Automatic Curriculum Learning (AutoCL) for Hard Exploration Environments: Rubik's Cube as an Example

Kamel BROUTHEN, Amir ALMAMMA, Massi-Nissa ABBOUD, Azzedine Idir AITSAID

Introduction

In the realm of Reinforcement Learning, while significant improvements have been realized, issues with slow convergence and high sample complexity show that better learning methods are necessary. Traditional Curriculum Learning (CL) introduces a structured progression for agents but is hindered by the manual intricacies of task sequence design. This research delves into **Automatic Curriculum Learning (AutoCL)**[4] as a pivotal solution, seeking to automate curriculum design and optimize skill progression. Through AutoCL, we aim to enhance the efficiency of RL agents, harnessing computational methodologies to dynamically adjust learning trajectories based on evolving agent parameters and task demands. In this project we compared the efficiency of training a PPO agent on a Rubic's Cube environment [2]using CL and AutoCL.

Background

Jumanji[2] Rubik's Cube Environment: In our experimentation we used a a Jax JIT-compatible Rubik's Cube environment on a 3x3x3 size set-up. Reward function is binary as shown in the figure below:

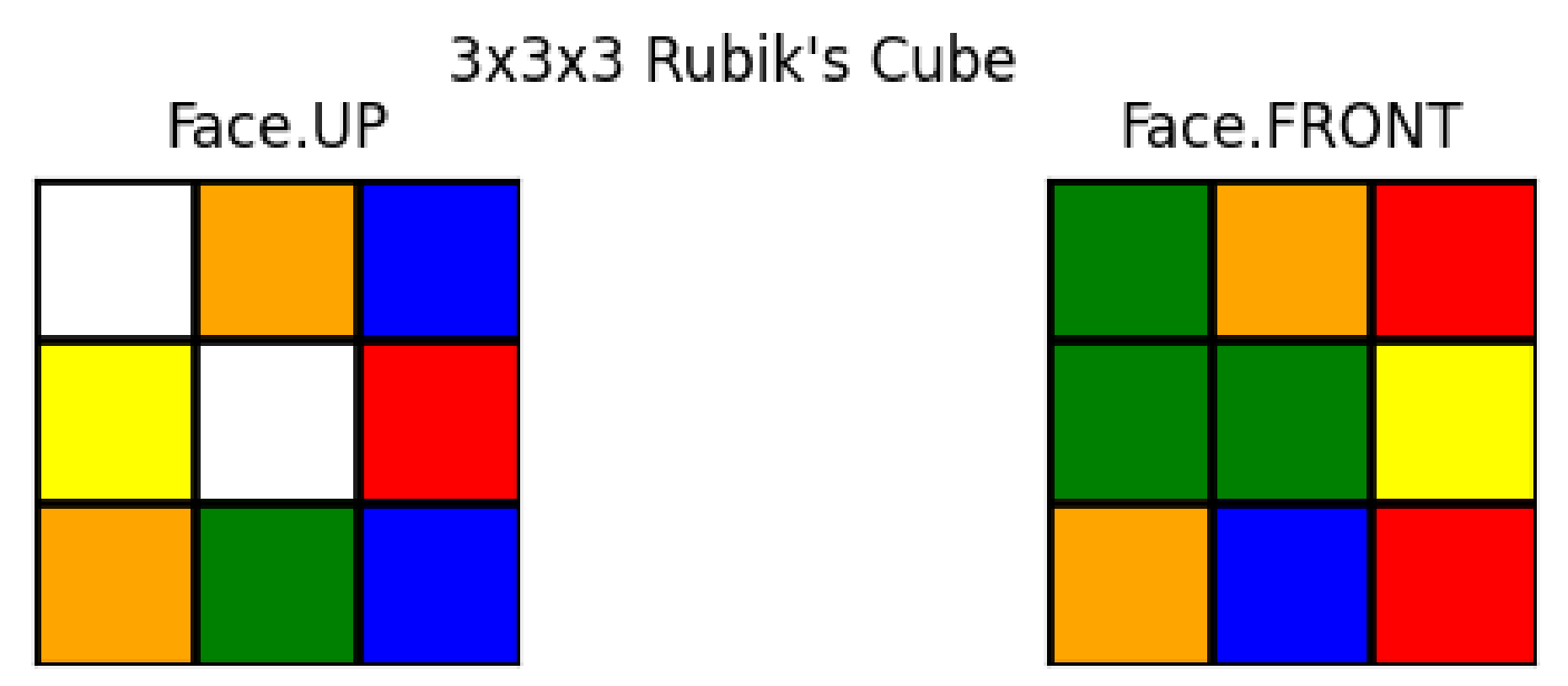

Figure 1. 2 Faces (Front and UP) of a 3x3x3 Rubik's Cube Environment in Jumanji [?]

Curriculum Learning (CL)[1] is grounded in the principle of sequencing learning tasks to ensure progressive learning. Just as in human education where we progress from simpler to more complex topics, Curriculum Learning presents models with easier tasks initially, gradually increasing the complexity. This ordered introduction can lead to faster convergence and potentially better generalization in neural networks. The main challenge lies in designing an appropriate curriculum, as determining the sequence and complexity of tasks isn't always straightforward.

Auto Curriculm Learning (AutoCL) for DRL is a family of mechanisms that automatically adapt the distribution of training data by learning to adjust the selection of learning situations to the capabilities of DRL agents[4]. It helps in efficiently improving performance on specific tasks, guiding the learning process from easy to hard tasks, training agents that can generalize across multiple situations, including transitioning from simulations to real-world scenarios, and organizing open-ended exploration for diverse behaviors. The figure below show how the teacher agent can act on task MDPs to generate tasks.

