JOURNAL OF
EMERGING INVESTIGATORS

Article

Pruning replay buffer for efficient training of deep reinforcement learning

Gavin An[1], Sai Qian Zhang[2]
[1] Madison West High School, Madison, Wisconsin
[2] Harvard University, Cambridge, Massachusetts

SUMMARY

Reinforcement learning (RL) is a type of machine learning that develops artificial intelligence by training an algorithm through multiple generations to understand what strategies to use in various situations. RL has applications in virtually every field, from transportation to research. However, RL is limited in that it is very resource intensive, partially because of the necessity of a large replay buffer, which contains the data learned from each episode. This study provides knowledge on replay buffer reward mechanics to inform the creation of new pruning methods for improving RL efficiency. Specifically, we develop a novel approach designed to reduce storage complexity of the replay buffer and training data and thus improve model efficiency. We create three algorithms, Threshold Replay Buffer Pruning (TRBP), Cluster Replay Buffer Pruning (CRBP), and Inverse Threshold Replay Buffer Pruning (ITRBP), for this purpose, testing three contradicting theories on reward mechanics. We hypothesized that TRBP's theory would be the most conducive to real-world conditions, which our results corroborated. These results indicated that TRBP can achieve a 2-fold reduction in replay buffer size with only a 5% reduction in score, while CRBP and ITRBP performed much worse. This supported the hypothesis that TRBP's reward thesis is the most accurate out of the three algorithms, as well as demonstrated that TRBP is a potentially effective replay buffer pruning algorithm.

INTRODUCTION

Reinforcement learning (RL) is an area of machine learning concerned with the development of algorithms capable of making decisions that maximize a defined reward in a given situation. In the real world, RL leads the way in developing self-driving technology and industrial automation (1, 2). It is used in healthcare, where RL systems can recommend treatments to patients and develop new medications (3). RL can control a traffic grid, tell investors how to manage their portfolios, and more effectively and accurately inform governments on the state of their economy (4, 5, 6). As such, RL is an immensely powerful tool.

A major limiting factor in RL is its high running cost, as training an effective model can take weeks to months and require terabytes of memory (7). Often, it is impractical to train on mobile devices or low-end computers, and algorithms involved in complex environments may take months to train to an acceptable accuracy. A major reason for such resource cost is the often-high amount of training data necessary to effectively and quickly train an RL model (8). This represents a major problem with current machine learning algorithms, as high resource and time costs limit the feasibility of RL in complex situations and edge computing. In order to achieve efficient RL training, it is helpful to reduce the amount of training data used during the RL training process. A smaller amount of training data will enable RL models to train in less time and with less memory.

The training data of an RL model is stored in a replay buffer as entries, with each entry having the following composition: (S1, A, S2, R) (9). This entry is a datum explaining how an action (A) being taken in a certain state (S1) will lead to a new state (S2), and what reward (R) that change in states will cause. A large replay buffer, currently necessary for effective model training, creates a hardware inefficiency in RL, requiring both more memory and more time to train. Reducing the size of the replay buffer through pruning can remedy this inefficiency. This study focused on reward (R), how it relates to the improvement that the entry will bring to the model, and how this information can be applied to replay buffer pruning **(Figure 1).**

There has been much work in the past on the subject of pruning reinforcement learning algorithms concerned with decreasing the amount of data, time, and resources required to effectively train an algorithm. However, current techniques

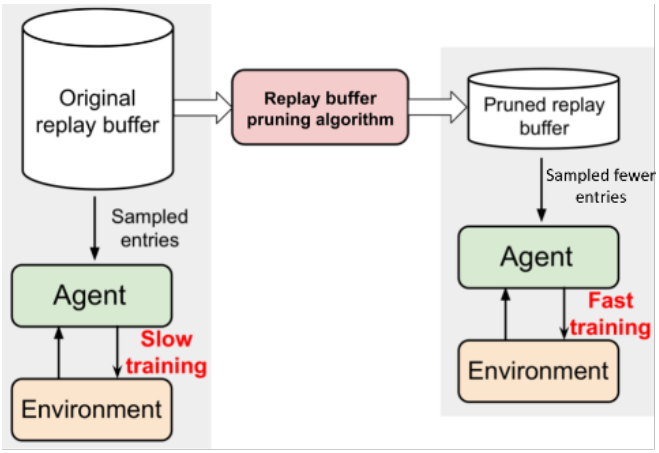

Figure 1: Explanation of replay buffer pruning and its theoretical effects on algorithm performance. Original replay buffer is larger and leads to slower training, but after it is passed through the pruning algorithm, it is condensed, leading to faster training due to fewer sampled entries.

