End-Term Report

Adithyaa RG ED21B005, Shobhith Vadlamudi ED21B069

Course: DA7400, Fall 2024, IITM
Date: 12/11/2024

1 Problem Description and Existing Methods

Comparative Study for Reinforcement Learning Paradigms in Physical Reasoning Tasks

Recapping from our mid-term report, we notice that human beings and many other animals are known for their profound reasoning aptitude. This is one of the first things any toddler or animal learns. This whole process happens subconsciously in our brains. It could be a simple arrangement of shapes, or how dropping an object causes it to fall.

We chose the environment I-PHYRE: Interactive Physical Reasoning, published in 2024 by Shiqian Et al. from Peking University. This environment is designed to evaluate the Physical Reasoning ability of any Reinforcement Learning agent. The environment is designed with blocks and balls. It has a downward acceleration due to gravity, leading to the necessity of physics-based reasoning. The action space involves removing blocks from the environment, to facilitate the ball to fall into a pit. The aim of this paper aligns with the general aim of reinforcement learning learning through environmental interactions.

We believed that the I-PHYRE environment is a good candidate to evaluate baseline and modern RL algorithms because:

- **Physical Modeling**: As described before, most existing environments are grid-based, with transition dynamics being the primary influence. They don't have extensive physics rules, to influence the agent's behavior. This environment provides us with this behaviour.
- **Multi-step Interventions**: The ball's motion is very dependent on the blocks present in the environment. Multiple obstacles might obstruct the ball's motion, and thus, the agent would need sequential multi-step operations. This is a slightly challenging aspect for regular agents, who learn a policy based on the current state and do not plan for multiple steps ahead.
- **Action Timing**: Along with multi-step sequential actions, another essential part of the action is the time step at which each block is removed. The agent has a 5-second window when they can remove blocks, which can cause significant differences in the final outcomes based on the precision of those actions.

Language as an Abstraction for Hierarchical Deep Reinforcement Learning

This is a 2019 paper from Google Research written by Jiang et al. dealing with a hierarchical agent, using language that makes it similar to a human feudal agent. The paper uses an environment designed to give language-based user feedback for each action executed in the environment. Based on this feedback, it tries to learn a high-level policy that outputs a language-based goal condition, which a low-level policy uses to interact with the environment.

Problem Statement

The environment of I-PHYRE is challenging to learn using Reinforcement Learning. It is a sparse reward environment, causing existing baselines to perform poorly in the environment. The paper claimed the sub-par performance of the baselines compared to humans and suggested using model-based and hierarchical ideas to solve this issue of reward sparsity and time-based actions. Our main proposals were modified to:

- Reproduce results from the tests conducted on I-PHYRE environment using the baseline RL algorithms: SAC, PPO, A2C, DDPG, DQN
- Implement and Evaluate an approach inspired by the Language as an abstraction paper to solve, by solving its issues of learning in an environment without pre-existing language based feedback, and try applying it in the I-PHYRE settings.

2 Mid-term Work

2.1 Setting up the Environment

Details about the environment are given. We notice that the code required to run the environment has been provided to us. To replicate their exact results, we use the same algorithms given by the authors. They use the baselines implemented by RLLib library. We were able to successfully set up the whole environment, and run these experiments, to reproduce these results.

2.2 Running Experiments

Three planning strategies were implemented to evaluate the agents in the environment:

- **Inadvance**: Agents observe the initial scene to predict the timing for eliminating each block. They follow a predetermined plan, executing actions at specific time steps to handle the multi-step, time-dependent nature of the environment.
- **Onthefly**: Agents plan continuously and interact with the environment, making decisions at each step. This follows a classic RL approach, proposing a distribution of actions to maximize rewards at each time step.
- **Combine**: Agents initially determine execution times for all actions, execute the earliest action, and update timing based on the changed state until the game concludes.

Results for each of the 40 environments were tabulated, comparing deviation and mean with the data from the paper.

We see that the mean rewards obtained from our simulations are generally lower than the rewards shown in the paper. This is because the models in the training phase were run for 800-900k steps for the paper. We have set 500 as the maximum number of steps per episode, therefore that comes to around 1600 episodes, however due to our time constraints we have ran each algorithm for 200 episodes only. From the training plot that they have given we see an increase in performance after 150k steps for many of the algorithms. We then implemented our proposed architecture on the environment.