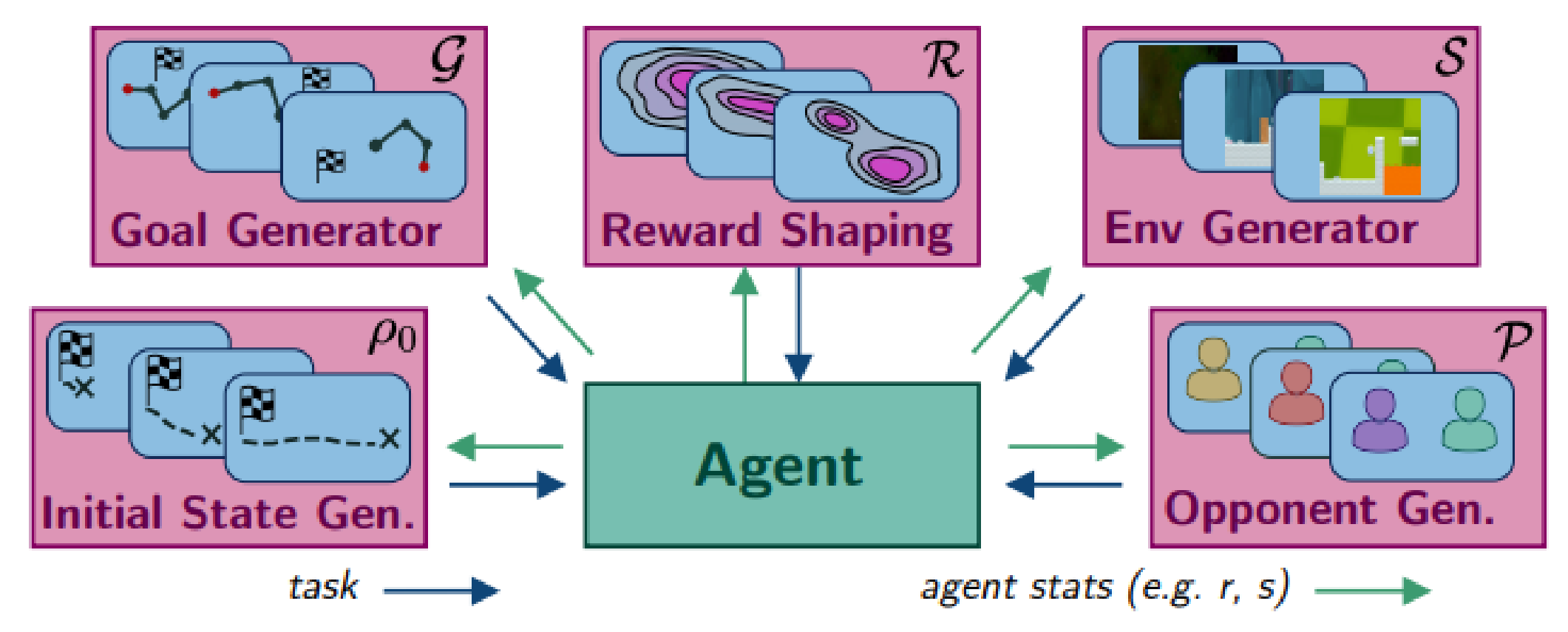

Figure 2. ACL Task Generation for Data Collection [4]

Motivation

Training an agent to solve the Rubik's cube presents a significant challenge due to the sparse rewards and vast search space associated with it. When using a straightforward approach with a PPO agent [5], we found that the learning didn't progress effectively. Our hypothesis is that the key to addressing this problem lies in Curriculum Learning (CL). Through the use of CL and eventually Automatic Curriculum Learning (AutoCL), we aim to demonstrate that an agent can be trained more effectively, even in environments with sparse rewards and large search spaces.

Implementation