DOI: https://doi.org/10.59720/23-068

focus on creating more efficient ways to train algorithms rather than pruning training data itself. Agapiou et al. developed a novel way of reducing training time with their Deep Q-Learning from Demonstrations (DQfD) algorithm (10). DQfD accomplishes this reduction by introducing demonstration data into the algorithm's pre-training phase, which allows the algorithm to imitate the demonstrator and apply its learned policy in the scene, giving it a head start in forming an effective score-maximizing strategy. DQfD introduces new, more effective training data as learning catalysts, but does not prioritize pruning existing training data to lower resource costs, leaving some room for efficiency improvements. Sokar et al. contributed to solving this problem as well with their dynamically adapting neural network that significantly reduced its size while improving its learning speed (11). They achieved their method, Dynamic Sparse Twin Delayed Deep Deterministic policy gradient (DS-TD3), through innovative sparse topologies and neural networks, allowing faster training while achieving a better performance than its counterparts. DS-TD3, like DQfD, also does not focus on pruning training data, but rather the neural networks themselves, leading to similar limitations. Group-Sparse Training (GST) focuses on achieving better performance on mobile devices, which are primarily limited by memory bandwidth (12). Using GST and reward-aware pruning (RWP), the compression ratios and stability of training are simultaneously increased. GST and RWP tackle the same mobile device efficiency problem as this paper does but focus on improving model compression methods rather than pruning the replay buffer of such algorithms. Livne et al. developed one of the first techniques, Policy Pruning and Shrinking (PoPS), which can train DNNs capable of retaining strong performance after each pruning process (13). This performance is possible due to the ability to identify and preserve the most important information of the model while removing redundancies, creating a compact yet effective version of the initial DNN. PoPS can effectively prune DNNs, but just as with other models, it lacks the ability to prune training data in the same way. The Rigged Reinforcement Learning Lottery (RLx2) applied ultra-sparse networks to achieve model compression (14). A variety of different mechanisms, such as a dynamic-capacity replay buffer and gradient-guided topology search scheme, work in tandem to achieve such results. RLx2 again focuses on the neural network itself, not on the similarly important training data.

In comparison, this paper focuses on a virtually unexplored field in AI, aiming at pruning training data contained in the replay buffer during the exploration phase rather than during the training phase. As such, the results, methodology, and algorithms of our paper could not be easily compared to those of other papers.

To test the importance of the characteristics of the reward, we built three novel pruning algorithms based on three different and contradictory theories about replay buffer reward. We built the first algorithm, Cluster Replay Buffer Pruning (CRBP), on the theory that every reward is equally valuable to the model, no matter its absolute value. Thus, the pruning algorithm ensures that representative entries are kept with high and low absolute value rewards. Intuitively, this theory could be justified as both extreme and neutral rewards having an important part in a model, with the extremes giving big-picture guidelines and neutral reward entries providing finer-tuned tweaks.

We built the second algorithm, Threshold Replay Buffer Pruning (TRBP), on the theory that the higher the absolute value of the reward or punishment, the more valuable the reward is to the training of the model. Thus, the pruning algorithm removes low absolute value reward entries first. Intuitively, this theory could be justified as higher rewards show what strategies work and what strategies really do not. This is more valuable to the training of a model than an entry that shows a neutral strategy.

We built the third algorithm, Inverse Threshold Replay Buffer Pruning (ITRBP), on the theory that the higher the absolute value of a reward, the less valuable the reward is to the training of the model. Thus, the pruning algorithm removes high absolute value reward entries first. Intuitively, this theory could be justified as extreme rewards being outliers, and less likely to be applicable to a model's situation than neutral rewards. More detailed information on all used models can be found in the appendix.

