Improving Reinforcement Learning Agent Training using Text-based Guidance: A Study using Commands in Dravidian Languages

Nikhil Chowdary Paleti, Sai Aravind Vadlapudi, Sai Aashish Menta, Sai Akshay Menta, Vishnu Vardhan Gorantla V N S L, Janakiram Chandu, Soman K P, and **Sachin Kumar S**
Amrita School of Artificial Intelligence, Coimbatore,
Amrita Vishwa Vidyapeetham, India.
s_sachinkumar@cb.amrita.edu, nikhil.paleti@outlook.com

Abstract

Reinforcement learning (RL) agents have achieved remarkable success in various domains, such as game-playing and protein structure prediction. However, most RL agents rely on exploration to find optimal solutions without explicit guidance. This paper proposes a methodology for training RL agents using text-based instructions in Dravidian Languages, including Telugu, Tamil, and Malayalam along with using the English language. The agents are trained in a modified Lunar Lander environment, where they must follow specific paths to successfully land the lander. The methodology involves collecting a dataset of human demonstrations and textual instructions, encoding the instructions into numerical representations using text-based embeddings, and training RL agents using state-of-the-art algorithms. The results demonstrate that the trained Soft Actor-Critic (SAC) agent can effectively understand and generalize instructions in different languages, outperforming other RL algorithms such as Proximal Policy Optimization (PPO) and Deep Deterministic Policy Gradient (DDPG).

1 Introduction

Reinforcement learning (RL) has developed by leaps and bounds in the past few years, there have been agents capable of beating world champions in games like go Silver et al. (2016), there have been agents capable of predicting protein structures Senior et al. (2020) and more recently there has also been an RL agent capable of optimizing computer code Mankowitz et al. (2023). However, most of the RL agents optimally find the best solution by themselves through exploration of the environment and there lacks a technique through which these agents can be guided so that we can control the path or trajectory that the agent takes while reaching the optimal solution.

There has been little work done to guide or help the RL agents get to the goal state through text-based instructions, especially in the Dravidian languages. The current literature Kaplan et al. (2017) and (Li et al., 2022) provide a basis for this approach where they have constructed a Bimodal embedding network to guide the RL agent on text-based instructions. However, existing literature doesn't compare various Reinforcement learning algorithms and they also don't consider the possibility of training the agents to understand the text instructions in multiple languages.

This paper aims to address this literature gap by proposing a methodology for training reinforcement learning agents in the lunar lander game using text-based embeddings in four languages: English, Telugu, Tamil, and Malayalam (K et al., 2021). By encoding instructions into meaningful numerical representations (Nagasai et al., 2021), the agents can effectively understand and respond to natural language instructions, leading to more immersive and intuitive user-agent interactions.

The proposed methodology for training RL agents with natural language guidance in the lunar lander game involves: 1) collecting a dataset of human demonstrations and textual instructions in multiple languages, 2) encoding the instructions into numerical representations using text-based embeddings, 3) employing RL algorithms with the embeddings as input to train the RL agent, optimizing for successful landings, 4) evaluating the effectiveness of the methodology for unseen paths. we show that

Third Workshop on Speech and Language Technologies for Dravidian Languages,
pages 33–42, Varna, Bulgaria, Sep 7, 2023.
https://doi.org/10.26615/978-954-452-085-4_005

the trained Soft Actor-Critic agent is capable of generalizing well to act according to the instruction given in any language.

The organization of the remainder of the paper is as follows: Section 2 details the related works. Section 3 provides a detailed description of the environment used for training the RL agents. Section 4 presents the proposed methodology and the results are discussed in section 5. Finally, we conclude in Section 6 while providing future directions for research.

2 Related Work

Existing work that combines reinforcement learning and natural language can be categorized into two tasks. In the first task, RL agents are trained in environments where the environment is rendered using only text descriptions, unlike the standard 2D or 3D environments that we traditionally see Jansen (2022). The second task focuses on helping or guiding Reinforcement learning agents through natural language, which the present work focuses on.

In Kaplan et al. (2017), a methodology was presented to use natural language to train a reinforcement learning agent to beat "MONTEZUMAS REVENGE", a game that standard RL agents like A3C fail to solve. To prepare a dataset, games were played manually by humans, and snapshots of the game state were taken. The snapshots along with text instructions were used to train CNN and RNN networks using cosine similarity loss to produce text and image embeddings of the game state. These embeddings were given to the RL agent as observations and a new reward function was designed which incorporated a similarity measure reward based on the text and image embeddings. The agent trained obtained a score of 3500 which outperformed the best model at that time by a score of 1000.

