Efficient Models Deep Reinforcement Learning for NetHack Strategies

Yasuhiro Onuki, Yasuyuki Tahara[a], Akihiko Ohsuga[b] and Yuichi Sei[c]
The University of Electro-Communications, Tokyo, Japan
ohnuki.yasuhiro@ohsuga.lab.uec.ac.jp, {tahara, ohsuga, seiuny}@uec.ac.jp

Keywords: Deep Reinforcement Learning, Game Agents, Additional Rewards, VAE, NLE.

Abstract: Deep reinforcement learning (DRL) has been widely used in agent research across various video games, demonstrating its effectiveness. Recently, there has been increasing interest in DRL research in complex environments such as Roguelike games. These games, while complex, offer fast execution speeds, making them useful as a testbeds for DRL agents. Among them, the game NetHack has gained of research attention. In this study, we aim to train a DRL agent for efficient learning with reduced training costs using the NetHack Learning Environment (NLE). We propose a method that incorporates a variational autoencoder (VAE). Additionally, since the rewards provided by the NLE are sparse, which complicates training, we also trained a DRL agent with additional rewards. As a result, although we expected that using the VAE would allow for more advantageous progress in the game, contrary to our expectations, it proves ineffective. Conversely, we find that the additional rewards are effective.

1 INTRODUCTION

Research on Deep Reinforcement Learning (DRL) has been conducted in video games such as Atari 2600, Minecraft, and NetHack. DRL research in games is useful not only for gameplay but also for game design and testing (Bergdahl et al., 2020).

With this context, DRL research in complex and stochastic environments such as Roguelike games has also attracted attention (Kanagawa and Kaneko, 2019; Hambro et al., 2022; Izumiya and Simo-Serra, 2021).

Roguelike games refer to games with similar characteristics to Rogue, which was released in 1980. Examples of Roguelike games include NetHack and the Pokémon Mystery Dungeon series. Roguelike games have characteristics including the following: the probability of encountering the same situation multiple times is extremely low, there is partial observability in which players cannot see areas they have not visited, and if the playing character dies, the game must be restarted from the beginning.

NetHack is a Roguelike game which released in 1987[1, 2]. The NetHack play screen, as shown in Figure 1, is represented the game's stats using letters and symbols. The goal of the game is to explore over 50 floors, collect amulets, and offer them to the gods. NetHack has the following characteristics, among others: eating too much can cause suffocation, there are instant-death traps, and the game is lengthy with over 50 levels to explore.

There is a study titled "The NetHack Learning Environment" (NLE) (Küttler et al., 2020) that used NetHack as a reinforcement learning (RL) environment. NetHack is known for being a challenging RL environment, making it difficult for DRL agents to succeed in the game (Piterbarg et al., 2023).

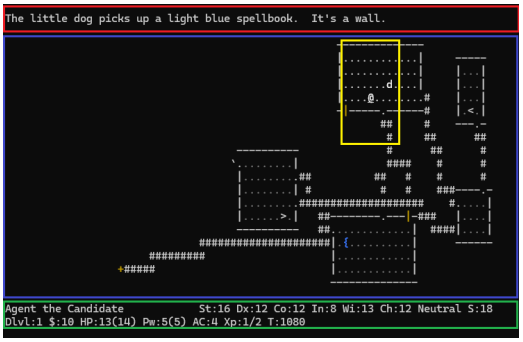

Figure 1: This is an example of playing NetHack scene.

The authors of NLE have recommended using the Score task, which rewards in-game score such as the number of enemies defeated, the dungeon depth (dungeon level) and other factors. However, the Score task

[a] https://orcid.org/0000-0002-1939-4455
[b] https://orcid.org/0000-0002-2552-6717
[c] https://orcid.org/0000-0001-6717-7028
[1] NetHack Homepage:https://www.nethack.org/
[2] NetHack Wiki:https://nethackwiki.com/

Onuki, Y., Tahara, Y., Ohsuga, A. and Sei, Y.
Efficient Models Deep Reinforcement Learning for NetHack Strategies.
DOI: 10.5220/0013253100003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1*, pages 556-563
ISBN: 978-989-758-737-5; ISSN: 2184-433X

only provides rewards to the agent when certain situations are met. As a result, the rewards in the Score task are sparse, making it difficult for the agent to learn the value of its actions.

