Designing an efficient online-learning Reinforcement Learning Setup with the use of Natural Language Instructions

Pandula Priyadarshana D.L.H.
Brettnacher Strasse 29, 14167, Berlin.
`dodamduwaleb@@uni-potsdam.de`

Abstract

In recent years, reinforcement learning methods have shown their strength by solving many complex problems. When comparing model-free and model-based RL methods, the latter is highly sample inefficient, but can be used to find optimal policies only using past states and rewards, even when the environment dynamics are unknown. Therefore, model-free RL methods provide a powerful set of tools that can be used to solve any sequential decision-making problem which can be formulated as a Markov decision process.

One popular method to make model-free methods sample efficient is to perform reward shaping. Reward shaping is about creating new reward functions that provide agents intermediate rewards in order to make them reach their goals faster. Reward shaping is a difficult and a time-consuming process, and thus, its applications are still very limited.

In this project, I propose a framework that can be used to integrate reward shaping process into any dynamical system on which RL methods are used to find optimal policies. In the proposed setup, the RL agent's learning phrase is made sample efficient by creating shaping rewards that are generated according to human-provided natural language instructions. This setup enables human instructors to observe the learning process of an RL agent and give natural language instructions to guide the agent to complete the assigned tasks quickly. Human instructors do not need to have any technical expertise to use this setup, and therefore, anyone who has knowledge about the environment can use this setup to influence the agent's behavior and make the agent reach its goal faster. Hence, this setup can be used to help any RL agent find optimal (or near optimal) policies with higher sample efficiency.

1 Introduction

Reinforcement learning (RL), which is a subfield of machine learning (ML), started to gain its popularity in the recent years due to many recent successes such as solving Go (Silver et al., 2016), beating world champions of Dota 2 (Berner et al., 2019), outperforming humans in all 57 Atari 2600 games (Badia et al., 2020). The core idea in RL is to study how past data can be used to improve future manipulations of a dynamical system. In other words, RL aims to find the correct sequence of inputs that should be fed into a dynamical system in order to maximize some objective, where we have no (or very limited) knowledge about how the system would react to the given set of inputs.

One can argue that the modern advancements in RL would not have been possible without the immensely powerful computing infrastructures to back them up. This fact is hard to deny because, for example, according to Silver et al., (2017) the agent who defeated the human Go champion had played around 4.9 million games during its training phrase, and the RL agent that beat Dota 2 world champions had played 45,000 years of gameplay within 10 real-time months (Berner et al., 2019) . These points clearly illustrate the sample-inefficiency of RL algorithms, and I believe that, this is one of the main reasons that limits the use of RL methods in industry.

During the past few years, many researches have been conducted to improve sample-efficiency in RL algorithms. Among them, experience transfer methods such as transfer RL (Gamrian et al., 2019), learning (or handcrafting) better state representations (Du et al., 2019), environment modeling and combining model-free and model-based RL methods (Weber et al., 2017; Pong et al., 2018), hierarchical RL methods (Pinsler et al., 2018), and designing/learning dense reward functions (Ng et al., 1999) have showed promising results.

Goyal et al. (2019) in their work propose a framework that is capable of generating dense reward functions by taking arbitrary natural language instructions and past trajectory information executed by an RL agent. This framework makes a prediction whether the agent had followed a given natural language instruction by looking at the executed trajectory and uses such predictions to generate intermediate rewards; therefore, generates dense reward functions. Their experiments show that the new dense reward function the framework generates speeds up the learning process of an RL agent, especially when the agent operates in a sparse reward setting.

One problem of the framework that Goyal et al. (2019) propose is that, the LanguagE-Action Reward Network (LEARN) model that lies at the heart of their approach is only trained once and not updated afterwards. This becomes problematic when we try to incorporate this framework into a stochastic environment. In a stochastic environment, the natural language instructions that we provide to an agent has to change according to the changes that takes place in the environment. For example, look at Figure 1 that shows three episodes from the same 'simple crossing' environment from MiniGrid (Boisvert et al., 2018). The path that the agent (red triangle) needs to follow in order to reach the goal (green box) in each episode is different, and therefore, the agent should receive different natural language instructions at each episode for the same environment. The framework of Goyal et al. (2019) fails to make the learning process sample efficient when applied on this environment as the LEARN model will only be trained on one version of this stochastic environment.

