Gradient Boosting Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural networks (NN) achieve remarkable results in various tasks, but lack key characteristics: interpretability, support for categorical features, and lightweight implementations suitable for edge devices. While ongoing efforts aim to address these challenges, Gradient Boosting Trees (GBT) inherently meet these requirements. As a result, GBTs have become the go-to method for supervised learning tasks in many real-world applications and competitions. However, their application in online learning scenarios, notably in reinforcement learning (RL), has been limited. In this work, we bridge this gap by introducing Gradient-Boosting RL (GBRL), a framework that extends the advantages of GBT to the RL domain. Using the GBRL framework, we implement various actor-critic algorithms and compare their performance with their NN counterparts. Inspired by shared backbones in NN we introduce a tree-sharing approach for policy and value functions with distinct learning rates, enhancing learning efficiency over millions of interactions. GBRL achieves competitive performance across a diverse array of tasks, excelling in domains with structured or categorical features. Additionally, we present a high-performance, GPU-accelerated implementation that integrates seamlessly with widely-used RL libraries. GBRL expands the toolkit for RL practitioners, demonstrating the viability and promise of GBT within the RL paradigm, particularly in domains characterized by structured or categorical features.

1 Introduction

Reinforcement Learning (RL) has shown great promise in various domains that involve sequential decision making. However, many real-world tasks, such as inventory management, traffic signal optimization, network optimization, resource allocation, and robotics, are represented by structured observations with categorical or mixed data types. These tasks can benefit significantly from deployment and training on edge devices due to resource constraints. Moreover, interpretability is crucial in these applications for regulatory reasons and for trust in the decision-making process. Current neural network (NN) based solutions struggle with interpretability, handling categorical data, and supporting light implementations suitable for low-compute devices.

Gradient Boosting Trees (GBT) is a powerful ensemble method extensively used in supervised learning due to its simplicity, accuracy, interpretability, and natural handling of structured and categorical data. Frameworks such as XGBoost [7], LightGBM [20], and CatBoost [36] have become integral in applications spanning finance [49], healthcare [54, 27, 43], and competitive data science [6]. Despite their successes, GBT has seen limited application in RL. This is primarily because traditional GBT libraries are designed for static datasets with predefined labels, contrasting with the dynamic nature of RL. The distribution shift in both input (state) and output (reward) poses significant challenges for the direct application of GBT in RL. Moreover, there is a notable lack of benchmarks or environments tailored for structured data, further hindering progress in this area.

In this paper, we introduce Gradient Boosting Reinforcement Learning (GBRL), a GBT framework tailored for RL. Our contributions are:

1. **GBT for RL**. We demonstrate the viability and potential of GBT as function approximators in RL. We present GBT-based implementations of PPO, A2C, and AWR, and show that GBRL is competitive with NNs across a range of environments. In addition, similarly to supervised learning, GBRL outperforms NNs on categorical tasks (see Figure 1).
2. **Tree-based Actor-Critic architecture**. Inspired by shared architectures in NN-based actor-critic (AC), we introduce a GBT-based AC architecture. This reduces the memory and computational requirements by sharing a common ensemble structure for both the policy and value. This approach significantly reduces runtime compared to existing GBT frameworks, thus removing the barrier to solving complex, high-dimensional RL tasks with millions of interactions.
3. **Modern GBT-based RL library**. We provide a CUDA-based [33] hardware-accelerated GBT framework optimized for RL. GBRL is designed to work as part of a broader system and seamlessly integrates with popular repositories such as Stable-baselines3 [39].This new tool offers practitioners a powerful option for exploring GBT in RL settings. [1]

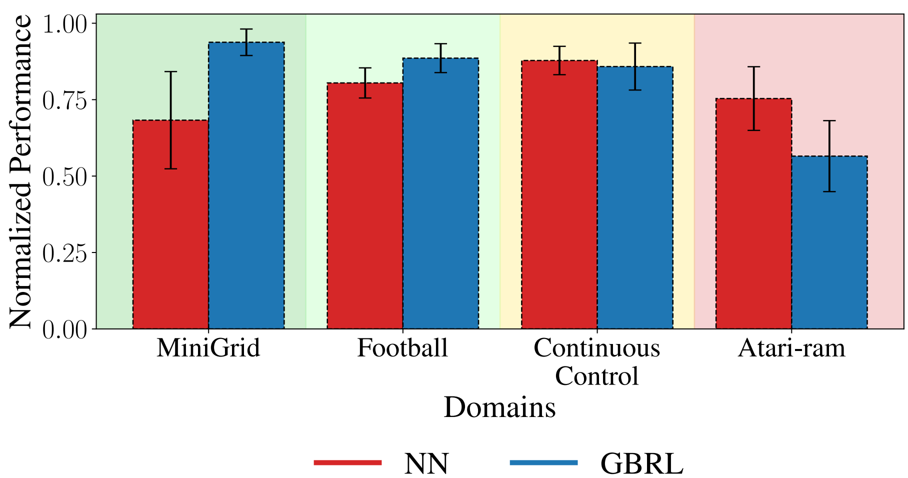


Figure 1: **PPO GBRL vs PPO NN.** Aggregated mean and standard deviation of the normalized average reward for the final 100 episodes. Rewards were normalized as: $\text{reward}_{\text{norm}} = \frac{\text{reward}}{\text{reward}_{\max\{\text{NN},\text{GBRL}\}}}$ per environment and then aggregated across each domain.

2 Related Work

Gradient boosted trees. Recent advances have extended GBT's capabilities beyond traditional regression and classification. In ranking problems, GBT has been used to directly optimize ranking metrics, as demonstrated by frameworks like StochasticRank [51] and recent advancements explored in Lyzhin et al. [26]. Additionally, GBT offer probabilistic predictions through frameworks like NGBoost [11], enabling uncertainty quantification [28]. The connection between GBT and Gaussian Processes [52, 45] offers further possibilities for uncertainty-aware modeling. Recently, Ivanov and Prokhorenkova [18] modeled graph-structured data by combining GBT with graph neural networks.

Despite their versatility, applying GBT in RL remains a relatively less explored area. Several works have employed GBT as a function approximator within off-policy RL methods, including its use in Q-learning [1] and in bandit settings to learn inverse propensity scores [24]. Recently, Brukhim et al. [5] proposed a boosting framework for RL where a base policy class is incrementally enhanced using linear combinations and nonlinear transformations. However, these previous works have not yet demonstrated the scalability and effectiveness in complex, high-dimensional RL environments

[1] We attached the GBRL repository as supplementary material and will release it after the review process.

requiring extensive interactions. In this work, we show how to adapt the framework of GBT to successfully solve large-scale RL problems.

Interpretability. Due to the inherent non-linearities, NNs are challenging to interpret and require sophisticated methods to do so. Interpreting NNs often involves either approximation with simpler models such as decision trees or using gradient-based techniques, which require additional forward and backward passes [16, 9, 44, 3, 37]. On the other hand, interpretability methods for GBT can take advantage of the structure of a decision tree for high speed, efficiency, and accuracy [25, 10].

Structured and categorical data. Previous work in RL has predominantly focused on using NNs due to their ability to capture complex patterns in high-dimensional data. Techniques such as Q-learning and AC methods have advanced significantly, demonstrating success in tasks involving raw sensory inputs like images, text, and audio. However, NNs that perform well on structured and categorical data typically have very specialized architectures [19, 46, 14, 2] and are not standard multi-layer perceptrons (MLPs) that are often used in many RL tasks and algorithms [34]. Even with these specialized architectures, Gradient Boosting Trees (GBT) often perform equally or better on structured and categorical datasets [19, 31, 14, 15].

Policy optimization through functional gradient ascent. In this approach, the policy is parameterized by a growing linear combination of functions [29]. Each linear addition represents the functional gradient with respect to current parameters. Kersting and Driessens [21] demonstrated the direct optimization of policies using the policy gradient theorem [48]. Similarly, Scherrer and Geist [41] proposed a functional gradient ascent approach as a local policy search algorithm. While these works lay theoretical groundwork, practical results on complex, high-dimensional RL environments have not been shown. To adapt GBT's to RL, we leverage the framework of functional gradient ascent. This combination enables a seamless integration of GBRL directly into existing RL optimization packages, such as Stable-baselines [39].

3 Preliminaries

We begin by introducing Markov Decision Processes (MDPs) and the AC schema. Then, we introduce GBT. In the following section, we show how to combine both of these paradigms into GBRL.

3.1 Markov Decision Process

We consider a fully observable infinite-horizon Markov decision process (MDP) characterized by the tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$. At each step, the agent observes a state $\mathbf{s} \in \mathcal{S}$ and samples an action $\mathbf{a} \in \mathcal{A}$ from its policy $\pi(\mathbf{s}, \mathbf{a})$. Performing the action causes the system to transition to a new state $\mathbf{s}'$ based on the transition probabilities $P(\mathbf{s}' \,|\, \mathbf{s}, \mathbf{a})$, and the agent receives a reward $\mathbf{r} \sim \mathcal{R}(\mathbf{s}, \mathbf{a})$. The objective is to find an optimal policy π^* that maximizes the expected discounted reward $J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \, \mathbf{r}_t]$, with a discount factor $\gamma \in [0, 1)$.

The action-value function $Q_\pi(\mathbf{s}, \mathbf{a}) := \mathbb{E}_\pi[\sum_{t'=0}^{\infty} \gamma^{t'} \mathcal{R}(\mathbf{s}_{t+t'}, \mathbf{a}_{t+t'}) | \, \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}]$ estimates the expected returns of performing action $\mathbf{a}$ in state $\mathbf{s}$ and then acting according to π. Additionally, the value function $V_\pi(\mathbf{s}) := \mathbb{E}_\pi[\sum_{t'=0}^{\infty} \gamma^{t'} \mathcal{R}(\mathbf{s}_{t+t'}, \mathbf{a}_{t+t'}) | \, \mathbf{s}_t = \mathbf{s}]$, predicts the expected return starting from state $\mathbf{s}$ and acting according to π. Finally, the advantage function $A_\pi(\mathbf{s}, \mathbf{a}) := Q_\pi(\mathbf{s}, \mathbf{a}) - V_\pi(\mathbf{s})$, indicates the expected relative benefit of performing action $\mathbf{a}$ over acting according to π.

