Revisiting Adam for Streaming Reinforcement Learning

Florin Gogianu, Adrian Catalin Lutu, Razvan Pascanu

Keywords: streaming RL, Adam, optimisation, eligibility traces, distributional RL

Summary

Learning from a sequence of interactions, as soon as observations are perceived and acted upon, without explicitly storing them, holds the promise of simpler, more efficient and adaptive algorithms. For over a decade, however, deep reinforcement learning walked the contrary path, augmenting agents with replay buffers or parallel sampling routines, in an effort to tame learning instability. Recently, this topic has been revisited by Elsayed et al. (2024), focusing on update computation through eligibility traces and modifications to the optimisation routine, resulting in the StreamQ algorithm. In this work we take a step back, investigating the efficacy of established updates, such as those implemented by DQN and C51 within this online setting. Not only do we find that they perform well, but through analysing how the optimisation algorithm generally, and Adam in particular, interacts with these updates, we contend that two properties are essential for robust performance: i) the derivative of the objective is to be bounded and ii) weight updates are variance-adjusted. Rigorous and exhaustive experimentation demonstrates that C51, which exhibits both characteristics, is competitive with StreamQ across a subset of 55 Atari games. Using these insights, we derive a variance-adjusted algorithm based on eligibility traces, termed Adaptive Q(λ), which approaches double the human baseline on the same subset, surpassing existing methods by all performance metrics.

Contribution(s)

1. We highlight the effectiveness of ADAM and established objective functions in the streaming RL setting.
   Context: Recent literature suggests that the severe non-stationarity of streaming environments requires specialized algorithms. We contrast this by adapting DQN and C51 to the streaming setup, showing that when appropriately tuned, standard optimisation techniques can be surprisingly effective.
2. We establish variance-adjusted optimisation algorithms, together with carefully tuned ε-values, to be important components in streaming RL agents.
   Context: We provide mechanistic insights into the role of variance-adjusted updates and ADAM's epsilon value. We bring evidence that large ε values can act as a filter for sparse and noisy gradient components, allowing for a more stable learning dynamics.
3. We propose Adaptive $Q(\lambda)$.
   Context: $AQ(\lambda)$ combines eligibility traces with the variance adaptation mechanism of ADAM and bounded error signal, resulting in an algorithm which improves upon the performance of existing streaming RL methods.
4. We perform a large empirical evaluation in the streaming RL setting, extensively benchmarking STREAMQ alongside DQN and C51 across 55 Atari games and setting new performance expectations for these algorithms.
   Context: To provide a rigorous demonstration of our algorithmic contributions, the 55 Atari games were selected and ordered based on the performance correlation framework established by Aitchison et al. (2023).

Revisiting Adam for Streaming Reinforcement Learning

Florin Gogianu[1], **Adrian Catalin Lutu**[1], **Razvan Pascanu**[2]
florin.gogianu@gmail.com

[1]**Bitdefender, Bucharest, Romania**, [2]**Mila, Montreal, Canada**

Abstract

Learning from a sequence of interactions, as soon as observations are perceived and acted upon, without explicitly storing them, holds the promise of simpler, more efficient and adaptive algorithms. For over a decade, however, deep reinforcement learning walked the contrary path, augmenting agents with replay buffers or parallel sampling routines, in an effort to tame learning instability. Recently, this topic has been revisited by Elsayed et al. (2024), focusing on update computation through eligibility traces and modifications to the optimisation routine, resulting in the StreamQ algorithm. In this work we take a step back, investigating the efficacy of established updates, such as those implemented by DQN and C51 within this online setting. Not only do we find that they perform well, but through analysing how the optimisation algorithm generally, and Adam in particular, interacts with these updates, we contend that two properties are essential for robust performance: i) the derivative of the objective is to be bounded and ii) weight updates are variance-adjusted. Rigorous and exhaustive experimentation demonstrates that C51, which exhibits both characteristics, is competitive with StreamQ across a subset of 55 Atari games. Using these insights, we derive a variance-adjusted algorithm based on eligibility traces, termed Adaptive Q(λ), which approaches double the human baseline on the same subset, surpassing existing methods by all performance metrics.

1 Introduction

Much of the performance gains in deep reinforcement learning (DRL) over the past decade can be traced back to scaling the amount of off-policy data the optimisation routine can use during training. Ever-increasing experience replay buffers or massively parallel sampling of the environment, allow stochastic gradient descent algorithms originally developed for empirical risk minimisation to play for their strengths, at the cost of large memory, sample and algorithmic complexity. It became rather the rule that we tackled environments that fitted in 4Kb of ROM with agents consisting of 7GB replay buffers and millions of parameters in an effort to meet the stochastic gradient descent requirements for stability (Bowling & Elelimy, 2025b).