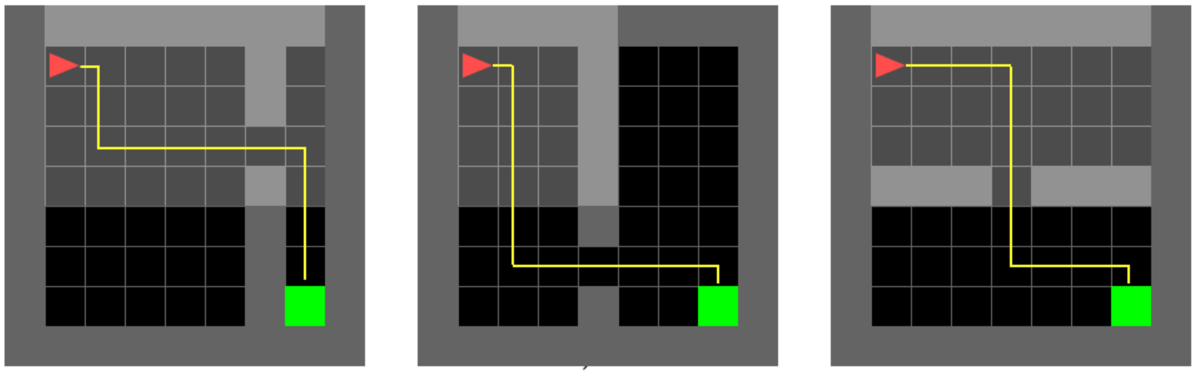


Figure 1: Three episodes of the Simple Crossing Environment: Yellow lines represent the paths that the agent (red arrow) needs to follow in each episode in order to reach the goal state (green box). Language instructions that the agent should receive in order to follow each of these paths would be different.

Another problem of Goyal et al. (2019) work is that the RL agent's past trajectories has to be translated into *action-frequency vectors* in order for the LEARN model to make predictions. The creation of action-frequency vectors needs human intervention, and therefore, this fact constrains the framework's usability. When we combine this point with stochastic environments, the proposed framework becomes unusable as the environments will be always changing. Therefore, the previously coded action-frequency vectors will have no validity after a certain time point since the new trajectories of the agent, once the environment changes, will be completely different.

In this project, my primary contribution is to propose a method to tackle above discussed issues and design an online learning version of the LEARN model so that the framework Goyal et al. (2019) proposes can be used in any environment. The newer version of the LEARN model that I proposed does not need action-frequency vectors as it directly learns from the trajectory inputs. It also carries the advantage of not needing to extensively train before integrating into the actual environment. Since the new version of the LEARN (called *'I-LEARN'*) will be learning online, the first training duration of the model can be cut short; therefore, the data collection process for the new method that I propose will not need to be outsourced to external parties as Goyal et al. (2019) has done during their work.

2 Proposed Approach

To implement the proposed approach of this project, I extend the core ideas developed by Goyal et al. (2019). The extended language-augmented Markov Decision Process (MDP+L) framework which is defined as $\langle S, A, R, T, \gamma, l \rangle$ is used where l is a natural language instruction that human instructors provide to guide the agent's intended behavior. In this project, any state s in the set of states S includes a sequence of three consecutive frames/images of the environment that describes agent's behaviour during those three consecutive time-points. Similar to Goyal et al. (2019), I employ reinforcement learning to find an optimal policy π^* that maximises the expected sum of rewards using a three-step approach:

STEP 1: Initial training the **Image-LanguagE-Aided Reward Network (I-LEARN)**

At this step, similar to Goyal et al. (2019) work, first I train a neural network (NN) using trajectory and language-instruction pairs. This network predicts whether the given language instruction explains with the bahaviour of the agent which is observed by the input trajectory. The main difference between LEARN and I-LEARN is that, even though LEARN requires action-frequency vectors as inputs, I-LEARN does not require that; I-LEARN is trained using trajectories of visual inputs, specifically, three consecutive visual inputs (image frames) that captures the behavior of the agent in the focused environment. Therefore, I-LEARN measures how closely the agent's behavior observed through the imagery input trajectory relates to the given language instruction when generating the prediction.

