agent, which is rewarded toward exploring novel states meets an irresistible distraction to keep on trying to figure out this seemingly random state.

For another example, consider a toy problem, as simple as a maze. Starting anywhere in the middle, we need to get out of maze and into the open. How will our AI agent behave here? Given the maze state space, it'll avoid banging into walls and start exploring the paths around them and make its way through it. When it reaches one of the many dead-ends of the maze, the curiosity element shall note that there's no more new explorations in this state and it will ward itself away from re-visiting this space. It shall continue to do so until it comes out of the maze.

Let's add the Noisy TV problem in this game space. What if one of the walls, has some random pictures moving across it. When our agent comes across this wall, it will get curious about exploring the pictures on the wall. The random motion of the pictures on the wall puts the now mobile agent to a halt so it can stop and predict the next picture on the wall. The randomness keeps the prediction error high and the agent will continue until it believes it can accurately predict the next state. While it's doing so, its goal of coming out of the maze via exploration is in jeopardy as it now explores the pictures on the wall. Like humans, the agent is procrastinating with such distractions while its goal of getting out of the maze remains a question mark. The unpredictable nature of the noisy TV keeps the agent busy and the agent now requires a push like a human would, when procrastinating.

Previous Work on this Problem

Savinov et al. 2018's attempt at solving this problem involves an episodic memory based model of granting RL rewards. The intuition behind their effort is basically to push the agent into exploring more and more states that it hasn't seen before. They ensure this by storing each state encountered. Now, along with the intrinsic reward, they provide a new reward which is equal in value to the difference between the current state and the most similar state seen in memory. Hence, this formulation aims towards pushing the agent to explore more and not get stuck at one point.

Burda, Edwards, Storkey, et al. 2018 recently published their own work on curiosity based models called: Random Network Distillation. The intuition behind this model is the same: it tries to provide incentives for the agent to explore more and more novel states and stop getting stuck at a problem. The technique involves predicting the output of a fixed and randomly initialized neural network on the next state, given the next state itself with the intuition that predictive models have low error in states similar to the ones they have been trained on. Hence, the agent's prediction will be less accurate in novel states than in the states visited previously. This technique makes Random Network Distillation immune to the Noisy TV Problem (i.e., Procrastination).

Our Approach and Research Question

We were inspired by the authors of Pathak et al. 2017, as they translated the human concept of curiosity to a reinforcement learning tool. Another notion that interests us (which is also the major contribution of that paper) is that the reward function no more remains artificially crafted and is derived naturally from the environment by the agent.

We wanted to progress on this challenge along with maintaining the natural intrinsic reward property. The only option that left us with is adding more information in the reward function but in a way that is derivable from the environment. Hence, we brainstormed how a human would solve the problem of procrastination. We came up with two ideas about how one does that:

- **Deadlines:** With some experience, we identified that humans tend to stop procrastinating when the deadlines are tougher or when they're approaching soon. Naturally, if the agent were to be given a penalty for each time step it wastes on the problem, it should become more restless and try as many things as possible to stop procrastinating and reach the final state.
- **Difference between current state and final state:** We identified that humans tend to stop or avoid procrastinating once they identify how far they are from the goal or when knowledge regarding the award at the goal inspires them. If an agent playing Mario receives a reward signal penalizing it for being in states that do not look similar to the final state, then naturally it should focus on coming closer and closer to the goal state.

Hence, we wanted to formulate new reward signals to solve the problem but also wanted to limit the reward signals to the intrinsic space: our reward functions are derivable from the environment, are not manually hard-coded for every state and do not cause scalability issues as the computation isn't to be done differently for higher dimensional games/environments.

