Minimalist Vision: Optimizing Information Extraction for RL using Computer Vision

Anirudh Singh Bhakar
anirudh21102@iiitnr.edu.in
Data Science and Artificial Intelligence
International Institute of Information Technology
Naya Raipur

Priyanshu Senabaya Deori
priyanshud21102@iiitnr.edu.in
Data Science and Artificial Intelligence
International Institute of Information Technology
Naya Raipur

***Abstract*—Deep Reinforcement Learning (DRL) has emerged as a powerful tool for solving complex sequential decision-making problems. However, traditional DRL methods often suffer from the curse of dimensionality, making them computationally expensive and time-consuming to train. To address this challenge, we propose a novel approach that efficiently combines DRL with autoencoders to encode high-dimensional observation spaces into a lower-dimensional latent space. This latent space representation is then used as input to the DRL agent, significantly reducing computational complexity and improving training efficiency. Our experimental results demonstrate that our proposed approach outperforms traditional DRL methods in terms of both training time and performance.**

Keywords: Deep Reinforcement Learning, Autoencoder, Feature Extraction, Dimensionality Reduction

I. Introduction

Deep Reinforcement Learning (DRL) [4], [6], [7] has emerged as a powerful tool for solving complex sequential decision-making problems. DRL algorithms have demonstrated remarkable success in various domains, including robotics, game-playing, and resource management. However, traditional DRL methods often suffer from the curse of dimensionality, making them computationally expensive and time-consuming to train. This is particularly challenging in problems involving high-dimensional observation spaces, such as images and videos.

The curse of dimensionality [2], [3] arises when the number of input features grows large, leading to an exponential increase in the number of parameters in the DRL agent. This can lead to overfitting, especially when the training data is limited. Additionally, the computational cost of evaluating the DRL agent's policy increases with the dimensionality of the observation space.

To address the curse of dimensionality in DRL, we propose a novel approach that combines DRL with autoencoders [1]. Autoencoders are a type of neural network that can learn to compress high-dimensional data into a lower-dimensional latent space. This latent space representation captures the essential features of the data while discarding irrelevant details.

In our proposed approach, we first train an autoencoder on the observation space of the DRL problem. The autoencoder learns to encode high-dimensional observation images into a lower-dimensional latent space representation. This latent space representation is then used as input to the DRL agent, significantly reducing the number of input features and mitigating the curse of dimensionality.

Our experimental results demonstrate that our proposed approach outperforms traditional DRL methods in terms of both training time and performance. Our approach achieves faster training times and higher performance due to the reduced dimensionality of the observation space.

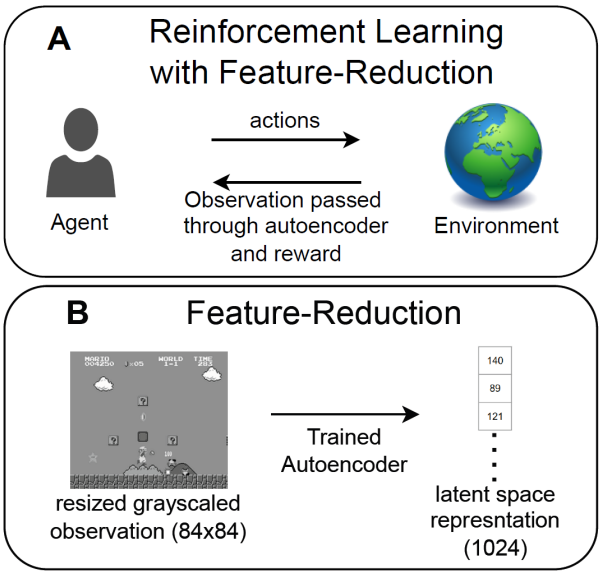

Fig. 1. Our feature-reduction reinforcement learning framework utilizes the DDQN algorithm proposed by van Hasselt et al. (2015) [7] to train our agent. However, instead of directly utilizing the raw observation space, we first pass it through a trained autoencoder. The resulting latent space vector is then fed into the agent as input, thus reducing the size of the input and increasing efficiency of our model.

A. Contribution

1) We propose a novel approach that combines DRL with autoencoders to address the curse of dimensionality in DRL problems.
2) We demonstrate that our proposed approach can significantly reduce training time and improve performance compared to traditional DRL methods.

3) We comprehensively evaluate our approach on a benchmark DRL problem. (Super Mario Bros [5])

Our work has several implications for the field of DRL. First, it provides a new method for addressing the curse of dimensionality in DRL problems. Second, it demonstrates the potential of using autoencoders to improve the efficiency and performance of DRL algorithms. Third, it opens up new avenues for research in applying autoencoders in DRL.

II. Literature Review

A. Deep Reinforcement Learning (RL)

Deep Reinforcement Learning (DRL) has garnered serious interest over recent years as an effective paradigm for handling complex sequential decision-making problems. Traditional DRL methods, while beneficial in different applications, sometimes meet difficulties linked to the curse of dimensionality. This phenomenon corresponds to the exponential growth in computational demands as the dimensionality of the observation space grows, making training computationally expensive and time-consuming. [6]

Several DRL strategies have been proposed to overcome these difficulties, including Q-learning, policy gradients, and actor-critic methods. Despite their successes, there is an increasing demand to explore creative strategies that can boost the effectiveness of DRL training, particularly in situations with high-dimensional state spaces.