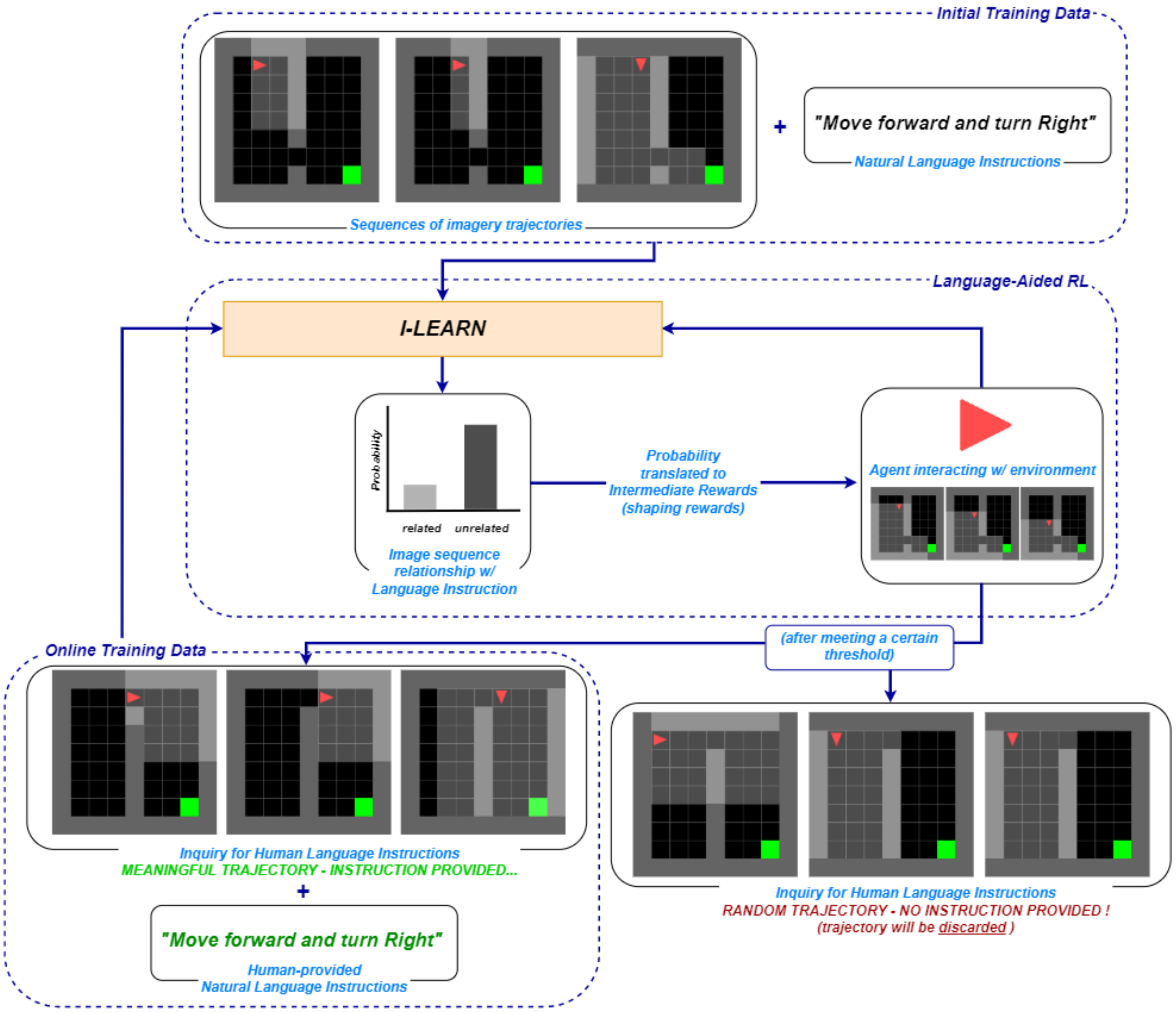
STEP 2: Performing **language-aided RL**

This step is exactly as of Goyal et al. (2019) work; here I employ a model-free RL method to solve the MDP+L problem formulated above. The I-LEARN model is used at this step to predict whether the agent is following the provided language instructions, and these predictions made by the I-LEARN model are used to generate shaping rewards. The shaping rewards generated in this manner contribute to make the training phrase of the agent more efficient (more on this point is discussed under section 4).

STEP 3: Online learning of I-LEARN

This step executes in parallel with STEP 2. Once the agent starts to interact with the environment, I design the framework to capture sequences of imagery trajectories from agent's behaviour and poses them for human input; the framework requests for new language instructions from a human instructor that would help the agent to interact with the environment efficiently. A human instructor provides new language instructions for the trajectories that he/she believes would aid the agent to reach its goal state quickly, and avoids providing instructions for any nonsensical/random trajectories. These requests for human inputs occur after a certain number of agent-environment interactions or after meeting a certain threshold (more about this discussed in Section 3), which can be tuned after taking the stochastic-nature of the environment into account, i.e., if the environment changes rapidly, the framework can be programmed to prompt for human input in shorter time intervals and vice-versa. When the framework receives new human language instructions, the posed trajectory and the input natural language instruction pairs are treated as training data samples, and used to retrain the I-LEARN model.

In a nutshell, the approach that I propose continuously trains the I-LEARN neural network based on imagery trajectory inputs and natural language instructions. Due to the continuous learning process, I-LEARN captures the stochastic nature of the environment and correct the reward shaping process accordingly. Therefore, at any given time point, I-LEARN is able to examine a visual trajectory of the agent and predict how closely the agent's behaviour aligns with the most recently provided language instruction, and therefore, generate intermediate rewards that are most relatable to the latest configurations of the environment. An overview of the proposed approach is described in Figure 2.