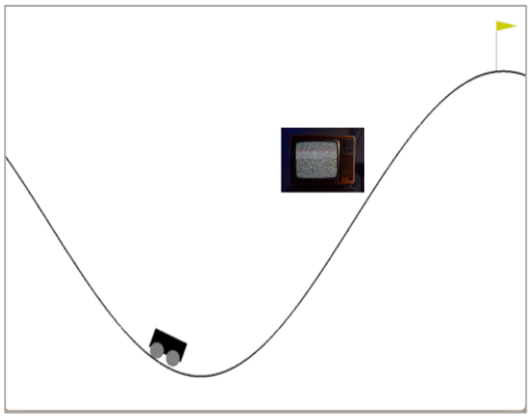

Figure 4: Noisy TV in Mountain Cart

Research Question/Falsifiable Claim: We wish to deduce whether the issue of Procrastination, which affects curiosity based models, can be solved by either adding a concept of time to reach the goal state or by informing the agent how far they are from the goal state. We wish to identify whether these notions can be used to *push* the agent away from the noisy TV. We hypothesize this knowing that we do not have a theoretical proof, and that we take this up as an exploratory project to test our hypothesis.

Our modification

We propose to modify the intrinsic reward function by adding these two notions:

- **Time/Deadline:** We will write a method to model the concept of time and deadlines in the game which the agent is required to reach the goal state in. This deadline will be added as a term in the reward function in order to inform the agent that it is wasting time at the Noisy TV and instead should focus on reaching other states.
  - Through this new reward signal, we will penalize the agent for every time step it spends in a game episode.
  - We believe that we will first need to train the agent without the TV but just with the notion of time and then in later epochs add in some noisy TV scenarios
- **Difference from the Goal State:** Another idea is to incorporate the difference between the current state and the goal state in the reward function. The intuition is to provide a reward signal which informs the agent how far away it is from the goal state.
  - This signal, we hypothesize, should push the agent towards moving to states that look more and more similar to the goal state.

  We formalize this reward as $R_{diff} = \alpha * \nabla(U(s'), U(s))^{-1}$, where $U(s)$ describes the utility of a particular state s as it's difference from the goal state.

  We will utilize combinations of these reward signals with the original formulation of the total reward as a sum of the extrinsic reward and the intrinsic reward:

$$R_{total} = R_{extrinsic} + R_{intrinsic} + (R_{time} \vee R_{diff})$$

$$R_{time} = \beta * F(steps)$$

$$R_{diff} = \alpha * \nabla(U(s'), U(s))^{-1}$$

Terms α and β are hyper-parameters that can be tuned based on need and the game environment being used. The function *F(x)* makes sure that the increase in the value of *x* causes an inverse affect on the the reward. We will test with two forms of *F(.)*: The inverse function, and negative function.