In contrast, few voices in the community started to argue for developing computationally bounded agents, with a reduced complexity compared to that of the environment (Javed & Sutton; Bowling & Elelimy, 2025a; Lewandowski et al., 2025). Thus, moving away from learning with batches sampled from pseudo-stationary buffers and towards learning from each interaction at a time, in a streaming fashion, without storing data or leveraging parallel environments.

Arguably, a major desideratum for this vision is the development of new RL objectives and optimisation algorithms that learn stably and efficiently purely from sequential data. Inroads have been opened in by van Seijen et al. (2016); van Hasselt et al. (2014); Javed et al. (2024) and recent results by Elsayed et al. (2024); Elelimy et al. (2025) suggest a way forward even for historically difficult

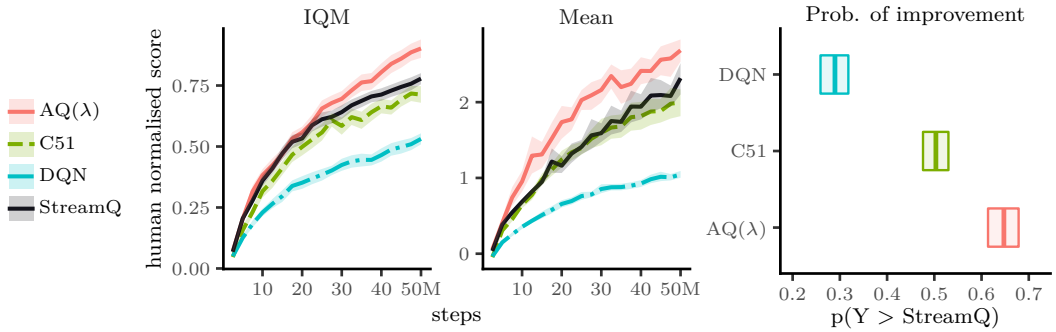

Figure 1: IQM and Mean human normalised score on 55 ATARI games. All agents learn solely from the current transition, without experience replay, or target network. C51 and AQ(λ) demonstrate the usefulness of bounded objectives and variance-adjusted optimisation.

control benchmarks such as the Arcade Learning Environment (ALE). These recent breakthroughs in the online setting have been attributed to the resurgence of eligibility traces (Sutton, 1988; Sutton & Barto, 2018) based methods (van Hasselt et al., 2014; Mahmood & Sutton, 2015; White & White, 2016; van Hasselt et al., 2021).

Indeed, eligibility traces for long held the promise of solving the central problem of propagating present or future rewards to past states and actions by updating value functions at every transition using multistep returns. Improved temporal credit assignment would in turn unlock lower sample complexity and help close the gap to batch-RL[1] algorithms. Elsayed et al. (2024) and Elelimy et al. (2025) both highlight the importance of eligibility traces among the other algorithmic novelties they employ, such as sparse initialisation of weight, novel regularized objectives and optimisers, as well as liberal use of normalisation for observations, rewards and activations, in breaking the "streaming barrier" towards an efficient online learner.

It has been remarked however that eligibility trace methods bear more than a passing similarity to stochastic gradient descent with momentum (SGD-M) (Polyak, 1964), even as recent as Elsayed et al. (2024). Furthermore, optimisation algorithms that also implement running averages of the derivative of the objective such as Adaptive Moment Estimation (ADAM) (Kingma & Ba, 2015) and to a lesser extent Root Mean Square Propagation (RMSProp) (Tieleman & Hinton, 2012), have been critical to training reinforcement learning (RL) agents with neural-network function approximation. In this light, we aim to revisit whether:

Q1: Are objectives and optimisers developed for batch reinforcement learning competitive in the online setting?

Deploying a careful and extensive empirical protocol, we find that objectives, update rules and optimisation routines commonly used in batch-RL, yield surprisingly strong results once adapted to the streaming setting. These strong empirical findings should contribute on their own to shaping the discussion around the somewhat nascent field of streaming *deep* reinforcement learning. They hint towards the generality and robustness of TD(0) methods when coupled with ADAM–style updates. And should establish stronger baselines for the development of new streaming algorithms.

Q2: What explains the performance of TD(0) methods and ADAM–style updates?

Key for obtaining these results is the peculiar set of hyperparameters we settled on for ADAM. Our top configurations invariably required a coefficient of the running average of the gradient close to 1.0 and a very large value for the numerical stability term. In section 4 we test several hypotheses that could explain this choice of values. Similarly, we notice a preference for algorithms that have

[1]Following terminology in Elsayed et al. (2024), not to be confused with offline-RL.

a bounded derivative of the objective, such is the case of Categorical DQN (C51) and some Deep Q-Network (DQN) versions.