	Game	DDPG-I	DQN-O	A2C-I	A2C-O	A2C-C	PPO-I	PPO-O	PPO-C	SAC-I	SAC-O	SAC-C
0	support	969.100	977.60	972.3	974.7000	974.80	973.4000	977.6000	972.60	973.60	977.60	971.800
1	hinder	-35.000	-35.00	-35.0	-45.0000	-25.00	962.5000	961.7000	962.50	972.50	-45.00	972.500
2	direction	-55.000	-25.00	-45.0	953.3000	-45.10	948.9000	-55.0000	948.60	956.90	953.30	-45.100
3	hole	945.100	-45.00	-55.0	-55.0000	-45.00	949.6000	-55.0000	959.90	959.80	-55.00	-55.100
4	fill	-25.000	-15.00	-35.0	-45.0000	-35.00	957.9000	-45.0000	-35.00	-35.00	-45.00	-45.000
5	seesaw	-25.000	-25.00	-35.0	-35.0000	-15.00	-35.0000	-35.0000	-35.00	-35.00	-35.00	-25.000
6	angle	-55.000	-35.00	-55.0	-55.0000	-35.00	950.0000	-55.0000	-55.00	968.00	-55.00	982.900
7	impulse	965.000	-25.00	-25.0	-35.0000	-35.00	967.8000	971.5000	968.00	969.30	972.10	967.300
8	pendulum	967.500	-25.00	-35.0	966.5000	-35.00	969.1000	-35.0000	965.70	969.80	-35.00	-35.000
9	spring	979.600	-15.00	-25.0	975.7000	-35.10	970.9000	-35.0000	971.00	981.50	-35.00	981.900
10	noisy_support	-35.000	-35.00	945.7	956.4000	-35.00	952.6000	957.6000	953.80	-55.00	956.70	-45.100
11	noisy_hinder	-65.000	-35.00	-45.0	-65.0000	968.90	948.9000	-65.0000	952.50	-65.00	-65.00	-45.000
12	noisy_direction	-65.000	-55.00	-65.0	-75.0000	-45.00	938.5000	933.3000	939.00	-75.00	933.30	-65.000
13	noisy_hole	945.100	-45.00	-55.0	-65.0000	-35.00	951.3000	-65.0000	958.80	-65.00	-65.00	-55.000
14	noisy_fill	-55.000	-35.00	-55.0	-55.0000	-35.00	-55.0000	-55.0000	-55.00	-55.00	-55.00	-45.000
15	noisy_seesaw	-45.000	-25.00	-35.0	-45.0000	-25.00	-45.0000	-45.0000	-45.00	-35.00	-45.00	-45.100
16	noisy_angle	-65.000	-65.00	-45.0	-75.0000	-45.00	-75.0000	-75.0000	-65.00	-65.00	-75.00	-55.100
17	noisy_impulse	965.000	-15.00	-45.0	959.5000	-35.00	956.2000	960.6000	956.60	-45.00	960.30	-45.000
18	noisy_pendulum	-45.000	-25.00	-35.0	959.9000	-35.00	-45.0000	-45.0000	958.40	-45.00	-45.00	-45.100
19	noisy_spring	-35.000	-15.00	-25.0	-45.0000	-35.10	963.3000	-45.0000	-35.00	-45.00	-45.00	-45.000
20	support_hinder	-55.000	-25.00	-35.0	-55.0000	-45.00	948.7000	-55.0000	-55.00	-55.00	-55.00	-55.000