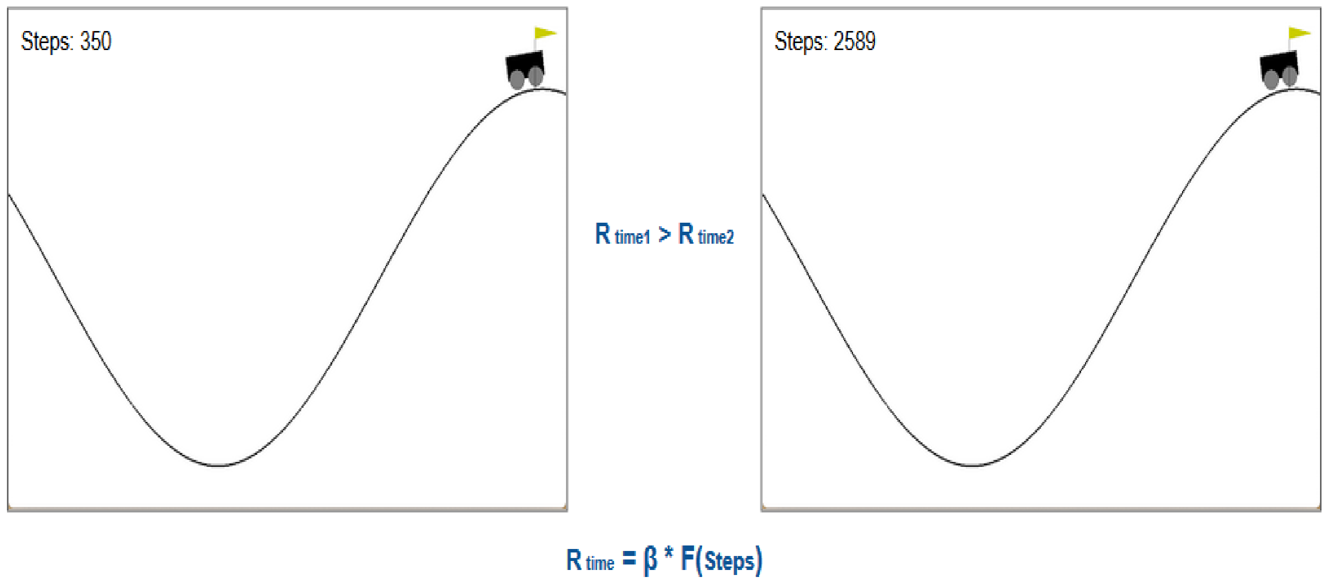

Figure 5: Deadline based reward

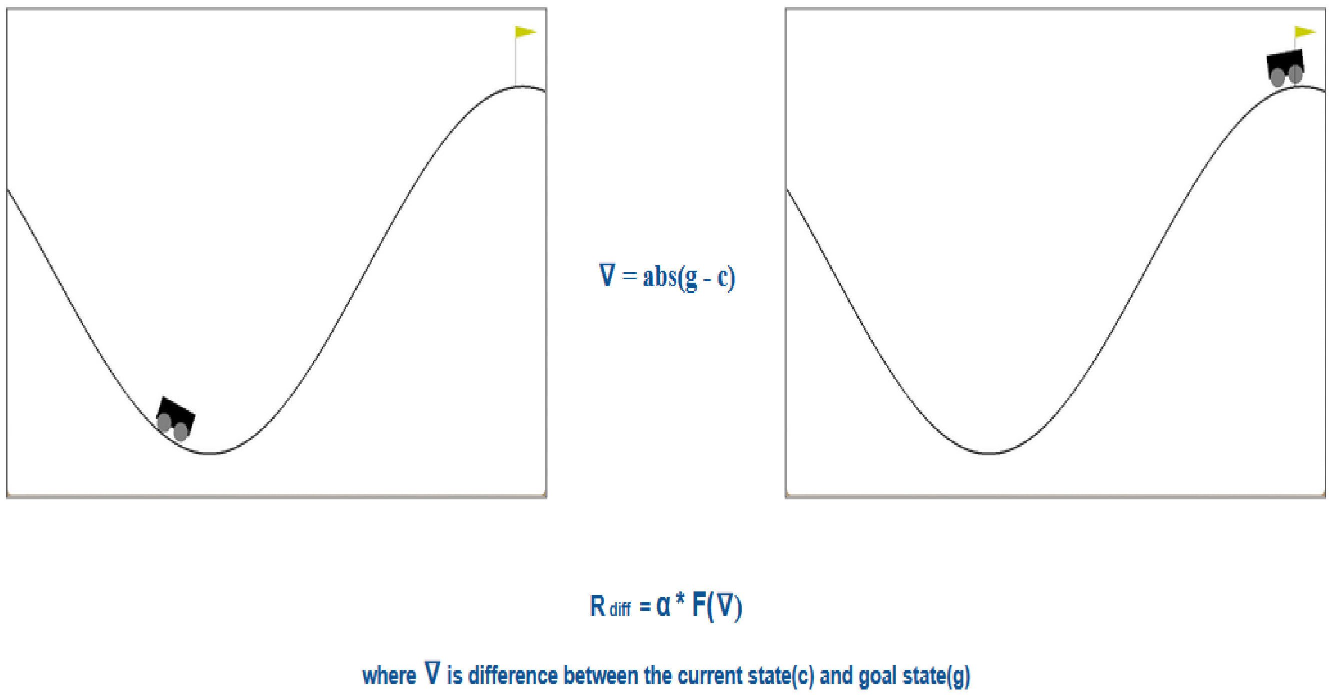
∇ = abs(g - c)
R diff = α * F(∇)
where ∇ is difference between the current state(c) and goal state(g)

Figure 6: State Difference based reward

Design Challenges and Experimental Setup

During the course of this project, we identified challenges which were also shared in the previous update. Following are the challenges we faced and the changes we made.