In recent years, research on RL using Variational Autoencoder (VAE) (Kingma and Welling, 2014) has progressed (Ha and Schmidhuber, 2018; Hafner et al., 2024). Of these, the high-performing agent "DreamerV3" (Hafner et al., 2024) has been developed.

DRL consumes a substantial amount of computational resources, especially in complex environments and tasks. The high usage of GPUs, CPUs, and memory consumption also leads to increased costs. Given the increasing costs, it is essential to reduce them in order to use resources efficiently in research and small-scale development project. Additionally, when training takes a long time, the cycles of experimentation and improvement slow down, making it crucial to reduce learning costs and achieve faster training.

In this study, we aim to develop an agent that progresses more efficiently in the NLE by using a DRL approach that reduces learning costs and enables efficient learning. To achieve this, we train agents with both the rewards from the NLE Score task and additional rewards, and compare them with agents that receive only the rewards from the Score task. Furthermore, the agent is divided into two parts: one for learning latent representations using a VAE and the other for learning action selection using a DRL agent, in order to train the agent efficiently.

As a result, although the integration of the VAE was expected to enhance the DRL agent's efficiency, the anticipated improvement fails to appear. In contrast, the additional rewards provide the necessary information for progressing more efficiently in the NLE, contributing to more efficient learning.

This paper proceeds as follows. In Section 2, related work is introduced. Section 3 provides a detailed explanation of the proposed method, and Section 4 presents the experimental methods and results. In Section 5, a discussion is provided, and Section 6 concludes the paper and discusses future study.

2 RELATED WORK

2.1 NetHack Learning Environment

The NetHack Learning Environment (NLE) (Küttler et al., 2020) is a study that uses the Gym Environment (Brockman et al., 2016) to set up NetHack as a RL environment and presents the results of baseline models. In NLE, the number of environment steps per episode is generally limited to a maximum of 5000 steps.

The observation space in NLE consists of elements such as glyphs, blstats and message. Glyphs represent the 2D symbolic dungeon observation space (the purple-framed area in Figure 1). Blstats represent the agent's coordinates and character attributes (the green-framed area and the white '@' symbol attributes in Figure 1). Message represent displayed messages (the red-framed area in Figure 1). NLE has an action space of 93 available actions and includes seven initial tasks, such as Score task, which feature different reward functions and action space.

The authors of NLE have considered it currently very difficult for machine learning approaches to solve NetHack, and have recommended using the Score task with in-game score as the reward function. Therefore, in this study, we use the Score task.

In the NLE baseline models, learning was conducted using IMPALA and RND. The observation space was composed glyphs, blstats, and 9×9 area centered around the agent (the yellow-framed area in Figure 1, hereafter referred to as glpyh99), and a reduced action space with 23 dimensions was used.

The baseline models were trained with 1 billion environment steps using the Score task, and "Monk-Human-Neutral-Male". The reported average dungeon level was 5.4, with a maximum of 11. However, NetHack requires descending as many as 50 dungeon levels, so it is essential to develop agents capable of exploring deeper levels.

In this study, we aim to advance the exploration of NLE while reducing learning costs and achieving efficient learning. We consider IMPALA (Espeholt et al., 2018) which DRL models use multi-learners and RND (Burda et al., 2019) which involve training multiple neural networks simultaneously. However, these method are associated with high learning costs and are deemed to have high learning costs and are limited in supporting efficient learning. Therefore, we decided to use different algorithms.