21	support_direction	945.800	-35.00	950.3	-65.0000	-25.00	-55.0000	-65.0000	957.60	-65.00	-65.00	-55.000
22	support_hole	-65.000	-35.00	-55.0	-65.0000	-45.10	-55.0000	-65.0000	-55.10	-65.00	941.60	-65.000
23	more_step_hole	-35.000	-25.00	-35.0	-45.0000	-45.00	973.4000	-45.0000	-45.00	-45.00	965.40	-45.000
24	hinder_fill	-55.000	-65.00	-65.0	-75.0000	-45.00	-75.0000	925.3000	-65.00	-75.00	-75.00	-35.100
25	impulse_spring	-35.000	-35.00	-35.0	-35.0000	-25.00	-35.0000	972.5000	-35.00	-35.00	973.10	-35.000
26	impulse_pendulum	965.700	-15.00	-35.0	970.9000	-15.00	966.3000	972.9000	966.10	967.60	973.50	966.600
27	activated_pendulum	-35.000	-45.00	-25.0	-45.0000	-35.00	-45.0000	-45.0000	-35.00	-35.00	961.70	-45.000
28	spring_flick	968.900	-25.00	-45.0	-45.0000	-35.10	958.8000	-45.0000	-45.00	966.80	964.80	-45.000
29	seesaw_angle	-45.000	-35.00	-35.0	-45.0000	-45.10	-45.0000	-45.0000	-45.00	-45.00	-45.00	-35.000
30	multi_ball_stack	-45.000	-15.00	-25.0	-45.0000	-45.00	-45.0000	-45.0000	-45.00	-45.00	955.40	-35.000
31	multi_ball_hinder	-55.000	-25.00	-35.0	949.8000	-45.10	-55.0000	-55.0000	-45.00	-55.00	-55.00	-55.000
32	multi_ball_redirect	-45.000	-15.00	-35.0	964.5000	-35.00	-45.0000	964.5000	961.20	-45.00	964.50	961.700
33	multi_ball_hole	-65.000	-45.00	-45.0	-65.0000	-25.00	-65.0000	-65.0000	-55.00	-65.00	-65.00	-65.000
34	multi_ball_fill	-45.000	-45.00	-45.0	-65.0000	-45.00	-65.0000	-65.0000	937.70	-65.00	-65.00	-55.000
35	multi_ball_lever	970.800	980.80	980.8	974.0000	990.80	970.5000	971.2000	965.70	990.80	965.80	970.800
36	multi_ball_angle	-55.000	-25.00	-55.0	-55.0000	-45.00	-55.0000	-55.0000	-55.00	-55.00	-55.00	-45.100
37	multi_ball_pendulum	-45.000	-25.00	-45.0	-55.0000	-35.00	-55.0000	-55.0000	-45.00	-55.00	-55.00	-55.100
38	multi_ball_spring	-25.000	-35.00	-35.0	-55.0000	-35.00	949.9000	955.6000	-45.00	-55.00	956.10	-35.000
39	multi_ball_spring_flick	-45.000	-15.00	969.9	953.5000	-35.00	-55.0000	-55.0000	-45.00	-55.00	-55.00	-55.000
40	Average	230.815	19.46	84.6	251.2175	39.97	478.0875	251.3575	405.39	228.54	352.13	156.615