3.2 Actor-Critic Reinforcement Learning

Actor-critic methods are a common method to solve the objective $J(\pi)$. They learn both the policy and value. In the GBRL framework, we extend three common AC algorithms to support GBT-based function approximators.

A2C [32] is a synchronous, on-policy AC algorithm designed to improve learning stability. The critic learns a value function, $V(\mathbf{s})$, used to estimate the advantage. This advantage is incorporated into the policy gradient updates, reducing variance and leading to smoother learning. The policy is updated using the following gradient: $\nabla_\theta J(\pi_\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(\mathbf{a} \,|\, \mathbf{s}) A(\mathbf{s}, \mathbf{a})]$.

PPO [42] extends A2C by improving stability. This is achieved through constrained policy update steps using a clipped surrogate objective. This prevents drastic policy changes and leads to smoother learning. To achieve this, PPO solves the following objective: $\nabla_\theta J(\pi_\theta) = \mathbb{E}[\nabla_\theta \text{clip}(\frac{\log \pi_\theta(\mathbf{a} \mid \mathbf{s})}{\log \pi_{\theta_{\text{old}}}(\mathbf{a} \mid \mathbf{s})}, 1 - \epsilon, 1 + \epsilon) A(\mathbf{s}, \mathbf{a})]$. Additionally, PPO enhances sample efficiency by performing multiple optimization steps on each collected rollout.

AWR [35] is an off-policy AC algorithm. Provided a dataset $\mathcal{D}$, AWR updates both the policy and the value through supervised learning. This dataset can be pre-defined and fixed (offline), or continually updated using the agents experience (replay buffer). At each training iteration k, AWR solves the following two regression problems:

$$V_k = \arg\min_V \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[\|G(\mathbf{s}, \mathbf{a}) - V(\mathbf{s})\|_2^2] \, , \; \pi_{k+1} = \arg\max_\pi \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[\log \pi(\mathbf{a} \mid \mathbf{s}) \exp(\frac{1}{\beta} A_k(\mathbf{s}, \mathbf{a}))] \, ,$$

where $G(\mathbf{s}, \mathbf{a})$ represents the monte-carlo estimate or TD(λ) of the expected return [47].

3.3 Gradient Boosting Trees as Functional Gradient Descent

Gradient boosting trees (GBT) [12] are a non-parametric machine learning technique that combines decision tree ensembles with functional gradient descent [30]. GBT iteratively minimizes the expected loss $L(F(\mathbf{x})) = \mathbb{E}_{\mathbf{x},\mathbf{y}}[L(\mathbf{y}, F(\mathbf{x}))]$ over a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. A GBT model, F_K, predicts outputs using K additive trees as follows:

$$F_K(\mathbf{x}_i) = F_0 + \sum_{k=1}^{K} \epsilon h_k(\mathbf{x}_i) \, , \tag{1}$$

where ϵ is the learning rate, F_0 is the base learner, and each h_k is an independent regression tree partitioning the feature space.

In the context of functional gradient descent, the objective is to minimize the expected loss $L(F(\mathbf{x})) = \mathbb{E}_{\mathbf{x},\mathbf{y}}[L(\mathbf{y}, F(\mathbf{x}))]$ with respect to the functional F. Here, a functional $F : \mathcal{H} \to \mathbb{R}$ maps a function space to real numbers. A GBT model can be viewed as a functional F that maps a linear combination of binary decision trees to outputs: $F : \text{lin}(\mathcal{H}) \to \mathbb{R}^D$, where $\mathcal{H}$ is the decision tree function class.

We start with an initial model, F_0, and iteratively add trees to F to minimize the expected loss. Similar to parametric gradient descent, at each iteration k, we minimize the loss by taking a step in the direction of the functional gradient $g_k^i := \nabla_{F_{k-1}} L(\mathbf{y}_i, F_{k-1}(\mathbf{x}_i))$. However, we are constrained to gradient directions within $\mathcal{H}$. Thus, we project the gradient g_k into a decision tree by solving:

$$h_k = \arg\min_h \| - \epsilon g_k - h(\mathbf{x})\|_2^2 \, . \tag{2}$$

4 Gradient Boosting Reinforcement Learning

In this work, we extend the framework of GBT to support AC algorithms in the task of RL. The objective in RL is to optimize the return J, the cumulative reward an agent receives. Unlike in supervised learning, the target predictions are unknown a priori. RL agents learn through trial and error. Good actions are reinforced by taking a step in the direction of the gradient $\nabla_\pi J$. This formulation aligns perfectly with functional gradient ascent; thus, in GBRL, we optimize the objective directly over the decision tree function class. This is achieved by iteratively growing the ensemble of trees $\{h_i\}$. The ensemble outputs θ, representing AC parameters such as the policy π and the value function. For example, $\theta = [\mu(\mathbf{s}), \sigma(\mathbf{s}), V(\mathbf{s})]$ for a Gaussian policy. At each iteration, a new tree h_k, constructed to minimize the distance to $\nabla_{\theta_{k-1}} J$, is added to the ensemble. Here, The resulting method is an application of GBT as a functional gradient optimizer $\theta_k \approx \theta_0 + \epsilon \sum_{m=0}^{k-1} \nabla_{\theta_m} J$.

However, RL presents unique challenges for GBT. RL involves a nonstationary state distribution and inherent online learning, causing gradients to vary in magnitude and direction. Large gradients in unfavorable directions risk destabilizing training or leading to catastrophic forgetting. Moreover, feedback in RL is provided through interactions with the environment and is not available a priori. This contrasts with supervised learning settings, where gradients decrease with boosting iterations, and targets are predefined. As a result, many of the key features that traditional GBT libraries rely on

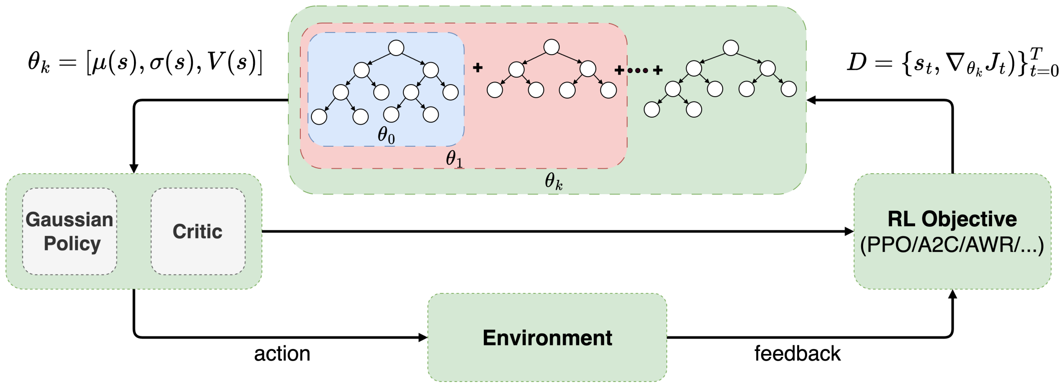


Figure 2: **The GBRL framework.** The actor's policy and critic's value function are parameterized by θ_k. For example, $\theta_k = [\mu(\mathbf{s}), \sigma(\mathbf{s}), V(\mathbf{s})]$ for a Gaussian policy. θ_k is calculated by summing all the outputs of trees in the ensemble. Starting from θ_0, at each training iteration, GBRL collects a rollout and computes the gradient $\nabla_{\theta_0} J$. This gradient is then used to fit the next tree added to the ensemble, which is updated to θ_1. This process repeats with each iteration fitting a new tree, refining the parameterization, and expanding the ensemble towards $\theta_k \approx \theta_0 + \epsilon \sum_{m=0}^{k-1} \nabla_{\theta_m} J$, an approximated scaled sum of gradients with respect to past parametrizations.

are not suitable. For example, GOSS [20], categorical feature encoding [36], early-stopping signals, pruning methods [53], and strategies to tackle online learning [55].

To address these challenges, we employ appropriate tools from the NN and GBT literature, such as batch learning [13, 40] to update the ensemble. At each boosting iteration, we fit a decision tree on a random batch sampled with replacement from the experience buffer. This approach helps handle non-stationary distributions and improve stability by focusing on different parts of the state space, allowing beneficial gradient directions to accumulate and minimizing the impact of detrimental ones. Additionally, GBRL fits gradients directly to optimize objectives, whereas traditional GBT methods require targets and need workarounds to utilize gradients effectively.

A common theme in AC algorithms is to utilize a shared approximation for the actor and the critic. We adopt this approach in GBRL, constructing trees where each leaf provides two predictions. GBRL predicts both the policy (distribution over actions) and the value estimate. The internal structure of the tree is shared, providing a single feature representation for both objectives and significantly reducing memory and computational bottlenecks. Accordingly, in GBRL we apply differentiated learning rates to the policy and value outputs during prediction, effectively optimizing distinct objectives within this shared structure. We present the full algorithm in Algorithm 1 and diagram in Figure 2.

5 Experiments

Our experiments aim to answer the following questions:

1. **GBT as RL Function Approximator**: Can GBT-based AC algorithms effectively solve complex high-dimensional RL tasks?
2. **Comparison to NNs**: How does GBRL compare with NN-based training in various RL algorithms?
3. **Benefits in Categorical Domains**: Do the benefits of GBT in supervised learning transfer to the realm of RL?
4. **Comparison to Traditional GBT libraries**: Can we use traditional GBT libraries instead of GBRL for RL tasks?
5. **Evaluating the shared AC architecture**: How does sharing the tree structure between the actor and the critic impact performance?