2.2 DRL Research on NLE

In recent years, research using Large Language Models (LLMs) in NetHack has been conducted (Klissarov et al., 2023). Klissarov et al. explored the capabilities and limitations of LLM (GPT-4-Turbo) as a zero-shot agent to investigate how LLMs function without prior training. Their results showed that LLM performed to a certain degree in zero-shot due to its expensive knowledge and versatility. However, in complex and dynamic environments like NLE, challenges remained in terms of strategic decision-making and planning abilities. The computational cost of LLMs is enormous, and in Section 1, they are mis-

aligned with the purpose of this study to reduce the learning cost. Thus, we decided not to use LLMs.

MiniHack (Samvelyan et al., 2021) is an environment based on NLE that provides an interface for RL experiments across a range of settings, from simple rooms to complex worlds. While simpler tasks in MiniHack were successfully completed, more challenging tasks were not. It was also found that RL methods struggle to generalize at scale. Consequently, DRL agents in complex environments must possess the ability to handle this complexity.

The NetHack Challenge (Hambro et al., 2022a) at NeurIPS 2021 competition saw several agents surpass the baseline model, but none completed the NLE. Additionally, agents that used symbolic approaches outperformed those using neural approaches. Among the neural approaches agents, those that combined symbolic methods with neural networks achieved the best results. Therefore, it is considered that neural network approaches require efficient exploration and accurate state understanding.

2.3 DRL with VAE

DreamerV3 (Hafner et al., 2024) adopted an approach that simulates the environment using World Mode (Ha and Schmidhuber, 2018) based on a VAE to determine optional actions based on future prediction. As a result, it has become a DRL agent with advantages such as efficient asmple use, fast learning, and a highly versatile design. DreamerV3 is also the first algorithm to collect diamonds in Minecraft from scratch without external tuning or prior knowledge. Inspired by its high performance as a DRL network that uses a VAE, this study incorporates a VAE into the DRL neural network. However, unlike DreamerV3, we do not perform future predictions. Instead, we aim to directly leverage the latent representations for decision-making to enable efficient agent decisions.

3 PROPOSED METHOD

The reward function for the Score task in the NLE only provides rewards when certain situations occur, resulting in sparse rewards for DRL agents. To address this, we propose additional rewards for the agent beyond those provided by the Score task. Additionally, we introduce a DRL agent that uses a VAE, with separate training for the VAE component and the other neural networks.

In this Section, we describe both the additional rewards and the neural network architecture of the DRL agent that incorporates the VAE.

3.1 Additional Rewards

In this study, we propose additional rewards, which are combined with the rewards from Score task and used for the agent's learning. The purpose and components of the additional rewards are as follows.

- The reward related to the agent's parameters: a positive reward is given to the agent if the parameters stay within a certain range, and a negative reward if they fall outside of that range.
  - Reward of Hit points (HP): $\min\left((\text{current HP}) \,/\, (\text{MAX HP}) - 0.5, 0.3\right)/100$.
  - Reward of Hunger status: Table 1.
- Rewards that assist in progressing through the game: These are provided as reference information to help advance the gameplay.
  - The dungeon level up: 20.
  - The unexplored areas have decreased except the above: (Number of reductions)/32.
  - The agent level up: 15.
  - The agent stay in the same position for the last 12/16 steps: -0.01.
- Rewards related to specific actions: These are given to Mitigation or addition of penalty, and increase the value of necessary actions.
  - The agent select the go down command when a down staircase is present in glyph99: 0.02.
  - The agent select the go down command when except the above: 0.005.
  - The agent select the kick command: 0.01.
  - The agent failed to specify the direction to kick when it was necessary: -0.02.
  - The agent walk towards the wall: -0.01.

Table 1: Reward of Hunger status.

Hunger stats	Select eat command?	reward
Satiated	Yes	-0.01
Not hungry	No	0.001
hungry	Yes	0.02
	No	-0.01
Weak	Yes	0.02
	No	-0.02
Fainting	Yes	0.02
	No	-0.03

3.2 Architecture of DRL with VAE

In this study, we propose a DRL agent's neural network using VAE.