The authors in Li et al. (2022) used a similar methodology to (Kaplan et al., 2017) but they replaced the RNN-based network for text embedding with a pre-trained Bert model. This model supports giving instructions using synonyms of original instructions and the model would still be able to understand the instructions. Through their experiments, the authors say that the agent is able to get to the goal state 24 times out of 100 test tasks with a bert-distance model and 17 times with a bert-cosine model.

In recent times, work has focused on using Large Language Models (LLMs) as RL agents. In Wang et al. (2023), the authors have presented Voyager which uses LLM as an RL agent. The LLM harnesses the world language learned to generate consistent action plans or executable policies. The methodology presented in Voyager relies on using a Black-Box LLM (GPT-4) and skips any need to train or finetune the model. The methodology is comprised of three components: An automatic curriculum which is based on the goal of "discovering as many diverse things as possible". A skill library to store and retrieve code generated by the LLM based on embeddings of the generated function descriptions. An iterative prompting mechanism that generates code for various tasks. Finally, there is also a self-evaluation component where the LLM acts as a critic to evaluate the generated function. Though the voyager agent performs well when compared with other similar agents, it, however, has its own downsides like significant cost incurred through GPT-4 API, Hallucinations, and inaccuracies in generating a new skill. Most of the drawbacks can be improved by employing a multimodal LLM that can benefit from both text and visual data or finetuning an LLM with knowledge about various aspects of the environment to reduce inaccuracies and hallucinations. We still require advancements in research in the domain to tackle other problems like the high inference time of LLMs and computational costs of finetuning.

Existing literature did not explore the possibility of using multiple languages to guide Reinforcement learning agents, and there has not been a comparative study on various reinforcement learning algorithms for natural language-guided learning. The present work aims to tackle these research gaps by presenting a methodology to train Reinforcement learning agents through text guidance in various languages including Dravidian languages and also perform a comparative study on the employability of various state-of-the-art Reinforcement learning algorithms for text guidance.

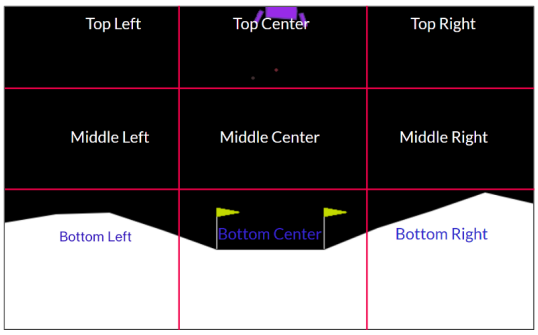

Figure 1: Modified Lunar Lander Environment

3 Environment

The environment used in this study is a modified version of the Lunar Lander environment which is a part of the Box2D environments from the open-source Python library "Gymnasium". The environment is a typical rocket trajectory optimization problem and the goal of the environment is to land the lander on the landing pad. The environment supports both continuous state configuration and discrete state configuration. The continuous state version of the original environment is considered in this study which contains eight observations: the coordinates of the lander in x & y, its linear velocities in x & y, its angle, its angular velocity, two booleans that represent whether each leg is in contact with the ground or not and the action space consists of two continuous actions: The first coordinate of an action determines the throttle of the main engine, while the second coordinate specifies the throttle of the lateral boosters.

The original environment is partitioned into 9 regions as shown in Figure 1 to construct the modified environment. The new goal of the environment is to trace the lander along the path that is given (An example path can be: Top center, Top right, Middle Right, and Bottom center) and finally land on the landing pad. A random path from the preconfigured list of paths is automatically assigned by the environment every time the environment is reset. The path given to the environment can contain locations of the environment in the following languages: English, Telugu, Tamil, and Malayalam. Refer to Appendix A for a detailed list of the paths that can be generated.

