Graph Backup: Data Efficient Backup Exploiting Markovian Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

Bootstrapped value estimation has become a widely adopted ingredient for modern reinforcement learning algorithms. These methods compute a target value based on observed data and predictions for future values. The approximation error of the target value, which comes from stochastic dynamics and inaccurate predictions, can significantly affect the data efficiency of RL algorithms. Multi-step methods, such as n-step Q learning and TD(λ), leverage the chain structure of the data, alleviating the effect of inaccurate predictions and allowing credit assignment across a longer time horizon. However, the main limitation of such multi-step methods is that they fail to exploit the graph structure of certain MDPs by only treating each trajectory independently, resulting in an inadequate estimate of the target value that misses the intersections between multiple trajectories. In this paper, we propose to treat the transition data of an MDP as a graph, and define a novel backup operator exploiting this graph structure. Comparing to multi-step backup, our graph backup method allows counterfactual credit assignment, and can reduce the variance that comes from stochastic environment dynamics. Our empirical evaluation on MiniGrid and Minatar shows graph backup can greatly improve data efficiency compared to one-step and multi-step backup.

1 Introduction

Deep Reinforcement Learning (DRL) methods have achieved super-human performance in a varied range of games [Mnih et al., 2015, Silver et al., 2016, Berner et al., 2019, Vinyals et al., 2019]. However, existing DRL methods require a huge amount of online interactions to solve each task. Modern DRL methods are developed under the assumption that we have a fast and perfect simulator. However, this assumption is often taken to extremes, resulting in a huge gap in the number of samples used by methods on standard DRL benchmarks and the number available for real-world tasks. In order to compete for state-of-the-art, the number of overall interactions used to train DRL methods is climbing quickly. However, most of the real-world tasks are hard to simulate, and generating new interaction data can be quite expensive. This makes it crucial to develop data-efficient RL approaches that solve sequential decision-making problems with limited environment interactions.

One of the key challenges to improve data efficiency is how to perform credit assignment, that is, how to assign rewards to previous actions. To implement credit assignment, value-based RL methods (such as DQN [Mnih et al., 2015]) learn a bootstrapped Q-value function with states and actions as arguments. Most policy-based methods employ an actor-critic design, which means that they also learn a value function utilising bootstrapping to increase the data efficiency for credit assignment. Bootstrapping methods combine the information from observed data and future value estimates in order to produce a better prediction of the value. This improved prediction is called the backup target and the process of computing this target is called backup. The value function is essentially trained with a supervised loss to mimic the backup target, and therefore the computation of this backup target

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

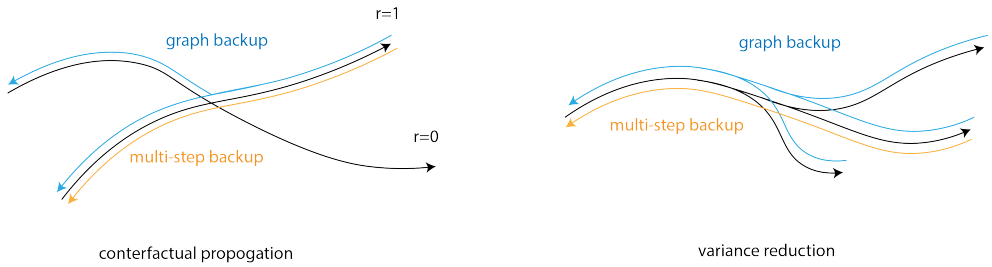

Figure 1: Benefits of introducing the graph structure into backup. The black curves represent the experienced trajectories. The blue curves represent the reward propagation paths of graph backup and orange curves are the paths of multi-step backup. The graph backup can propagate rewards in counterfactual paths and aggregate information branching from the same starting state. However, the multi-step methods can only follow existing trajectories.

is important, as it influences the quality of the learned value function and hence the overall policy. In this paper we aim to improve bootstrapping methods for target value estimation, improving credit assignment and hence data efficiency. The key idea is to leverage the Markovian property of the MDP that generates the data, merging identical states in different trajectories. The state merging process will produce a data graph, which is then used to guide the target value estimation.

Vanilla DQN uses a one-step backup target, where the target is given by the next step reward and next state value estimate. This has several disadvantages: first, if the next step value estimation is inaccurate, this target can be far away from the ground truth; second, from the credit assignment perspective, the reward can only be propagated one-step forward until the target network gets updated. To speed up credit assignment over longer temporal spans, multi-step methods propagate information through a segment of experience trajectory [Moriarty and Miikkulainen, 1995, Hessel et al., 2018a, Sutton and Barto, 2018, Hernandez-Garcia and Sutton, 2019]. However, the structure of the data in RL contains more information rather than independent trajectories. Given the standard Markovian assumption on the transition function in an MDP, the transition probability is independent of previous states. Therefore, after observing the trajectory $(s_1, a_1, s_2, a_2, s_3, a_3)$ and $(s'_1, a'_1, s_2, a'_2, s'_3, a'_3)$ we can infer $(s_1, a_1, s_2, a'_2, s'_3, a'_3)$ (a cross-over of the two trajectories) is a possible trajectory. To fully utilise the data sampled from an MDP, we should exploit this structure and treat the data as a bipartite graph with state and action nodes.