For the VAE, beta-VAE (Higgins et al., 2017) is adopted. The VAE network is shown in Figure 2,

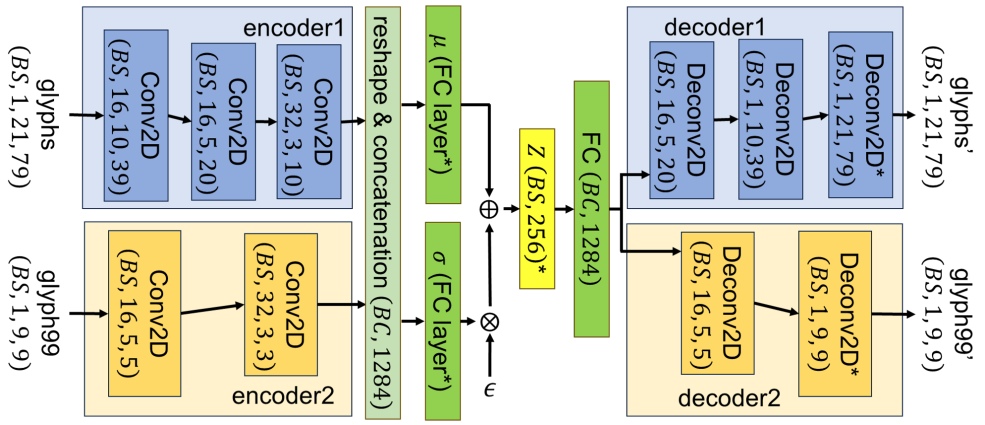

Figure 2: This is proposed VAE Networks. BS means Batch Size.

where the inputs are glyphs and glyph99, and separate encoders and decoders are used for each.

The DRL agent's neural network is shown in Figure 3, with inputs consisting of the encoder's output from the VAE, along with blstats and message. The DRL agent's network is based on PPO agents, and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is used to retain information from previous steps within the same episode.

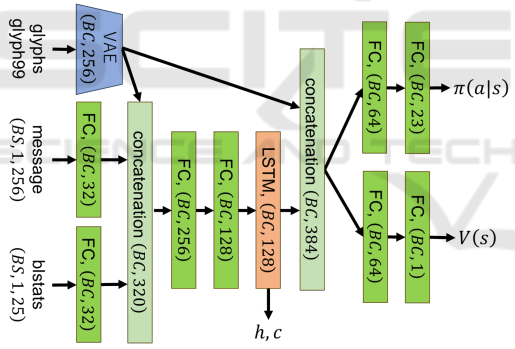

Figure 3: This is proposed DRL Networks. BS means Batch Size.

4 EXPERIMENTS

In this study, we used Windows 11 with WSL2, an NVIDIA GeForce RTX 3090 as the GPU, and an Intel i9-10850K as the CPU. Due to computational resource constraints, the agents were trained using a single setting of "Monk-Human-Neutral-Male". Additionally, for all learning, rewards were clipped using $\tanh(r/16)$, and the Adam optimizer (Kingma and Ba, 2015) was used. The action space was set to a 23-dimensional action space, as in NLE.

4.1 PPO vs DQN

First, we compared and decided on the algorithm to be used as the DRL agent, choosing between Deep Q-Network (DQN (Volodymyr et al., 2015): to adopted prioritized experience replay (Schaul et al., 2016), Double Q-Network (Van Hasselt et al., 2016), and Dueling Network (Wang et al., 2016)) and Proximal Policy Optimization (PPO) (Schulman et al., 2017).

We trained the DQN for 10,000 episodes (25 million environment steps), and the PPO for 25 million environment steps. The neural network for DQN and PPO is shown in Figure 4, and the main hyperparameters are listed in Table 2. For DQN, the Q-values (the output of the neural network) were calculated following the implementation of the Dueling Network.

Table 2: Main Hyperparameters of DQN and PPO. '*' used the triangular mode of CyclicLR.

