Evaluating the Impact of Prioritized Experience Replay on Deep Q-Learning Performance in Tetris

Neil Adrian Baltar
College of Computing and Information Technology (CCIT)
National University (N.U.)
Manila, Philippines
baltarnb@students.national-u.edu.ph

Carl Arvin Hipolito
College of Computing and Information Technology (CCIT)
National University (N.U.)
Manila, Philippines
hipolitocc@students.national-u.edu.ph

Xavier Pagdanganan
College of Computing and Information Technology (CCIT)
National University (N.U.)
Manila, Philippines
pagdangananxp@students.national-u.edu.ph

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm for developing intelligent agents capable of learning optimal behaviors through interaction with dynamic environments [1]. Among various RL approaches, the Deep Q-Network (DQN) algorithm has gained prominence for its ability to approximate complex value functions using deep neural networks, enabling effective decision-making in high-dimensional state spaces [2]. The success of DQN in domains such as Atari 2600 games [3] and robotic control has inspired numerous studies aimed at improving its learning efficiency, stability, and convergence rate.

A critical component in the DQN framework is the *experience replay* mechanism, which stores past transitions (s, a, r, s') in a replay buffer for randomized sampling during training [4]. This approach mitigates temporal correlations in sequential data, enhances sample efficiency, and stabilizes learning. However, traditional uniform sampling in replay memory treats all experiences equally, which may limit learning effectiveness, especially when some transitions provide more significant learning signals than others [5]. To address this limitation, Prioritized Experience Replay (PER) was introduced to bias the sampling probability toward transitions with higher temporal-difference (TD) errors [6]. By focusing on more informative experiences, PER aims to accelerate learning and improve the overall policy performance.

Tetris serves as an ideal testbed for reinforcement learning research due to its discrete state space, complex long-term dependencies, and stochastic dynamics [7]. The game requires strategic decision-making to optimize line clearances and prevent premature termination, making it a challenging benchmark for evaluating learning-based algorithms. Although several RL studies have explored Tetris using algorithms such as Q-learning, Sarsa, and policy-gradient methods, the comparative analysis of experience replay strategies within a DQN framework remains relatively underexplored.

This study aims to investigate the performance differences between *standard replay memory* and *prioritized experience replay* in training DQN agents to play Tetris. Both replay strategies are implemented under identical network architectures and hyperparameters to ensure a fair comparison. The agents are trained in the Gymnasium-based Tetris environment, with performance evaluated using metrics such as average episode reward, number of cleared lines, and training stability. The results of this comparative analysis are expected to provide insights into how replay memory design influences the efficiency and effectiveness of deep reinforcement learning agents.

The remainder of this paper is organized as follows: Section II discusses related literature on reinforcement learning and experience replay techniques. Section III describes the methodology, including environment configuration, DQN architecture, and replay memory implementations. Section IV presents the experimental setup and results. Finally, Section V concludes the paper and outlines potential directions for future work.

II. Review of Related Literature

A. Reinforcement Learning (RL)

Reinforcement Learning (RL) is a subfield of machine learning in which an agent learns to make sequential decisions by interacting with an environment and receiving scalar feedback signals known as rewards. The primary objective of the agent is to maximize cumulative rewards over time by learning an optimal policy that maps states to actions [10]. RL problems are typically modeled as Markov Decision Processes (MDPs), which define the set of states, actions, transition dynamics, and reward functions governing the agent–environment interaction [11].

Unlike supervised learning, where labeled data guide the model, RL relies on experience-based exploration and feedback to improve performance. This makes RL particularly suited for dynamic and uncertain environments where explicit labels are unavailable [12]. However, RL also faces several challenges, including the trade-off between exploration and exploitation, sparse or delayed rewards, and high-dimensional state spaces [13]. These challenges are especially apparent in game-based environments, where agents must plan actions over long horizons to achieve success.

The theoretical foundations of RL have been extensively discussed by Sutton and Barto [10], who emphasized the importance of temporal difference learning, reward prediction, and policy optimization. These principles continue to guide modern RL applications in domains such as robotics, game playing, and autonomous control systems.

B. Game Environments for Reinforcement Learning

In reinforcement learning (RL) research, game environments play a crucial role as testbeds for developing, benchmarking, and evaluating algorithms. These environments provide controlled yet complex scenarios that allow agents to learn optimal policies through repeated interactions. The design of the environment—including its reward structure, state representation, and action space—significantly influences the learning performance and generalization capabilities of RL models [**?**]. Classic game environments such as Atari 2600, OpenAI Gym, and MuJoCo have served as standard platforms for testing reinforcement learning algorithms, fostering reproducibility and consistent performance evaluation across studies [**?**].