We implemented GBT-based versions of A2C, PPO, and AWR within Stable Baselines3. We refer to our implementations as PPO GBRL, A2C GBRL, and AWR GBRL. We evaluated GBRL against the

Algorithm 1 Gradient Boosting for Reinforcement Learning (GBRL)

Initialize: θ_0, ϵ_{actor}, ϵ_{critic}, experience buffer $\mathcal{B}$, total training iterations K, number of updates U, batch size N, $k \leftarrow 1$
while $k < K$ **do**
  Collect trajectory $\tau^{(k)} = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_T, \mathbf{a}_T)^{(k)}$ and rewards $(\mathbf{r}_0, \ldots, \mathbf{r}_T)^{(k)}$ using $\pi_{\theta_{k-1}}$
  Add trajectory $\tau^{(k)}$ and rewards to the experience buffer $\mathcal{B}$
  for each update $u = 1$ to U **do**
    Sample a batch from the experience buffer $\mathcal{B}$
    Compute gradients g according to AC algorithm (e.g., PPO, A2C, AWR)
    Construct dataset $D = \{(\mathbf{s}_n, g_n)\}_{n=0}^{N}$ and fit a decision tree h_k
    for each dimension $d = 0$ to D **do**
      if $0 \leq d < D$ **then**
        Update $\theta_k^{(d)} = \theta_{k-1}^{(d)} + \epsilon_{\text{actor}} h_k(\mathbf{s})$
      else
        Update $\theta_k^{(d)} = \theta_{k-1}^{(d)} + \epsilon_{\text{critic}} h_k(\mathbf{s})$
    $k \leftarrow k + 1$
Output: AC parameters $\theta_K^{(d)}(\mathbf{s})$ for $d = 0, 1, \ldots, D$

equivalent NN implementations. Where available, we utilize hyperparameters from RL Baselines3 Zoo [38]; otherwise, we optimize the hyperparameters for specific environments. The AWR NN implementation is based on the original paper [35].

We conducted experiments on a range of RL domains. We test classic control tasks, high-dimensional vectorized problems, and finally categorical tasks. We use 5 random seeds per experiment on a single NVIDIA V100-32GB GPU. We present the cummulative non-discounted reward, averaged across the last 100 episodes. We normalize the plots for simple visual comparison between GBRL and the corresponding NN implementations. The normalized score is computed as $\text{score}_{\text{norm}} = \frac{\text{score}_{\text{GBRL}} - \text{score}_{\text{NN}}}{\text{score}_{\max\{\text{NN,GBRL}\}} - \text{score}_{\min\{\text{NN,GBRL}\}}}$. We provide the full learning curves, implementation details, compute resources, un-normalized numerical results, and hyperparameters in the supplementary material.

Classic Enviroments. We evaluate GBRL's ability to solve classic RL tasks using Continuous-Control and Box2D environments, provided via Gym [50]. We trained agents for 1M steps (1.5M for LunarLander-v2) and provide the results in Figure 3. For exact values, refer to Table 2.

Considering the algorithmic objective, we observe that GBRL and NN present similar performance when optimized using PPO. In contrast, the other methods demonstrate inconclusive results. In certain environments, such as MountainCar, GBRL outperforms NN with all AC methods. On the other hand, in Pendulum NN is better.

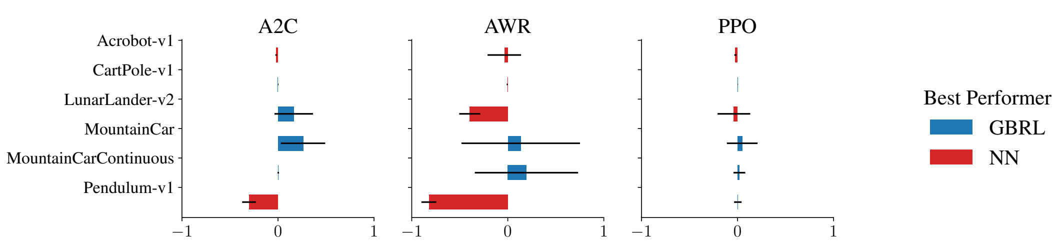
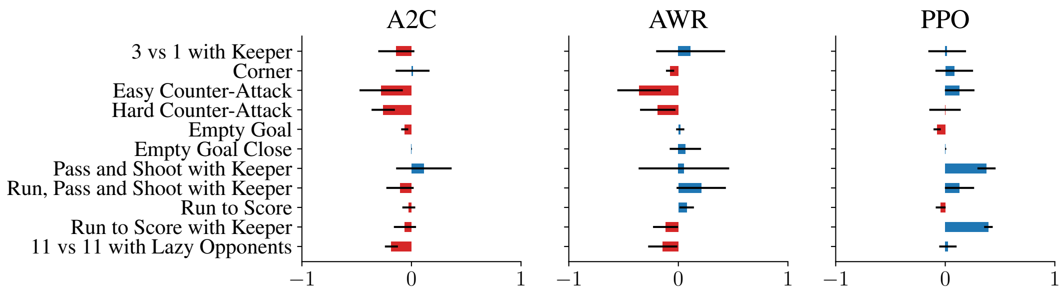


Figure 3: **Continuous-Control and Box2D environments.** Normalized comparison between GBRL and NN. PPO, the best performing method, shows similar performance with GBRL and NN function classes.

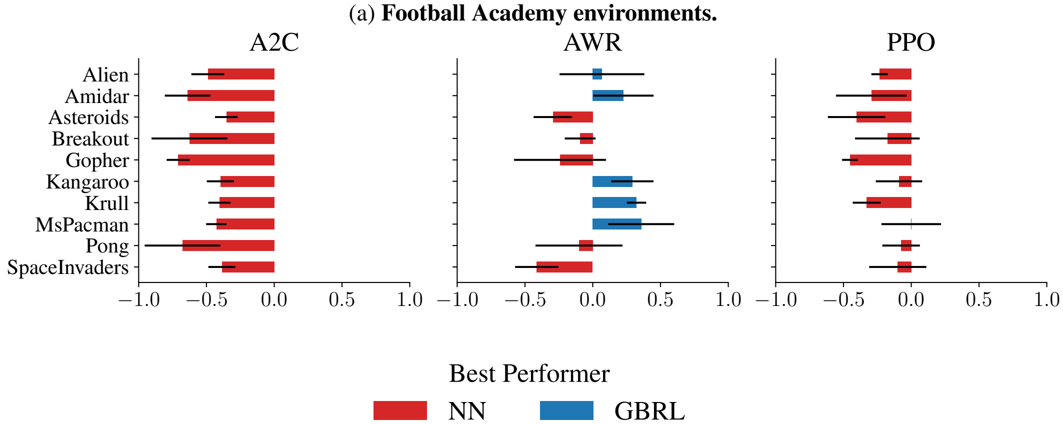
High-Dimensional Vectorized Environments. The decision-tree function class operates on individual features at each step. Consequently, this function class is not well-suited for handling pixel-based representations, which require more complex feature interactions. Therefore, we evaluted GBRL in the Football [22] and Atari RAM [4] domains. These offer high-dimensional vectorized representations. We trained agents in both environments for 10 million timesteps. The complete results are reported in Tables 3 and 4 and illustrated in Figure 4.

The results portray the following phenomenon. While both tasks may seem similar, there is a distinct difference. The features in the football domain are manually constructed and represent identifiable information, such as the location of the ball and the players. However, the Atari RAM domain provides a flattened view of the system RAM, which is unstructured.

At their core, binary decision trees are if-else clauses. This function class is naturally suited to work with structured data. These insights are emphasized in the football domain. Here, PPO GBRL greatly outperforms PPO NN across most environments and exhibits equivalent performance on the rest. In addition, as Atari RAM is unstructured, we observe that, as can be expected, in most cases GBRL underperforms NN, except for AWR. However, AWR NN underperformed considerably compared to the other NN implementations.



(a) **Football Academy environments.**



(b) **Atari-ramNoFrameskip-v4 environments.**

Figure 4: **High-Dimensional Vectorized Environments.** GBRL outperforms NN on the structured Football domain using PPO. NN outperforms on unstructured tasks, such as Atari RAM.

Categorical Environments. The football experiment suggests that GBRL outperforms when assigned structured data. Here, we focus on categorical environments. This is a regime where GBT excels in supervised learning. In these experiments, we evaluated the MiniGrid domain [8]. It consists of 2D grid worlds with goal-oriented tasks that require object interaction. We trained in PutNear, FourRooms, and Fetch tasks for 10M timesteps, matching the reported PPO NN in RL Baselines3 Zoo. We trained the remaining environments for 1M timesteps. We give the results in Figure 5. For exact numbers, see Table 5.

In MiniGrid, GBRL outperforms or is on-par with NN in most tasks. Specifically, PPO GBRL is significantly better than PPO NN. We observe the same trend when comparing between environments. These results emphasize that GBRL is a strong candidate for problems characterized by structured data, specifically when using PPO as the algorithmic backend.

GBRL vs Traditional GBT Libraries. Here, we compare GBRL with Catboost and XGBoost. We focus on the PPO variant. When comparing to the standard libraries, we utilize their built-in

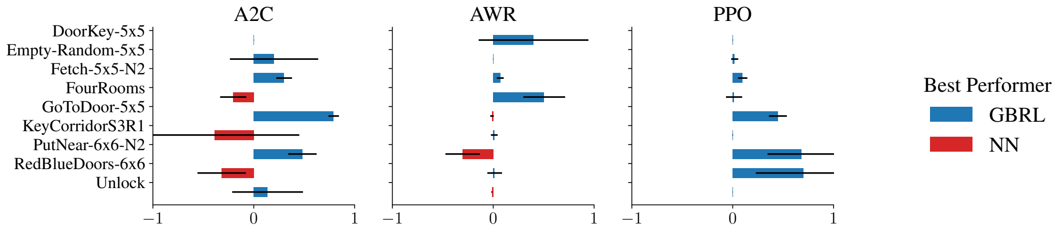


Figure 5: **MiniGrid environments.** GBRL combined with the PPO backend outperforms NN across a range of categorical environments.

options for incremental learning, vectorized leaves, and custom loss functions. As both CatBoost and XGBoost do not support differential learning rates, we used separate ensembles for the actor and the critic. For the comparison, we use the CartPole-v1 environment, training for 1M steps. The results are shown in Figure 6.