We have created an environment for the student, designed to interact with a specified teacher[3]. Within this environment, the reward system is based on the absolute difference between the current mean reward and the previous mean reward for a given task. The observation provided to the agent represents the mean reward of the task[3]. The environment also maintains a dynamic record of training data and task-specific reward buffers. The Diagram below illustrates the teacher/student interaction during training.

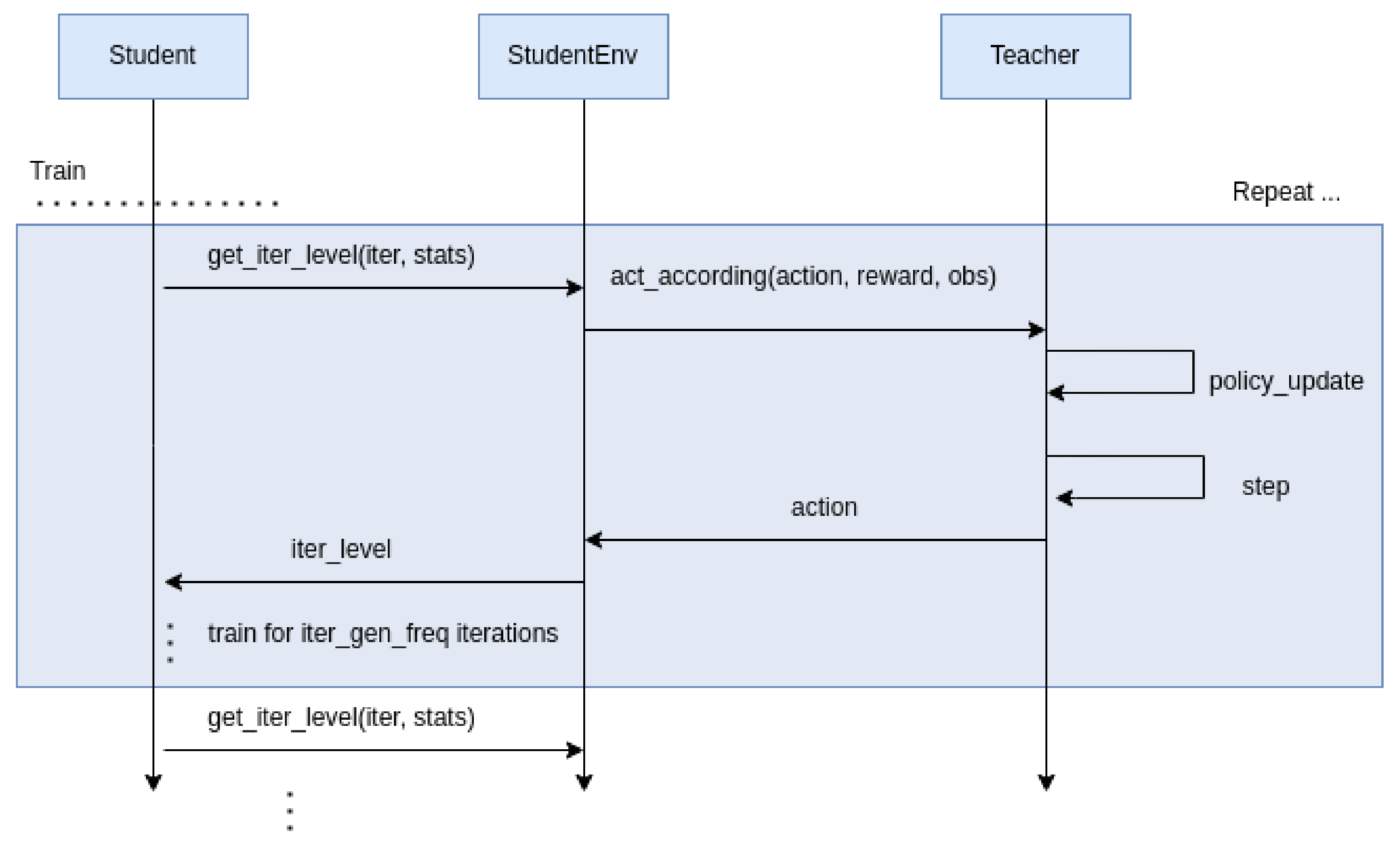
Student
StudentEnv
Teacher
Train
Repeat ...
get_iter_level(iter, stats)
act_according(action, reward, obs)
policy_update
step
action
iter_level
train for iter_gen_freq iterations
get_iter_level(iter, stats)