Hyperparameter	DQN	PPO
Learning Rate	5×10^{-4}	$[10^{-4}, 3 \times 10^{-4}]$*
Batch size	64	512
Weight decay	10^{-5}	-
Discount (γ)	0.96	0.99
GAE param (λ)	-	0.95
Num. Actor	-	20
Horizon (T)	-	1536
Num. Epochs	-	5
Epsilon clip	-	0.2
c_1 / c_2	-	1 / 0.002

Figure 5 shows reward progression for DQN and PPO, and Figure 6 shows dungeon levels and agent levels. After training each agent, we evaluated them 500 episodes, and the results are shown at the top of Table 3. From Table 3, PPO agent outperformed DQN in all metrics, including rewards of the Score task, plus rewards, dungeon level and agent level.

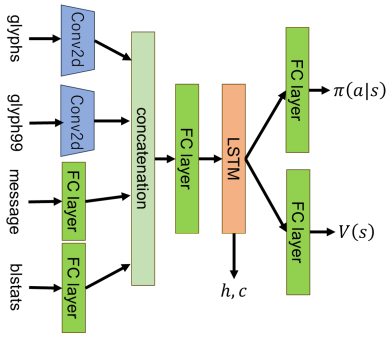

Figure 4: This is DQN and PPO Networks.

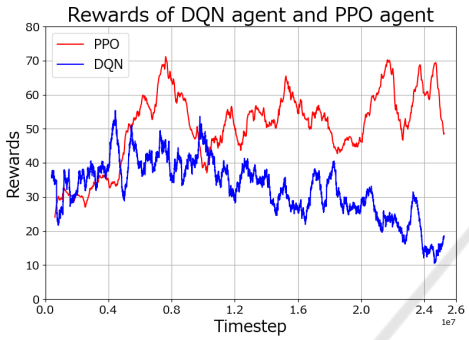

Figure 5: This is the transition of rewards for DQN and PPO.

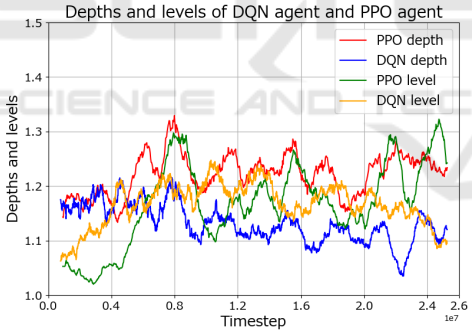

Figure 6: This is the transition of dungeon level (depth) and agent levels (levels) for DQN and PPO.

These results indicate that PPO can progress learning more effectively than DQN. Therefore, in the following experiments, we adopt PPO as the DRL algorithm.

4.2 Comparison of Three PPO Agents

In this section, we compare three types of PPO agents: a PPO agent with only the rewards from the Score task, a PPO agent with both the rewards from the Score task and additional rewards, and a PPO agent using both the additional rewards and the VAE.

In the following, we evaluate the effectiveness of the additional rewards, and then proceed to train the PPO agent using both the additional rewards and the VAE. In this study, aiming to progress through NLE, we focus particularly on dungeon levels.

4.2.1 Additional Rewards

To evaluate the effectiveness of the additional rewards, we trained two PPO agents for 100 million environment steps: one with only the rewards from the Score task, and the other with both additional rewards and the rewards from the Score task. We used the hyperparameters listed in Table 2, and the same neural network architecture in the previous section.

Figure 7 shows the progression of rewards for the PPO agent with only the rewards from the Score task and the PPO agent with both the additional rewards and the rewards from the Score task, Figure 8 shows the progression of dungeon levels, and Figure 9 shows the progression of agent levels. After training each agent, we evaluated them 500 episodes, and Table 3 shows the results at the bottom.

As a result, the DRL agent with the additional rewards outperformed the one with only the rewards from the Score task in terms of rewards from the Score task, dungeon level, and other metrics. Therefore, we consider the additional rewards in Section 3.1 to be effective and proceed to compare a DRL agent using VAE that receives both the additional rewards and the rewards from the Score task.