In this work, we propose an extension to the multi-step target, which leverages the markovian property and graph-structured nature of many MDPs to improve bootstrap target estimation. The method, named *graph backup*, has two advantages over multi-step targets, illustrated in Fig. 1. The first is the ability to propagate the rewards in a counter-factual manner: Suppose two trajectories have a crossover and one of the trajectories received a reward in one of its later states. The multi-step backup can only propagate the reward within the states in the current trajectory, while the graph backup can bring rewards to the earlier states of both trajectories. In this case, the graph backup propagates reward through an imaginary trajectory that is the recombination of two existing trajectories. The second advantage of graph backup is the reduction in the variance of the target value, enabled by averaging the possible downstream future trajectories, branching from the same upstream trajectory. The branching itself can either be caused by the stochasticity of the environment dynamics or the stochastic policy.

In this work, we propose two implementations of graph backup, building on DQN: *GB-limited* conducts a depth- and breadth-limited expansion of the data graph, starting from the source state. This is a strict extension of a multi-step method called tree backup [Precup et al., 2000], where the tree is not a state tree but a state chain with action leaves. The second implementation, *GB-Q*, converts the target computation into the problem of estimating the optimal q values in a tabular MDP. By running tabular Q-learning on the replay buffer, GB-Q is able to compute the infinite-step graph backup targets. Note that GB-Q does not construct an explicit graph so can be easier to implement.

To summarise, the contributions of this paper are three-fold: (1) we propose graph backup, a novel backup method that leverages the graph structure of the replay data to produce a more accurate target

value, (2) we propose two implementations of graph backup: GB-limited and GB-Q, and (3) we test the performance of graph backup empirically in MiniGrid and Minatar tasks, showing better sample-efficiency and final performance than the one-step and multi-step backup baselines.

2 Related Work

The idea of multi-step backup algorithms (e.g. TD(λ), n-step TD) dates back to early work in tabular reinforcement learning [Sutton, 1988, Sutton and Barto, 2018]. Two approaches to multi-step targets are n-step methods and eligibility trace methods. The n-step target method is a natural extension to using a one-step target that brings the rewards and value estimations of n steps into future into consideration. For example, the n-step SARSA [Rummery and Niranjan, 1994, Sutton and Barto, 2018] target for step t is simply the sum of n-step rewards and the value in $t+n$, namely $R_{t+1} + R_{t+2} + ... + R_{t+n-1} + V(S_{t+n})$. Our graph backup is an extension of an n-step backup target, tree backup, which will be described in preliminaries.

Eligibility trace methods instead estimate the λ-return, which is an infinite weighted sum of n-step returns. The advantage of the eligibility traces method is it can be computed in an online manner without explicit storage of all the past experiences, while still computing accurate target value estimates. However, in the context of off-policy RL, the eligibility traces is not widely applied because of the use of a reply buffer means all past experiences are already stored. In addition, the eligibility traces are designed for the case with a linear function approximator, and when the value function is a neural network, it is non-trivial to apply eligibility traces. Recently, van Hasselt et al. [2021] proposed an extension of eligibility traces methods called expected eligibility traces. Similar to graph backup, it allows information propagation across different episodes and thus enables counterfactual credit assignment and variance reduction. However, similar to the original eligibility traces methods, it is a better fit for the online and linear case, whereas graph backup is designed for the non-linear and off-policy cases.

Since a learned model can be treated as a distilled replay buffer, we can view model-based reinforcement learning as related to our work [Schrittwieser et al., 2020, Hessel et al., 2021, Farquhar et al., 2018, Hafner et al., 2021, Kaiser et al., 2020, Ha and Schmidhuber, 2018]. The model can be used for planning or generating imaginary training data, which both enable counterfactual credit assignment. Methods doing explicit planning [Schrittwieser et al., 2020, Farquhar et al., 2018] usually use the model for a tree expansion and compute a backup target for their value functions, which is similar to graph-backup where we instead use real data for the expansion. Deep model-based methods can deal with high-dimensional data leveraging the generalization of the learned model. However, training an accurate model itself can be difficult and the model error can keep accumulating during the tree expansion. In addition, dealing with the stochasticity for model-based methods is non-trivial while stochasticity is well-handled by graph backup methods by nature.