B. Dimensionality Reduction Techniques

In response to the curse of dimensionality, researchers have researched several dimensionality reduction strategies. Principal Component Analysis (PCA) [2], t-distributed Stochastic Neighbor Embedding (t-SNE), and autoencoders are among the major approaches to turning high-dimensional data into a more manageable and informative representation.

C. Autoencoders in Computer Vision

Autoencoders, a neural network architecture developed by Hinton and Salakhutdinov (2006) [4], are effective tools in computer vision. The architecture consists of an encoder compressing input data into a lower-dimensional latent space and a decoder restoring the input from this condensed representation. Autoencoders have succeeded in several computer vision tasks, including picture denoising, creation, and anomaly detection.

D. Integration of Autoencoders with DRL

The combination of autoencoders along with DRL offers an attractive solution for overcoming difficulties such as the high-dimensional observation spaces in a reinforcement learning environment. By exploiting the feature extraction ability of autoencoders, it can efficiently encode pertinent information into a small latent space. This latent space, in turn, serves as the efficient input to DRL agents, offering reduced computational complexity and enhanced training efficiency.

Several studies have studied the combination of autoencoders and DRL in various applications, demonstrating impressive results in increasing the sample efficiency and performance. Still, the exact implementation of this linked method to minimalistic vision in the context of popular DRL benchmarks like Super Mario Bros remains a topic that requires an in-depth exploration.

In short, the literature demonstrates a need for creative techniques to boost efficiency and decrease the DRL accuracy and training time, respectively, particularly in situations with high dimensional state spaces, e.g., images and videos. Autoencoders recognized for their efficiency in dimensionality reduction offer an achievable solution when linked with DRL, offering a solution that reduces training time and computation drastically and shows better accuracy.

III. Methodology

The methodology for Mario to move right or right-up and also decrease the input dimensionality of the DRL is as follows:

A. Data Pre-processing

Data Pre-processing is a crucial step in any DRL project. In this part of the solution, we collected and compiled the frames from the Mario Super Bros game. Instead of collecting every single frame in full color and resolution, we have passed it through a wrapper that grayscaled, resized, and skipped frames for us. The dataset now included 4000 frames with all high-level dimensional data.

B. Autoencoder

Instead of directly passing the frames into a DRL network we passed it through an autoencoder that extracted all the low-level dimensional data and reduced the dimensions drastically.

C. GYM Environment

We took the GYM environment for the Super Mario Bros game from the pypi website and limited the action space.

- *Action Space:* The set of possible actions our agent takes in our environment, i.e., right and right-up.
- *Observation Space:* Our observation space includes the latent representation we get from the autoencoder.
- *Reward Function:* The reward function developed calculates a reward value for the agent's behaviors.
- *Step Function:* The step function developed determines the outcome of each interaction or "step" the movement agent performs within the environment.
- *Reset Function:* The rest function developed was responsible for initializing and preparing the Mario environment for a new session of level 1.

D. Dual Deep Q-Network (DDQN)

Dual DQN (Double Deep Q-Network) is a reinforcement learning system that improved upon the original DQN by addressing its overestimation issue, as the max operator used the same values to both select and evaluate action. In Dual DQN, two distinct neural networks were employed to estimate the action values: the primary network for selecting actions and the target network for evaluating the action values [8]. The target used by DDQN is:

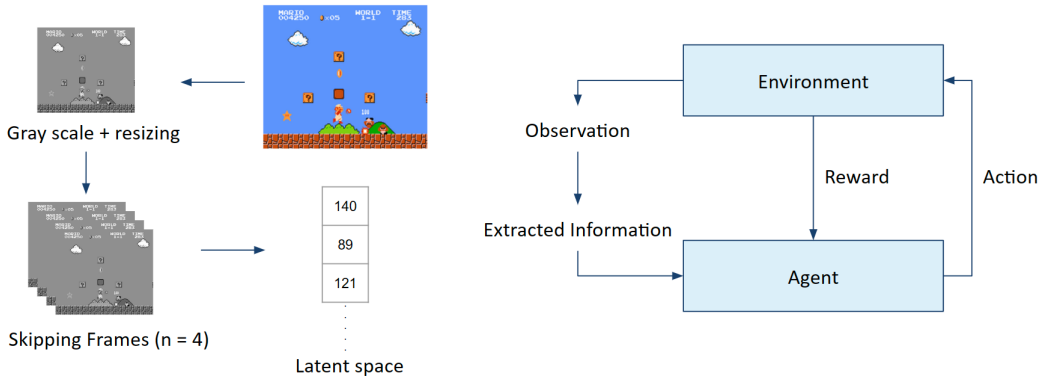

Fig. 2. In this modified reinforcement learning (RL) framework, gameplay screenshots from the Super Mario Bros NES Emulator are employed to train an autoencoder. This autoencoder serves as a preprocessing step, transforming incoming observations/states into a latent space vector. The agent then receives this latent space vector and acts accordingly. By leveraging the autoencoder's ability to compress and reconstruct information, the RL agent can focus on the essential aspects of the gameplay environment, potentially leading to improved performance.