4.2.2 Learning of DRL Architecture with VAE

Next, we trained a PPO agent using the DRL architecture with the additional rewards rom Section 3.1 and the VAE from Section 3.2, to compare whether it could progress further than a standard PPO agent. We trained a PPO agent using beta-VAE and additional rewards for 100 million environment steps.

In Figure 2, Conv2D, Deconv2D and FC without '*' have ReLU as the activation function. Additionally, in Figure 3, ReLU was used before the first concatenation, and ELU was used thereafter.

In the first 16 iterations, we trained only the VAE, setting the number of VAE epochs to eight. After that, we trained both VAE and PPO, setting the number of VAE epochs to two, the number of PPO actors to 24, and the learning rate to the range $[2 \times 10^{-4}, 3 \times 10^{-4}]$.

Figure 7 shows the progression of rewards for the PPO agent with the additional rewards and the VAE, Figure 8 shows the progression of dungeon levels, and Figure 9 shows the progression of agent levels.

Figure 7 shows the transition of rewards during training. The rewards for PPO and Proposed (Additional rewards) initially increased, then decreased, and subsequently increased again. On the other hand,

Table 3: This table shows the average values from the evaluation conducted after 500 episodes of training. “Reward” mean score task’s reward, and “Plus Reward” refers to the Score task’s reward excluding the penalty. The number in parentheses to Dungeon level represents the maximum value of Dungeon level.

Agent	Reward	Plus Reward	Dungeon level	Agent level
DQN (25M step)	-33.49	10.18	1.03(3)	1.00
PPO (25M step)	54.36	62.55	1.24(4)	1.18
PPO (100M step)	41.00	46.88	1.18(5)	1.08
Proposed (Additional Reward 100M step)	**58.76**	**66.01**	**1.28(6)**	**1.18**
Proposed (VAE+Additional Reward 100M step)	33.59	42.79	1.16(4)	1.09

Figure 7: This is the transition of rewards for 3 agents. “Addr” means Additional rewards. The thin dashed line represents the sum of the rewards from the Score task and the additional rewards.

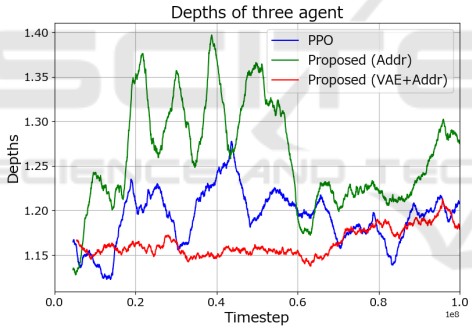

Figure 8: This is the transition of dungeon level for 3 agents.

the rewards for Proposed (VAE and Additional rewards) consistently grew but did not surpass those of the other agents.

Figure 8 shows the transition of dungeon levels. For PPO and Proposed (Additional rewards), the levels went up and down repeatedly but started to increase around 60 million steps. On the other hand, for Proposed (VAE and Additional rewards), the dungeon levels began to increase around 60 million steps and reached values comparable to PPO at approximately 100 million steps.

Figure 9 shows the transition of agent levels. For PPO and Proposed (Additional rewards), the agent levels exhibited a trend similar to the dungeon levels. On the other hand, for Proposed (VAE and Additional rewards), the agent levels showed a gradual increasing trend but did not surpass those of the other agents.

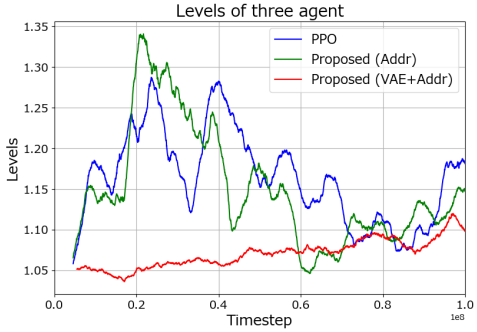

Figure 9: This is the transition of agent level for 3 agents.

After training the agent, we evaluated them 500 episodes, and Table 3 shows the results at bottom.