A newly shaped reward function is defined through which the lander receives a reward of

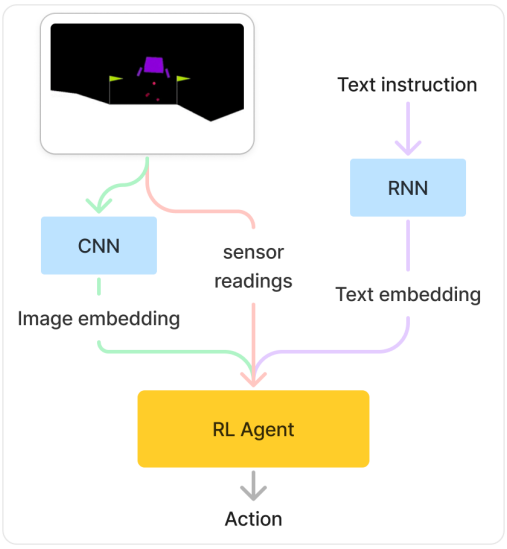

Figure 2: High-level overview of methodology

0 if it moves closer to the target location along the path, -1 if it moves away or deviates from the path, a negative reward based on the angle tilted if its more than 45 degrees, 100 for each leg in contact with the landing pad, and -20 if it tries to land without going along the path. This will effectively punish the agent if it is not following the trajectory given and will reward the agent if it follows along the trajectory and successfully lands. With the mentioned changes, the modified environment still has 8 observations and 2 actions but there is an added feature to generate to path and a new shaped reward function that we employ.

4 Methodology

A high-level overview of the methodology is presented in Figure 2. The 8 sensor observations from the lander along with image embedding of the current state and text embedding of the current target location will go in as input to the Reinforcement learning Agent which gives an action to be taken. The text instruction to the RNN can be in the following languages: English, Telugu, Tamil, and Malayalam. The training of these networks can be split into two stages, in the first stage the embedding networks CNN and RNN are trained together (Kumar et al., 2015) using a cosine similarity loss, and then in the second stage, the RL agent is trained.

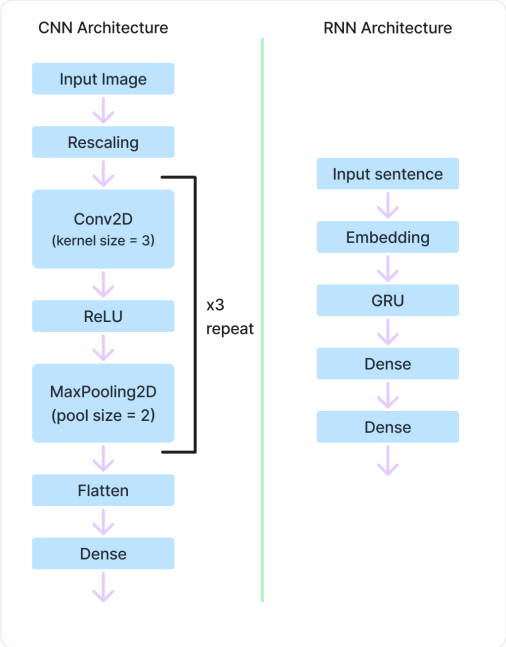

Figure 3: Left: Architecture of CNN network Right: Architecture of RNN network

4.1 Bimodal Embedding networks

To capture the current state of the environment a CNN-based network is used and to capture the text instruction an RNN-based network is used. Both of these networks are trained together based on cosine embedding loss. Each of the networks outputs an embedding of length 10. The architecture of the two networks is presented in Figure 3.

4.1.1 Dataset

The dataset used to train the embedding network consists of image and text pairs.

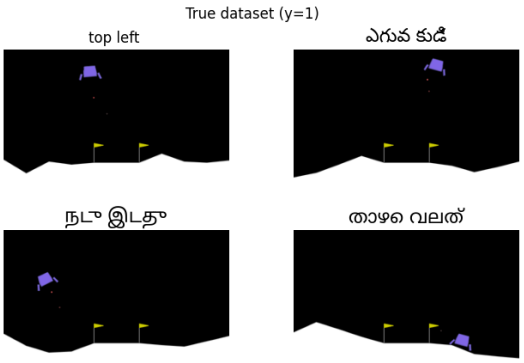

Figure 4: True class of embeddings dataset

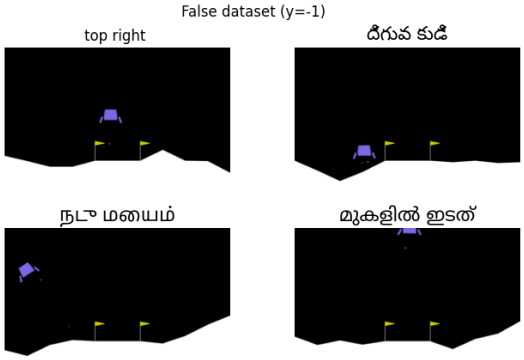

Figure 5: False class of embeddings dataset

The game was played manually and at each timestep, a snapshot of the game state was saved. The snapshots were then manually assigned a text description based on the 9 regions in the modified environment. The dataset consists of two classes: The true class (y = 1) is shown in Figure 4 and represents image, text pairs where text is an accurate description of the image. The false class (y = -1) shown in Figure 5 represents the image, text pairs where text is a false description of the image.