As seen, standard GBT libraries are unable to solve RL tasks in a realistic timeframe. GBRL, however, efficiently solves the task while also remaining competitive with NN across a range of environments.

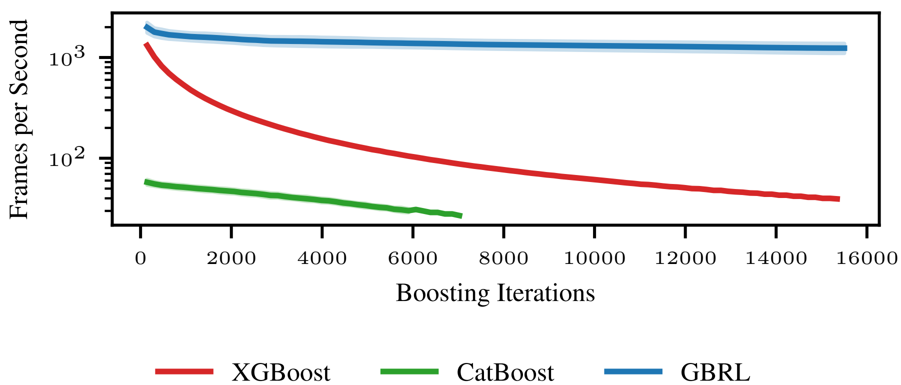


Figure 6: **Comparing to standard GBT libraries.** CatBoost and XGBoost are intractable in RL. Specifically, CatBoost lacks GPU support for custom losses, leading to low FPS and early termination.

Evaluating the shared AC architecture. Finally, we evaluated the benefits of using a shared AC architecture by training PPO GBRL on three MiniGrid environments. We train agents with shared and non-shared architectures for 10M timesteps and compare the score, GPU memory usage, and FPS. We provide the aggregated results in Figure 7, and environment-specific breakdowns in the supplementary.

The benefit of the shared structure is clear both in terms of GPU memory consumption and computation speed. By sharing the tree structure, GBRL requires less than half the memory and almost triples the training FPS. This is achieved without any negative performance on the resulting policy, as seen in the reward plot.

Result summary. The performance of GBRL varied across RL algorithms, but environments like MiniGrid highlight the potential advantages of using GBT in RL. The results suggest that GBT's strengths in handling structured and categorical data from supervised learning can effectively transfer to the RL domain. Conversely, GBRL underperformed in Atari-RAM environments, indicating that certain environments, characterized by unstructured observations, are less suited for GBTs.

The results can be explained by the findings of Grinsztajn et al. [15], which suggest that NNs have an inductive bias toward overly smooth solutions and that MLP-like architectures are not robust to uninformative features. The optimal solutions for Atari-RAM might be smoother, which could explain the better performance of NNs. On the other hand, McElfresh et al. [31] argue that GBT outperforms NNs on 'irregular' datasets. Tree-based models excel in handling irregular patterns and categorical data, aligning with GBRL's success in environments like MiniGrid.

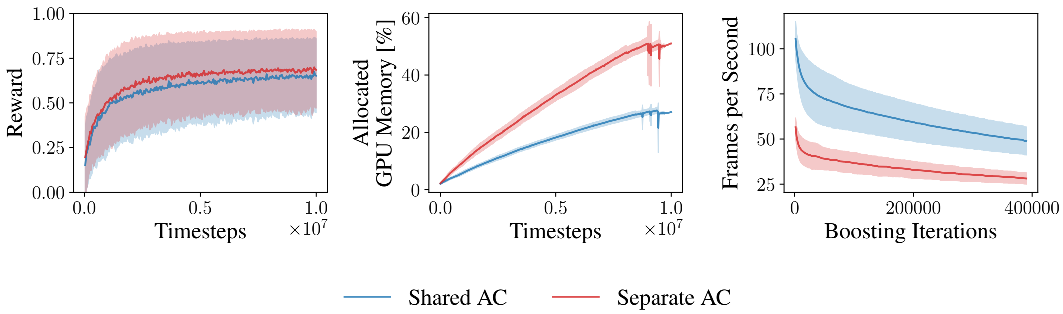


Figure 7: **Shared Actor-Critic.** Sharing the tree structure significantly increases training efficiency and memory, without impacting on the score.

Comparing different algorithmic backbones, we find PPO to be the strongest. PPO GBRL excelled in the MiniGrid and Football domains, and performed comparably with NN in classic control tasks. PPO GBRL's success can be attributed to its alignment with GBRL's incremental learning strategy. On the other hand, A2C's single gradient update per rollout may limit its effectiveness and contribute to its underwhelming performance in many environments. Similarly, AWR's design for multiple sample updates results in very large ensembles, creating a trade-off between large, slow, and memory-intensive ensembles, and lighter, less performant versions.

6 Conclusion

Historically, RL practitioners have relied on tabular, linear, and NN-based function approximators. But, GBT, a widely successful tool in supervised learning, has been absent from this toolbox. We present a method for effectively integrating it into RL and demonstrate domains where it excels compared to NNs. GBRL is a step toward solutions that are more interpretable, well suited for real-world tasks with structured data, or capable of deployment on low-compute devices.

The choice of an RL method depends on the task characteristics: tabular and linear approaches are suitable for small state spaces or simple mappings, while NNs handle complex relationships in unstructured data. GBT thrives in complex, yet structured environments. In such cases, we observe the advantage of GBRL over NNs, reflecting its already known benefits in supervised learning.

A crucial component of GBRL is our efficient adaptation of GBT for AC methods, which allows the simultaneous optimization of distinct objectives. We optimized this approach for large-scale ensembles using GPU acceleration (CUDA). Furthermore, GBRL integrates seamlessly with existing RL libraries, promoting ease of use and adoption.

7 Limitations and Future Directions

In this work, we integrated the highly popular GBT, typically used in supervised learning, into RL. Our results show that GBT is competitive across a range of problems. However, we identified several limitations and compelling areas for further research. First, a significant challenge lies in the continuous generation of trees. As the policy improves through numerous updates, the size of the ensemble increases. This unbounded growth has implications for memory usage, computational efficiency, and the feasibility of online real-time adaptation. The problem is exacerbated by off-policy methods that build many trees per sample. Moreover, the redundancy of trees, especially those from early stages, suggests that the final policy could be represented with a much smaller ensemble. Consequently, developing strategies for tree pruning, ensemble compression, or dynamically managing ensemble size could offer crucial optimizations without compromising performance.

Another key challenge lies in effectively integrating GBT with additional state-of-the-art RL algorithms such as DDPG [23] or SAC [17]. These require differentiable Q-functions to update the policy. Since GBTs are not differentiable, new solutions are needed to incorporate them into these algorithms. One such possible direction can be probabilistic trees, where each node represents the probability of traversing the graph.

References

[1] D. Abel, A. Agarwal, F. Diaz, A. Krishnamurthy, and R. E. Schapire. Exploratory gradient boosting for reinforcement learning in complex domains, 2016.

[2] S. O. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning, 2020.

[3] O. Bastani, Y. Pu, and A. Solar-Lezama. Verifiable reinforcement learning via policy extraction, 2019.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL `http://dx.doi.org/10.1613/jair.3912`.

[5] N. Brukhim, E. Hazan, and K. Singh. A boosting approach to reinforcement learning, 2023.

[6] T. Chen. Machine learning challenge winning solutions, 2023. `https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions`.

[7] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, Aug. 2016. doi: 10.1145/2939672.2939785. URL `http://dx.doi.org/10.1145/2939672.2939785`.

[8] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

[9] Q. Delfosse, S. Sztwiertnia, M. Rothermel, W. Stammer, and K. Kersting. Interpretable concept bottlenecks to align reinforcement learning agents, 2024.

[10] A. Delgado-Panadero, B. Hernandez-Lorca, M. T. Garcia-Ordas, and J. A. Benitez-Andrades. Implementing local-explainability in gradient boosting trees: Feature contribution. *Information Sciences*, 589:199–212, Apr. 2022. ISSN 0020-0255. doi: 10.1016/j.ins.2021.12.111. URL `http://dx.doi.org/10.1016/j.ins.2021.12.111`.

[11] T. Duan, A. Avati, D. Y. Ding, K. K. Thai, S. Basu, A. Y. Ng, and A. Schuler. Ngboost: Natural gradient boosting for probabilistic prediction, 2020.

[12] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451. URL `https://doi.org/10.1214/aos/1013203451`.

[13] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38 (4):367–378, 2002. ISSN 0167-9473. doi: https://doi.org/10.1016/S0167-9473(01)00065-2. URL `https://www.sciencedirect.com/science/article/pii/S0167947301000652`. Nonlinear Methods and Data Mining.

[14] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. Revisiting deep learning models for tabular data, 2023.

[15] L. Grinsztajn, E. Oyallon, and G. Varoquaux. Why do tree-based models still outperform deep learning on tabular data?, 2022.

[16] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, D. Pedreschi, and F. Giannotti. A survey of methods for explaining black box models, 2018.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[18] S. Ivanov and L. Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks, 2021.

[19] L. Katzir, G. Elidan, and R. El-Yaniv. Net-dnf: Effective deep modeling of tabular data. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=73WTGs96kho.

[20] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[21] K. Kersting and K. Driessens. Non-parametric policy gradients: a unified treatment of propositional and relational domains. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 456–463, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390214. URL https://doi.org/10.1145/1390156.1390214.

[22] K. Kurach, A. Raichuk, P. Stańczyk, M. Zając, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, and S. Gelly. Google research football: A novel reinforcement learning environment, 2020.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2019.

[24] B. London, L. Lu, T. Sandler, and T. Joachims. Boosted off-policy learning. In F. Ruiz, J. Dy, and J.-W. van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206, pages 5614–5640, 2023.

[25] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839, 2020.

[26] I. Lyzhin, A. Ustimenko, A. Gulin, and L. Prokhorenkova. Which tricks are important for learning to rank?, 2023.