Initial Training Data
Sequences of imagery trajectories
+
"Move forward and turn Right"
Natural Language Instructions
Language-Aided RL
I-LEARN
Probability
related
unrelated
Image sequence relationship w/ Language Instruction
Probability translated to Intermediate Rewards (shaping rewards)
Agent interacting w/ environment
(after meeting a certain threshold)
Online Training Data
Inquiry for Human Language Instructions
MEANINGFUL TRAJECTORY - INSTRUCTION PROVIDED...
+
"Move forward and turn Right"
Human-provided Natural Language Instructions
Inquiry for Human Language Instructions
RANDOM TRAJECTORY - NO INSTRUCTION PROVIDED !
(trajectory will be discarded)

Figure 2: The proposed framework is an enhanced version of the Goyal et al. (2019) framework; it is capable of directly using imagery data as inputs and the model has an online learning process to capture the dynamics of stochastic environments

3 Description of the Model

3.1 Image-LanguagE-Aided Reward Network

During the initial training process, the I-LEARN model uses pairs of image sequences and natural language instructions as training data. In each pair, the image sequence includes three consecutive frames from the environment and these frames represent some behaviour trajectory of the agent; the natural language instruction tied to a particular image sequence describes the agent's observed behaviour trajectory.

The primary use of the I-LEARN model is to predict how closely the agent's behaviour trajectory (observed through a given image sequence) relates with an input natural language instruction. Therefore, the I-LEARN model acts as a binary classification model. In order to produce the intended result, I generate positive training data samples by creating (image sequence, language instruction) pairs where the language instruction of each pair accurately describes the corresponding image sequence. Similarly, negative training data samples are created by matching irrelevant image sequences and natural language descriptions with each other.

When creating positive training data samples, I only couple image sequence and natural language instruction pairs that leads the agent reach its goal quickly. This is done having the primary focus of the I-LEARN model in mind, i.e., to create intermediate rewards and make the RL algorithm become more sample-efficient. Training the I-LEARN model on such positive data samples allows it to accurately identify the trajectories that relate to language instructions which guides the agent to the goal-state; therefore, these predictions can then be used to generate intermediate rewards during the language-aided RL phrase. The I-LEARN model produce these predictions as probability values, and they are later translated into shaping rewards.

3.2 The Neural Network Architecture

The inputs to the neural network are pairs of image sequence and natural language instructions. The image sequence and language instruction information in each pair are separately transformed into embedding vectors first. Thereafter, the resulting embedding vectors are joined together to form a single vector. The concatenated vector is passed through a series of fully connected layers, and at the final (output) layer, a probability distribution over two classes ('RELATED' and 'UNRELATED') are produced. These two classes signify whether the agent's trajectory observed through the input image sequence relates to the input natural language instruction or not.

In order to transform the image sequence into an embedding vector, I use the pre-trained ResNet50 (He et al., 2016) neural network. ResNet50 is a convolutional neural network that is 50 layers deep and trained on the ImageNet (Deng et al., 2009) dataset. While keeping all other pretrained weights unchanged, I used the last layer of the ResNet50 network to generate the image embedding vector.

In a similar manner, in order to create sentence embedding vectors for natural language instructions, I used one of the models available in the *SentenceTransformers* framework (Reimers et al., 2918), namely the 'roberta-large-nli-stsb-mean-tokens' model, which is optimized for semantic textual similarities and is based on The RoBERTa model (Liu et al., 2019).

The input image sequence is (244 x 244 x 3 x 3) dimensional and is transformed into a (2048 x 3) dimensional image embedding vector at the end. The input natural language instruction is also converted into a one-dimensional sentence embedding vector of length 1024. Afterwards, these two vectors are combined to generate a one-dimensional concatenated vector of length 7168. This concatenated vector is then passed through three fully connected layers having ReLu activation function and reducing dimensions. Finally, a softmax layer is placed as the output layer to generate the probability distribution over 'RELATED' and 'UNRELATED' classed. A simplified illustration of the I-LEARN model architecture is presented in Figure 3.