There are 876 image and text pairs for a language and in total, across four languages there are a total of 3,504 pairs. The images are 200px in height, 300px in width and have RGB channels. The text description consists of two words representing the position of the rover in the image according to the regions presented in Figure 1.

4.2 Reinforcement Learning

The Reinforcement Learning (RL) agent is responsible for determining the best action possible given the current environment state. For Guided reinforcement learning, along with the sensor information, the two embedding vectors are also taken in by the algorithm as input. There is a wide spectrum of algorithms (Sreedevi and Rao, 2019) that have the capacity to learn in continuous observation space, among them Deep Deterministic Policy Gradients (DDPG), Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) have been considered in this study. The implementations of the algorithms are taken from the open source python library “StableBaselines3” presented by Raffin et al. (2021).

4.2.1 PPO

The authors in Schulman et al. (2017), have presented a new family of policy gradient methods that optimize a "surrogate" by performing a stochastic gradient ascent. In contrast to the standard policy gradient when an update happens per data sample, a novel objective function is proposed that can perform minibatch updates. It has its roots in TRPO but is much simpler to implement. The authors show that this new algorithm strikes a balance between sample complexity, simplicity, and wall time.

4.2.2 DDPG

The authors, Lillicrap et al. (2019), have adapted the technique based on Deep Q-Learning technique for the continuous action space domain. DDPG is a model-free algorithm that can solve more than 20 simulated physics tasks using the same neural network architecture and hyperparameters. To solve the exploration problem in continuous action spaces, the authors have used noise generated using the Ornstein-Uhlenbeck process. The same was adapted for our environment.

4.2.3 SAC

Standard model-free RL algorithms suffer from high sample complexity and brittle convergence properties which requires careful hyperparameter tuning. In Haarnoja et al. (2018), the authors propose an off-policy actor-critic algorithm that maximizes the expected reward while also maximizing entropy. It tries to achieve the goal while also being as random as possible. This feature of the SAC algorithm enables the agent to find out optimal solutions even when the environment is changing or even when there is an obstacle in the standard optimal path, the agent will learn to maneuver around it.

4.3 Training

The embedding networks were trained on a standard Google Colab instance with a T4 GPU. The networks were trained using a batch size of 64 for 600 epochs using Adam optimizer with a learning rate 1e-4. The training loss is presented in Figure 6.

The Reinforcement learning agents were put to training on a lambda labs instance equipped with an Nvidia A10 GPU, which has a compute capability of 8.6, 30vCPUs, 200GiB RAM, and 1.4 TiB SSD. Default Stable baselines 3 parameters were used to train the models as they have been already tuned to work with diverse sets of environments.

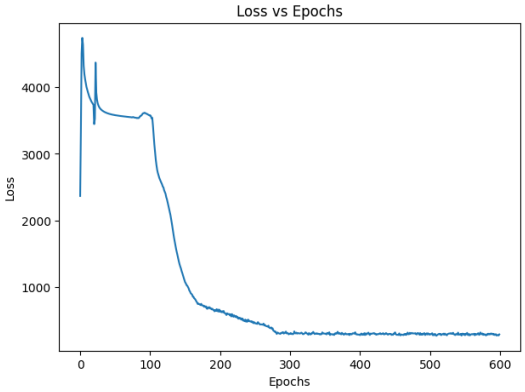

Figure 6: Embeddings networks loss

Algorithm	Telugu Language	Multi Language
DDPG	30 Million	30 Million
PPO	40 Million	40 Million
SAC	30 Million	50 Million

Table 1: Training steps for RL algorithms

Two experiments were run using RL algorithms, In the first experiment the agent received instructions only in the Telugu language, and in the second experiment, the agent received instructions in all 4 languages (English, Telugu, Tamil, and Malayalam). Table 1 describes various training step lengths that were used to train.