[27] H. Ma, J. Cao, Y. Fang, W. Zhang, W. Sheng, S. Zhang, and Y. Yu. Retrieval-based gradient boosting decision trees for disease risk assessment. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3468–3476, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539052. URL https://doi.org/10.1145/3534678.3539052.

[28] A. Malinin, L. Prokhorenkova, and A. Ustimenko. Uncertainty in gradient boosting via ensembles, 2021.

[29] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/file/96a93ba89a5b5c6c226e49b88973f46e-Paper.pdf.

[30] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/96a93ba89a5b5c6c226e49b88973f46e-Paper.pdf.

[31] D. McElfresh, S. Khandagale, J. Valverde, V. Prasad C, G. Ramakrishnan, M. Goldblum, and C. White. When do neural nets outperform boosted trees on tabular data? In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 76336–76369. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/f06d5ebd4ff40b40dd97e30cee632123-Paper-Datasets_and_Benchmarks.pdf.

[32] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. 2016. doi: 10.48550/ARXIV.1602.01783. URL https://arxiv.org/abs/1602.01783.

[33] NVIDIA, P. Vingelmann, and F. H. Fitzek. Cuda, release: 10.2.89, 2020. URL https://developer.nvidia.com/cuda-toolkit.

[34] K. Ota, D. K. Jha, and A. Kanezaki. Training larger networks for deep reinforcement learning, 2021.

[35] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.

[36] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features, 2019.

[37] Y. Qing, S. Liu, J. Song, H. Wang, and M. Song. A survey on explainable reinforcement learning: Concepts, algorithms, challenges, 2023.

[38] A. Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

[39] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[40] S. Ruder. An overview of gradient descent optimization algorithms, 2017.

[41] B. Scherrer and M. Geist. Local policy search in a convex space and conservative policy iteration as boosted policy search. In T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 35–50, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44845-8.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[43] H. Seto, A. Oyama, S. Kitora, H. Toki, R. Yamamoto, J. Kotoku, A. Haga, M. Shinzawa, M. Yamakawa, S. Fukui, and T. Moriyama. Gradient boosting decision tree becomes more reliable than logistic regression in predicting probability for diabetes with big data. *Scientific Reports*, 12(1):15889, Oct. 2022.

[44] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences, 2019.

[45] F. Sigrist. Gaussian process boosting, 2022.

[46] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training, 2021.

[47] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[48] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[49] Z. Tian, J. Xiao, H. Feng, and Y. Wei. Credit risk assessment based on gradient boosting decision tree. *Procedia Computer Science*, 174:150–160, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.06.070. URL https://www.sciencedirect.com/science/article/pii/S1877050920315842. 2019 International Conference on Identification, Information and Knowledge in the Internet of Things.

[50] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis. Gymnasium, Mar. 2023. URL https://zenodo.org/record/8127025.

[51] A. Ustimenko and L. Prokhorenkova. Stochasticrank: Global optimization of scale-free discrete functions, 2020.

[52] A. Ustimenko, A. Beliakov, and L. Prokhorenkova. Gradient boosting performs gaussian process inference, 2023.

[53] K. Wang, J. Lu, A. Liu, G. Zhang, and L. Xiong. Evolving gradient boost: A pruning scheme based on loss improvement ratio for learning under concept drift. *IEEE Trans. Cybern.*, 53(4): 2110–2123, Apr. 2023.

[54] S. Wassan, B. Suhail, R. Mubeen, B. Raj, U. Agarwal, E. Khatri, S. Gopinathan, and G. Dhiman. Gradient boosting for health iot federated learning. *Sustainability*, 14(24), 2022. ISSN 2071-1050. doi: 10.3390/su142416842. URL `https://www.mdpi.com/2071-1050/14/24/16842`.

[55] C. Zhang, Y. Zhang, X. Shi, G. Almpanidis, G. Fan, and X. Shen. On incremental learning for gradient boosting decision trees. *Neural Processing Letters*, 50(1):957–987, Aug 2019. ISSN 1573-773X. doi: 10.1007/s11063-019-09999-3. URL `https://doi.org/10.1007/s11063-019-09999-3`.

Appendix

This appendix provides supplementary materials that support the findings and methodologies discussed in the main text. It is organized into four sections to present the full experiment results, implementation details, hyperparameters used during the experiments, training progression plots, and experimental plots, respectively. These materials offer detailed insights into the research process and outcomes, facilitating a deeper understanding and replication of the study.

A Implementaion Details and Hyperparameters

Included in this section are implementation details, information regarding compute resources, and tables containing the hyperparameters used in our experiments enabling the reproducibility of our results. Table 1 lists GBRL hyperparameters for all experiments.

A.1 Environments

The Football domain consists of a vectorized 115-dimensional observation space that summarizes the main aspects of the game and 19 discrete actions. We focus on its academy scenarios, which present situational tasks involving scoring a single goal. A standard reward of +1 is granted for scoring, and we employed the "Checkpoints" shaped reward structure. This structure provides additional points as the agent moves closer towards the goal, with a maximum reward of 2 per scenario. The Atari-ram environment consists of a vectorized 128-dimensional observational space representing the 128 byte RAM state and up to 18 discrete actions. We trained agents in both domains for 10M timesteps.

The MiniGrid environment [8] is a 2D grid world with goal-oriented tasks requiring object interaction. The observation space consists of a 7x7 image representing the grid, a mission string, and the agent's direction. Each tile in the observed image contains a 3D tuple dictating an object's color, type, and state. All MiniGrid tasks emit a reward of +1 for successful completion and 0 otherwise.

We trained our NN-based agents on a flattened observation space using the built-in one-hot wrapper. For GBRL agents, we generated a 51-dimensional categorical observational space by encoding each unique tile tuple as a categorical string to represent the observed image. Categorical features were added for the agent's direction (up, left, right, down) and missions. All agents were trained for 1M timesteps, except for PutNear, FourRooms, and Fetch tasks, which were trained for 10M based on the reported values for PPO NN in RL Baselines3 Zoo.

A.2 Compute Resources

All experiments were done on the NVIDIA NGC platform on a single NVIDIA V100-32GB GPU per experiment. Training time and compute requirements vary between algorithms and according to hyperparameters. The number of boosting iterations has the largest impact on both runtime and memory. GBRL experimental runs required from 1GB to 24GB of GPU memory. Moreover, runtime varied from 20 minutes for 1M timesteps training on classic environments and up to 5 days for 10M timesteps on Atari-ram. NN experimental runs required up to 3GB of GPU memory and runtime ranged from 10 minutes and up to 3 days. The total compute time for all experiments combined was approximately 1800 GPU hours. Additionally, the research project involved preliminary experiments and hyperparameter tuning, which required an estimated additional 168 GPU hours.

B Detailed Results Tables

This section contains tables presenting the mean and standard deviation of the average episode reward for the final 100 episodes within each experiment. More specifically, Table 2 presents results for Continuous Control & Block2D environments, Tables 3 and 4 present results for the high-dimensional vectorized environments, and Table 5 presents results for the categorical environments.

	batch size	clip range	ent coef	gae lambda	gamma	num epochs	num steps	num envs	policy lr	value lr
Acrobot	512	0.2	0.0	0.94	0.99	20	128	16	0.16	0.034
CartPole	64	0.2	0.0	0.8	0.98	1	128	8	0.029	0.015
LunarLander	256	0.2	0.0033	0.98	0.999	20	512	16	0.031	0.003
MountainCar	256	0.2	0.033	0.98	0.999	20	512	16	0.031	0.003
MountainCar Continuous	256	0.2	0.033	0.98	0.999	20	512	16	0.031	0.003
Pendulum	512	0.2	0.0	0.93	0.91	20	256	16	0.031	0.013
Football	512	0.2	0.0	0.95	0.998	10	256	16	0.033	0.006
Atari-Ram	64	0.92	8e-5	0.95	0.99	4	512	16	0.05	0.002
MiniGrid	512	0.2	0.0	0.95	0.99	20	256	16	0.17	0.01

(a) PPO. For continuous action spaces we used log std init = -2 and log std lr = lin_0.0017. We utilized gradient norm clipping for Gym environments. Specifically, 10 for the value gradients and 150 for the policy gradients.

	ent coef	gae lambda	gamma	num steps	num envs	policy lr	value lr	log std init	log std lr
Acrobot	0.0	1	0.99	8	4	0.79	0.031	-	-
CartPole	0.0	1	0.99	8	16	0.13	0.047	-	-
LunarLander	0.0	1	0.995	5	32	0.16	0.04	-	-
MountainCar	0.0	1	0.99	8	16	0.64	0.032	-	-
MountainCar Continuous	0.0	1	0.995	128	16	0.0008	2.8e-6	0	0.0004
Pendulum	0.0	0.9	0.9	10	32	0.003	0.056	-2	0.00018
Football	0.0004	0.95	0.998	128	8	0.87	0.017	-	-
Atari-Ram	0.0009	0.95	0.993	128	8	0.17	0.013	-	-
MiniGrid	0.0	0.95	0.99	10	128	0.34	0.039	-	-

(b) A2C

	batch size	ent coef	gae lambda	gamma	train freq	gradient steps	num envs	policy lr	value lr	log std init	log std lr
Acrobot	1024	0.0	0.95	0.99	2000	150	1	0.05	0.1	-	-
CartPole	1024	0.0	0.95	0.99	2000	150	1	0.05	0.1	-	-
LunarLander	1024	0.0	0.95	0.99	2000	150	1	0.05	0.1	-	-
MountainCar	64	0.0	0.95	0.99	2000	150	1	0.64	0.032	-	-
MountainCar Continuous	64	0.0	0.95	0.99	2000	150	1	0.089	0.083	-2	lin_0.0017
Pendulum	1024	0.0	0.9	0.9	1000	50	1	0.003	0.07	-2	0.0005
Football	512	0.03	0.95	0.99	750	10	1	0.09	0.00048	-	-
Atari-Ram	1024	0.0	0.95	0.993	2000	50	1	0.0779	0.0048	-	-
MiniGrid	1024	0.0	0.95	0.99	1500	25/100*	1	0.0075	0.005	-	-

(c) AWR. For all envs, buffer size = 50,000, β = 0.05. *MiniGrid environments used 100 gradient steps for tasks trained for 1M steps, and 25 gradient steps for tasks trained for 10M steps, for a reduced tree size.