We chose to create three separate algorithms because it would allow us to test the performance of each type of reward entry (high absolute value of R to low) individually, which would allow us to conclude which was most important to a model's performance. We hypothesized that TRBP would be the most effective because it prioritizes more extreme value entries. We believed the theory behind it was most plausible, and that extreme entries provide more information to the model on which strategies were successful than more neutral value entries.

We tested the algorithms using a game called Lunar Lander, provided by OpenAI, in which the player controls a small lander and attempts to land it on the moon (15). Landing closer to the targets and at a more level angle results in a higher score. After each episode, the game resets so the model can play again. We found that when TRBP pruned 50% of the replay buffer, the score decreased by only 5%, while the scores of CRBP and ITRBP were noticeably lower. These results demonstrated the viability of reward-based replay buffer pruning as a pruning strategy and showed that the theory behind TRBP's algorithm is most accurate.

RESULTS

We tested the algorithm with the Lunar Lander game. One episode represents one training cycle, in which 64 games of Lunar Lander are played before the model is updated to learn from its experience. We ran these algorithms for 300 episodes, 3 times each, and took the average of the three at 25-episode intervals. We set the algorithms to have a pruning ratio of 50%, meaning that 50% of the entries in the replay buffer would be removed each update. Comparing the effectiveness and scores of these pruning algorithms with the scores of the unpruned algorithm and a pruning algorithm which randomly removes entries, Random Replay Buffer Pruning (RRBP), as controls indicated which reward theory was more accurate.

DOI: https://doi.org/10.59720/23-068

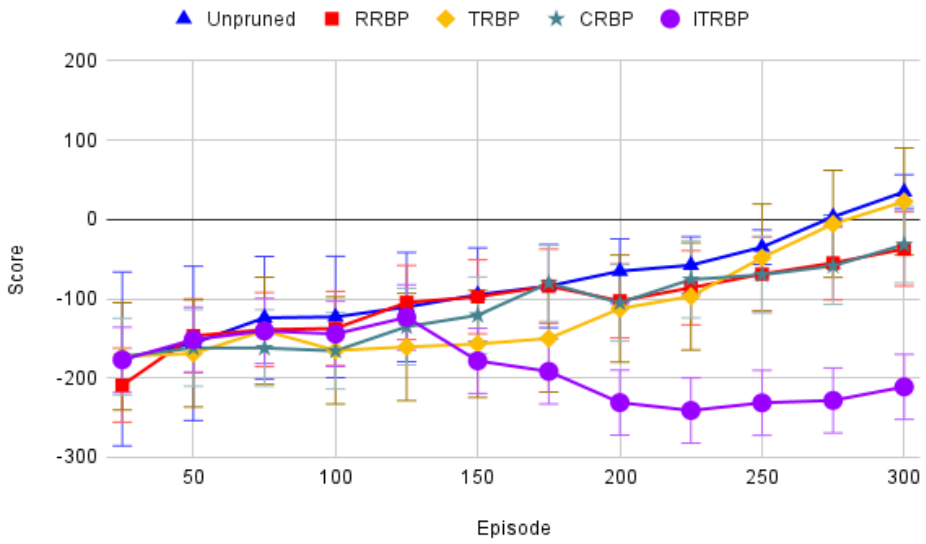

Figure 2: The unpruned model and TRBP achieved the best scores over time. Line graph showing all algorithms' performance in the form of score. RL was run on Lunar Lander for 300 episodes with various pruning buffer algorithms. Score data was collected and averaged from 3 runs every 25 episodes, then plotted. Error bars represent standard deviation.