The training results of the Multi-Language SAC agent which was trained for 50 Million steps are presented in Figure 7. The agent is able to achieve a score close to 200 with an episode length close to 500 after training.

5 Results and discussion

Figure 8 shows a visualization of the input text embeddings plotted using UMAP McInnes et al. (2020). The plot depicts similar instructions being plotted together in the low dimensional space indicating that our embedding network has learned to build connections among the vocabulary used for training it. Table 2 shows the top 5 cosine similarity values computed between the embedding of ఎగువ

Figure 7: Episode length and reward for Multi Language SAC

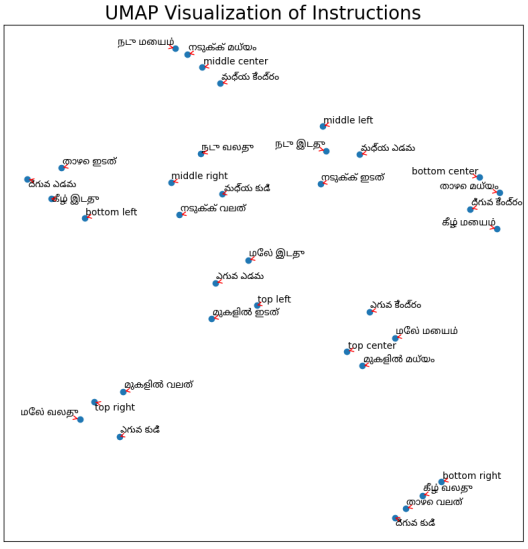

Figure 8: UMAP visualization of input instructions

Instruction	Cosine Similarity to ఎగువ ఎడమ (top left)
మధ్య ఎడమ (middle left)	0.98
ఎగువ కుడి (top right)	0.06
దిగువ ఎడమ (bottom left)	0.05
ఎగువ కేంద్రం (top center)	0.02
మధ్య కుడి (middle right)	-0.08

Table 2: Cosine similarity of text embeddings

ఎడమ which means "top left" to other Telugu text instructions. The similarity values further strengthen the claim about the model understanding the underlying patterns in the input text.

As discussed before, 9 unique paths were used for training and a total of 36 paths are made from the 9 paths by translating them to various Dravidian languages. Appendix A provides an overview of the paths used in training. For the first experiment, only 9 paths in the Telugu language were considered and 3 RL agents were trained. The SAC agent obtained an average reward of 192.94, the PPO agent obtained an average reward of -184.53 and the DDPG agent obtained an average reward of -344.46.

In order to test the ability of RL agents to generalize to unseen paths, a few experiments are conducted as presented in Table 3. The paths included transitions that the agent never got to see during training. The unseen paths included transitions like going from middle center to middle right which tests the agent on its capability to fly the lander in the opposite direction and also transitions like going from middle left to top left which tests the agent's capability to fly up in the opposite direction. It should be noted that the agent was never instructed to fly in the opposite direction during training.

The results in Table 3 demonstrate the capability of SAC to generalize well to unseen instructions. Across all the experiments it has maintained a positive reward while the PPO and DDPG agents struggled to perform well.

S.No	Path	SAC Average Reward	PPO Average Reward	DDPG Average Reward
1	ఎగువ కేంద్రం, మధ్య కుడి, ఎగువ కుడి, మధ్య కేంద్రం, దిగువ కేంద్రం (top center, middle right, top right middle center, bottom center)	181	-418.5	-180.5
2	ఎగువ కేంద్రం, మధ్య కుడి, మధ్య కేంద్రం, దిగువ కేంద్రం (top center, middle right, middle center, bottom center)	187	-95.4	-453.7
3	ఎగువ కేంద్రం, ఎగువ కుడి, ఎగువ కేంద్రం, ఎగువ ఎడమ, మధ్య కేంద్రం, దిగువ కేంద్రం (top center, top right, top center, top left, middle center, bottom center)	182	-299.7	-668.2
4	ఎగువ కేంద్రం, ఎగువ ఎడమ, ఎగువ కేంద్రం, ఎగువ కుడి, మధ్య కుడి, మధ్య కేంద్రం, దిగువ కేంద్రం (top center, top left, top center, top right, middle right, middle center, bottom center)	185	-265.7	-441.9
5	ఎగువ కేంద్రం, మధ్య కుడి, మధ్య కేంద్రం, మధ్య ఎడమ, దిగువ కేంద్రం (top center, middle right, middle center, middle left, bottom center)	181	-87.9	-558.2

Table 3: Rewards of the unseen paths tested on RL agents trained on Telugu language instructions.