Table 1: **GBRL hyperparameters** - NN represented by an MLP with two hidden layers.

Table 2: Continuous-Control and Box2D environments: Average episode reward for the final 100 episodes.

	Acrobot	**CartPole**	**LunarLander**	**MountainCar**	**MountainCar Continuous**	**Pendulum-v1**
NN: A2C	-82.27 ± 3.29	500.00 ± 0.0	-43.01 ± 106.26	-148.90 ± 24.10	92.66 ± 0.32	$\mathbf{-183.64 \pm 22.32}$
GBRL: A2C	-90.73 ± 2.98	500.00 ± 0.0	$\mathbf{47.93 \pm 41.00}$	-124.42 ± 5.74	93.15 ± 1.19	-538.83 ± 66.25
NN: AWR	-102.53 ± 57.25	500.00 ± 0.0	$\mathbf{282.48 \pm 1.96}$	-160.65 ± 53.97	18.93 ± 42.34	$\mathbf{-159.64 \pm 9.42}$
GBRL: AWR	-118.12 ± 33.54	497.54 ± 3.11	76.03 ± 56.62	-146.68 ± 24.53	44.38 ± 45.94	-1257.61 ± 98.10
NN: PPO	-74.83 ± 1.22	500.00 ± 0.0	261.73 ± 6.93	-115.53 ± 1.39	85.81 ± 7.51	-249.31 ± 60.00
GBRL: PPO	-87.82 ± 2.16	500.00 ± 0.0	248.72 ± 59.10	-110.55 ± 15.60	89.42 ± 5.73	-246.89 ± 20.61

C Training Plots

This section presents learning curves depicting model performance throughout the training phase. Figures 8 to 11 show the training reward as a function of environment steps of the agents trained in the experiments. The column order is: A2C, AWR, and PPO.

Table 3: Football Academy environments: Average episode reward for the final 100 episodes.

	3 vs 1 with keeper	Corner	Counterattack Easy	Counterattack Hard	Empty Goal	Empty Goal Close
NN: A2C	1.78 ± 0.10	1.00 ± 0.17	$\mathbf{1.58 \pm 0.35}$	$\mathbf{1.43 \pm 0.17}$	$\mathbf{1.93 \pm 0.05}$	2.0 ± 0.0
GBRL: A2C	1.59 ± 0.17	1.01 ± 0.07	1.11 ± 0.14	1.00 ± 0.05	1.81 ± 0.03	2.00 ± 0.00
NN: AWR	1.50 ± 0.37	1.01 ± 0.04	$\mathbf{1.59 \pm 0.36}$	1.18 ± 0.21	1.90 ± 0.08	1.92 ± 0.17
GBRL: AWR	1.66 ± 0.34	0.92 ± 0.05	0.95 ± 0.05	0.92 ± 0.05	1.93 ± 0.07	2.0 ± 0.0
NN: PPO	1.61 ± 0.05	0.95 ± 0.02	1.43 ± 0.15	1.23 ± 0.18	$\mathbf{1.98 \pm 0.01}$	1.99 ± 0.00
GBRL: PPO	1.63 ± 0.19	1.05 ± 0.20	1.64 ± 0.09	1.23 ± 0.07	1.84 ± 0.06	2.0 ± 0.0

	Pass & Shoot keeper	Run Pass & Shoot keeper	Run to Score	Run to score w/ keeper	Single Goal vs Lazy
NN: A2C	1.41 ± 0.37	1.77 ± 0.08	1.87 ± 0.12	1.25 ± 0.23	$\mathbf{1.65 \pm 0.04}$
GBRL: A2C	1.60 ± 0.21	1.60 ± 0.14	1.82 ± 0.10	1.15 ± 0.08	1.31 ± 0.11
NN: AWR	1.26 ± 0.46	1.15 ± 0.14	1.81 ± 0.14	1.25 ± 0.34	1.28 ± 0.27
GBRL: AWR	1.35 ± 0.37	1.53 ± 0.40	$\mathbf{1.98 \pm 0.01}$	0.99 ± 0.16	1.03 ± 0.12
NN: PPO	1.31 ± 0.13	1.64 ± 0.16	1.91 ± 0.09	1.13 ± 0.06	1.68 ± 0.09
GBRL: PPO	$\mathbf{1.87 \pm 0.09}$	1.85 ± 0.08	1.83 ± 0.04	$\mathbf{1.95 \pm 0.02}$	1.73 ± 0.06

Table 4: Atari-ramNoFrameskip-v4 environments: Average episode reward for the final 100 episodes.

	Alien	Amidar	Asteroids	Breakout	Gopher
NN: A2C	$\mathbf{1802.24 \pm 323.12}$	$\mathbf{304.62 \pm 55.61}$	$\mathbf{2770.46 \pm 271.97}$	$\mathbf{76.69 \pm 30.08}$	$\mathbf{3533.84 \pm 118.50}$
GBRL: A2C	595.08 ± 43.51	48.71 ± 14.65	1402.66 ± 161.67	11.52 ± 2.34	502.20 ± 341.88
NN: AWR	739.82 ± 303.06	86.32 ± 40.16	$\mathbf{2308.68 \pm 257.72}$	26.57 ± 9.91	1471.93 ± 716.65
GBRL: AWR	829.99 ± 166.48	125.53 ± 25.25	1592.63 ± 109.96	17.32 ± 1.89	913.06 ± 79.95
NN: PPO	$\mathbf{1555.32 \pm 107.59}$	$\mathbf{310.93 \pm 80.13}$	$\mathbf{2309.46 \pm 145.66}$	$\mathbf{32.88 \pm 15.74}$	$\mathbf{2507.84 \pm 108.37}$
GBRL: PPO	1163.86 ± 76.54	186.32 ± 50.63	1514.34 ± 317.46	19.96 ± 1.93	1215.04 ± 81.01

	Kangaroo	Krull	MsPacman	Pong	SpaceInvaders
NN: A2C	$\mathbf{2137.6 \pm 425.64}$	$\mathbf{9325.38 \pm 777.12}$	$\mathbf{2007.64 \pm 116.52}$	$\mathbf{15.39 \pm 4.26}$	$\mathbf{462.30 \pm 35.56}$
GBRL: A2C	948.8 ± 483.80	5291.4 ± 433.35	989.68 ± 100.02	-12.80 ± 11.10	265.36 ± 44.64
NN: AWR	1214.8 ± 313.42	4519.78 ± 522.11	892.31 ± 289.36	-10.25 ± 2.11	842.00 ± 130.51
GBRL: AWR	$\mathbf{1809.26 \pm 37.51}$	$\mathbf{6419.26 \pm 387.76}$	$\mathbf{1641.84 \pm 284.19}$	-11.68 ± 3.79	397.85 ± 566.38
NN: PPO	2487.4 ± 829.65	9167.3 ± 294.30	2069.22 ± 202.48	18.50 ± 1.60	479.77 ± 65.07
GBRL: PPO	2160.8 ± 826.92	6888.66 ± 756.18	2069.22 ± 538.62	15.40 ± 6.55	434.84 ± 31.83

Table 5: MiniGrid environments: Average episode reward for the final 100 episodes.

	DoorKey-5x5	Empty-Random-5x5	Fetch-5x5-N2	FourRooms	GoToDoor-5x5
NN: A2C	0.96 ± 0.00	0.77 ± 0.42	0.43 ± 0.03	0.62 ± 0.19	0.05 ± 0.04
GBRL: A2C	0.96 ± 0.00	0.96 ± 0.00	0.62 ± 0.02	0.51 ± 0.07	0.78 ± 0.02
NN: AWR	0.57 ± 0.52	0.96 ± 0.00	0.90 ± 0.26	0.19 ± 0.12	0.95 ± 0.01
GBRL: AWR	$\mathbf{0.96 \pm 0.00}$	0.97 ± 0.00	$\mathbf{0.95 \pm 0.01}$	$\mathbf{0.54 \pm 0.05}$	0.94 ± 0.01
NN: PPO	0.78 ± 0.40	0.96 ± 0.00	0.89 ± 0.03	0.53 ± 0.03	0.60 ± 0.06
GBRL: PPO	0.96 ± 0.00	0.96 ± 0.00	$\mathbf{0.96 \pm 0.01}$	0.56 ± 0.04	$\mathbf{0.96 \pm 0.00}$

	KeyCorridorS3R1	PutNear-6x6-N2	RedBlueDoors-6x6	Unlock
NN: A2C	0.75 ± 0.42	0.01 ± 0.00	$\mathbf{0.30 \pm 0.22}$	0.77 ± 0.43
GBRL: A2C	0.39 ± 0.48	$\mathbf{0.18 \pm 0.018}$	0.0 ± 0.0	0.90 ± 0.09
NN: AWR	0.93 ± 0.00	$\mathbf{0.60 \pm 0.13}$	0.83 ± 0.00	0.96 ± 0.00
GBRL: AWR	0.94 ± 0.00	0.36 ± 0.01	0.84 ± 0.03	0.95 ± 0.00
NN: PPO	0.76 ± 0.42	0.001 ± 0.00	0.17 ± 0.40	0.97 ± 0.00
GBRL: PPO	$\mathbf{0.95 \pm 0.00}$	$\mathbf{0.44 \pm 0.19}$	$\mathbf{0.88 \pm 0.02}$	0.97 ± 0.00