Though most of the algorithms started by achieving around the same in-game score, we saw more variations as the game proceeded and the algorithms began to learn (Figure 2). The unpruned algorithm improved its score by 210.72 over the course of 300 episodes, at an average rate of 0.70 per episode. It reached a positive score by episode 275 and an end score of 34.8. RRBP improved its score by 171.74 over the course of 300 episodes, at an average rate of 0.57 per episode. It never reached a positive score and achieved an end score of -37.23. CRBP improved its score by 141.2 over the course of the 300 episodes, at an average rate of 0.47 per episode. It never reached a positive score and had an end score of -31.61. TRBP improved its score by 195.3 over the course of 300 episodes, at an average rate of 0.65 per episode. It reached its first positive score at 300 and had an end score of 22.7. ITRBP saw a score decrease of 34.09 over the course of 300 episodes, at an average rate of -0.11 per episode. It never reached a score greater than -100 and had an end score of -211.09.

We calculated standard deviation by determining how much the average score of a given algorithm's run deviated from the average score of all three runs. The standard deviations of the first four algorithms (unpruned, RRBP, CRBP, TRBP) were similar: 17.97, 22, 19, and 26.11 respectively. However, the average standard deviation of ITRBP was much higher, reaching 49.41. It is unclear why the neutral-prioritizing algorithm varied so dramatically compared to the other algorithms, especially RRBP, but the reason could be tied to the poor performance of the algorithm. Perhaps luck is a bigger factor in the score of an algorithm with few good strategies than one that has a clearer plan of how to act.

Comparing these algorithms established TRBP as the most effective at increasing RL efficiency. TRBP ended with a slightly lower score than the unpruned algorithm, but with a much higher score than cluster pruning and random pruning, which both had similar scores. All performed dramatically better than inverse threshold pruning **(Figure 2)**.

By the end of the 300 episodes, TRBP performed better than RRBP. We saw a clear improvement of almost 60 between the end scores of TRBP and RRBP. TRBP pruning achieved a roughly 14% higher total improvement, and a score after 25 episodes (the first recorded, or initial score) that was 36.37 higher than random pruning. TRBP also performed better than cluster pruning in almost all areas. TRBP achieved a 54.31 increase in end score, with a 38% higher total improvement, though both algorithms had almost identical scores after 25 episodes. In comparison to the unpruned algorithm, TRBP only performed slightly worse, despite having half the training data. While total improvement was around 7% less, the end score of TRBP was only 12.1 less than the unpruned algorithm, with an initial score difference of only 3.32. In contrast, RRBP saw a decrease of 18.4% in total improvement, with an end score difference of 72.03 and an initial score difference of 33.05. Meanwhile, the ITRBP pruning algorithm performed poorly. Despite having a similar initial score to other algorithms, its score consistently deteriorated after the 150-episode mark. It is the only algorithm to have achieved a lower end score than initial score, and to have never risen above -100 **(Figure 2)**.

DISCUSSION

The results of our experiments corroborate our hypothesis that TRBP's reasoning is most accurate under the tested conditions. Since the algorithm built on prioritizing extreme rewards performed better than those prioritizing all rewards equally, it can be inferred that in Lunar Lander, extreme rewards are more helpful to optimizing score than neutral rewards are. This conclusion is further strengthened by the fact that ITRBP, which operates with a theory opposite to TRBP and only learns on neutral reward entries, performed very poorly, likely for the same reason. On the other hand, the performance of CRBP, which values both types of rewards equally, was not very different from that of RRBP. Additionally,

DOI: https://doi.org/10.59720/23-068

this data shows that TRBP is a relatively viable algorithm in the conditions tested to reduce the amount of training data required for a model to learn. It performs at a level similar to the unpruned algorithm with half the training data and runs much more accurately than RRBP, legitimizing itself as well as replay buffer pruning.

There were two abnormalities in the results. TRBP only began improving over RRBP from Episode 230 onwards **(Figure 2).** This may be due to the nature of reinforcement learning. At the beginning of the simulation, all entries in the replay buffer will have very low rewards, since the algorithms have not created a good strategy yet. Since TRBP prioritizes high absolute value of rewards, it will always prioritize the extremely low rewards rather than the entries with improved rewards that grow closer to zero but are still negative. However, as the simulation progresses, the algorithm's rewards will also increase, until it is able to prioritize high value rewards as well as low entry rewards. This causes it to improve at a much faster rate. Another abnormality is that the standard deviation of ITRBP was higher than that of other algorithms. This could be due to ITRBP's poor performance. If the algorithm is unable to create a good strategy, luck may play a far greater role in its performance than that of an algorithm with a good strategy.