The second set of experiments consisted of training the RL agents using instructions from all 4 languages: English, Telugu, Tamil, and Malayalam. The total number of instructions considered is 36 (9 from each language). The trained SAC agent obtained an average reward of 187.19, the PPO agent obtained -262.8, and the DDPG agent obtained a -419.37 reward. We can observe that the performance of the Agent trained on multi-language instructions is lower compared with the agent trained on a single language. Though the SAC agent received 20M additional training steps for multiple languages, it obtained less average reward than a single language agent. PPO and DDPG showed similar performance to single language agents, failing to converge.

For the agents trained on instructions from multiple languages first a test was performed to evaluate the agents on using combinations of languages. Presented in Table 4, for multi-language combination paths, the SAC agent obtained an average score of 186.9 while PPO and DDPG obtained -312 and -326.8 respectively. These results indicate that the SAC agent is not confusing among languages and is able to reach the goal state successfully even if the input consisted of instructions from multiple languages.

The final set of tests as presented in Table 5 consisted of evaluating the RL agent train on multi-language instructions on unseen paths. These paths again consisted of transitions that were never seen during training and this time the paths included instructions from multiple languages. The results again show that SAC is able to generalize well to the unseen paths compared to PPO and DDPG. These tests can be viewed by accessing https://www.youtube.com/watch?v=oxADf4oV74w

6 Conclusion and Future Works

We have successfully demonstrated a methodology to guide the reinforcement learning agents through text instructions in multiple Dravidian languages. PPO, DDPG, and SAC algorithms were put to use for the task and results showed that SAC generalized well even for unseen paths. When unseen instructions from a mix of Dravidian Languages were given

S.No	Path	SAC Average Reward	PPO Average Reward	DDPG Average Reward
1	top center, మధ్య కేంద్రం, கீழ் மையம் (top center, middle center, bottom center)	190.6	-185.6	-453.1
2	முகளில் மധ്യം, நடு மையம், దిగువ కేంద్రం (top center, middle center, bottom center)	195.1	-263.5	-159.1
3	மேல் மையம், முகளில் വലത് middle right, దిగువ కేంద్రం (top center, top right, middle right, bottom center)	181.5	-423.6	-218.8
4	எగువ కేంద్రం, top left நடு இடது, താഴെ മധ്യം (top center, top left, middle left, bottom center)	180.7	-375.4	-476.4

Table 4: Evaluating Agents on paths constructed from multiple languages

S.No	Path	SAC Average Reward	PPO Average Reward	DDPG Average Reward
1	top center, మధ్య ఎడమ, மேல் இடது, முகளில் മധ്യം, top right, നടുക്ക് മധ്യം, താഴെ മധ്യം (top center, middle left, top left, top center, top right, middle center, bottom center)	180	-710.4	-230.2
2	முகளில் மധ்யం, நடு இடது, எగువ ఎడమ top center, நடு மையம், దిగువ కేంద్రం (top center, middle left, top left, top center, middle center, bottom center)	165	-617.4	-252.9
3	எగువ కేంద్రం, middle left, மேல் இடது, முகளில் மധ്യം, நடு மையம், దిగువ కేంద్రం (top center, middle left, top left, top center, middle center, bottom center)	148	-564.5	-198.1
4	மேல் மையம், எగువ ఎడమ, முகளில் மധ്യం, top right, നടുക്ക് വലത് మధ్య కేంద్రం, bottom center (top center, top left, top center, top right, middle right, middle center, bottom center)	-69	-366.7	-382.3
5	எగువ కేంద్రం, നടുക്ക് വലത്, நடு மையம், நடு இடது, bottom center (top center, middle right, middle center, middle left, bottom center)	175.3	-188.6	-278.3

Table 5: Rewards of the unseen paths tested on RL agents trained on Multi-language instructions.

to the SAC agent, it obtained an average reward of 119.8 while an SAC agent trained on Telugu language instructions alone obtained an average reward of 183.2 for unseen Telugu instructions. The correctness of the embeddings was also verified through the UMAP plot and cosine similarity.