Figure 1: The results obtained from our simulations

Table A4: **All results of humans and different RL agents on `I-PHYRE` benchmark, measured by rewards of gameplay.**

Game	Human	Random	DDPG-I	DQN-O	A2C-I	A2C-O	A2C-C	PPO-I	PPO-O	PPO-C	SAC-I	SAC-O	SAC-C
support	975.96	977.6	968	970.3	973.6	970.6	974.3	973.6	977.6	973	969.6	977.6	975.9
hinder	971.57	-45	-25	972.5	962.5	962.5	962.5	962.5	962.5	962.5	962.5	-45	972.5
direction	971.89	953.3	-55	947.3	958.4	963.3	959.7	949	953.3	948.9	963.3	953.3	970
hole	966.08	-55	955.1	952.5	950.1	-55	-55	949.7	962.5	960	959.6	953.9	966
fill	970.89	-45	-35	-45	957.1	-45	-35	958.4	-45	958.6	-45	-45	969.5
seesaw	436.36	-35	-25	-35	-35	-35	-35	-35	-35	-35	-35	-35	-35
angle	770.47	-55	-45	948.2	952.9	949.3	962.9	950	949.4	950	958.3	951.6	-45
impulse	970.47	972.3	966.5	973.5	967.5	971.3	967.8	967.8	972.3	967.7	966.9	972.1	-35
pendulum	972.56	-35	969	-35	966.9	-35	971.7	968.7	-35	971.2	968.2	-35	-35
spring	973.84	970.1	969.6	-35	970.7	-35	970.6	970.9	976.5	970.9	984.2	-35	-35.1
noisy_support	977.09	957.6	949	947.9	-45	957.6	-45	952.5	954.4	953.8	952.9	957.6	960.3
noisy_hinder	967.33	-65	-55	952.5	951.9	942.3	962.5	938.9	942.5	952.5	-65	-65	-45
noisy_direction	972.81	928.7	-45	-25	940.6	-75	950.2	928.5	-75	938.6	940.1	-75	-55
noisy_hole	966.43	-65	955.4	-55	950.3	946	959.5	941.3	942.5	951.5	-65	-65	-65
noisy_fill	971.47	-55	-45	-15	-45	-55	-35	-45	-55	-55	-55	-55	-55.1
noisy_seesaw	455.88	-45	-25	-45	-45	-45	-45	-45	-45	-45	-45	-45	-35
noisy_angle	565.88	-75	-45	-25	-75	-75	931.4	935.3	-75	941.3	-75	-75	-45
noisy_impulse	968.36	-45	-35	-35	956.7	-45	957	956.2	962.3	956.9	-45	-35	-45
noisy_pendulum	972.59	-45	-35	-15	-45	-45	-45	961.4	-45	-45	-45	-45	-45.1
noisy_spring	974.47	-45	-35	983.1	960.8	-45	-45	961.9	966.6	963.3	966.1	-45	984.3
support_hinder	961.54	-55	-55	-25	946.1	-55	-45	946.8	-55	945.9	-45	-55	-35
support_direction	963.07	-65	956	-45	-35	-65	951.3	-55	-65	936.2	-65	-65	-55
support_hole	957.88	-65	-45	-25	-65	945	-45	-65	-65	-65	-65	-65	-45.1
more_step_hole	969.51	-45	-45	-25	-45	965	-45	973.5	963.6	-45	-45	-45	-45
hinder_fill	954.2	-75	-55	-35	-65	-75	-55	-65	-75	-65	938.3	-75	-55
impulse_spring	281.96	-35	-35	-25	-35	-35	-35	-35	-35	-35	-35	-35	-35
impulse_pendulum	975.79	972.8	966.3	-25	966.3	-35	969.2	966.5	973.4	966.2	967.5	972.3	968.7
activated_pendulum	577.84	-45	-45	-15	-45	-45	-45	-45	-45	-45	-45	-35	-45
spring_flick	935.83	-45	966	966	-45	974.9	-35	-45	-45	-45	958.6	963.9	975.1
seesaw_angle	409.22	-45	-35	-15	-45	-45	-45	-45	-45	-45	-45	-45	-35
multi_ball_stack	612.59	-45	-35	-25	-35	969.9	-35	958.5	-45	-45	-45	957.3	-45.1
multi_ball_hinder	487.67	948	-45	-25	-45	-55	-35	-55	-55	-55	-55	-45	-55
multi_ball_redirect	919.63	964.7	-35	-15	961.1	962.4	961.2	-45	964.5	962.4	955.5	964.5	-35
multi_ball_hole	673.51	-65	-45	-45	-65	-65	-55	-55	-65	-55	-65	-65	-65
multi_ball_fill	957.84	-65	-45	-45	936.8	-65	-55.1	-65	936.1	-45	-65	-65	-65
multi_ball_lever	980.33	969.7	990.8	990.8	968.9	969.9	970.8	969.1	974.5	970.7	980.8	976.9	980.8
multi_ball_angle	927.3	-55	-45	-45	-55	951.5	-45	-55	-55	-55	-55	-55	969.7
multi_ball_pendulum	928.52	-55	-45	-35	-55	-55	-45	-55	-55	-55	-55	-55	-45
multi_ball_spring	896.18	-55	-45	-25	-45	-55	960.5	947.9	-55	950.9	960.6	-55	-45.1
multi_ball_spring_flick	624.17	-55	956	-25	-55	945.9	-45	-55	-55	-45	-55	-55	-55
average reward	844.17	200.87	260.19	242.99	429.36	352.69	383.45	528.10	377.74	504.33	378.45	227.15	233.93
success rate	87.55%	25.00%	30.00%	27.50%	47.50%	40.00%	42.50%	57.50%	42.50%	55.00%	42.50%	27.50%	27.50%