There are several potential biases that could have influenced the results of this experiment. First, each algorithm was only run three times, which may not be enough to sufficiently remove the element of random chance within the experiment. Additionally, the algorithms were only run for 300 episodes each, which may not have been long enough to determine the true effectiveness of each algorithm. The environment of the game, Lunar Lander, is also a limitation. Lunar Lander is a straight-forward game in that the parameters for success are clear – angle of landing, and distance to target. Minimizing both will always lead to a higher score, and thus is a better strategy. In more complex games, like racing or strategy games, the parameters of success are less clear. The consequences of important choices in these games may take longer to fully realize, and thus seem neutral at first. Thus, Lunar Lander's simplicity may make TRBP more helpful than it would be in more complex games. Finally, the pruning ratio (that is, the percentage of the replay buffer that is removed) was tested at only 50%. The various algorithms could perform differently at different levels of pruning, such as 25% or 90%. If replicated, future experiments should increase the number of runs per algorithm, increase the number of episodes per run, and vary the pruning ratio and environment each algorithm was tested on.

Future work could be done on developing more complex algorithms around replay buffer pruning, as well as discovering new mechanics and interactions between the various parts of the replay buffer to better inform these future algorithms. As for TRBP, it could be further tested and expanded upon to become more effective. Replay buffer pruning is a powerful tool in the RL sphere that should be investigated further, especially to increase efficiency and decrease the size of RL models.

This study touched upon the power of replay buffer pruning and the mechanics of reward within the replay buffer. The study also shows the effectiveness of TRBP as a pruning model under the conditions studied, demonstrates that extreme reward entries are more valuable to the training of a model than neutral reward entries, and serves as a proof of concept for future work on this topic.

MATERIALS AND METHODS

The baseline machine learning model was created using PyTorch in Google Colab, with a batch size of 64 and a buffer size of 100,000. In order to increase the speed at which the code would run, the provided GPU notebook setting in Colab was used. A basic RL model and algorithm were created, consisting of three linear layers: an input, a hidden, and an output. Then, four novel pruning algorithms were applied and tested on the model: the three experimental algorithms, and the control algorithm that randomly pruned. These algorithms were built and coded in the same Python environment that the baseline model was created in, and more details about them can be found in the appendix. All algorithms were tested with a 50% pruning ratio. This ratio was achieved by applying an r parameter (the maximum or minimum absolute value of a reward before pruning for ITRBP and TRBP respectively) of about 1.6 for ITRBP and TRBP, while four clusters were used for CRBP. Each algorithm was run 3 times to help eliminate the randomness of the experiment, with each run consisting of 300 episodes, with data collected and averaged every 25 episodes. Results were compared to each other and the unpruned algorithm. More details on the model can be found on the GitHub page in the appendix.

Statistics

We graphed and recorded the data using Google Sheets. We calculated standard deviation by determining how much the average score of a given algorithm's run deviated from the average score of all three runs.

Received: April 3, 2023
Accepted: October 17, 2023
Published: October 25, 2023

REFERENCES

1. Heaven, Will Douglas. "The Big New Idea for Making Self-Driving Cars That Can Go Anywhere." MIT Technology Review, MIT Technology Review, 27 May 2022, www.technologyreview.com/2022/05/27/1052826/ai-reinforcement-learning-self-driving-cars-autonomous-vehicles-wayve-waabi-cruise/. Accessed 10 Mar, 2023.
2. "Business value of autonomous systems" Microsoft AI, www.microsoft.com/en-us/ai/autonomous-systems-solutions. Accessed 10 Mar, 2023.
3. Liu, Siqi, et al. "Reinforcement Learning for Clinical Decision Support in Critical Care: Comprehensive Review." Journal of Medical Internet Research, vol. 22, no. 7, Jul. 2020, https://doi:10.2196/18477
4. Tan, Tian, et al. "Cooperative Deep Reinforcement Learning for Large-Scale Traffic Grid Signal Control." IEEE Transactions on Cybernetics, vol. 50, no. 6, 2020, pp. 2687–2700. https://doi:10.1109/tcyb.2019.2904742
5. Mohammad, Shareefuddin, et al. "Embracing advanced AI/ML to help investors achieve success: Vanguard