Zhu et al. [2020] proposed the method of associative memory to leveraging graph-structured data in an MDP in the literature of episodic reinforcement learning. Episodic reinforcement learning is a psychobiological-inspired field that tries to store experience that gains high returns and use them for better decision making [Min and Kim, 2017, Pritzel et al., 2017]. Associative memory treats the experience data as a graph rather than unrelated items [Zhu et al., 2020]. This allows counterfactual reward propagation and can improve data efficiency. Since based on different fields, graph backup has many differences from the associative memory method. Here we only list a few: 1) Associative memory specifies a one-to-one mapping from state-action pairs to next states and is valid only for the deterministic case. While graph backup not only works in stochastic cases but is also able to reduce the variance of the target value. 2) The historical highest return in associative memory is not generated in a bootstrapping way and only comes from rewards actually being seen. 3) The highest return is used in an auxiliary loss term rather than a backup target.

3 Preliminaries: One-step and Multi-step Backup

To clarify notation, given an MDP $\mathcal{M}$ we denote $\mathcal{A}$ as the action space, $\mathcal{S}$ to be state space, $\mathcal{R} \subset \mathbb{R}$ to be reward space. Capital $A_t \in \mathcal{A}$ or $S_t \in \mathcal{S}$ are used to denote the specific actions/states observed. We can then denote a trajectory of states, actions and rewards as $\tau = (S_1, A_1, R_1, S_2, A_2, R_2, ...)$.

For a transition (S_t, A_t, R_t, S_{t+1}) the loss function of DQN methods is defined as the mean square loss between the predicted q-value and the backup target G^{A_T} for (S_t, A_t):

$$L(\boldsymbol{\theta}|S_t, A_t) \stackrel{\text{def}}{=} \left|q_{\boldsymbol{\theta}}(S_t, A_t) - G^{A_t}\right|^2, \tag{1}$$

where $q_{\boldsymbol{\theta}}$ represents the online network parametrized by $\boldsymbol{\theta}$. The backup target is an estimation of the optimal q value $q^*(S_t, A_t)$. Vanilla DQN uses one-step bootstrapped backup, which makes gradient descent an analog to the update of the tabular Q learning:

$$G^{A_t}_{t:t+1} \stackrel{\text{def}}{=} R_{t+1} + \gamma \max_{a'} q_{\boldsymbol{\theta}'}(S_{t+1}, a'). \tag{2}$$

The one-step target makes the propagation of the reward to previous states slow and the use of a separate frozen network amplifies the problem of slow credit assignment. Using a separate target network with parameters θ' to compute the target G is a key ingredient to stabilise the training of the DQN, and the weights of the target network are synchronized with the (learning) online network periodically rather than being updated each training step. For example in Rainbow [Hessel et al., 2018b], this target network update frequency is set to be every $t_{\text{target}} = 8000$ steps. However, using a slowly updated target network makes propagation of reward further delayed: even ideally, to propagate a reward n steps forward, $n \times t_{\text{target}}$ number of training steps are needed. This motivates the use of multi-step target in DQN, which achieved better data efficiency [Hessel et al., 2018b, Hernandez-Garcia and Sutton, 2019], possibly at the cost of biases value targets (in the case of n-step Q-learning).

One example of an off-policy multi-steps target is the Tree Backup target [Precup et al., 2000]. Tree backup is designed for general purpose off-policy evaluation, meaning it aims to estimate the value of any target policy π by observing the behaviour policy μ. When the target policy is the optimal policy given by $q_{\boldsymbol{\theta}'}$, tree backup recursively applies one-step backup to the trajectory, bootstrapping with the target value network when the input action a isn't that taken in the trajectory (A_t):

$$G^a_{t:t+n} \stackrel{\text{def}}{=} \begin{cases} R_{t+1} + \gamma \max_{a'} G^{a'}_{t+1:t+n}, & \text{if } t < n \text{ and } a = A_t \\ q_{\boldsymbol{\theta}'}(S_t, a), & \text{otherwise} \end{cases}. \tag{3}$$

It is worth noting that, despite what its name might suggest, the tree backup does not expand a tree of transitions, and it still, like other multi-step methods, only leverages the chain structure of trajectories. The method is called tree backup because it has leaves corresponding to the actions that were not selected in the current trajectory. In Fig. 3, we showed the backup diagram of the 3-step tree backup, where yellow squares are these leaf actions.

4 Graph Backup

In this section, we will introduce a new backup operator, *graph backup*, extending tree backup. This graph backup allows counterfactual credit assignment and variance reduction while also having the benefits of multi-step backup. We will also propose two implementations of graph backup, *GB-limited* and *GB-Q*, with different computational complexity and robustness against value estimation error of other states.

4.1 Limitations of multi-step backups

The multi-step backup target leverages the chain structure of trajectory data. However, treating the data from several trajectories as independent chains does not fully exploit the Markovian property of the data. For one-step and multi-step backup, the backup target for the same state can vary according to the following states in its trajectory. However, for an MDP, if two observations are the same, then their corresponding state transition probability and values should also be the same, independent of previous states. Therefore, in principle, these two states should be able to share the information of their trajectories to build a more accurate target estimation.