Figure 3. Teacher-Student interactions on Training

Experimentation

CL: We designed a standard curriculum where the level of difficulty increases from 1 to 5, we changed the number of training iterations in each level while observing the reward improvement.

Auto-CL: We implemented an Online Teacher algorithm with **ϵ-decay-greedy** policy, then we tested the performance on the algorithm with different **ϵ-greedy** values, and gradually increasing the number of training iterations from 300 up to 6000, while observing the reward improvements, the experiments showed better results an ϵ value decreasing from 0.3 to 0.0 at the end of training and with 1500 training iterations.

Comparative Results

Curriculum Learning (CL) demonstrates a distinct advantage over the conventional Proximal Policy Optimization (PPO) method of problem-solving.

Moreover, when subjected to 1500 iterations, AutoCL with Online Algorithm in epsilon decay-greedy policy exhibits superior performance in comparison to both CL and direct resolving. Accompanying visual aids elucidate the progression of reward during both the training and evaluative stages. An analysis of the probability distribution reveals an intriguing pattern: in the initial phase of training, there is a pronounced inclination for the teacher to sample from simpler tasks. As training progresses, however, the teacher's preference shifts towards more intricate challenges.

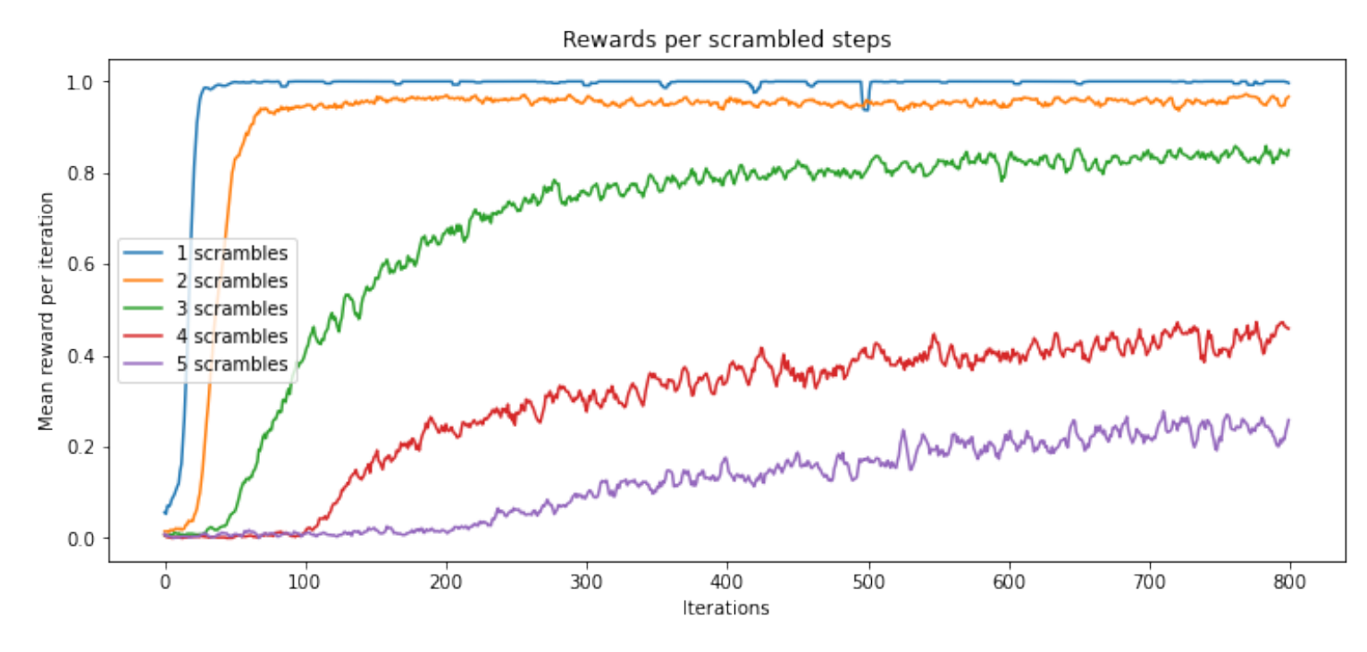

Figure 4. Reward progress of naive learning on 1 to 5 scrambles of Rubik's Cube in training