The results indicate that the DRL agent employing additional rewards and VAE underperformed the standard PPO agent in terms of dungeon level progression. Although particular emphasis was placed on dungeon level performance, the DRL agent with additional rewards and VAE was ineffective compared to the conventional PPO agent.

5 DISCUSSION

In this study, we have placed particular emphasis on dungeon level from the perspective of progressing in NLE. The rewards from the Score task are comprehensive measures based on factors such as the number of defeated enemies and the amount of coins picked up, which allows for higher rewards by remaining on the same floor. Conversely, in NetHack, there are challenges like “score runs,” where players aim to complete the game with lower in-game scores, and “level 1 clears,” where players complete the game without leveling up the hero at all. Therefore, we consider it inappropriate to view the rewards from the Score task as a direct measure of dungeon progression. Consequently, in this study, we focus on dungeon level, considering that in a hierarchical game like NetHack, the depth of the dungeon reached and progress made are closer indicators of progress than other metrics.

As shown in Table 3, in the evaluation, The PPO with additional rewards achieved the highest dungeon level and reward, followed by the PPO only the rewards from the Score task, and finally the PPO with VAE and additional rewards.

Therefore, the findings suggest that using VAE as the neural network for DRL may limit efficient progress. In contrast, it is shown that the additional rewards are effective for progress in NLE.

5.1 Disscusion on DQN and PPO

In the process of learning, the reward progression of the DQN agent showed an increasing trend initially, followed by a decreasing trend, with negative values in the evaluation reward. This is considered to be because DQN adopts the ε-greedy method.

Figures 7, 8 and 9 show that the PPO agent with a standard neural network experienced particularly sharp variations in reward, indicating instability in learning. This instability is believed to be due to the sensitivity of the agent to hyperparameters. This sensitivity can explain why the PPO agent trained for 25 million steps outperformed the agent trained for 100 million steps in terms of reward and dungeon level.

5.2 Disscusion on Additional Rewards

The DRL agent with a standard neural network, when given additional rewards, outperformed the agent without additional rewards in terms of metrics such as reward and dungeon level. In environments with sparse rewards, we argue that providing additional rewards to the DRL agent improves the reward function provided by the environment and effectively contributes to progress in the game.

5.3 Disscusion on VAE Architecture

The PPO agent with VAE showed a consistent upward trend in rewards. However the reward increase occurred at a slower pace compared to other agents, and it did not reach the levels achieved by the other agents. This is assumed that this results from using VAE to map the state into a latent representation, which the DRL agent then uses as input. Therefore, it is necessary to transform the VAE encoder's output into a form that the DRL agent can more easily understand.

The training time was approximately 57.54 hours for PPO using VAE and 59.67 hours for standard PPO, with the VAE-based PPO finishing the training faster. The CPU usage was nearly the same for both agents, while the GPU usage was slightly lower (around 1%) for PPO with VAE. Therefore, using a VAE is considered to contribute to more efficient training.

In this study, we provided the latent representations of glyphs and glyph99 generated by a VAE directly as inputs to the DRL agent. However, the results did not surpass those achieved by standard PPO. Therefore, it is necessary to explore how best to utilize VAEs in the context of NLE. We are currently examining what input information should be used for the VAE and how the encoder outputs of the VAE can be effectively employed. Additionally, we aim to investigate whether it is possible to design VAE outputs in a form that is easier for DRL to understand, rather than simply utilizing a VAE in its standard form.

6 CONCLUSIONS

In this study, we developed an efficient DRL agent and examined its ability to progress in NLE. As efficient DRL agents, we developed and trained two agents: one agent that added additional rewards to the rewards from NLE Score task, and another agent that used both additional rewards and beta-VAE in the neural network.

The results suggest that, despite expectations for more efficient learning and improved performance in the DRL agent through VAE integration, there is a departure from these initial predictions. Conversely, we have demonstrated that using additional rewards is effective for promoting efficient learning.

There are three main challenges for future work.

The first is the adjustment of additional rewards and hyperparameters. We believe that the additional reward scheme requires revision. Furthermore, we plan to develop an algorithm that automatically adjusts both additional rewards and hyperparameters, aiming to create an agent capable of autonomously achieving optimal performance.