For example, consider a 5 step MDP with two actions a and b, as illustrated in Fig. 2. The agent observes two trajectories with action (a, a, a, a) and (b, b, b, b), where states overlaps in step 1, 3 and 5. The two trajectories can be merged into a graph since they share some state nodes. After merging

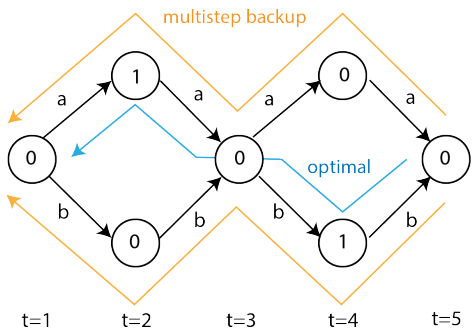

Figure 2: An example of graph-structured data where each node is a unique state and numbers inside indicate the rewards received. There are two actions a and b, labelled on the edges. The path of reward propagation for multistep backup is marked as orange and the optimal path is blue. The optimal backup path can propagate rewards generated in two different trajectories to the state in step 1.

these two trajectories, we can get two extra possible trajectories: (a, a, b, b) and (b, b, a, a). Setting $\gamma = 1$ and $\forall(s, a), q_{\boldsymbol{\theta}'}(s, a) = X$, X is a random variable depended on how $\boldsymbol{\theta}'$ is initialised. Here we assume $X \sim \mathcal{U}(-1, 1)$ to simplify the example.

Now we try to compute the backup target at step 1. For action a, the single step target $G^a_{1:2} = 1 + X$ and multi-step target $G^a_{1:5} = 1 + X$ gives the same result. For action b, since multi-steps target managed to propagate reward seen in step 4, the target would be $G^b_{1:5} = 1 + X$, which is different from one-step backup target $G^b_{1:2} = X$. Although a multi-step target seems to be more efficient than the one-step target, by looking at the graph-structured data, it is not difficult to see a better target would be $2 + X$, which corresponds to the action sequence (a, a, b, b).

In fact, when we merge the two trajectories into a graph, we automatically get the information about two more trajectories (a, a, b, b) and (b, b, a, a). Such information is derived from the Markovian property and is not leveraged by the multi-step target. The only way for multi-step DQN to exploit the trajectory crossover information is to wait until the target network gets updated so that the value of (S_3, b) is corrected: $q_{\boldsymbol{\theta}'}(S_3, b) = 1$. This is similar to the situation where the one-step target is dealing with multi-step reward propagation. The update of the target network is slow and there is no guarantee that DQN can perform this cross trajectory information exchange robustly.

4.2 Introducing Graph Backup

Inspired by the previous example, we propose the graph backup operator, to propagate temporal differences across the whole data graph rather than a single trajectory. The differences between one-step backup, multi-step backup and graph backup are illustrated in Fig. 3.

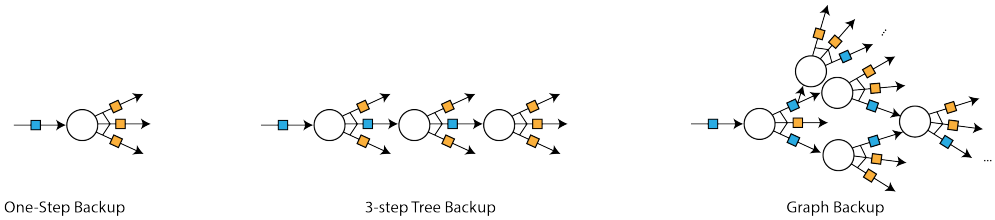

Figure 3: Three backup diagrams for different targets. The blue squares represent the state-action pairs that have been tried. The orange squares are actions where the target net evaluation happened. The circles are states in the trajectory/data graph. The links between arrows represent the max operator. For the graph backup, the same state action pair can have different following states because of the transition stochasticity.

We want a backup method that can work with stochastic transitions, which means a single state-action pair can lead to different states. This means it's not obvious how to perform recursive backups to the next state, as there could be multiple next states. We propose to estimate the transition probability to each state using visitation counts, and to use the estimated transition probabilities to compute the empirical mean over all possible state value estimates weighted by the likelihood of transitioning to that state.

Denoting the set of all seen transitions to be $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S}$, a counter function $f : \mathcal{T} \to \mathbb{N}^+$ maps each transition $T = (s, a, r, s')$ to its frequency $f(T)$. The graph backup target for a state-action pair (s, a) is then the average of recursive one-step backup of all outgoing transitions, and similar to tree backup, if the (s, a) has not been seen, the target would be estimated directly by the target network. Define $\mathcal{T}_{s,a} \stackrel{\text{def}}{=} \{(\hat{s}, \hat{a}, \hat{r}, \hat{s}') \in \mathcal{T} | \hat{s} = s, \hat{a} = a\}$. We can then define the *graph backup* value estimate as

$$G_s^a \stackrel{\text{def}}{=} \begin{cases} \sum_{T \in \mathcal{T}_{s,a}} \frac{f(T)}{c(s,a)} \left(r + \gamma \max_{a'} G_{s'}^{a'} \right) & \text{if } c(s,a) > 0 \\ q_{\boldsymbol{\theta}'}(s, a) & \text{otherwise} \end{cases} \tag{4}$$

where $c(s, a) = \sum_{T \in \mathcal{T}_{s,a}} f(T)$ is the normalizer.