Figure 2: Results obtained from the paper

```
Comparison of Averages between our simulations and paper's results:
        Mean (Paper)  Mean (Our Results)  Mean Difference
DDPG-I    260.1925           230.8150          29.3775
DQN-O     242.9900            19.4600         223.5300
A2C-I     429.3550            84.6000         344.7550
A2C-O     352.6850           251.2175         101.4675
A2C-C     383.4500            39.9700         343.4800
PPO-I     528.0975           478.0875          50.0100
PPO-O     377.7375           251.3575         126.3800
PPO-C     504.3250           405.3900          98.9350
SAC-I     378.4500           228.5400         149.9100
SAC-O     227.1500           352.1300        -124.9800
SAC-C     233.9300           156.6150          77.3150
```

Figure 3: Mean rewards comparison

```
Standard deviation between corresponding columns
A2C-C     483.507273
A2C-I     623.389236
A2C-O     637.497892
DDPG-I    274.574735
DQN-O     419.064110
PPO-C     440.922059
PPO-I     318.201911
PPO-O     567.758434
SAC-C     533.375768
SAC-I     366.194839
SAC-O     466.300086
```

Figure 4: Column-wise Standard deviation

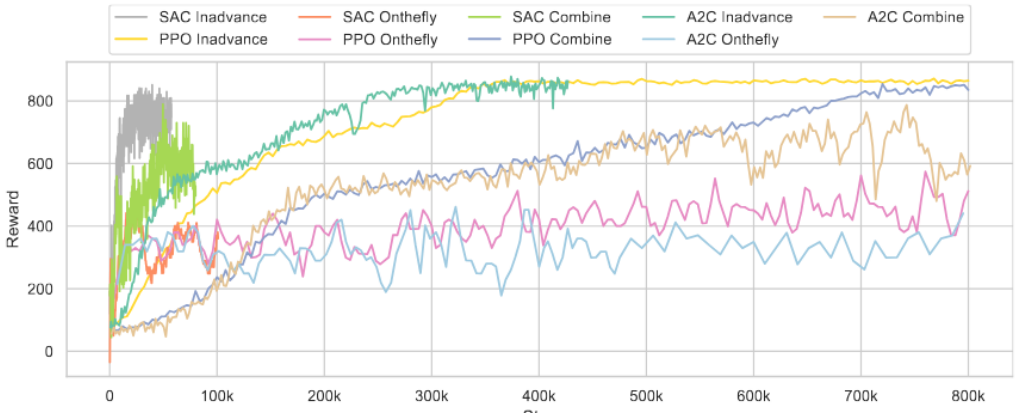


Figure 5: The training curves of different RL agents on the basic split

2.3 Results

Some of the results we had observed are:

- **Planning in Advance**: Agents using the "planning in advance" strategy converge faster, more stably, and effectively due to a more concise representation of the action space, but this limits real-time decision-making.

- **On-the-fly Planning**: On-the-fly planners experience oscillations in training due to challenges in learning sparse action distributions over extended periods, resulting in unstable training curves.
- **Combined Strategy**: While slower to converge, the combined strategy eventually reaches similar reward levels to advance planning by blending temporal planning with updating mechanisms, enhancing adaptability.
- **Generalization Across Splits**: All strategies generalize similarly across different data splits, with the combined strategy offering comparable generalization and adaptability, particularly for interactive physical reasoning tasks.

3 Proposed Architecture

3.1 Structure

As seen in 6, we aim to use the rendered images of the environment as an input, which would be a **partially observable input**. This seemed to be an intuitive idea because humans generally solve most of these reasoning based tasks due to their vision capabilities. This input contains image information, which we try to capture using a **Language Model**. The language model produces an output, based on the image and the prompt we give the same. We design this prompt to mainly identify a goal for a particular frame of image, to do the necessary action. As seen in the figure, we train an encoder-decoder structure here to encode this goal conditioning we get from the language model. We aim to use this as a latent embedded vector, as an input to a baseline, and learn a policy by interacting with the environment.