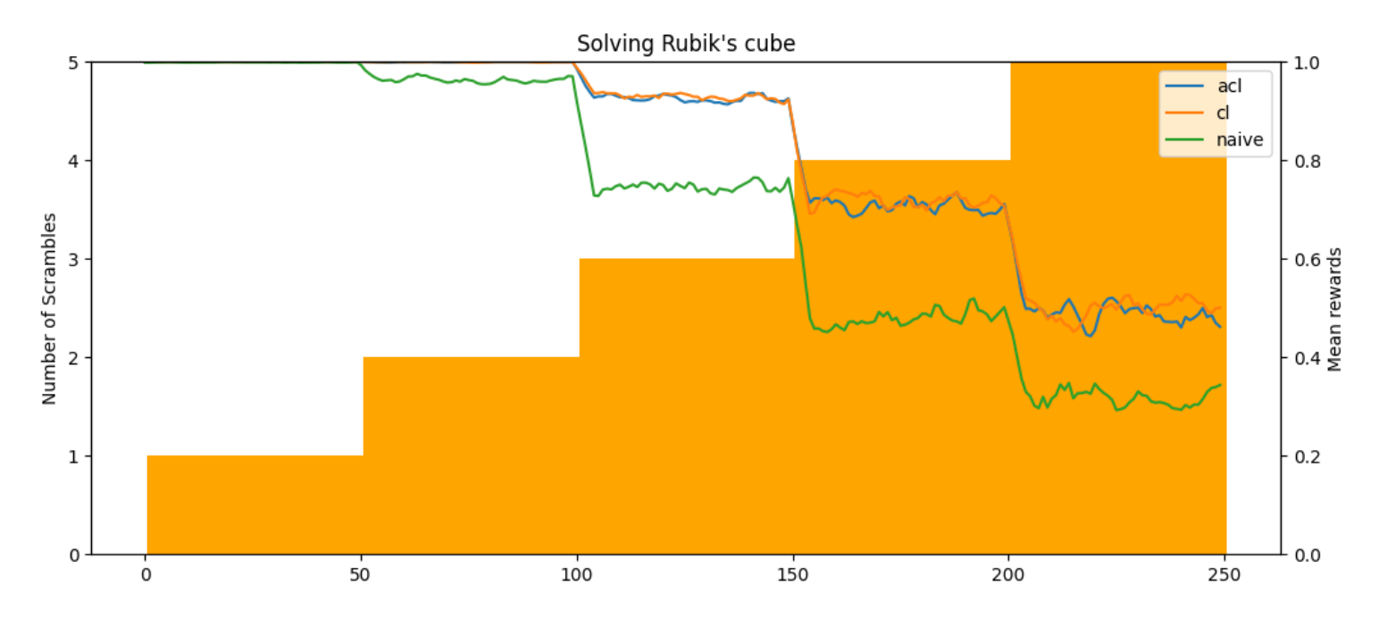

Figure 5. Comparing Auto-CL, CL and naive learning on 50 iterations for each level