Challenges

- The code-base for the original problem requires running on much powerful machines than our personal computers and a different OS to run on.

- The original authors used convolutional neural networks (CNNs), which require strong GPUs to run. This is because the environments they ran on (Mario and Vizdoom), are high dimensional in the visual space and require a lot of compute.

Changes from the original implementation

To tackle these challenges, we made some changes:

- Firstly, to make the dimensions smaller, we shifted to a simpler game: The Mountain Cart game from Open AI Gym (Brockman et al. 2016). It is a 2 dimensional game (as can be seen in figures 5 and 6) where each state is represented by two floating point numbers which inform about the location of the cart on the mountain.
- In this implementation, we only offer a sparse +1 reward at the end of each episode when the cart reaches the goal.
- Next we wrote our own implementation of the curiosity model in Pathak et al. 2017. However, to work for the new Mountain Cart environment, we did not need heavy algorithms such as A3C or Convolutional Neural Networks.
- In our implementation, the reinforcement learning agent is a Deep Q Network (DQN) as in Mnih, Kavukcuoglu, et al. 2013. We implemented experience replay and target network techniques to make the network converge as suggested in the DQN paper. We use a one layer Multi Layer Perceptron as our neural network with the states as input.
- In this mountain cart game, we implement the TV in two ways:
  - **Method 1:** As mentioned above, each state in the Mountain Cart environment is represented as two floating point numbers. To implement the noisy TV, we will simply show the agent a random pair of floating point numbers. Hence, the agent's prediction error will always be high and that will induce procrastination.
  - **Method 2:** In this method, we extend the state space of the game. We start representing it as three numbers instead of two. This new third number is populated as a random one when the noisy TV is encountered and 0 when the noisy TV is absent.
- It is also to be noted, that the state representation consists of just two numbers which define the location of the cart. Hence, there are no elements in the state representation which are not affected by the execute action. Therefore, in our implementation, the inverse dynamics model is the identity function since we do not need to extract any specific feature representation.
- For the DQN, we use a one layer perceptron, which has 64 neurons in that layer. We use ReLU activation function. Our learning rate is set to 0.01, experience replay memory of size is 10000 and has to have at least 500 transitions stored for the agent to start training. We use a batch size of 128. We use a discount factor of 0.95 and an exploration quotient of 0.05.

Metrics and Methodology

Metrics: The only metric that we use is the *average number of timesteps* taken by the agent to reach the goal state over five runs, each consisting of 100 game episodes. Two averages are reported for each run: one for the whole episode, and one for the portion of the episode when the noisy TV was running. The reason for averaging over five runs is to make sure that we average out the stochasticity/randomness that occurs due to things such as neural network weight initialization, agent's exploration quotient, etc..

Following are our steps:

- **Step1:** We first run our implementation of the work in Pathak et al. 2017 on the Mountain Cart environment without the noisy TV. We store the required metrics and use this as our baseline.
- **Step2:** We then proceed to introduce our modifications one by one in the reward function. We run this new agent on the Mountain Cart game without the noisy TV. This is done to test and demonstrate that our modifications do not affect the original implementation adversely in normal scenarios. This means there will be metrics from three models: Original reward + time-reward, Original reward + diff-reward, Original reward + time-reward + diff-reward.
- **Step3:** Next, we introduce the noisy TV problem and record the metrics for the curiosity model implementation without our modifications. We record the metrics for both the ways we proposed to introduce the noisy TV problem.
- **Step4:** We then test our three modified models on these noisy TV environments.