In Eq. (4), the graph structure does not seem to appear explicitly. To better gain intuition of the graph backup, in Appendix B we explicitly describe the data graph generated from an MDP and link that to Eq. (4). In practice, maintaining this explicit graph structure can speed up the computation because otherwise, we will have to do frequent transition lookups in the replay buffer. However, in this work, we will stick to the definition based on transition counts because using describing the graph structure explicitly will complexify the notations.

4.3 Practical Implementations of Graph Backup

Having defined the theoretical graph backup value estimate we now describe two concrete ways of implementing this value estimate in deep RL algorithms.

GB-limited: Limited Expansion of Graph Backup A naive way to implement the graph backup is to follow the definition exactly and do an exhaustive recursive expansion of the graph. However, the computational cost of doing so can quickly blow up with the size of the replay data[1]. One solution to this problem is to take inspiration from the n-step tree back up, and limit the depth and breadth of recursion, using the target network estimate when reaching depth limits. This is the motivation behind our first implementation of graph backup *GB-limited*, which expands the data graph with both a breadth limit b and depth limit d.

During the expansion of the graph, the number of boundary nodes can grow exponentially. To make this growth to be linear, we will sample b transitions from $\mathcal{T}_{s,a}$ according to their frequency f, as opposed to expanding all transitions. To constrain depth, when a depth limit is hit, the intermediate target values will be estimated by target network $q_{\boldsymbol{\theta}'}$ directly. If we set $b = 1$, GB-limited will do a tree backup with trajectories sampled from the data graph rather than real trajectories.

The pseudocode for GB-limited target computation is shown in Algorithm 1. Besides computational benefits, GB-limited has the implicit benefit of restricting the error of target network prediction. An expanded expression of graph backup target can have a lot of nested $\max$ operators, which can propagate the most optimistic target network error to the whole graph. The problem can be alleviated by discounting factors or other tricks such as double DQN [van Hasselt et al., 2016] or using a soft-Q target policy [Haarnoja et al., 2018]. In this work, we will only investigate the role of expansion limit and leave potential solutions to future work.

GB-Q: In-Buffer Q-learning GB-limited can give an approximation of the graph backup target, but what if we want to compute the exact value of the target without expansion limits? In fact, we can deal with this with dynamic programming, by storing previous computations of intermediate target values for later usage. A simple approach is to initialise all target estimations with the target network, and then run tabular Q learning with transitions uniformly sampled from replay buffer, effectively treating the replay buffer as a tabular MDP. The converged optimal q-value estimates on this MDP

[1] In fact, if there are loops in the graph, the situation can be even worse as the algorithm will never converge.

are then the graph backup targets, which is proven in Appendix A. It is worth noting that a single run of tabular Q learning can give us target values for all the seen states, so the computational overhead can actually be quite low when the replay size is small. In the online learning setting, we also need to consider the online initialization of the new states, and the target estimation should also be fine-tuned accordingly. The pseudo-code for full online DQN with graph backup is shown in Algorithm 2. The blue text shows operations GB-Q to the standard DQN algorithm.

Algorithm 1 GB-limited

Input: source state S_{source}, source action A_{source}, depth limit d, breath limit b, frequency mapping $f : \mathcal{T} \to \mathbb{N}^+$

Initialize the set containing states on the boundary of expansion $\mathcal{S}_{\text{new}} \leftarrow \{S_{\text{source}}\}$
Initialize the list of expanded state-action pairs l, denoting the largest element to be l_{max}
for $i = 0$ to d **do**
    Find all transitions on boundary $\mathcal{T}_{\text{new}} \leftarrow \{t | \forall t = (s, a, r, s') \in \mathcal{T}, s \in \mathcal{S}_{\text{new}}\}$
    Sample b transitions from $\mathcal{T}_{\text{new}}$ with $p(t) \propto f(t)$, getting $\mathcal{T}_{\text{pruned}} = \{t_1, t_2, ..., t_b\}$
    Append state-action pairs to list l, $\{l_{\text{max+1}}, l_{\text{max+2}}, ...\} = \{(s, a) | \forall (s, a, r, s') \in \mathcal{T}_{\text{pruned}}\}$
    Update boundary states $\mathcal{S}_{\text{new}} = \{s' | \forall (s, a, r, s') \in \mathcal{T}_{\text{pruned}}\}$
end for
Set S_{expanded} be the set containing all the states in list l
Initialize the target values $\bar{G}_s^a = q_{\boldsymbol{\theta}'}(s, a), \forall s \in S_{\text{expanded}}, a \in \mathcal{A}$
for (s, a) in $l_{\text{max}}, l_{\text{max-1}}, ..., l_1$ **do**
    $\bar{G}_s^a \leftarrow \sum_{(s,a,r,s') \in \mathcal{T}} \frac{f(s,a,r,s')}{c(s,a)} \left(r + \gamma \max_{a'} \bar{G}_{s'}^{a'} \right)$
end for
return $\bar{G}_{S_{\text{source}}}^{A_{\text{source}}}$

Algorithm 2 DQN with GB-Q (Online Case). Blue text denotes additions of GB-Q to standard DQN.