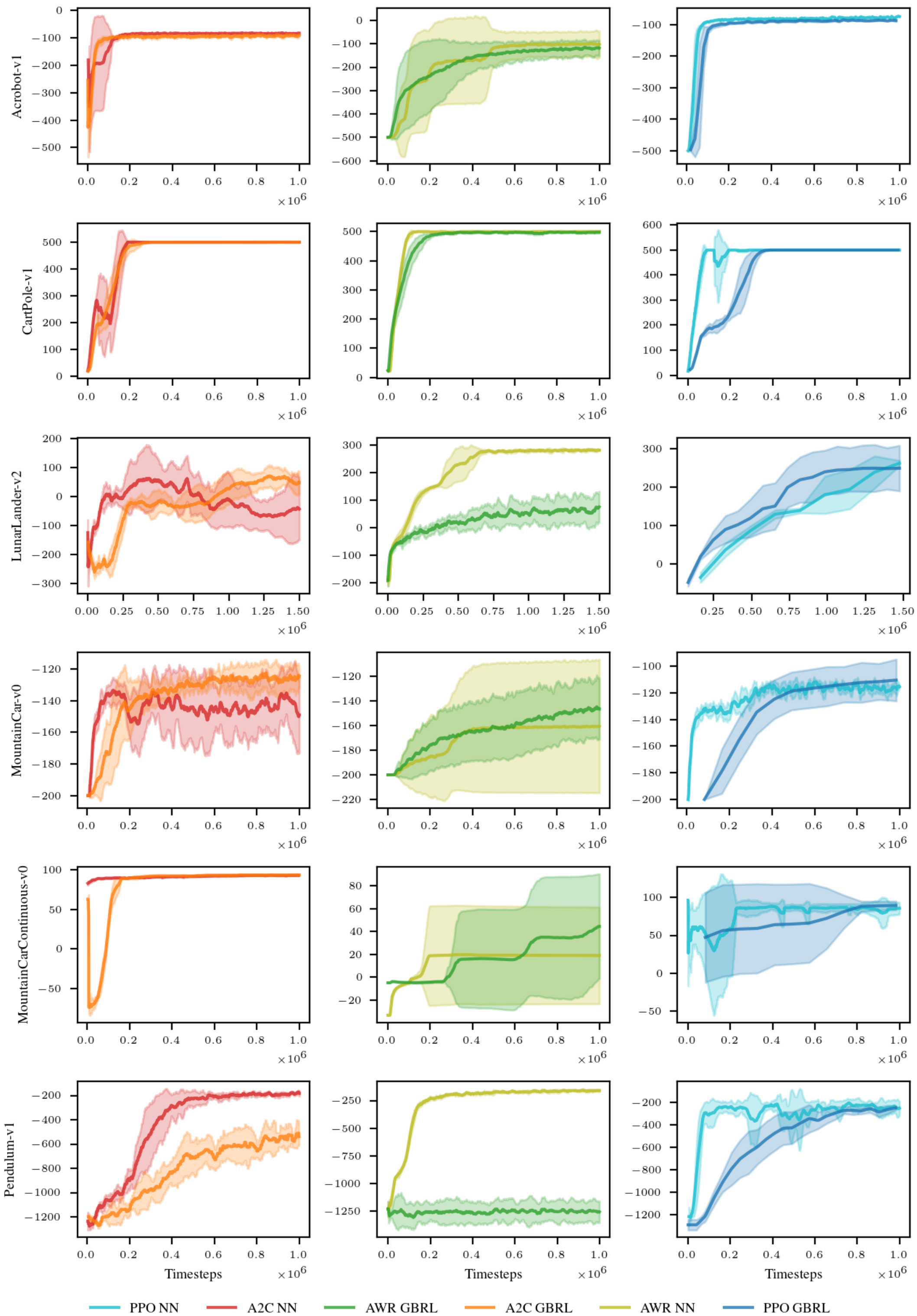


Figure 8: Classic Control and Box2D environments: Training reward as a function of environment steps.

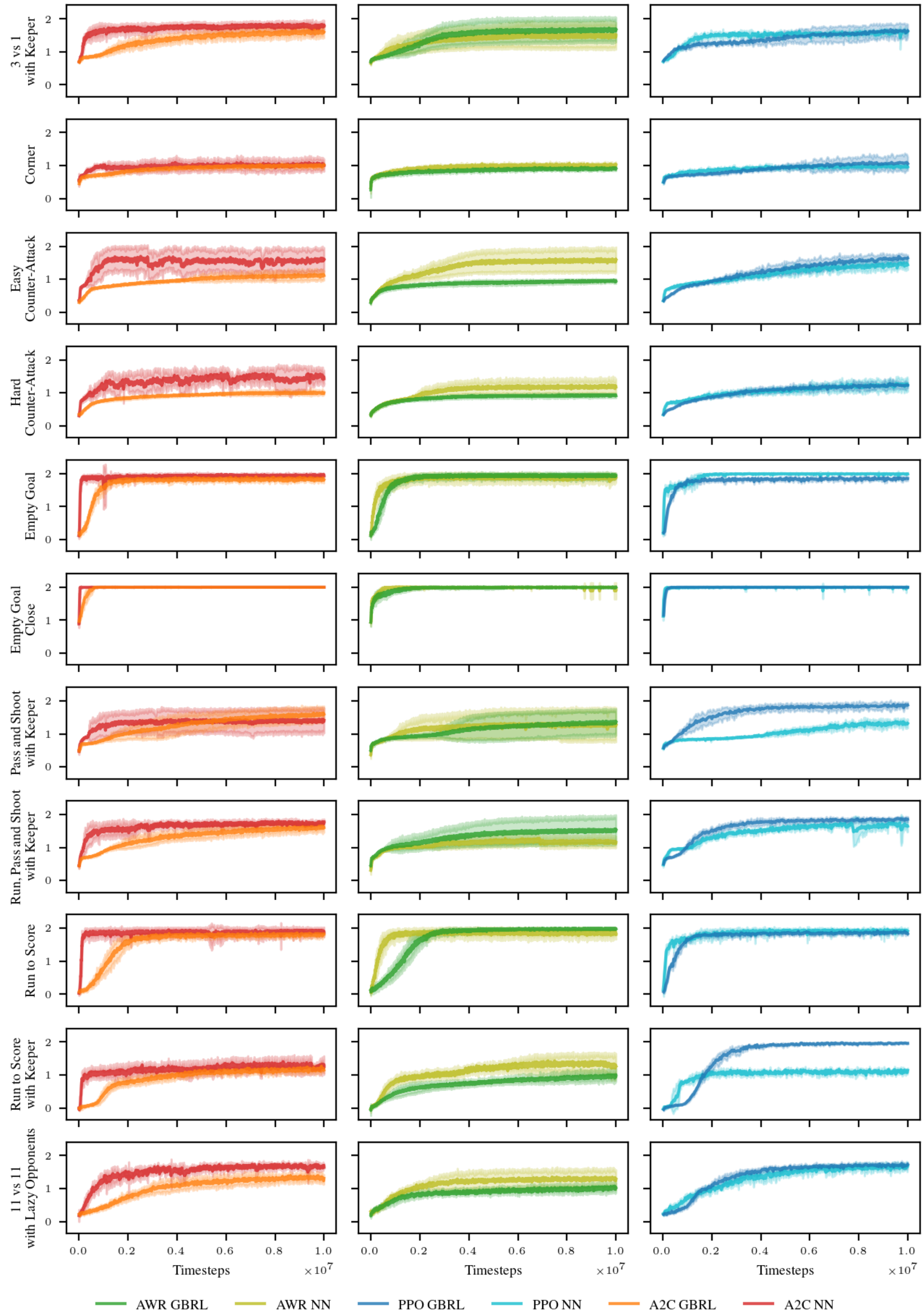


Figure 9: Football Academy environments: Training reward as a function of environment step.

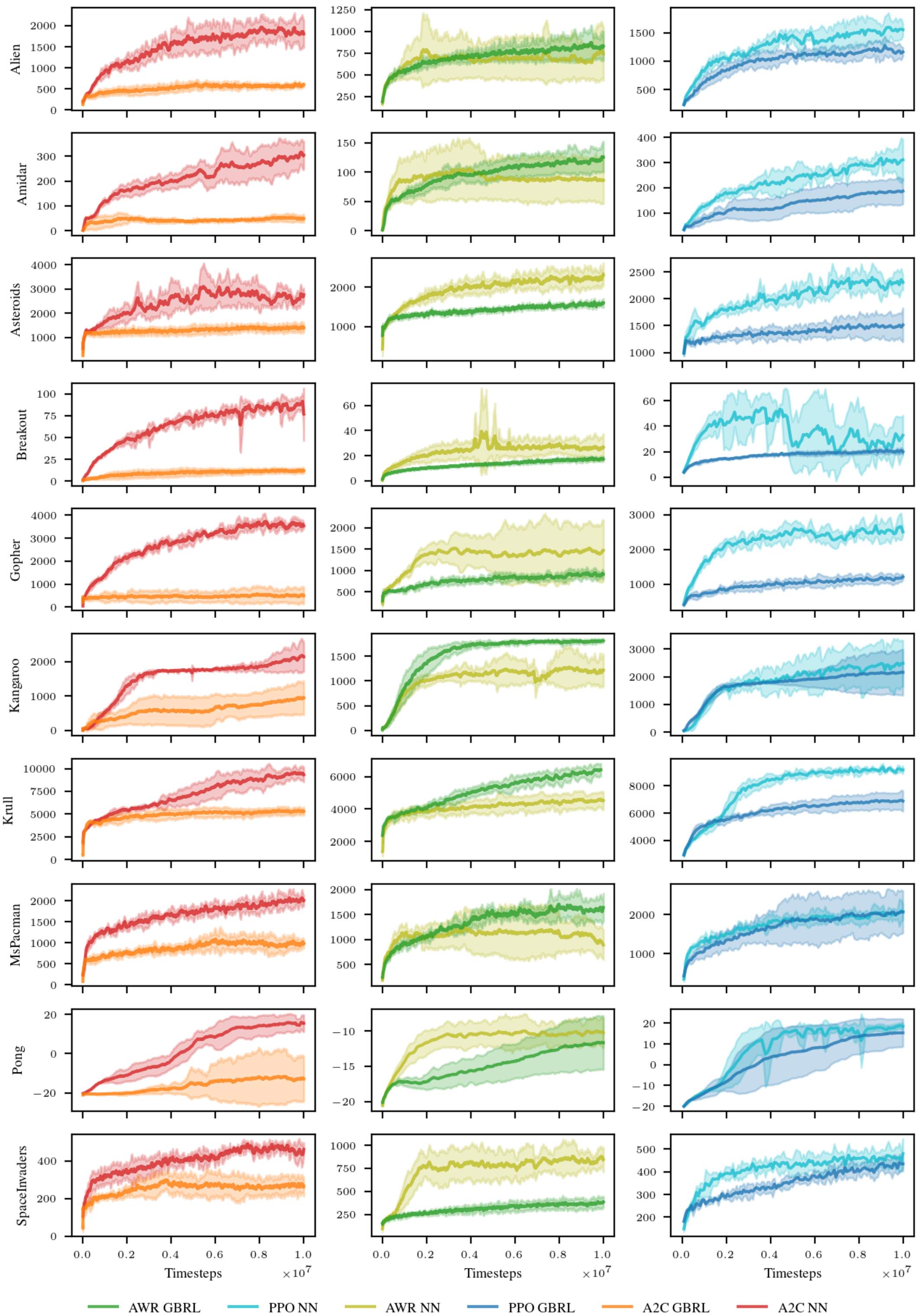


Figure 10: Atari-ramNoFrameskip-v4 environments: Training reward as a function of environment step.