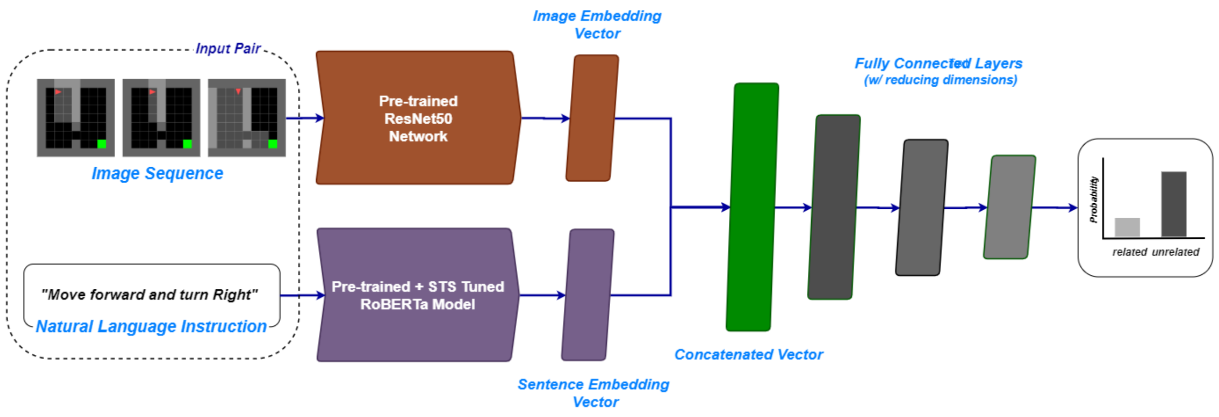

Input Pair
Image Sequence
"Move forward and turn Right"
Natural Language Instruction
Pre-trained ResNet50 Network
Pre-trained + STS Tuned RoBERTa Model
Image Embedding Vector
Sentence Embedding Vector
Concatenated Vector
Fully Connected Layers (w/ reducing dimensions)
Probability
related unrelated

Figure 3: The neural network architecture of I-LEARN model

During the initial training phrase, the I-LEARN model is trained using backpropagation with Adam optimizer (Kingma and Ba, 2014) over 500 epochs to minimize the binary cross-entropy loss.

3.3 The Online Learning Setup

After training the I-LEARN model, it is incorporated into the training framework of the RL agent to generate intermediate shaping rewards (more on this in section 4). To handle the drawback of Goyal et al. (2019) work with regards to not being able to successfully generate shaping rewards in stochastic environment, I implement a continuous update loop for the I-LEARN model by following the below process:

- Calculate the moving average of the shaping rewards over last N episodes: ***MovAvgShapRwrd***
- Compute the shaping rewards generated in the most recent episode: ***LstEpiShapRwrd***
- If $|$***(LstEpiShapRwrd - MovAvgShapRwrd)***$| >$ ***some threshold*** $\rightarrow$ Prompt last 3 image frames and ask for language instructions (prompt for human)
- **OR**, If number of episodes since last human input $>$ **some threshold** $\rightarrow$ Prompt last 3 image frames and ask for language instructions (prompt for human)

When the configurations of an environment change due to its stochastic nature, this directly affects the generation of shaping rewards. By measuring the deviation between a moving average of shaping rewards and the shaping rewards generated at the most recent episode, it allows the proposed framework to detect this change; hence prompt for human input. When all episodes generate small amount of shaping rewards (close to zero), the difference measured as above is not significant. In such cases, even if the environment changes, the measured deviation does not trigger for human input. Therefore as a solution, I define a episodic time interval so that the framework will prompt for human input once the agent passes a predefined number of episodes.

When the request for human input gets triggered, the framework pulls the three most recent imagery observations from the environment and pose them to a human expert who is knowledgeable about the environment. By looking at these three images, the human expert has the chance either to: 1) decide the observed trajectory of the agent is random (not helpful to reach the goal-state), and therefore, avoid giving any language instructions, or 2) provide natural language instructions to describe how the agent has behaved in the posed image sequence since he/she trusts that the observed behaviour helps the agent to reach the goal state quickly. The framework uses the information gathered through above two scenarios to construct new training data samples as:

- **Positive samples**:
  - (prompted **useful** image-sequence, human-provided instruction) $\rightarrow$ ***RELATED-CLASS***
- **Negative samples**:
  - (prompted **random** image-sequence, random instruction) $\rightarrow$ ***UNRELATED-CLASS***

Once the framework gathers a sufficient amount new samples, the I-LEARN model is retrained using the new data. The old training data is not used again since they do not represent the properties of the new environment configurations.

3.4 Training Data Preparation

For the initial training of the I-LEARN model, I generate sequences of image trajectories from the environment. The image trajectories that I generate are from one specific environment, treating it environment as a static environment. I do this because, before starting to work on stochastic environments, I want to make sure that the RL algorithm that I use will be able to find an optimal policy on the selected static environment (otherwise, the complete experimentation fails). Once that step is completed and when I want to test the proposed framework on a stochastic environment, the specific environment that I create the initial training data from is chosen as the first environment that the agent gets to interact with during the learning phrase. This is because the information that the I-LEARN model acquires during the initial training phrase will allow it to generate intermediate rewards during the initial stages of the experimentation phrase.