Input: batch size M, exploration constant ϵ, learning rate for tabular Q Learning α, number steps for running Tabular Q learning T_{table} target network update frequency t_{target}

Initialise online network weights θ at random
Observe the first state s
Initialise the set stores seen states $\mathcal{S}_{\text{seen}} \leftarrow \emptyset$
for $t = 0$ to T **do**
    With probability ϵ select random action a, otherwise select $a = \text{argmax}_{a'} q_{\boldsymbol{\theta}}(s, a')$
    Execute action a, observe reward r and state s'
    if $s' \notin \mathcal{S}_{\text{seen}}$ **then**
        Initialise the target $\hat{G}_{s',a'} = q_{\boldsymbol{\theta}'}(s', a')$ for all actions a'
        Update seen states $\mathcal{S}_{\text{seen}} \leftarrow \mathcal{S}_{\text{seen}} \cup \{s'\}$
    end if
    Store (s, a, r, s') in replay buffer
    if $t \mod t_{\text{target}} = 0$ **then**
        Update target network weights $\theta' \leftarrow \theta$
        Reinitialise targets $\hat{G}_{s_j,a'} = q_{\boldsymbol{\theta}'}(s_j, a')$ for all a' and observed $s_j \in \mathcal{S}_{\text{seen}}$
    end if
    for $i = 0$ to T_{table} **do**
        Sample a transition $(\bar{s}, \bar{a}, \bar{r}, \bar{s}')$ from replay buffer
        Update target $\hat{G}_{\bar{s}}^{\bar{a}} \leftarrow \hat{G}_{\bar{s}}^{\bar{a}} + \alpha(\bar{r} + \gamma \max_{a'} \hat{G}_{\bar{s}'}^{a'} - \hat{G}_{\bar{s}}^{\bar{a}})$
    end for
    Sample random minibatch of transitions (s_k, a_k, r_k, s'_k) from replay buffer
    Perform a gradient descent step on $|q_{\boldsymbol{\theta}}(s_k, a_k) - \hat{G}_{a_k}^{s_k}|^2$ with respect to θ
    Update current state variable $s \leftarrow s'$
end for

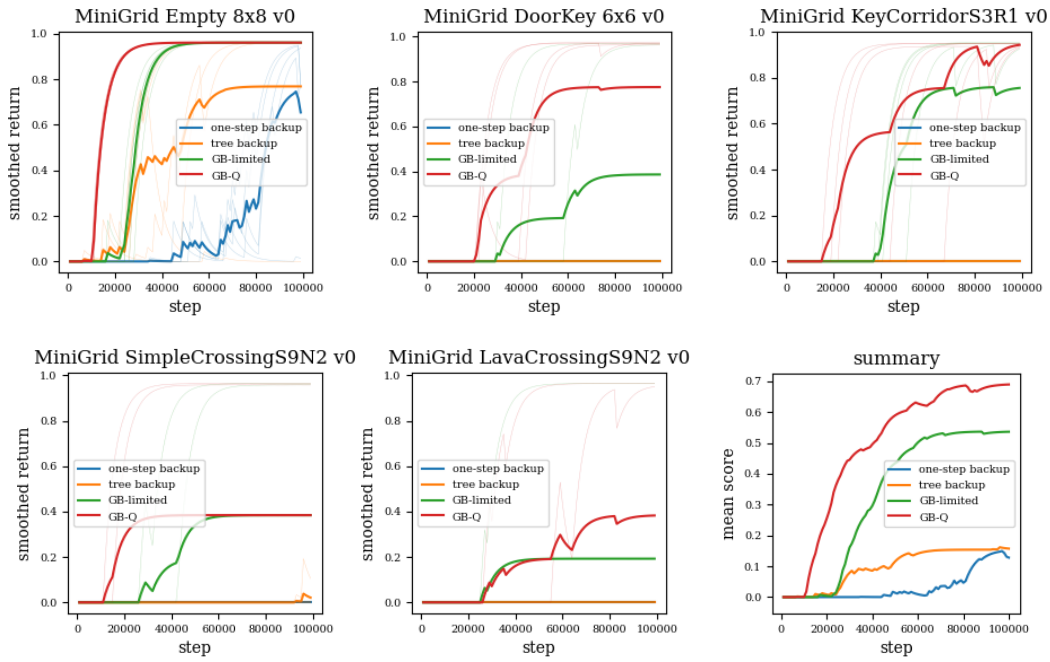

Figure 4: MiniGrid Results. The training curves of the different backup settings on 5 singleton MiniGrid tasks. The faded curves are those of individual runs and the solid lines the average over five runs. The summary plot presents the mean scores of all tasks. Curves are plotted with an exponential moving average for better readability.