DOI: https://doi.org/10.59720/23-068

Reinforcement Learning for Financial Goal Planning." Arxiv, Oct. 2021, https://doi:10.48550/arXiv.2110.12003
6. Charpentier, Arthur, et al. "Reinforcement Learning in Economics and Finance." Computational Economics, Apr. 2021, https://doi:10.1007/s10614-021-10119-4
7. Sharir, Or, et al. "The Cost of Training NLP Models: A Concise Overview." Arxiv, 19 Apr. 2020, https://doi:10.48550/arXiv.2004.08900
8. Yang, Shuo, et al. "Dataset Pruning: Reducing Training Data by Examining Generalization Influence." Arxiv, 27 Feb. 2023, https://doi.org/10.48550/arXiv.2205.09329
9. "Replay Buffers." TensorFlow, www.tensorflow.org/algorithms/tutorials/5_replay_buffers_tutorial.
10. T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Learning from demonstrations for real world reinforcement learning," arXiv preprint arXiv:1704.03732, 2017.
11. G. Sokar, E. Mocanu, D. C. Mocanu, M. Pechenizkiy, and P. Stone, "Dynamic sparse training for deep reinforcement learning," arXiv preprint arXiv:2106.04217, 2021.
12. J. Lee, S. Kim, S. Kim, W. Jo, and H.-J. Yoo, "Gst: Group-sparse training for accelerating deep reinforcement learning," arXiv preprint arXiv:2101.09650, 2021.
13. D. Livne and K. Cohen, "Pops: Policy pruning and shrinking for deep reinforcement learning," IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 4, pp. 789–801, 2020.
14. Y. Tan, P. Hu, L. Pan, and L. Huang, "Rlx2: Training a sparse deep reinforcement learning model from scratch," arXiv preprint arXiv:2205.15043, 2022.
15. Sweeney, Tim. "Lunar Lander - Open Ai." Lunar Lander - Open AI, Weights & Biases, 15 Oct. 2020, wandb.ai/timssweeney/lunar-lander/reports/Lunar-Lander-Open-AI--VmlldzoyNzg5NjE.

APPENDIX

Github Code Link: https://github.com/destroyer000lucky/Gymnasium/blob/main/ReplayBufferPruningCode.ipynb

Random Reply Buffer Pruning (RRBP)

RRBP runs through each entry in the replay buffer and generates a random number between zero and one. In the case of a 50% pruning ratio, if the number was above 0.5, then the entry would be deleted. Otherwise, it would continue to be used for training the model.

Cluster Replay Buffer Pruning (CRBP)

CRBP takes in two inputs: the number of clusters (groups that the replay buffer is split into), and the percentage pruned from each one. The algorithm then sorts entries based on reward into a corresponding cluster, each representing an equal interval between zero and one as well as one cluster each for rewards greater than one or less than negative one. Then, each cluster is randomly pruned to the percentage specified by the user.

Threshold Replay Buffer Pruning (TRBP)

TRBP takes in a threshold variable r. It runs through each entry in the replay buffer, comparing its reward to the r. If the reward is between -r and r, then the entry is removed. Otherwise, it continues training the model. In this experiment, r was selected to be a constant value achieving a pruning ratio of 50% in order to be consistent with the other algorithms. In this case, this constant was 1.6.

Inverse Threshold Reserve Buffer Pruning (ITRBP)

IRBTP takes in a threshold r. It runs through each entry in the replay buffer, comparing its reward to the r. If the reward is not between -r and r, then the entry is removed. Otherwise, it continues training the model. In this experiment, r was selected to be a constant value achieving a pruning ratio of 50% to be consistent with the other algorithms. In this case, this constant was 1.6.