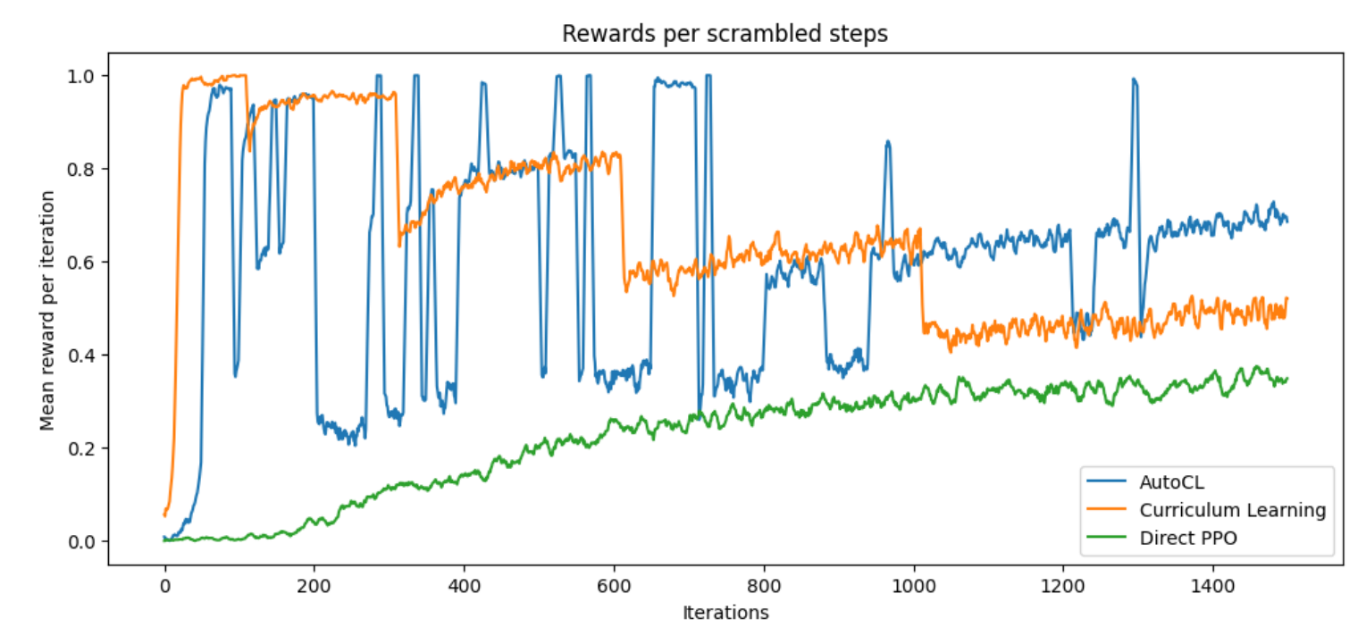

Figure 6. Reward progress of Auto-CL, CL and naive learning on 1500 training iterations