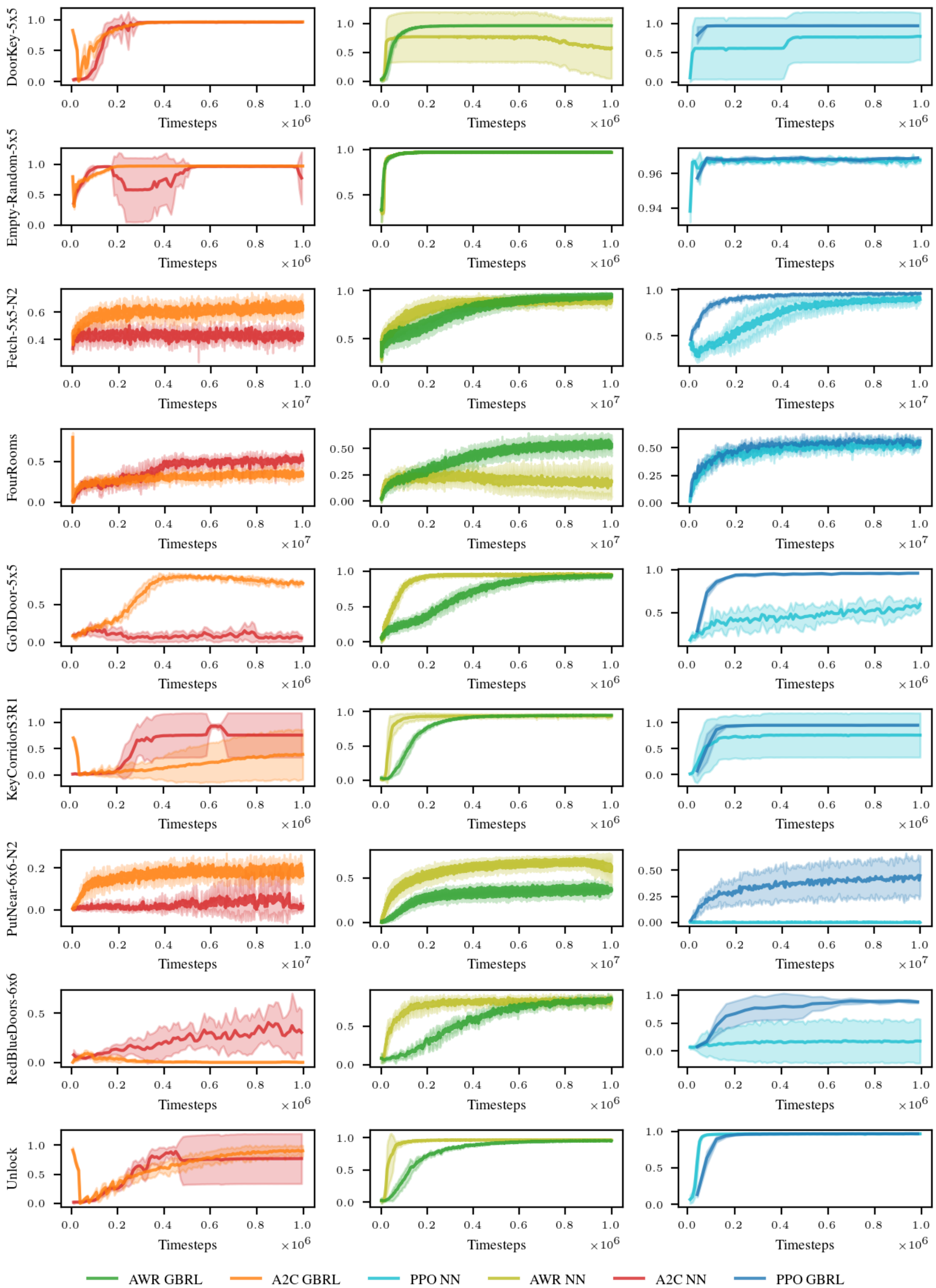


Figure 11: MiniGrid environments: Training reward as a function of environment step.

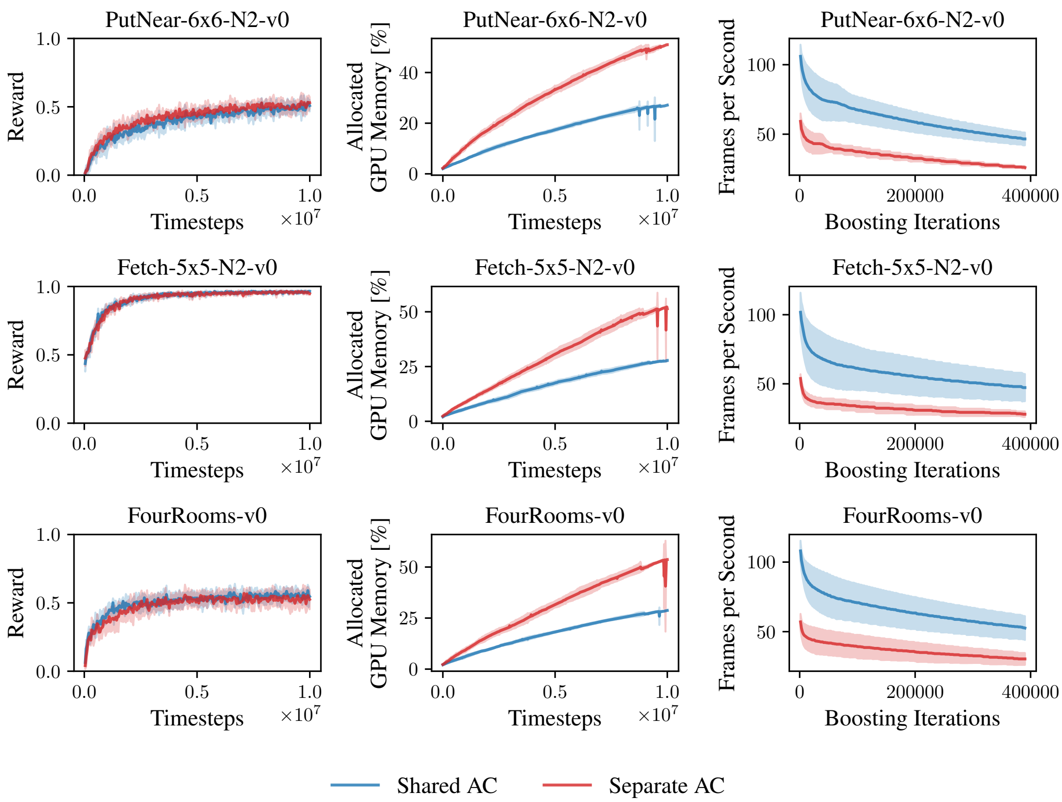


Figure 12: **Sharing actor critic tree structure significantly increases efficiency while retraining similar performance**. Training reward, GPU memory usage, and FPS, are compared across 10M environment (5 seeds, 3 MiniGrid environments)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope. The abstract provides a concise overview of the problem, the proposed solution, and the key contributions, including the introduction of GBRL, which extends the advantages of GBT to the RL domain. The paper demonstrates competitive performance with NN-based methods, especially in domains with structured or categorical features, and presents a high-performance, GPU-accelerated implementation that can be integrated with RL libraries. While the paper includes some aspirational goals related to future applications, these are clearly distinguished from the results shown and serve as motivation for further research..

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have included a limitations section, in which we discuss the computational efficiency and the memory limits of our current method. Additionally, we discuss the challenges in implementing our methods to other popular RL algorithms.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We do not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully discloses the architecture, the GBRL method, hyperparameters used, and the code will be released publicly.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is attached as supplimentary material and will be released publicly.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We share our hyperparameters and code required to reproduce our results in the appendix and as supplemental material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We report mean and standard deviation for the average episodic reward over the last 100 training episodes across five different agents trained random seeds per environment for all our experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide information on computer resources in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Yes, our work conforms with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: The primary focus of this paper is on the algorithmic development and performance evaluation of Gradient Boosting Trees in Reinforcement Learning (GBRL). While the paper does not explicitly discuss societal impacts, GBRL has the potential to positively influence various domains, such as inventory management, traffic signal optimization, network optimization, resource allocation, and robotics. These domains have direct implications for the day-to-day lives of many people. The enhanced performance and the capability of GBRL to be deployed on edge devices could bring AI to new applications, potentially leading to significant societal benefits. However, as this work is foundational research, it does not address specific societal impacts or applications directly.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We are the original owners of the models and algorithm code. We credit the repository and authors of all datasets/models we based our implementations on.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We will release the GBRL code publicly. The code is documented, with instructions provided on how to install, use, and incorporate within RL libraries. Additionally, details regarding the training process, license, limitations, and other relevant information are included in the documentation. The documentation will be made available alongside the code upon release.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.