Finally, we distil these observations and our analysis into an algorithm based on eligibility traces we term Adaptive Q(λ) that improves on existing methods. To conclude this section we summarise the contributions we bring:

- **The unreasonable effectiveness of ADAM.** We show evidence that objectives and optimisers developed in the batch-RL setting are very strong performers in the streaming protocol once they are well tuned.
- **Mechanistic insights.** We identify and discuss several important properties for streaming RL: the variance-adjusted updates based on long gradient histories, the role of bounded derivatives of the objective, and the large effects of ε on performance.
- **Adaptive Q(λ).** We propose a new eligibility-trace update rule that improves on existing methods.
- **Extensive benchmarking.** We present a thorough empirical study, evaluating DQN and C51 which we adapt to the streaming setting, alongside STREAMQ, on 55 Atari games.

2 Background

This work is set in the classical episodic reinforcement learning setting and, for convenience follows the conventions defined by Elsayed et al. (2024), Elelimy et al. (2025) and others: an *agent* interacting with an *environment* generates a time-indexed sequence $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_T$, by following its state-conditioned action generating behaviour, the policy $A_t \sim \pi(\cdot|S_t)$. Its objective is to maximise the sum of future discounted return $G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$, with γ controlling how much weight to put on the most immediate rewards as opposed to the ones received later in the trajectory. It does so by estimating the expected return of being in state $S_t = s$ and following the policy π, which we call the value function $v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]$. When the estimator of v_π uses function approximation, we denote it $v(s, \boldsymbol{w})$, with $\boldsymbol{w}$ a parameter vector. Generally, we don't only care about estimating the utility of a state, also known as the *value prediction* problem. Moreover, we are interested in its counterpart, the *control* problem, with the corresponding action-value function $q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$ and its function approximation equivalent $q(s, a, \boldsymbol{w})$. Optimising the objective in the control problem is then a matter of finding the optimal policy $\pi^\star$ that maximises the state-action value function: $q_\pi(s, a) = \max_\pi q_\pi(s, a)$.

Temporal difference learning. The straightforward approach to estimating the expected discounted sum of rewards is to wait till the end of the episode and compute it for every preceding state. This estimate of the return makes for a learning target towards which we can adjust the current value function: $\boldsymbol{w}_{t+1} \doteq \eta\left(G_t - v\left(S_t, \boldsymbol{w}_t\right)\right) \nabla_{\boldsymbol{w}} v(S_t, \boldsymbol{w}_t)$. The downside of Monte-Carlo (MC) methods is the policy remains unchanged until the end of the episode. One way towards more frequent updates is to compute a learning target based on the estimator we are training, evaluated at the very next state, and the reward we received by taking an action at the current step: $R_{t+1} + \gamma v(S_{t+1}, \boldsymbol{w})$. The resulting objective function to minimize then becomes the temporal-difference (TD) error: $\delta \doteq R_{t+1} + \gamma v(S_{t+1}, \boldsymbol{w}) - v(S_t, \boldsymbol{w})$, of which we will make heavy use in this work. For completion, the TD error relevant for the control problem is $\delta \doteq R_{t+1} + \gamma \max_a q(S_{t+1}, a, \boldsymbol{w}) - q(S_t, A_t, \boldsymbol{w})$. Because the target uses the prediction at the immediate next state S_{t+1}, we call it a TD(0) method. But we can bootstrap the learning target farther into the future, giving way to n-step methods, where the return is estimated by: $G_{t:t+n} \doteq \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n v(S_{t+n}, \boldsymbol{w})$. Choosing n allows for interpolating between the two main estimators so far: the one-step look-ahead in TD(0) and MC.

On λ-returns. The vast majority of batch-RL algorithms are squarely set in the *forward*-view we just described. For a given state the agent "looks" forward in time at the future rewards received by the current policy and decides how to update the value estimate based on them. This works out nicely for TD(0) and small values of n for n-step return methods, but quickly becomes tricky to implement efficiently for other multistep methods, especially in conjunction with resampling strategies (Daley & Amato, 2019). One such alternative is the λ-return, an estimator of the return

that further balances the bias-variance tradeoff by computing an weighted average over all n-step returns along a trajectory, $G_t^\lambda \doteq (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_{t:t+n} + \lambda^{T-t-1}G_t$. Setting $\lambda = 0$ recovers TD(0), while with $\lambda = 1$ the estimator becomes the MC return. This estimator is rarely encountered in pure value-based methods with neural network approximation because it requires recomputing each n-step return every time.