For the initial training data, I use a total of 142 image sequences where each image sequence includes three consecutive frames from the environment. These 142 image sequences corresponds to seven natural language instructions. Using these 142 image sequences and seven language descriptions, I create the initial training dataset of size 994 with 142 positive samples (correct image sequence and language description pairs) and 852 negative samples (incorrect image sequence and language description pairs).

4 Introducing language-based shaping rewards into the RL training process

The language based shaping rewards are introduced to the RL framework using the I-LEARN model. At any given time point that the RL agent lives in, I select the three most recent game frames from the history to form an image sequence. Afterwards, I query the most matching language instruction for selected image sequence based on the most recently formed training dataset. After receiving the language instruction, I couple it with the selected image sequence and send it through the I-LEARN model to predict whether the image sequence and language instruction are related or not.

The prediction that the I-LEARN model generates shows the probability of that image sequence being described by the selected language description. Recall that when creating the training dataset, I make sure to only include correct (related) image sequence and language description pairs as positive samples, and also, the trajectories observed in the positive samples always help the agent to reach the desired goal quickly. By leveraging this information, at each time-step, I translate the generated probability values into intermediate rewards using below criteria:

- **IF:** $P(RELATED) > 0.5 \rightarrow$ **intermediate reward** $= +1$, **ELSE: intermediate reward** $= 0$

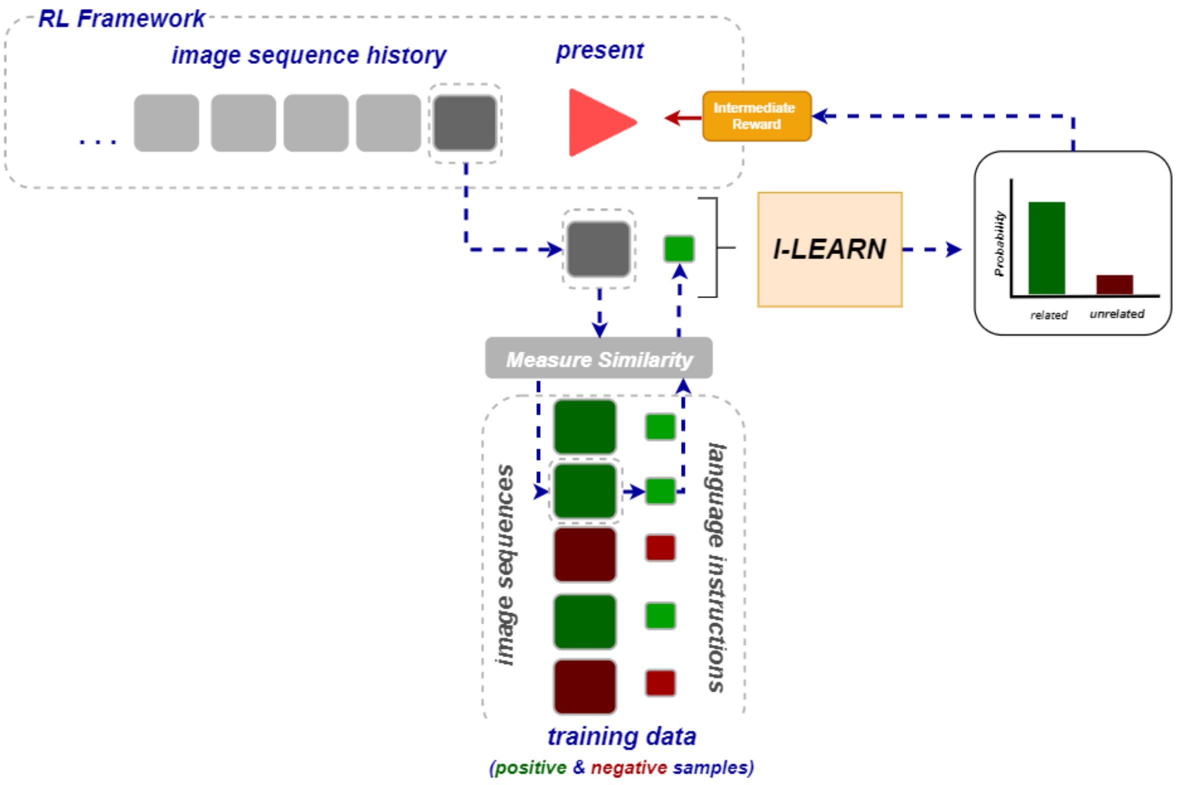


Figure 4: How language-based intermediate rewards are added to the RL training process

In order to understand how this approach works in an RL context, let's consider an example. Imagine that an input *(image sequence,language description)* pair to I-LEARN model outputs a $P(RELATED) > 0.5$. This means that, the image sequence and language instruction are related to each other and they belong to a positive training sample. Since positive training samples always correspond to efficient agent trajectories in the environment, providing an intermediate reward at this time point to the agent encourages it to revisit the same environment state in future. In this manner, the I-LEARN model generates intermediate rewards to incentivize the agent to visit environment states that are included in the positive training samples and discourages the agent to visit environment states of negative training samples.