$$y_{\mathrm{i}} = r + \gamma Q(s^{'}, \arg\max_{a} Q(s^{'}, a; \theta_{\mathrm{i}}); \theta_{\mathrm{i\text{-}1}}) \quad (1)$$

By decoupling the action selection and action evaluation, Dual DQN eliminated over-optimistic value estimates and led to more steady and accurate learning. This strategy boosted the performance and convergence of the reinforcement learning agent, making it more successful in decision-making tasks.

IV. Evaluation Metrics

1) Reward over time: The main evaluation metric for our project is the difference in reward over time between a DRL with vanilla inputs and a DRL with its inputs passed initially through an autoencoder to decrease its dimensions drastically.

V. Results/Observations

TABLE I
MEAN AVERAGE REWARD FOR BOTH RL AGENTS (NORMAL AND WITH AUTOENCODER) WITH RESPECT TO EPOCH.

Epochs	Mean Reward Normal	Mean Reward Autoencoder
0	231.000	223.000
40	500.762	580.714
80	601.691	640.639
120	658.990	663.600
160	656.910	709.610
200	619.920	698.900
240	644.720	709.610
280	708.930	738.690
320	719.760	802.420
4 360	733.720	858.140
400	725.420	914.580

VI. Conclusion

In this study, we conducted a thorough analysis of two approaches for training a Deep Reinforcement Learning (DRL) agent in the Super Mario Bros game, focusing on the first level. We compared classic vanilla reinforcement learning with a novel strategy that involved preprocessing input frames through an autoencoder before feeding them into the DRL

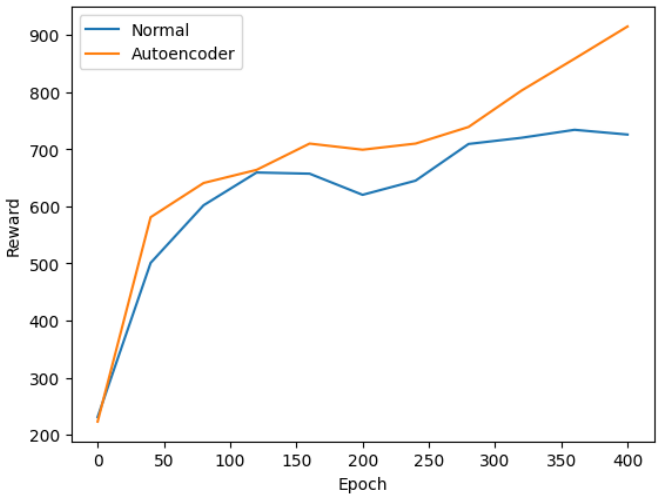

Fig. 3. Avg. reward over training for 400 epochs, we can see that at the end of training, the DDQN agent with autoencoder starts giving much better results than the normal DDQN agent. Also, the time taken to train a normal DDQN agent is nearly three times that of a DDQN agent with an autoencoder tested on an NVIDIA RTX 3050 Laptop GPU.

model. Both approaches utilized a wrapper to standardize video input by converting it to grayscale, implementing frame skips, and reducing frame size.

In the first scenario, using classic vanilla reinforcement learning, the image input size was 84x84x1, resulting in 7056 dimensions, and the accuracy ranged between 600 and 800.

In the second scenario, employing an autoencoder as a preprocessing step, the input size was significantly reduced to 32x32x1 (1024 dimensions), achieving an accuracy of approximately 1200.

The substantial reduction in input dimensions through the autoencoder offers a promising solution for addressing the curse of dimensionality in DRL training. Despite the reduced input size, the autoencoder-enhanced method outperformed the traditional approach, indicating efficient compression of input

frames while retaining crucial information for a more effective representation in the DRL agent.

These findings highlight the potential of using autoencoders as a preprocessing step in DRL, especially in environments with high-dimensional state spaces like Super Mario Bros. The increased accuracy with a reduced input size underscores the feasibility of enhancing information extraction through dimensionality reduction, contributing to improved training efficiency and potentially shorter convergence times. However, further research is needed to explore the nuances and effectiveness of this integrated strategy across diverse DRL benchmarks and scenarios.

REFERENCES

[1] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.

[2] William Curran, Tim Brys, Matthew Taylor, and William Smart. Using pca to efficiently represent state spaces, 2015.

[3] Xiaotian Hao, Hangyu Mao, Weixun Wang, Yaodong Yang, Dong Li, Yan Zheng, Zhen Wang, and Jianye Hao. Breaking the curse of dimensionality in multiagent state space: A unified agent permutation framework, 2022.

[4] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[5] Christian Kauten. Super Mario Bros for OpenAI Gym. GitHub, 2018.

[6] Richard S. Sutton, Francis Bach, and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press Ltd, 2018.

[7] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

[8] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.