Eligibility traces. λ-returns can however be efficiently implemented if we take on the *backward*-view where each update to the value function depends on the current TD-error and some running statistic of past events. The key insight is to have a *trace* vector $\boldsymbol{z}_t$ that mirrors and is the size of $\boldsymbol{w}$. While $\boldsymbol{w}$ stores the long-term coefficients required for estimating $v_\pi(s)$, the role of $\boldsymbol{z}$ is to record whenever a component of $\boldsymbol{w}$ was sensitive in producing an estimate. This record of the parameter "activity" is decayed towards 0 by λ and reset in terminal states. If the TD-error is non-zero, then $\boldsymbol{w}$ will be updated according to the values of the error and the corresponding components of the eligibility vector $\boldsymbol{z}$. Specifically, if we denote the sensitivityof the value function with respect to the weights $\boldsymbol{g}_t \doteq \nabla_{\boldsymbol{w}} v(S_t, \boldsymbol{w})$ and initialise $\boldsymbol{z}_0 \doteq \mathbf{0}$, then the update becomes:

$$\begin{aligned} \boldsymbol{z}_t &\doteq \gamma\lambda\boldsymbol{z}_{t-1} + \boldsymbol{g}_t \\ \boldsymbol{w}_{t+1} &\doteq \boldsymbol{w}_t + \eta\delta_t\boldsymbol{z}_t \end{aligned} \tag{1}$$

With linear function approximation the eligibility trace update in Eq. (1) can be equivalent to those of forward-view algorithms implementing λ-returns (Sutton, 1988; van Seijen & Sutton, 2014).

Objectives. Generally, in the forward-view, we set the TD-error objective for minimisation by formulating it as a mean squared error (MSE) function, $\mathcal{L}(\boldsymbol{w}) \doteq \delta^2$ and optimising it with semi-gradient descent. Very often in practice MSE is replaced by SmoothL1Loss, $L_\kappa(\delta) = \mathbb{1}_{|\delta|<\kappa}\left(0.5\delta^2/\kappa\right) + \mathbb{1}_{|\delta|\geq\kappa}\left(|\delta| - 0.5\kappa\right)$. For $\kappa = 1$ it smoothly transitions from the squared loss to the mean absolute error as the magnitude of δ increases.

Variance-adjusted optimisation methods. Most consequential to achieving the first strong results on challenging control problems using neural networks was the use of a new class of optimisation algorithms that divides the derivative of the objective function or an estimate of its first moment, by an estimate of the second moment. An early example was the use of RMSprop (Tieleman & Hinton, 2012) for successfully training DQN (Mnih et al., 2015). To our knowledge, Bellemare et al. (2017) introduces the use of ADAM in DQN-style methods and Hessel et al. (2018); Ceron & Castro (2021) confirm its advantages over RMSprop on a variety of agents. In this work we refer as *variance-adjusted methods* to algorithms that use the second moment of the gradient to scale the update. ADAM in particular approximates the first and second moments using β-normalised exponential moving averages. Ignoring bias-correction and letting $\boldsymbol{g}_t \doteq \nabla_{\boldsymbol{w}}\delta_t^2$, the update is:

$$\begin{aligned} \boldsymbol{m}_t &\doteq \beta_0\boldsymbol{m}_{t-1} + (1-\beta_0)\boldsymbol{g}_t & \boldsymbol{\rho} &\doteq \boldsymbol{m}_t/(\sqrt{\boldsymbol{v}_t} + \varepsilon) \\ \boldsymbol{v}_t &\doteq \beta_1\boldsymbol{v}_{t-1} + (1-\beta_1){\boldsymbol{g}_t}^2 & \boldsymbol{w}_t &\doteq \boldsymbol{w}_{t-1} - \eta\boldsymbol{\rho} \end{aligned} \tag{2}$$

Comparing to Eq. (1), we notice that both compute a running statistic of the gradient with respect to $\boldsymbol{w}$. However, whereas $\boldsymbol{z}$ accumulates the derivative of $v(S_t, \boldsymbol{w})$ and is reset periodically, $\boldsymbol{m}$ is a long-running average of the derivative of the objective and is generally not reset. While both are running statistics, their roles and dynamics are different. The eligibility trace is intended to accumulate the signal and tends to increase in magnitude, dampen oscillations and speed up learning, not unlike SGD-M (Polyak, 1964; Rumelhart et al., 1986), which was already pointed out by Elsayed et al. (2024). ADAM's EMA instead approximates the expected value of the gradients and tends to stay at the same scale as the signal.