The work presented in this paper can be extended by using various architectures for embedding networks and making them more efficient. Another direction that can be explored is the use of MultiModal Multilanguage Large Language Models which are capable of understanding images and text in multiple languages, providing access to good computational infrastructure one can try training these LLMs to understand Dravidian languages and also act as Reinforcement Learning agents.

References

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

Peter Jansen. 2022. A systematic survey of text worlds as embodied natural language environments. In *Proceedings of the 3rd Wordplay: When Language Meets Games Workshop (Wordplay 2022)*, pages 1–15, Seattle, United States. Association for Computational Linguistics.

Sreelakshmi K, Premjith B, and Soman Kp. 2021. Amrita_CEN_NLP@DravidianLangTech-EACL2021: Deep learning-based offensive language identification in Malayalam, Tamil and Kannada. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 249–254, Kyiv. Association for Computational Linguistics.

Russell Kaplan, Christopher Sauer, and Alexander Sosa. 2017. Beating atari with natural language guided reinforcement learning.

S. Sachin Kumar, B. Premjith, M. Anand Kumar, and K. P. Soman. 2015. Amrita_cen-nlp@sail2015: Sentiment analysis in indian language using regularized least square approach with randomized feature learning. In *Mining Intelligence and Knowledge Exploration*, pages 671–683, Cham. Springer International Publishing.

Xin Li, Yu Zhang, Junren Luo, and Yifeng Liu. 2022. Pre-trained bert for natural language guided reinforcement learning in atari game. In *2022 34th Chinese Control and Decision Conference (CCDC)*, pages 5119–5124.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning.

Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, et al. 2023. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263.

Leland McInnes, John Healy, and James Melville. 2020. Umap: Uniform manifold approximation and projection for dimension reduction.

L. S. Nagasai, V. J. Sriprasath, V. V. SajithVariyar, V. Sowmya, K. Aniketh, T. V. Sarath, and K. P. Soman. 2021. Electric vehicle steering design and automated control using cnn and reinforcement learning. In *Soft Computing and Signal Processing*, pages 513–523, Singapore. Springer Singapore.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. 2020. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

A. G. Sreedevi and Thipparaju Rama Rao. 2019. Reinforcement learning algorithm for 5g indoor devicetodevice communications. *Transactions on Emerging Telecommunications Technologies*, 30.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models.

Appendix

A Preconfigured Paths

The modified environment has a set of 36 preconfigured paths from which one is randomly assigned every time the environment is reset. There are a total of 9 unique paths to go from the start location to the landing pad which is listed in Table 6.

S.no	English Instruction
1	top center, middle center, bottom center
2	top center, top right, middle right, middle center, bottom center
3	top center, top right, middle center, bottom center
4	top center, top left, middle left, middle center, bottom center
5	top center, middle left, bottom center
6	top center, middle right, bottom center
7	top center, top left, middle left, bottom center
8	top center, top left, middle center bottom center
9	top center, top right, middle right bottom center

Table 6: Instructions used for training

All the other paths are translations of these 9 paths in Dravidian languages: Telugu, Tamil, and Malayalam. The help of google translate has been taken to get the translations in various Dravidian languages. Table 7 lists out the translations of locations in the languages considered.

S.no	English	Translation
1	top center	Telugu: ఎగువ కేంద్రం Tamil: மேல் மையம் Malayalam: മുകളിൽ മധ്യം
2	top left	Telugu: ఎగువ ఎడమ Tamil: மேல் இடது Malayalam: മുകളിൽ ഇടത്
3	top right	Telugu: ఎగువ కుడి Tamil: மேல் வலது Malayalam: മുകളിൽ വലത്
4	middle center	Telugu: మధ్య కేంద్రం Tamil: நடு மையம் Malayalam: നടുക്ക് മധ്യം
5	middle left	Telugu: మధ్య ఎడమ Tamil: நடு இடது Malayalam: നടുക്ക് ഇടത്
6	middle right	Telugu: మధ్య కుడి Tamil: நடு வலது Malayalam: നടുക്ക് വലത്
7	bottom center	Telugu: దిగువ కేంద్రం Tamil: கீழ் மையம் Malayalam: താഴെ മധ്യം
8	bottom left	Telugu: దిగువ ఎడమ Tamil: கீழ் இடது Malayalam: താഴെ ഇടത്
9	bottom right	Telugu: దిగువ కుడి Tamil: கீழ் வலது Malayalam: താഴെ വലത്

Table 7: Translation of locations