5 Experiments

Experiment Setting We test the effects of different backup targets based on the setting of vanilla DQN. Our aim is to understand whether graph backup produces more data-efficient learning than previous backup methods, and which implementation of graph backup works better in different settings. We compare GB-Q and GB-limited to tree backup and one-step backup. The GB-limited and tree backup both use a depth limit of 5. The breath limit for GB-limited is 50 for MiniGrid and 10 for Minatar.

For the architecture, the q network has 2 convolutions layers and 2 dense layers, and we follow the hyper-parameters of Rainbow [Hessel et al., 2018b] with target network update frequency of 8000, ϵ-greedy exploration with $\epsilon = 0.02$. The learning rate is 0.001 for MiniGrid and 0.0000625 for Minatar. The discounting factor is 0.95 for MiniGrid and 0.99 for Minatar. The replay frequency is 1 for MiniGrid and 4 for Minatar Since we tested the algorithm in a data-efficient setting, the size of the replay buffer is set to be equal to the overall training steps.

MiniGrid We first compare the methods in singleton MiniGrid tasks. Every single run (out of 5) has a different but fixed random seed within the whole training process. This means the overall number of possible states is small and the data graph is thus quite dense. The reward of MiniGrid is only given at the end of the episode, which makes credit assignment a critical problem. The training curves of the different backup settings are shown in Fig. 4, with the faded curves being those of individual runs and the solid lines the average over five runs.

Among the 5 tasks, one-step backup and tree backup only managed to converge within 1e5 steps for the easiest empty room task. For other tasks with more complex navigation (SimpleCrossing and LavaCrossing) and interaction with objects (DoorKey and KeyCorridor), only the graph backup converged this low data regime. Among the graph backup family, the GB-Q performs slightly better than GB-limited. This is potentially because the GB-Q implements an infinite step backup where the reward propagation is not strongly dependent on the target network, while for GB-limited and tree backup, the rewards can only be propagated 5 steps forward within each target network update cycle.

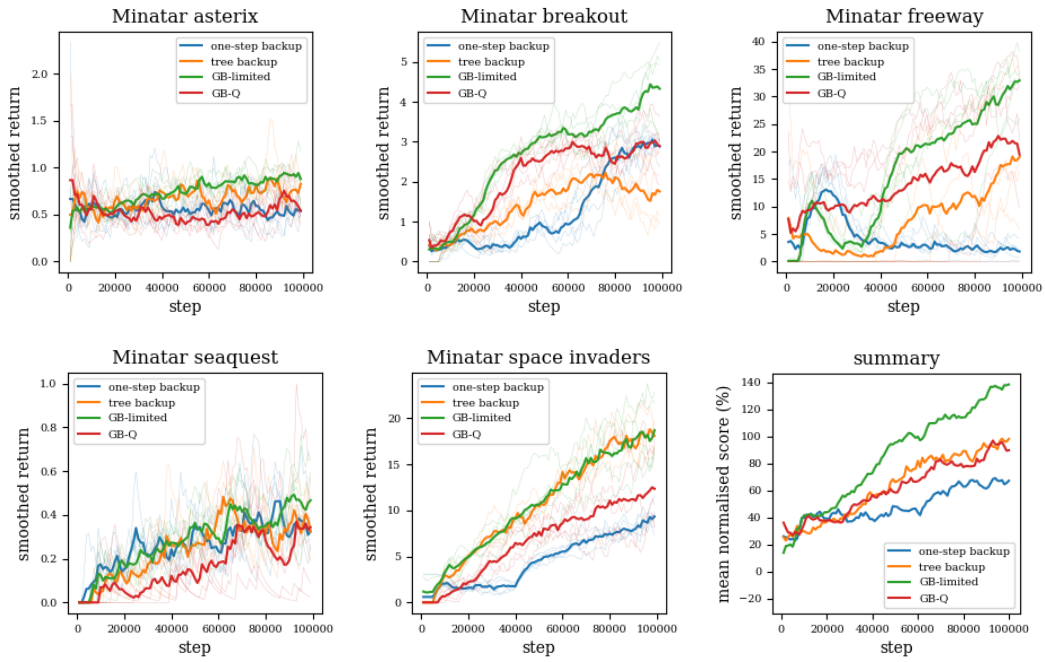

Figure 5: Minatar Results. The training curves of the different backup settings on 5 different Minatar tasks. The faded curves are those of individual runs and the solid lines the average over five runs. The summary plot presents the mean scores of all tasks, normalized by the average performance of all agents. Curves are plotted with an exponential moving average for better readability.