Distributional RL. Rather than estimating only the expected return, distributional reinforcement learning (Bellemare et al., 2017) models the full distribution of the return G_t, represented by the random variable $Z(s, a, \boldsymbol{w})$ such that $q(s, a, \boldsymbol{w}) \doteq \mathbb{E}[Z(s, a, \boldsymbol{w})]$. One way to approximate it is using a discrete set of K fixed atoms $\{z_i\}_{i=1}^K$ spanning a specific range of possible returns, where the model estimates the probability mass $p_i(s, a, \boldsymbol{w}) \doteq \Pr\left(Z(s, a, \boldsymbol{w}) = z_i\right)$ associated with each atom. The learning objective then is to minimize the Kullback-Leibler divergence between the current distribution and a target distribution formed by the sample $R_{t+1} + \gamma Z(S_{t+1}, A^\star, \boldsymbol{w})$, where $A^\star \doteq \arg\max_a q(S_{t+1}, a, \boldsymbol{w})$, yielding the C51 algorithm.

3 Empirical setup

Adapting old algorithms to new setups. We closely follow the setup introduced by Elsayed et al. (2024), using the environment and normalisation wrappers provided by the authors, with no modification, for all the algorithms studied here. Our implementation of STREAMQ closely follows the reference, and we were able to confirm that it performs on par with the original work in MINATAR and the eight ATARI environments the authors originally evaluated it.

We adapt DQN and C51 to the online setting by removing the target network and the replay buffer. Distributional algorithms such as C51 (Bellemare et al., 2017) often have a much larger final layer, the size of $A \times K$, where K is the number of bins or quantiles. In order to compare estimators of the same capacity, in all our experiments we scale the layer before the output for non-distributional methods such that the number of weights is similar for all algorithms.

Table 1: Changes to DQN and C51 for the online setting

Preserved from STREAMQ	Other changes
▷ Sparse initialisation of weights	▷ No target network
▷ Observation and reward normalisation	▷ No experience replay
▷ Architecture and liberal use of LayerNorm	▷ Equalized network capacity

For clarification, we note here that Elsayed et al. (2024) also adapt DQN to the streaming setup by setting the replay buffer to a size of 1, along with an update frequency of 1, and call it DQN1. We will refer to this agent several times in this work. In comparison, our DQN implementation also removes the target network, uses the same architecture and initialisation as STREAMQ and the same environment wrappers.

Benchmarks and hyperparameter tuning sets. For our empirical study we settle for MINATAR (Young & Tian, 2019) and ALE (Bellemare et al., 2013), two of the better established benchmarks in discrete control. For the MINATAR experiments we always train on all five games included in the benchmark. Although Ceron & Castro (2021) establishes MINATAR as a good predictor of ALE performance, our experiments suggest significant differences in algorithm rankings in the online setting. Previous efforts in the online setting (Elsayed et al., 2024; Elelimy et al., 2025) evaluated their proposals on a limited subset of ATARI games. While we were not able to train on the full 57 game set either, we take further steps to make sure our results are predictive for the entire benchmark. Aitchison et al. (2023) studies how predictive is the performance on each ATARI game to the entire ALE benchmark, and it further identifies the most predictive subset of 5 games, called ATARI-5. We use this subset for all our hyper-parameter searches, with one change: we replace *DoubleDunk*, in which all agents flat-line, with the most predictive game not in the original 5-set, *Amidar*. We call this subset ATARI-5*. The 55 games in the main comparison are also selected based on Aitchison et al. (2023), in the order of their correlation to the aggregate performance.

Evaluation. We perform 7 training runs with different initialisations for each game in the main ATARI experiment and use 9 seeds for the MINATAR experiments. Hyperparameter tuning was performed with 3 to 5 runs. When tuning on ATARI-5* we only train for 12.5M steps making for 25% of the usual budget. Reporting the performance we follow Agarwal et al. (2021) in using stratified bootstrapping for confidence interval estimation, interquartile mean for aggregates, along with estimating the probability of improvement.

4 Are established objectives and optimisers effective in streaming RL?

We developed our initial observations and intuitions on the MINATAR environments. A relatively sparse grid-search on a subset of games with DQN and C51 revealed two important findings that we kept confirming for the rest of the project. First, that both algorithms gain in performance with increasingly large values of β_0, the coefficient that controls the "length" of the history of gradients in ADAM's exponential moving average of the gradient. The default value, of $\beta_0 = 0.9$ is used throughout batch-RL, and defines a relatively small sliding window, which ensures the trace captures

gradient information from the most recent several tens of steps. We settled for $\beta_0 = \beta_1 = 0.999$ for all our experiments instead. Second, we noticed a strong increase in performance with larger values of ε. We defer the discussion of this choice of hyperparameters to Sec. 5.