The second is incorporating state-of-the-art methods and symbolic-based approaches. In this study, we used the fundamental DRL algorithms, DQN and PPO, and compared their performances. However, in NLE, symbolic-based approaches have also demonstrated effective results, and there are several high-performing DRL algorithms. Therefore, we would like to consider combining symbolic approaches with DRL or employing the latest DRL algorithms.

The third is considering comparisons with other methods and generalizability. In this study, due to computational resource limitations, we conducted training under the single setting of "Monk-Human-Neutral-Male" for 100 million steps. In contrast, many studies in NLE involve training for 1 billion

steps (Küttler et al., 2020; Izumiya and Simo-Serra, 2021). Therefore, considering the cost of implementing and training other algorithms, we did not perform comparisons with other methods. Thus, we aim to further enhance the efficiency and speed of training, conduct training for 1 billion environment steps, and compare our method with others. Additionally, we plan to include settings other than "Monk-Human-Neutral-Male" to improve generalizability in NLE.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Numbers JP22K12157, JP23K28377, JP24H00714.

We acknowledge the assistance for the ChatGPT (GPT-4o and 4o mini) was used for proofreading, which was further reviewed and revised by the authors.

REFERENCES

Bergdahl, J., Gordillo, C., Tollmar, K., and Gisslén, L. (2020). Augmenting automated game testing with deep reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, pages 600–603.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. In *arXiv preprint arXiv:1606.01540*.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2019). Exploration by random network distillation. In *International Conference on Learning Representations*.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. (2018). IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1407–1416.

Ha, D. and Schmidhuber, J. (2018). World models. In *arXiv preprint arXiv:1803.10122*.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. (2024). Mastering diverse domains through world models. In *arXiv preprint arXiv:2301.04104*.

Hambro, E., Raileanu, R., Rothermel, D., Mella, V., Rocktäschel, T., Küttler, H., and Murray, N. (2022). Dungeons and data: A large-scale nethack dataset. In *Advances in Neural Information Processing Systems*, volume 35, pages 24864–24878.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. In *Neural Comput.*, volume 9, pages 1735–1780. MIT Press.

Izumiya, K. and Simo-Serra, E. (2021). Inventory management with attention-based meta actions. In *2021 IEEE Conference on Games (CoG)*.

Kanagawa, Y. and Kaneko, T. (2019). Rogue-gym: A new challenge for generalization in reinforcement learning. In *arXiv preprint arXiv:1904.08129*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Klissarov, M., D'Oro, P., Sodhani, S., Raileanu, R., Bacon, P.-L., Vincent, P., Zhang, A., and Henaff, M. (2023). Motif: Intrinsic motivation from artificial intelligence feedback. In *arXiv preprint arXiv:2310.00166*.

Küttler, H., Nardelli, N., Miller, A. H., Raileanu, R., Selvatici, M., Grefenstette, E., and Rocktäschel, T. (2020). The nethack learning environment. In *34th Conference on Neural Information Processing Systems*.

Piterbarg, U., Pinto, L., and Fergus, R. (2023). Nethack is hard to hack. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Samvelyan, M., Kirk, R., Kurin, V., Parker-Holder, J., Jiang, M., Hambro, E., Petroni, F., Küttler, H., Grefenstette, E., and Rocktäschel, T. (2021). Minihack the planet: A sandbox for open-ended reinforcement learning research. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *arXiv preprint arXiv:1511.05952*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2094–2100. AAAI Press.

Volodymyr, M., Koray, K., David, S., A., R. A., Joel, V., G., B. M., Alex, G., Martin, R., K., F. A., Georg, O., Stig, P., Charles, B., Amir, S., Ioannis, A., Helen, K., Dharshan, K., Daan, W., Shane, L., and Demis, H. (2015). Human-level control through deep reinforcement learning. In *Nature*, number 518, pages 529–533.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, volume 48, pages 1995–2003.