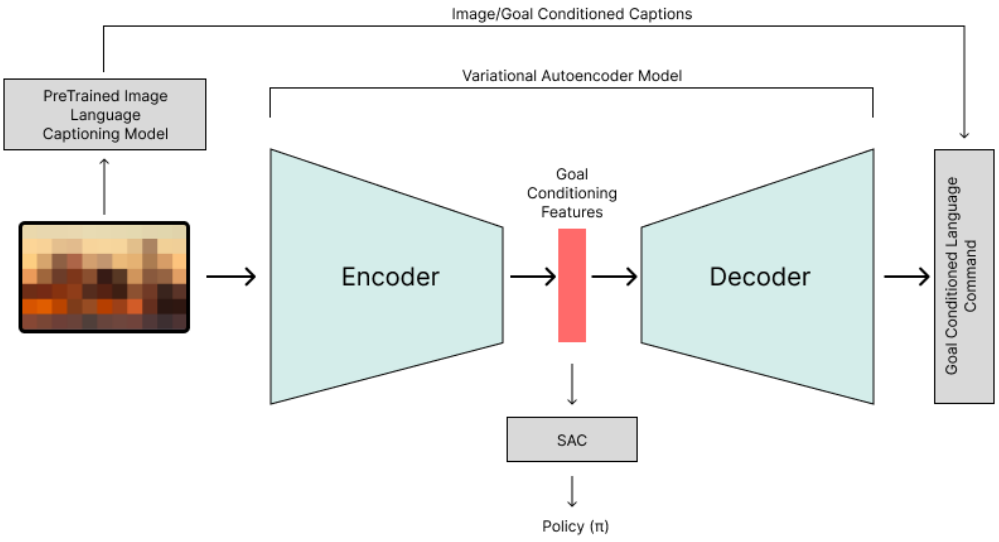


Figure 6: Proposed Architecture

3.2 Execution

While testing out this architecture, we designed it to have the following features:

- We chose the CarRacing environment from the Gym Box2D environments. We chose this as it is a well-tested RL environment with an image as a partial observable input. Since the I-PHYRE environment only gives a position value for each of the 12 objects as an input, we decided to test it out on CarRacing first.

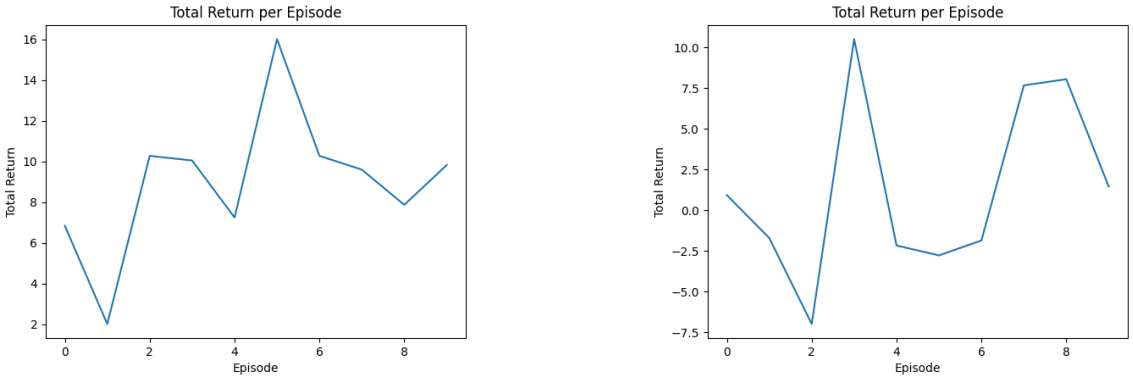
- For the language model, we chose a Visual Question Answer model BLIP which was able to process questions based on the image and give answers based on the same. The answers for each corresponding question were pretty accurate, implying it was a good candidate for the task; however, it gave one-word responses for most of the prompts.
- The encoder is a convolutional neural network used to encode and extract information from the image and encode it to a latent embedding.
- The latent embedding is used to learn a text-based output. This is executed with the help of an LSTM, with the goal-conditioned output from the VQA used to train this encoder-decoder structure.
- We used an SAC as the policy to train at the lower level based on the encoded latent goal representation. SAC is implemented using the StableBaselines3 library and was chosen due to its use as the baseline for the low-level policy in Language as an Abstraction paper.