Fig.2 illustrates the final results on the five MINATAR environments. Surprisingly, DQN and C51 both outperform STREAMQ. Significant is the performance of DQN when compared to DQN1 from Elsayed et al. (2024) which always underperforms STREAMQ, given that they share the optimiser and the objective function. We explain this difference because of our decision to level the field by allowing all methods to use the environment wrappers and the same estimator architecture, as well as to our specific choice of hyperparameters for ADAM.

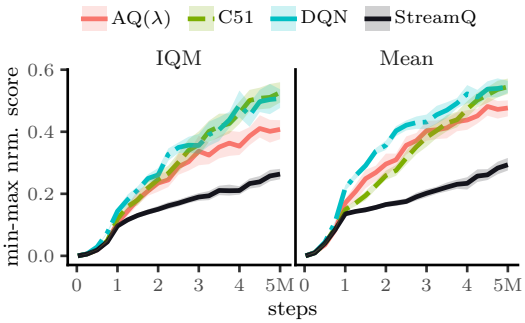

Figure 2: Aggregated normalised score on five MINATAR environments. Classical RL objectives using ADAM are strong performers.

Moving on to the 55 ATARI games selected as described above, the conclusions change to some degree and Figs. 1 and 8 paint a more nuanced view. Whereas DQN1 was failing catastrophically in the eight ATARI games selected by the authors, ours showcases a decent performance, although not to the level of STREAMQ. However, C51 manages to raise to the level of STREAMQ and is given a 50% chance of improving over it by the Mann-Whitney U-statistic (Mann & Whitney, 1947; Agarwal et al., 2021).

Given these results, we must answer the original question in the affirmative. The classical objective functions developed for batch-RL, when coupled with a properly tuned ADAM*, demonstrate a reasonable performance in the two benchmarks we evaluated on.*

5 What drives the performance of established objectives with ADAM?

Undoubtedly, one of the main reasons we are able to train these classic agents in the streaming setup in the first place, are the normalisation techniques and the weight initialisations introduced by Elsayed et al. (2024). The separate impact of reward, observation and activation normalisation and the sparse weight initialisation scheme, have been already thoroughly ablated in prior work, and we refer the reader to it.

We turn, in what follows, on what we believe are some of the factors that make these classical algorithms perform well in the streaming setup when compared to STREAMQ, which, in addition, employs eligibility traces and a novel optimisation algorithm that avoids updates that could overshoot the TD target, Overshooting-bounded Gradient Descent or ObGD.

Bounded objectives. The choice of objective function profoundly impacts optimisation stability in reinforcement learning, especially under the severe non-stationarity of the streaming setup. MSE is affected by unbounded gradients with respect to predictions. Consequently, when confronted with large Temporal Difference (TD) errors, the linear derivative of MSE translates these errors into massive, destabilizing gradient steps. Recent results by Palenicek et al. (2025) demonstrate that bounding gradient norms is critical for maintaining a stable effective learning rate under non-stationary targets and bootstrapping. Crucially, employing an objective function with strictly bounded gradients allows for the effective gradient update to be theoretically upper bounded. As shown by Palenicek et al. (2025), the Cross Entropy loss implemented by C51 is representative for this type of objective with bounded derivative, and we believe much of the performance it demonstrates in ATARI (Fig. 1) and MINATAR (Fig. 2) is because of this property. An increase in performance is also shown in Fig. 3, when changing the objective function from MSE to SmoothL1Loss.

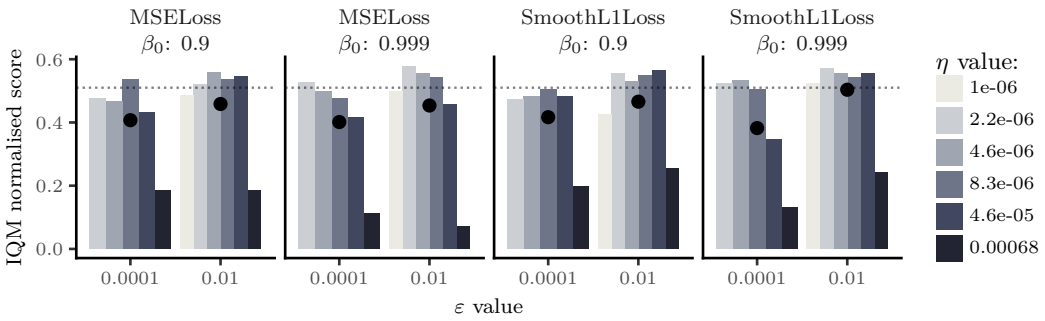

Figure 3: On aggregate, DQN performs better with larger β_0 values and bounded objectives such as SmoothL1Loss. • denotes the IQM score over η values given a value of ε. Each bar is the IQM of 5 games $\times$ 3 seeds $\times$ last 10 evaluations.

