Model and Dynamics-Free Reinforcement Learning for 2048

Sarvesh Babu Sri Jaladi

Introduction/Task

- 2048 is a tile-based combining game where equal tiles can be combined to create a single tile equal to the sum of their individual tile scores. The **aim is to combine (or equivalently survive) as many tiles as possible** and reach the highest possible tile without the game ending.
- We attempt to **create an agent** that can successfully **beat the game** (reach the 2048 tile) utilizing only **model-free and dynamics-free methods**.
- We use a **harsh environment**: **invalid moves result in game terminating** (unlike previous work).
- Starting from a very basic baseline, we continuously utilize our existing results to **combine different novel RL methods** in order to try and **specifically solve this task** and improve results **based on shortcomings**.

Related Work

- Existing work utilizing Reinforcement learning for 2048 has been **successful**, but relies on **model-based methods**, including beam search, monte-carlo tree search, and look-ahead methods. Some methods even reach up to the $2^{15} = 32768$ tile 72% of the time. While some **deep RL methods are used**, however, they all **rely on model and dynamics-based methods** as well to boost performance [5] [3].
- One paper works utilizing **model-free deep reinforcement learning** (using a q-network), but is only able to **reach the** 512 **tile inconsistently**. This paper utilizes one-hot encoding for each tile, the starting point for this project. With **additional model-based approaches**, such as approximate dynamic programming, they **reach the** 1024 **tile consistently** [2]

Baseline

- Baseline 1 is a **DQN** with **sparse reward system** using ϵ**-greedy exploration**
- This baseline gauges **difficulty of task** and how well **generic deep model-free RL** can perform
- Baseline 2 is **greedy moves** (utilizing common corner-based strategy)
- This baselines is meant to compare against good hard-coded performance.
- Shortcomings include **state representation** (one-hot encoding flattened), **replay buffer** representation of **larger tiles**, extremely **volatile performance**, **slow performance growth**.

Future Work

- **Better exploration** to potentially help stumble upon better strategies.
- **Weighted replay buffer** to weight certain transitions/episodes greater
- **Dense reward system** incorporating different components of the task.
- **Behavior cloning** techniques to help the model start stronger.
- **Artificial state creation** based on model's actions.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017. URL `http://arxiv.org/abs/1707.01495`.

[2] Antoine Dedieu and Jonathan Amar. Deep reinforcement learning for 2048 - mit. URL `https://web.mit.edu/people/adedieu/pdf/2048.pdf`.

[3] Hung Guei. On reinforcement learning for the game of 2048. *arXiv preprint arXiv:2212.11087*, 2022.

[4] Su Young Lee, Sung-Ik Choi, and Sae-Young Chung. Sample-efficient deep reinforcement learning via episodic backward update. *CoRR*, abs/1805.12375, 2018. URL `http://arxiv.org/abs/1805.12375`.

[5] Shilun Li and Veronica Peng. Playing 2048 with reinforcement learning. *CoRR*, abs/2110.10374, 2021. URL `https://arxiv.org/abs/2110.10374`.

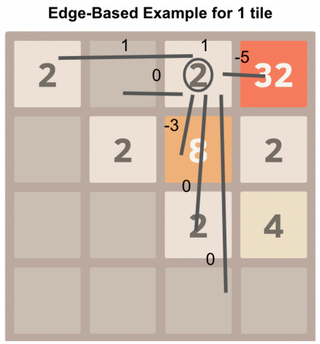
[6] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey, 2020.

Approaches

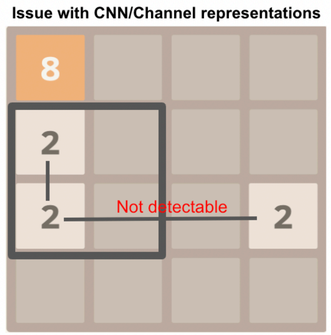
We utilize a **DQN** (Deep Q-Network) which takes a representation of the board as a state-input and gives 4 **values as output** (one for each action). The highest action is selected and training is done via **Mean Squared Error** on the target value: $r + \gamma \cdot \text{DQN}(s')$.

State Representations : We first focus on identifying state representations most conducive to fast learning. Note that all representations use log_2(tile value) for scaling. Target value (log) is appended to end of vector (in 1D case) or a channel of targets is appended (in 3D case). Utilizing related work and knowledge of the game, we identify 5 representations for states.

- **Basic**: take log values of 16 tiles and flatten to get $\mathbb{R}^{16}$ representation
- **One-Hot Flat**: convert each log tile into a one=hot vector (size 20) where the 1 is located at the index equal to the value of the log tile, and flatten to get $\mathbb{R}^{20*4*4}$ representation
- **Edge-Based Flat**: Utilize other log tiles in same row and column of current tile (based on which are (in)active) to generate a $\mathbb{R}^{8*4*4}$ representation
- **One-Hot Channeled/Edge-Based Channeled** : Same as one-hot and edge-based respectively but reshaped into sizes 20 x 4 x 4 and 8 x 4 x 4 respectively, creating 3D inputs.