3.3 Performance Evaluation

Due to the input being an image observation, initial training of 10000 steps was trained on both SAC and our proposed architecture on a GPU. The time taken for

- Soft Actor-Critic (SAC): 5.05 minutes
- Our Architecture: 28 minutes

Performance evaluated after 10000 steps of training also suggests that our environment is not learning properly, and SAC, in many instances, learns better over the same number of steps as shown in 7.



(a) SAC (b) Ours

Figure 7: Overall returns in 10 runs

s

Hypothesized Possible Causes for the same included:

- Sample Inefficiency: Since we use an image to get information about the environment, it becomes a partially observable environment, and thus would require a lot of samples to train the model. Many papers discussed about 2hrs of human gameplay, which includes about a million frames. This was not feasible on our local devices.
- Along with this, another primary reason could be the number of parameters that have to be learned. We need the agent to learn the parameters of the SAC, the encoder and the

decoder. This significantly higher number of parameters to tune could be causing delayed convergence and bad performance for a 10000-step trained agent.

- Along with increased parameters, there is an increase in the hyperparameters due to the 3 different networks with different structures and independent learning rates. It would be difficult to tune these due to possible correlations between these modules.

Possible Failures in the Language Model were:

- The pre-trained language model used for this evaluation, as described previously, was giving single-word responses for most of the input prompts. This could lead to the possible loss of information while training the latent vector.
- Another issue was that the prompt was chosen to represent any kind of action in the whole environment. This could have led to possible generalization in the question, thus affecting the performance of the agent.
- The lack of information from the language model could lead to improper embedding of the inputs to be conditioned on a goal due to minimal changes to the output vector from the language model, due to reduced vocabulary size

4 Modified Algorithm

The problem of increased parameters and hyperparameters is addressed with the use of a much-simplified architecture, as seen below. The output from the language model is directly fed into the decoder, which embeds into a smaller latent vector to a lower-level SAC policy and learns a policy.

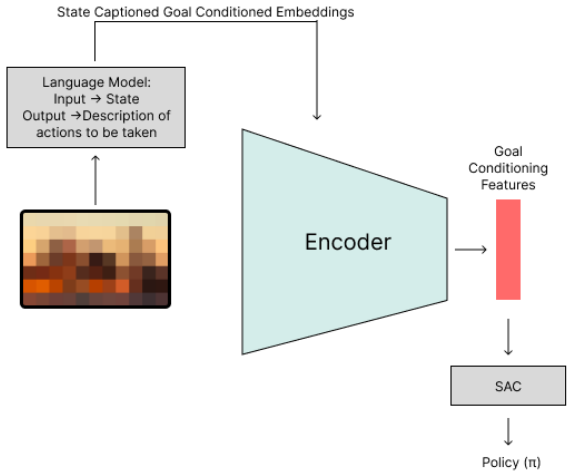


Figure 8: Modified New Algorithms

- Along with the simplified structure, instead of an image input, we feed the language model the current state and a detailed description of the environment for extra context.
- The BLIP model was replaced with the BERT model, which takes in text output to produce an encoded vector.
- The encoder is a neural network with ReLu activation to induce nonlinearity.

- This was directly tested out on the I-PHYRE environment instead of a Gym Environment due to the output being in our desired framework.

The agent's training on the environment using SAC is still about 3 times faster than our modified algorithm. We train 30000 steps on the I-PHYRE games and see if our agent can learn the same.

4.1 Results

A significant reduction in training time was observed, mainly due to fewer parameters to train. However, no noticeable difference was observed in the episode length/speed as training progressed. This suggested no learning had happened. The models at 30k steps have been evaluated in the corresponding environment. The model converges very quickly to a particular action, explaining the same average reward of -0.85 for every environment; it is never successfully completing the environment as a positive reward is obtained only after successful completion.