The longer the history of the gradients, the better. We explain in Sec. 2 that both ADAM and eligibility traces include mechanisms for storing gradient statistics. Although different in intent and dynamics, ADAM's $\boldsymbol{m}$, like the eligibility vector $\boldsymbol{z}$ accumulates gradient information with a window size determined by β_0. It maybe came as no surprise that in our initial experiments with MINATAR we noticed a strong preference towards large β_0 values, beyond the usual $\beta_0 = 0.9$ found in the literature. In Fig. 3, on the ATARI-5* subset, we notice a similar trend, where the robust median (IQM) over learning rates is higher for $\beta_0 = 0.999$. Similarly, in Fig. 4, on the same ATARI-5*, we plot on the left the two (ε, η) combinations that performed better for $\beta_0 = 0.9$ and observe that they perform worse than the ones with $\beta_0 = 0.999$.

ADAM's ε as a step-size scaling factor. The update in Eq. (2) would suggest that the increased performance as ε grows is an artefact of the interplay between the numerical stability term and step-size η. Indeed, in the $\eta/(\sqrt{\boldsymbol{v}} + \varepsilon)$ relation, as ε increases, the effective step size gets smaller, assuming $\sqrt{\boldsymbol{v}}$ constant. Note also that the updates scale higher with decreasing ε as the value of $\sqrt{\boldsymbol{v}}$ decreases. From this we could argue that increasing ε the size of the update decreases with the possible effect of a more stable (and slow) optimisation process. However, a second intuition is that this scaling effect should be compensated by picking a different step size η.

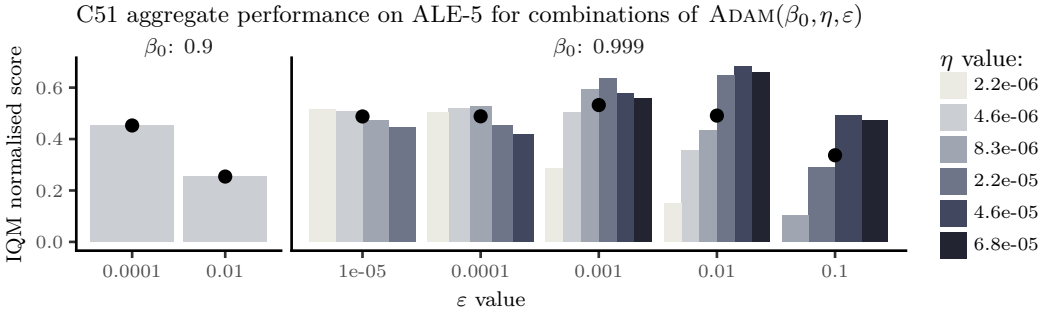

Figure 4: Performance increases with higher values of ε and there is no obvious scaling of the step size η that can compensate for it. • denotes the IQM score over η values given a value of ε. Each bar is the IQM of 5 games $\times$ 3 seeds $\times$ last 10 evaluations.

In order to test this hypothesis, we set the following experiment. We train C51 agents on the ATARI-5* subset for 12.5M steps, amounting to 25% of the standard training run. For five values of $\varepsilon \in \{1e{-}5, ..., 1e{-}1\}$ we do a grid search over η values. The bounded derivative of the C51 objective allows us to ignore possible stabilisation effects induced by ε in the presence of outlier target values

and further isolate the relative interplay between η and ε if it exists. Indeed, for this hypothesis to hold fully, we would expect a step-size η that performs equally well for each ε value. We observe in Fig. 4 that indeed there is an optimal step size for each ε value. However, the performance increases with ε instead of being constant at this optimal step size, therefore invalidating our initial hypothesis. An even clearer picture of the requirement to carefully tune ε is observed in Fig. 7, the result of a grid-search on MINATAR using Quantile Regression (QR), another distributional algorithm, that implements Huber quantile loss, also an objective with a bounded derivative. Details are available in App. A.2. We must conclude that the performance excess cannot be fully explained away by a proportional scaling of the step size and an additional mechanism must be at play.