1) OpenAI Gym and Gymnasium: OpenAI Gym, introduced by Brockman et al. [**?**], is one of the most widely adopted environments for reinforcement learning research. It provides a unified interface for a diverse set of tasks, ranging from simple control problems like CartPole to complex Atari games. The modular structure of Gym enables researchers to focus on algorithm development without worrying about environment implementation details. In 2022, Gymnasium was introduced as a community-maintained successor that offers improved compatibility, performance, and API standardization [**?**]. Gymnasium preserves backward compatibility with Gym while enhancing environment stability and observability, making it suitable for modern reinforcement learning frameworks.

Mathematically, an RL environment such as those in Gymnasium can be described as a Markov Decision Process (MDP), represented by the tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$$

where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ the set of actions, $P(s' \mid s, a)$ the transition probability of moving from state s to s' given action a, $R(s, a)$ the reward function, and $\gamma \in [0, 1]$ the discount factor. The agent interacts with the environment by selecting an action a_t in state s_t, receiving a reward $r_t = R(s_t, a_t)$, and transitioning to a new state s_{t+1} according to P. This iterative loop continues until a terminal state is reached or a predefined horizon is met [**?**].

2) Tetris as a Reinforcement Learning Benchmark: Tetris is a challenging testbed for reinforcement learning due to its high-dimensional state space, delayed rewards, and stochasticity introduced by random piece generation. Unlike standard environments with immediate feedback, Tetris rewards agents based on long-term strategies such as line clearing and survival duration. The game can also be configured with different action spaces (e.g., discrete rotations and translations) and reward schemes (e.g., score-based or survival-based), making it highly flexible for experimental setups [**?**].

Early studies on Tetris RL employed heuristic or handcrafted evaluation functions, but recent work has explored deep reinforcement learning approaches, particularly using Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) [**?**]. These algorithms aim to balance exploration and exploitation in a dynamic and non-stationary environment. However, due to the complex state dependencies and sparse rewards of Tetris, training stable and high-performing agents remains an open challenge. Gymnasium-based Tetris environments facilitate experimentation by providing standardized interfaces for customizing rewards, action constraints, and observation modes, allowing systematic evaluation of algorithmic modifications and parameter tuning [**?**].

3) Environment Modification and Reward Shaping: Environment modification, including reward shaping, is a common strategy to enhance learning efficiency. Reward shaping involves altering the reward function to provide more frequent feedback or guide the agent toward desired behaviors [**?**]. In Tetris, shaping rewards by giving intermediate rewards for actions such as clearing lines, avoiding holes, or maintaining a low stack height can significantly affect learning dynamics. Similarly, limiting action spaces to only valid moves can reduce exploration complexity, helping algorithms such as PPO and DQN learn more efficiently within feasible boundaries. However, these modifications must be applied carefully, as excessive shaping may bias the policy toward suboptimal strategies [**?**].

Overall, Gymnasium and similar frameworks have become indispensable for RL experimentation, enabling consistent and replicable research workflows. Tetris, as implemented within these environments, provides an ideal setting for studying the interplay between algorithmic design, environment modification, and parameter optimization in reinforcement learning.

C. Related Studies

III. Methodology

A. Environment and Problem Formulation

The experimental environment utilized in this study is Tetris-Gymnasium, a modern and modular reinforcement learning framework specifically designed for Tetris research and fully compliant with the Gymnasium API standard [1]. Developed to address limitations in existing Tetris environments—such as inadequate documentation, outdated codebases, and limited configurability—Tetris-Gymnasium provides a standardized, transparent, and extensible platform for evaluating RL algorithms under challenging sparse-reward conditions. The environment was selected for its adherence to contemporary best practices in RL research, including modular architecture, comprehensive documentation, and compatibility

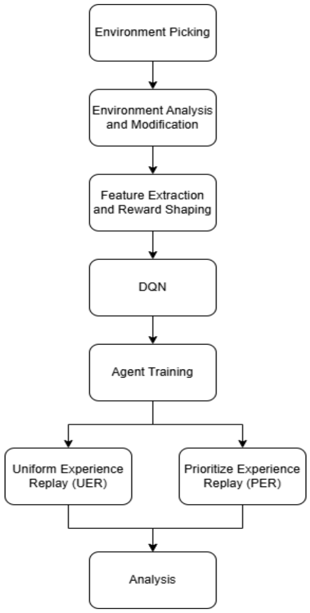

Fig. 1. Training Methodology

with state-of-the-art training frameworks. This ensures reproducibility and facilitates direct comparison with prior work in the domain.