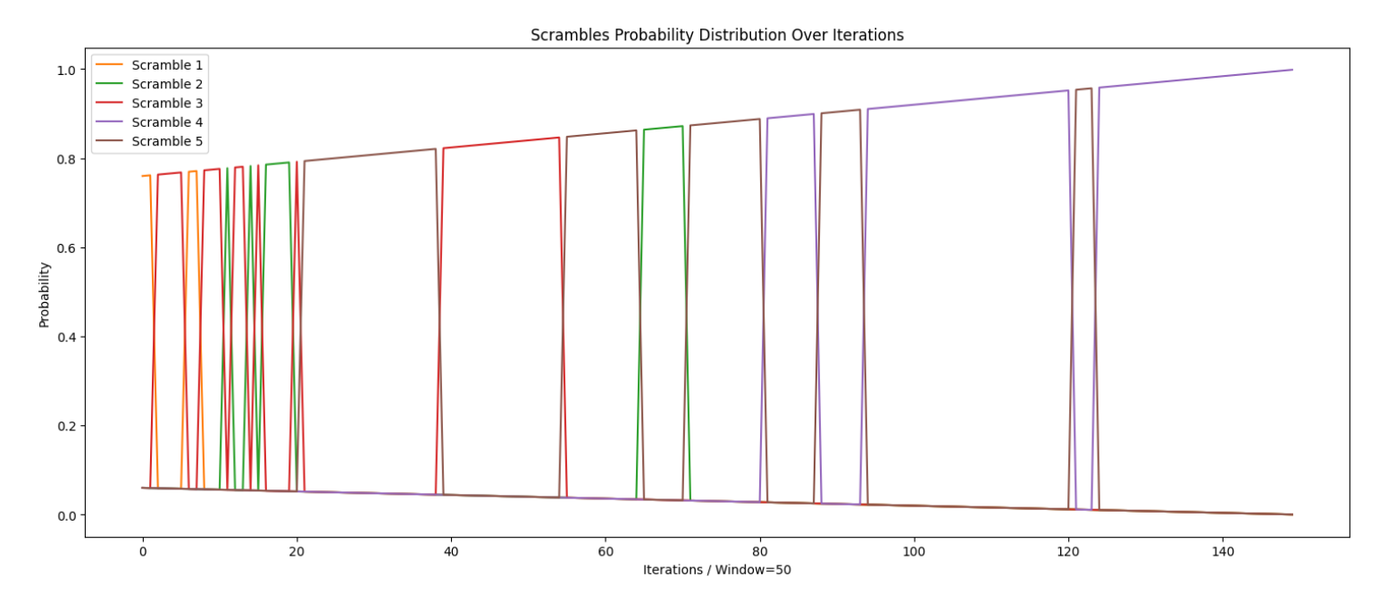

Figure 7. Sampling probability of different levels during Auto-CL training

Conclusion

In the course of our research, we concentrated our experimental adjustments primarily on the variations in the number of scrambles. Nevertheless, our methodology demonstrated superior performance when compared to both Curriculum Learning (CL) and naive learning techniques. These encouraging findings suggest the potential for further exploration into the optimal selection of teacher policy, particularly when augmented by greater computational resources.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[2] Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Vincent Coyette, Paul Duckworth, Laurence I. Midgley, Tristan Kalloniatis, Sasha Abramowitz, Cemlyn N. Waters, Andries P. Smit, Nathan Grinsztajn, Ulrich A. Mbou Sob, Omayma Mahjoub, Elshadai Tegegn, Mohamed A. Mimouni, Raphael Boige, Ruan de Kock, Daniel Furelos-Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. Jumanji: a diverse suite of scalable reinforcement learning environments in jax, 2023.

[3] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[4] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

This work is made possible by the support of School of AI Algiers, Ecole Nationale Supérieure d'Informatique, Algeria