Figure 4 provides an overview on how language-based shaping rewards are incorporated into the RL training process.

5 The Experiment

To put the proposed framework into test, I conduct an experiment on the 'simple crossing' environment included in the MiniGrid-World package (Boisvert et al., 2018). As seen in Figure 5, the objective for the agent is to navigate through a grid world and arrive at the goal state. At initialization, the grid world environment always place the agent and the goal state on two sides of a wall and there is only one opening that connects these two sides with each other. The agent has three actions: 'turn-left', 'turn-right' and 'move-forward'. The agent receives a reward of '+1' only when it reaches the goal state zero reward otherwise.

As the first step, I prepare the training data as described in Section 3.4. The full training dataset is split into two to construct the training and validation sets. After training the I-LEARN model with this data, I incorporate it into the RL training framework in order to test the proposed framework.

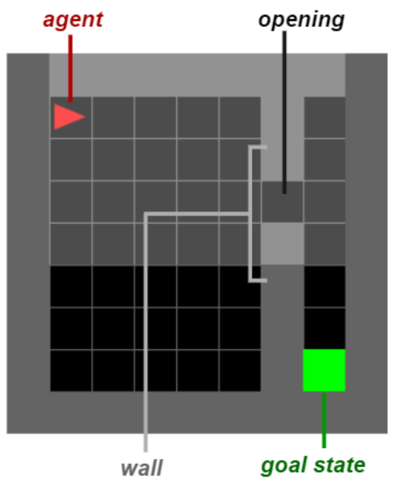


Figure 5: The simple crossing environment

To introduce stochastic environments, I program the simple crossing environment to randomize its configurations every time the agent ends an episode; therefore, different environment configurations are observed in each episode. Specifically, the positioning of the wall and the opening on the wall keep changing from episode to episode (Figure 1 illustrates this scenario).

With regards to the RL method, I apply the Q-learning algorithm in order to find an optimal policy on this environment. Since I am working with imagery inputs, I adopt the deep Q-network based approach Mnih et al. (2013) had used to train their agent on the Atari environment. First I train the agent without providing it language-based shaping rewards and let it find a policy that can accumulate a certain amount of rewards within a predefined amount of time-steps.

After completing the above step, I move forward to evaluate the proposed framework; I start generating language-based shaping rewards and introduce these intermediate rewards to the deep-Q learning network. I let this experiment run for the same number of training iterations that the pure deep-Q learning network trained, and then I measure the amount of rewards that the proposed network accumulates within the same predefined amount of time-steps that I tested the pure deep-Q learning network on.

6 Results of the Experiment

Unfortunately, I could not successfully complete the first step of the experiment, and therefore, the results are inconclusive. I let the pure deep-Q network run over 500,000 time steps with the stochastic environment setup (as described in Section 5) and observed that the agent could not score at least a single reward point in any of the episodes. Since finding a good policy in stochastic environments can be challenging, I made the test environment static by fixing it to have the same environment configuration, i.e., wall positioning and wall-opening, that was used in the initial training data. After this change, in about 300,000 time steps (1,000 episodes), the agent occasionally started to score rewards in multiple episodes. But when the learned deep-Q network was put into the test, the agent failed again as it did not score at least a single reward in any of the test episodes. Afterwards, I continued to train the deep-Q network for about a week (over 700,000 time steps and 2,400) episodes, but still the network did not learn a good enough policy to score any reward during the testing phrase.

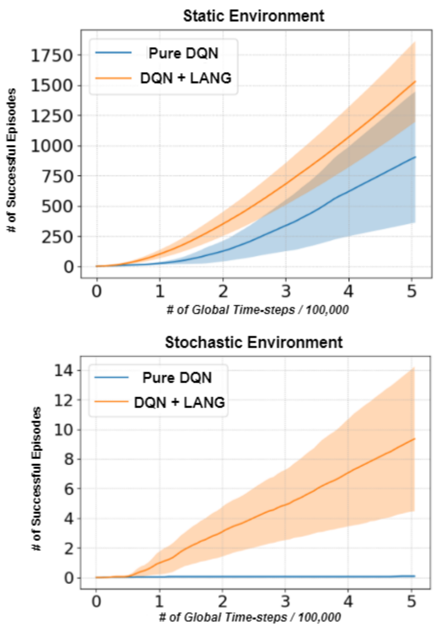


Figure 6: Comparing the performance of pure-DQN and language-aided DQN setups (fictitious; adopted from Goyal et al., 2019)

Since I could not complete the first part of the experiment, I did not apply the proposed framework into test. However, I did code almost all parts of the proposed framework and they are included in the zip file that I submit together with this report. Mentioned that, I do like to discuss a fictitious result that I was hoping to obtain by this experiment.