Minatar We perform experiments on Minatar. Minatar is a collection of miniature Atari games with a symbolic representation of the objects. The game state is fully specified by the observation of a 10 by 10 image, where each pixel corresponds to an object. The game dynamics are by nature stochastic, which differs from vanilla Atari games whose random seeds are generated from the players' action sequences [Hausknecht and Stone, 2015], and hence produce deterministic transitions if the agent can see all the history. We set the overall number of interactions to be 100'000, which is inspired by the Atari100K benchmark [Kaiser et al., 2020]. As suggested by its name, Atari100K limits the number of interactions to 100K, which is equivalent to 2 hours of game-play in Atari. Since the human performance score reported by Mnih et al. [2015] is also given by human experts after 2 hours of game-play, Atari100K is considered as a test-bed for human-level data-efficient learning.

The experimental results for Minatar are shown in Fig. 5. We can see in the plot that GB-limited consistently outperform the tree backup and vanilla one-step backup. The wilcoxon signed rank test gives p-values of 0.0002 and 0.003 for the hypotheses "GB-limited is better than one-step backup" and "GB-limited is better than tree backup". GB-Q in Minatar tasks is inferior to GB-limited, which can be explained by the nosier target estimation. Unlike MiniGrid, the rewards of Minatar tasks are denser, and the variance of the returns are also higher. A graph backup operator without expansion limitation might cause the target value to be overly optimistic.

6 Conclusion

In this work, we motivate the introduction of a novel bootstrapped value estimation operator, graph backup. This backup method utilises the graph-structured nature of MDP transition data to enable counterfactual credit assignment and variance reduction. We propose two implementations of the graph backup and demonstrate graph backup surpasses multi-step and one-step backup in MiniGrid and Minatar tasks. For future work, we hope to expand graph backup to higher-dimensional and continuous state-space environments. For higher-dimensional environments, the data graph might be much sparser, so we potentially need either a preprocessed or a learned discrete representation of the state.

References

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL `http://arxiv.org/abs/1912.06680`.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. Treeqn and atreec: Differentiable tree-structured models for deep reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=H1dh6Ax0Z`.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2455–2467, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html`.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR, 2018. URL `http://proceedings.mlr.press/v80/haarnoja18b.html`.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=0oabwyZbOu`.

Matthew J. Hausknecht and Peter Stone. The impact of determinism on learning atari 2600 games. In Michael Bowling, Marc G. Bellemare, Erik Talvitie, Joel Veness, and Marlos C. Machado, editors, *Learning for General Competency in Video Games, Papers from the 2015 AAAI Workshop, Austin, Texas, USA, January 26, 2015*, volume WS-15-10 of *AAAI Workshops*. AAAI Press, 2015. URL `http://aaai.org/ocs/index.php/WS/AAAIW15/paper/view/9564`.

J. Fernando Hernandez-Garcia and Richard S. Sutton. Understanding multi-step deep reinforcement learning: A systematic study of the DQN target. *CoRR*, abs/1901.07510, 2019. URL `http://arxiv.org/abs/1901.07510`.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3215–3222. AAAI Press, 2018a. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204`.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA,*

February 2-7, 2018, pages 3215–3222. AAAI Press, 2018b. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204.

Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado van Hasselt. Muesli: Combining improvements in policy optimization. *arXiv preprint arXiv:2104.06159*, 2021.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1xCPJHtDB.

Byeong-Jun Min and Kyung-Joong Kim. Learning to play visual doom using model-free episodic control. In *IEEE Conference on Computational Intelligence and Games, CIG 2017, New York, NY, USA, August 22-25, 2017*, pages 223–225. IEEE, 2017. doi: 10.1109/CIG.2017.8080439. URL https://doi.org/10.1109/CIG.2017.8080439.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

David E. Moriarty and Risto Miikkulainen. Efficient learning from delayed rewards through symbiotic evolution. In Armand Prieditis and Stuart J. Russell, editors, *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 396–404. Morgan Kaufmann, 1995. doi: 10.1016/b978-1-55860-377-6.50056-6. URL https://doi.org/10.1016/b978-1-55860-377-6.50056-6.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 759–766. Morgan Kaufmann, 2000.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2827–2836. PMLR, 2017. URL http://proceedings.mlr.press/v70/pritzel17a.html.

Tabish Rashid, Bei Peng, Wendelin Boehmer, and Shimon Whiteson. Optimistic exploration even with a pessimistic initialisation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1xGP6VYwH.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. Citeseer, 1994.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016. doi: 10.1038/nature16961. URL https://doi.org/10.1038/nature16961.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2753–2762, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3a20f62a0af1aa152670bab3c602feed-Abstract.html`.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2094–2100. AAAI Press, 2016. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389`.

Hado van Hasselt, Sephora Madjiheurem, Matteo Hessel, David Silver, André Barreto, and Diana Borsa. Expected eligibility traces. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9997–10005. AAAI Press, 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/17200`.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575 (7782):350–354, 2019. doi: 10.1038/s41586-019-1724-z. URL `https://doi.org/10.1038/s41586-019-1724-z`.

Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic reinforcement learning with associative memory. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=HkxjqxBYDB`.