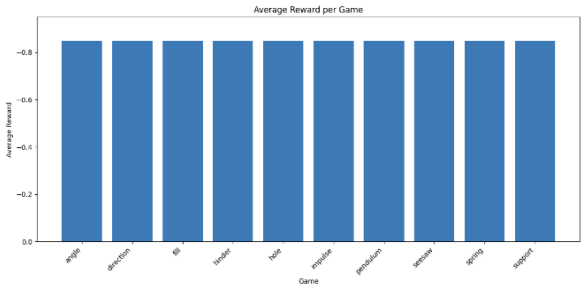
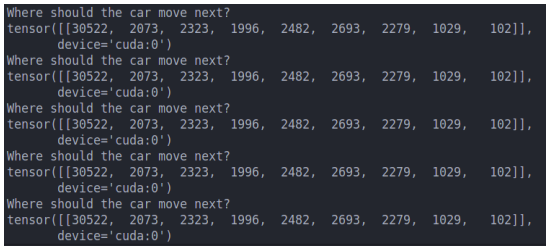


Figure 9: Resulting Returns

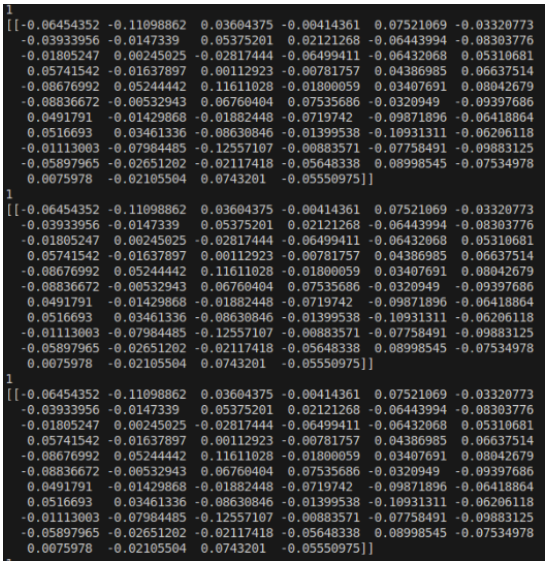
4.2 Possible Reasons for Failure

- While hyperparameter tuning could still helps with the performance, the inability to learn any policy suggests that the issue could be due to the language model. A regular SAC can still learn these environments, meaning the issue could be with the language model itself.
- The environment embeddings given by the language model could be the same, based on the state input and the prompt, which could lead to repeated training of the policy on the same embedded vector instead of different states.
- The other possible issue could be due to the generated embeddings getting clipped due to the size of the generated output, causing a loss of information.

As seen in 10, the language embedding for each input state remains the same. This is what causes the agent to have delayed learning. On the original CarRacing environment, it could be attributed to the language model not giving complex output, but the same phenomenon was seen while using BERT for I-PHYRE. Here, however, the input prompt was also significantly more detailed. This indicates that the language model is not able to extract the required information from the given input and, in most cases, is clouding the existing information available about the environment, converging at the same problem of learning from the same set of embeddings repeatedly.



(a) First Proposal



(b) Modified Proposal

Figure 10: Possible Failure Due to Language Model
s

5 Conclusion

- These results suggests that generative pre-trained models are not able to give extra information to the agent to learn, and in many cases, affect existing information.
- This is due to the inability of the embeddings to represent the goal for the next time step effectively as we don't see them change
- In the Language as an abstraction paper, they were using a curated environment, that was visual based and could give language feedback
- The fact that I=PHYRE and CarRacing are both sparse reward environments, we don't get significantly different states in every step. This leads to worse embedded states, due to repeated scenes of similar states. This might be fixed in a higher number of epochs, but can not be guaranteed.

Thus, language-based solutions don't seem to be the answer and, in some cases, might make it harder to learn the parameters. The possibility of improvements could be achieved by using a more robust language model that uses state information and environmental description to predict future states and make more comprehensive observations. GPT4 could maybe perform better.
However, if this is due to the sparse reward nature of the environment, maybe incorporating a world model idea by itself can be used to solve the I-PHYRE environment, where a language model might not really be necessary, as its prediction and inference aspect is performed by the world model .

6 Code repository

The codes used to generate the above results are given in the following GitHub repository : Code repo.