As seen in Figure 6, the language-aided DQN setup reaches a higher number of successful episodes when compared with the pure-DQN setup for the same number of time steps. This result applies to both static and stochastic environment setups. In static environments, the performance difference between pure-DQN and langauge-aided DQN setups grows at a constant rate whereas in stochastic environment, the performance of pure-DQN setup is extremely low compared with language-aided DQN setup, and therefore, the difference between them grows at an exponential rate.

7 Related Work

Even though language might not be essential to complete many RL tasks, it can be used to describe the structure of the task since contextual information can be used to define features, states and entities of an environment.

Natural language can be used to make the representation space more interpretable to humans as well as for RL agents. Andreas et al. (2018) investigate how natural language descriptions can be used to parameterize the policy of an agent that tries to navigate through a 2D environment. During the initial training stage, the agent follows a given set of trajectories conditioned on language instructions. They show that not only for RL tasks, even classification and text editing tasks can also be successfully be performed by models that have linguistic parameterizations.

Language can be used to construct priors about the structure of a task. Hu et al. (2019) examine how latent natural language instructions can be used to represent complex actions in hierarchical decision-making tasks. They show that agents who use natural language as a latent variables always outperform the ones that purely imitate human actions.

Fu et al. (2019) examine how language commands can be grounded to define reward functions using inverse RL methods. They use deep neural networks to denote reward functions which are designed by grounding natural language commands.They show that such neural network models learns reward functions that can be transferred into novel tasks. This shows that language representations can be use to efficiently structure the computations within models.

8 Conclusion

In this project, I propose a setup that can transform any RL method to be sample-efficient with the use of natural language instructions. The core idea of this framework is developed based on the work of Goyal et al. (2019). The Image-LanguagE-Aided Reward Network (I-LEARN) model and its online training process that I propose through this work can be successfully integrated with any RL method in order to make them sample efficient. While the previous work of Goyal et al., (2019) has only shown promising results on static environments, the setup that I propose demonstrates similar capability in improving the sample-efficiency of RL algorithms on both static and stochastic environments.

References

Andreas, J., Klein, D. and Levine, S., 2018, June. Learning with Latent Language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1 (Long Papers) (pp. 2166-2179).

Badia, A.P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D. and Blundell, C., 2020. Agent57: Outperforming the atari human benchmark. *arXiv preprint* arXiv:2003.13350.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C. and Józefowicz, R., 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint* arXiv:1912.06680.

Deng, J. et al., 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. pp. 248–255.

Du, S.S., Kakade, S.M., Wang, R. and Yang, L.F., 2019, September. Is a Good Representation Sufficient for Sample Efficient Reinforcement Learning?. In *International Conference on Learning Representations*.

Fu, J., Korattikara, A., Levine, S. and Guadarrama, S., 2018, September. From Language to Goals: Inverse Reinforcement Learning for Vision-Based Instruction Following. In *International Conference on Learning Representations*.

Gamrian, S. and Goldberg, Y., 2019, May. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *International Conference on Machine Learning* (pp. 2063-2072).

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Hu, H., Yarats, D., Gong, Q., Tian, Y. and Lewis, M., 2019. Hierarchical decision making by generating and following natural language instructions. In *Advances in neural information processing systems* (pp. 10025-10034).

Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprin*t arXiv:1412.6980.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint* arXiv:1907.11692.

Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint* arXiv:1412.6980.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. GitHub repository, 2018. URL https://github.com/maximecb/gym-minigrid

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *arXiv preprint* arXiv:1312.5602.

Ng, A.Y., Harada, D. and Russell, S., 1999, June. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML* (Vol. 99, pp. 278-287).

Prasoon Goyal, Scott Niekum, and Raymond J. Mooney., 2019. Using Natural Language for Reward Shaping in Reinforcement Learning. In *IJCAI*, 2019.

Pinsler, R., Akrour, R., Osa, T., Peters, J. and Neumann, G., 2018, May. Sample and feedback efficient hierarchical reinforcement learning from human preferences. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 596-601).

Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Rezende, D.J., Badia, A.P., Vinyals, O., Heess, N., Li, Y. and Pascanu, R., 2017. Imagination-augmented agents for deep reinforcement learning. In *Advances in neural information processing systems* (pp. 5690-5701).

Reimers, N. and Gurevych, I., 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint* arXiv:1908.10084.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587), pp.484-489.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Levine, S., 2018. Temporal difference models: Model-free deep rl for modelbased control. *In ICLR*, 2018.