The observation space in Tetris-Gymnasium is structured as a dictionary containing four primary components. First, the board state is represented as a two-dimensional array of dimensions × h×w, where h and w denote the height and width of the playing field, respectively, with default values of 20 and 10. The board is padded by a boundary equal to the size of the largest tetromino (typically four cells for the I-piece) to accommodate piece rotation near the edges. Each cell in the board array encodes the occupancy state: empty cells are represented by 0, while occupied cells and boundary padding are assigned distinct identifiers. Second, an active tetromino mask—also of dimensions × h×w—provides binary values indicating the spatial location of the currently falling piece, enabling the agent to distinguish between settled blocks and the active tetromino. Third, the holder component, represented as a one-dimensional array, encodes the tetromino currently stored in the hold feature (if available), facilitating strategic piece management. Fourth, the queue component, also represented as a one-dimensional array, contains the sequence of upcoming tetrominoes, allowing the agent to anticipate future pieces and plan accordingly. This comprehensive observation structure captures both immediate board configuration and lookahead information, supporting sophisticated decision-making strategies.

The action space is discrete and corresponds to the set of valid manipulations available to the player at each time step. These actions include moving the active piece left, right, or down; rotating the piece clockwise or counterclockwise; performing a hard drop to instantly place the piece; and swapping the current piece with the held tetromino. The exact composition of the action space is governed by a configurable ActionsMapping object, which allows environment customization to focus on specific aspects of gameplay. In the default configuration, the action space can be formally denoted as:

A=
move$_{l}eft, move_{r}ight, move_{d}own, rotate_{c}w,$
rotate$_{c}cw,$
hard$_{d}rop, swap.$

This discrete formulation simplifies policy parameterization and is particularly well-suited to value-based methods such as Deep Q-Networks (DQN), which require enumerable action spaces for Q-value estimation.

The reward function is governed by a customizable RewardsMapping component, which allows researchers to tailor the reward signal to specific learning objectives. In standard configurations, rewards are primarily determined by game progression metrics, such as the number of lines cleared in a single action. Positive rewards incentivize desirable outcomes, such as clearing multiple lines simultaneously (rewarding efficient play), while negative rewards may be applied for undesirable states, such as increasing board height or creating holes (empty cells beneath occupied ones). The flexibility of the reward mapping enables experimentation with dense versus sparse reward structures, reward shaping techniques, and multi-objective optimization. For instance, reward shaping can be employed to provide intermediate feedback that accelerates learning in early training phases while still aligning with the ultimate objective of maximizing cumulative line clears and prolonging episode duration. The modular design of the reward system supports hypothesis-driven experimentation, allowing researchers to isolate the impact of reward design on agent performance.

Episode termination occurs when a newly spawned tetromino cannot be placed at the top of the board due to collision with previously settled blocks—a condition commonly referred to as topping out. At this juncture, the episode concludes, and a terminal reward signal (often negative) is issued to discourage actions leading to premature termination. Consequently, the task is episodic: each training episode begins with an empty board and a random initial tetromino, and terminates upon reaching the failure condition. This episodic structure provides clear temporal boundaries for credit assignment and facilitates the computation of discounted return over finite horizons, which is essential for temporal-difference learning algorithms.

The Tetris-Gymnasium environment presents a particularly challenging benchmark for reinforcement learning due to several intrinsic properties. First, the combinatorial state space is vast—even with a modest 20 × 10 20×10 board, the number of possible configurations is exponentially large, necessitating function approximation via deep neural networks. Second, rewards are sparse and delayed; meaningful positive feedback (line clears) occurs infrequently, requiring the agent to learn long-horizon dependencies and credit assignment across multiple actions. Third, the environment exhibits non-Markovian characteristics when the queue and holder information are excluded, as optimal action selection depends on anticipating future piece arrivals. Finally, the task involves multi-objective trade-offs—balancing immediate line clears against long-term board stability—making it an ideal testbed for evaluating the sample efficiency, stability, and generalization capabilities of modern RL algorithms such as DQN and its variants.

=======================================

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "$\mathrm{Wb/m^2}$" or "webers per square meter", not "$\mathrm{webers/m^2}$". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "$\mathrm{cm^3}$", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a+b=\gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [7].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is “Heading 5”. Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract”, will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 2”, even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 2. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks . . .”. Instead, try “R. B. G. thanks. . .”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first . . .”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[2] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proc. AAAI Conf. Artificial Intelligence*, 2016, pp. 2094–2100.

[4] L. Lin, “Self-improving reactive agents based on reinforcement learning, planning, and teaching,” *Machine Learning*, vol. 8, no. 3–4, pp. 293–321, 1992.

[5] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.

[6] M. Hessel *et al.*, “Rainbow: Combining improvements in deep reinforcement learning,” in *Proc. AAAI Conf. Artificial Intelligence*, 2018, pp. 3215–3222.

[7] A. F. Karakovskiy and J. Togelius, “The Tetris benchmark for reinforcement learning,” in *IEEE Conf. Computational Intelligence and Games*, 2012, pp. 85–92.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.