Expected Outcome

Following are the expected outcomes of the previously defined steps:

- **Step1:** We expect this step to provide us with a solid baseline. Since Pathak et al. 2017 showed excellent performance on Super Mario Bros., without any rewards, we expect it to work just fine for the Mountain Cart game as well.
- **Step2:** We expect to see comparable performance as the baseline here. Our modification should not affect the original model adversely.
- **Step3:** Here, we argue that the procrastination problem will not cause the agent to get eternally stuck in the mountain cart game but cause a lot of delay in reaching the goal. This is because of the minimalist state representation in this game. The agent will get confused and stuck, but since the state representation is trivial, it will overcome the problem after some delay. Our preliminary results (which will be shared with the final report) confirm this as well.
- **Step4:** We expect the agents in this step to perform much better than the unmodified agent - i.e., reach the goal faster on an average.

Results and Interpretation

We implemented Pathak et al. 2017's curiosity model as the baseline. As shown in Fig. 7, it takes an average of about 1600 timesteps to reach the goal. We improved upon this with our goal based utility difference model to reduce the timesteps to about 500. As mentioned, we tune α to weigh the importance of our modification. When we use $\alpha = 5$, we achieve even better results and reduce the required number of timesteps to around 375.

For the next experiment, we introduced the Noisy TV problem into the Mountain Car game using method-1 described earlier by generating the next state randomly. Here, we noticed that our modification with $\alpha = 1$, is not able to outperform the baseline. However, with $\alpha = 5$, we were able to significantly reduce the timesteps as can be seen in Figure-8.

For the final experiment, we introduced Noisy TV problem into Mountain Car using method-2 described earlier by adding a random number to the state. We noticed that this is a diluted version of the noisy TV problem as the models are not very affected by it. As can be seen in Figure-9 all models perform pretty well.

Deadline based modification: We are not able to share results with the deadline modification as each episode of that took unreasonable amount of time (tens of millions of timesteps) to reach the goal state. Hence, we consider this particular modification useless in solving the procrastination problem.

With the goal based utility difference method, we notice that what this basically does is tell the agent to move closer to the state. However, we believe that writing this modification for higher dimensional game such as Mario, etc., will not be trivial and require more tuning to avoid issues like Mario dying, etc.. Also, with values of $\alpha > 1$, we basically end up overriding the original intrinsic reward function and that is why it works much better than the curiosity driven model in a simplistic game such as Mountain Car.

Limitations and Future Directions

- We haven't tested for high dimensional games such as Mario, VizDoom, etc.
- Our modification with the deadline did not produce good results at all. It took unreasonable amount of time to reach the goal even once.
- In the goal based utility difference modification, we might not always be aware of what the goal state looks like.