[1,0,1,-5,1,-3,0,0] -> Note using log_2(tile)



CNN Filters can't detect all connections

Sparse Reward System: Reward $R(z)$ for reaching tile z for first time, $-y$ for each move (incentivizing fast progress), and $-v$ for an invalid move or terminal state. VERY conducive of suicidal agents! Avoid by satisfying the following constraint:

$$-y \cdot (\text{MaxMovesToTarget}(z)) + R(z) > -v : \forall z \in \{1, 2, 3, ...16\} \Rightarrow \text{Use } R(z) = R(z-1) \cdot 2.2$$

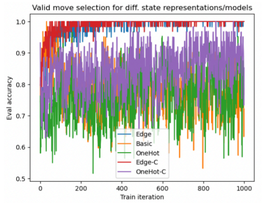
Curriculum Learning: Curriculum learning is a method of RL where the final task (reaching 2048 in this case) is slowly built up to in order to allow the model to learn effective manners of reaching each intermediate goal [6]. CL ideally helps the agent learn to effectively, quickly get up to each target, slowly learning better strategies that extrapolate better higher for higher targets.

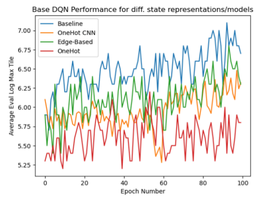
HER: HER (Hindsight Experience Replay) is a method to increase the replay buffer by selecting visited states as goals, increasing actions leading to larger magnitude rewards [1]. In 2048, this can help alleviate delayed reward issue by setting smaller target value goals that were reached.

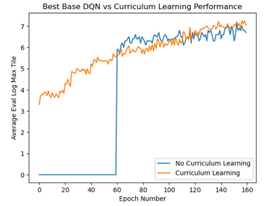
EBU: EBU (Episodic Backward Update) is a technique where during training on the replay buffer, episodes are propogated backwards [4]. EBU shows major boosts with delayed rewards. 2048 has extremely delayed rewards (actions can show impact hundreds of moves later).

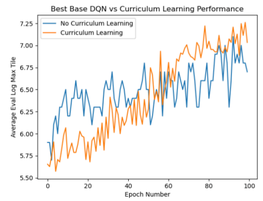
No EBU, Replay Buffer Inflation: When not utilizing EBU, there is no need to keep the replay buffer episodic. We thus add ALL actions from a state into the replay buffer since there are only 4. Only the taken action will continue the episode, but the other 3 actions will be simulated for one transition and this transition is added to the replay buffer.

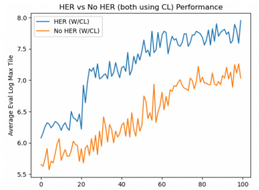
Results

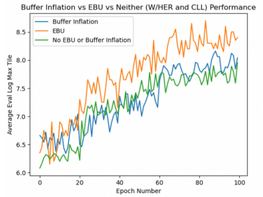


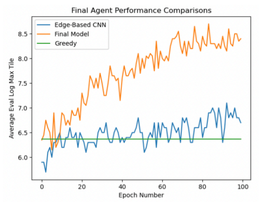












Experiment	Best Avg. Log(Max_Tile)	Best Tile
Greedy	6.37	128
DQN (OneHot)	6.2	128
DQN (OneHot CNN)	6.48	256
DQN (Edge-Based)	6.8	256
DQN (Edge-Based CNN)	7.1	256
Curriculum Learning (CL)	7.26	512
CL + HER	7.95	512
CL + HER + Buffer Inflation	8.16	512
CL + HER + EBU	8.7	1024

Conclusion Analysis

- **Identity function** over different cases is **difficult** for model to learn. Edge-based state representation + CNN is better helps alleviate problem and speeds up/benefits learning.
- **CNN works well** despite lacking all connections due to **congestion of tiles** past early stages.
- **Curriculum learning** does not immediately improve performance, but **improves** model **consistency and learning ability** at later stages through better foundational learning.
- **HER** provides expected benefits, **improving performance quicker** and allowing for **greater chance of reaching higher tiles**.
- **Replay Buffer Inflation** provides **smaller benefits** due to most added transitions being already learned well, since they are primarily invalid or very poor actions (especially at higher tiles).
- **EBU** provides **significant benefits** due to **extremely delayed rewards** in the system.
- **Utilized methods improve performance**, but still **struggle to get near model-based methods**. Due to **challenging state representations**, **delayed sparse rewards**, and **computational requirements**, this **task is extremely challenging in a model/dynamics-free environment**.