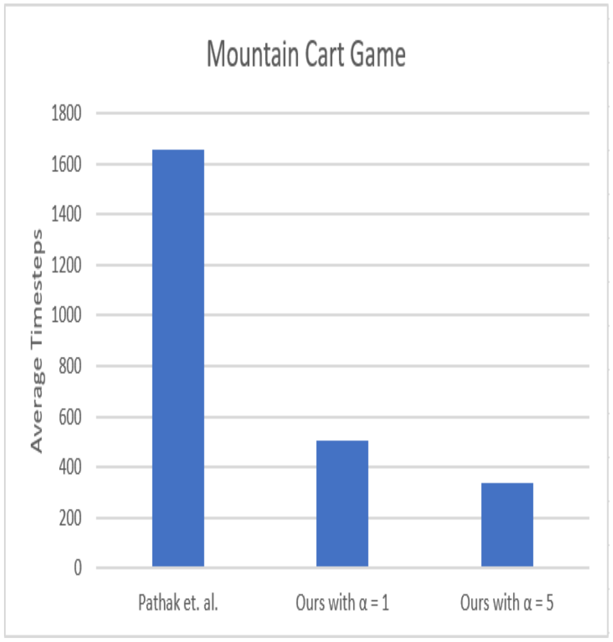

Figure 7: Results for the original Mountain Cart Game

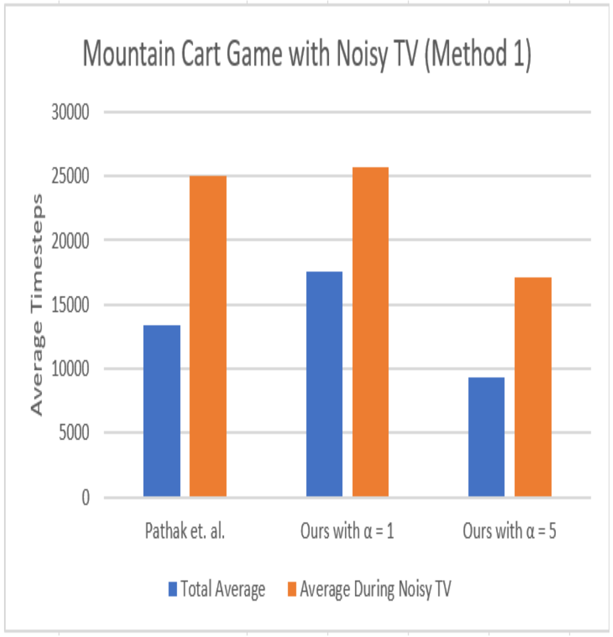

Figure 8: Results for the Mountain Cart Game with Noisy TV Method 1

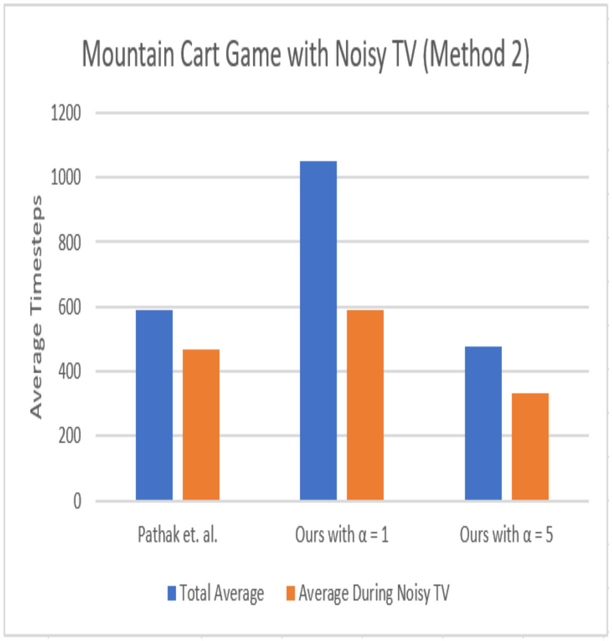

Figure 9: Results for the Mountain Cart Game with Noisy TV Method 2

References

[Bro+16] Greg Brockman et al. “OpenAI Gym”. In: *CoRR* abs/1606.01540 (2016). arXiv: `1606.01540`. URL: `http://arxiv.org/abs/1606.01540`.

[Bur+18a] Yuri Burda, Harrison Edwards, Deepak Pathak, et al. “Large-Scale Study of Curiosity-Driven Learning”. In: *CoRR* abs/1808.04355 (2018). arXiv: `1808.04355`. URL: `http://arxiv.org/abs/1808.04355`.

[Bur+18b] Yuri Burda, Harrison Edwards, Amos J. Storkey, et al. “Exploration by Random Network Distillation”. In: *CoRR* abs/1810.12894 (2018). arXiv: `1810.12894`. URL: `http://arxiv.org/abs/1810.12894`.

[Mni+13] Volodymyr Mnih, Koray Kavukcuoglu, et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). arXiv: `1312.5602`. URL: `http://arxiv.org/abs/1312.5602`.

[Mni+16] Volodymyr Mnih, Adrià Puigdomènech Badia, et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *CoRR* abs/1602.01783 (2016). arXiv: `1602.01783`. URL: `http://arxiv.org/abs/1602.01783`.

[Pat+17] Deepak Pathak et al. “Curiosity-driven Exploration by Self-supervised Prediction”. In: *ICML*. 2017.

[Sav+18] Nikolay Savinov et al. “Episodic Curiosity through Reachability”. In: *CoRR* abs/1810.02274 (2018). arXiv: `1810.02274`. URL: `http://arxiv.org/abs/1810.02274`.