ADAM's ε is an SNR filter. RL has the particularity of having to deal with a long tail of features that are rarely encountered by the agent: entities that appear only sparsely or after bottleneck states, objects and tokens that are relevant only to certain parts of the game. It is justified to assume these features are correlated with sparsely or noisily updated components of the weight vector. Updating both the sparse and the dense or the noisy and the stable directions of the parameter space with the same scale is likely to produce estimation errors, cause instabilities, and result in the policy diverging. Ideally, an optimiser should therefore be able to adjust the step-size of each component such that rare or noisy gradients result in conservative updates.

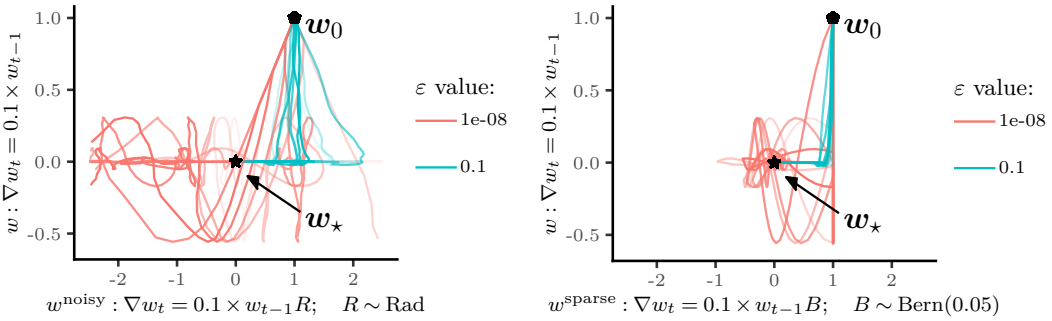

Figure 5: Large values for ADAM's ε allow for stable learning on problems which are noisy or sparse. The y-axis component receives a stable gradient signal in both cases, whereas on the x-axis the gradient can either change sign randomly (left) or be zero 95% of the time (right).

We hypothesise ADAM manifests a seldom-discussed implicit mechanism for scaling the updates differently across gradient components with variable noise or sparsity characteristics. Consider the denominator of the update, $\sqrt{\boldsymbol{v}} + \varepsilon$ with $vt \approx \boldsymbol{g}^2$ and ignore the bias correction. For frequently updated or stable features, the elements $\sqrt{\boldsymbol{v}_i}, i \in \mathcal{D}_{\text{dense/stable}}$ dominate the denominator and ADAM scales down the effective step size for the directions associated with these features. In contrast, for components of $\boldsymbol{v}$ associated with sparse or noisy features, will result in ε dominating the scaling term. Larger values of ε will be effective in this regime for bounding the maximum step size.

We propose two small toy examples to illustrate this behaviour in Fig. 5. Both optimisation problems depicted have just two parameters. The component on the y-axis always receives a constant gradient of $0.1 \times w_{t-1}$ in both problems. In the left panel, the derivative with respect to the second weight w^{noisy} has a similar form, but changes sign randomly at every step (on the x-axis). In the right panel we formulate a sparse variation of this problem where the component on the x-axis is set to zero 95% of the time. Therefore, in both problems the non-constant component of $\boldsymbol{v}$ will tend to be small, either because of noise, or because of sparse updates.

Indeed, for small values of ε, ADAM is overly sensitive to the sparse or noisy component, amplifying the updates whenever a component of $\boldsymbol{v}$ is small, along the corresponding dimension, resulting in unstable optimisation. In contrast, large values of ε that dominate the non-constant components of $\boldsymbol{v}$ allows for a filtering effect on the sparse or noisy dimension, resulting in stable and early convergence.

6 Adaptive Q(λ)

Previous sections highlight the importance of i) having a long memory of the sensitivity of the value function with respect to the decisions it produced as in eligibility traces or ADAM's EMA, ii) having an ε-modulated mechanism that adjusts the update based on the variance of this sensitivity and iii) that error signals should be bounded, like is the case of C51and SmoothL1Loss.

We combine these three ideas into Adaptive Q(λ). Taking $Q(\lambda)$ (Sutton, 1988) as starting point, the algorithm uses a momentum-based eligibility trace $\boldsymbol{z}_t$, alongside an exponential moving average of the squared gradients of the state-action value function, $\boldsymbol{v}_t$. We use this estimate of the second order moment of the gradient of the state-action value function to scale the eligibility trace before producing an update, similar to what ADAM is doing. By normalizing the eligibility trace vector $\boldsymbol{z}_t$ with this variance estimate, we ensure that the magnitude of the updates remains stable across different parameters, even in highly non-stationary streaming environments.

We noticed however that this update was having stability issues. Many of our training runs resulted in agents crashing because of large updates, despite the extensive normalisation techniques used in our setup. To further improve the convergence of our method, we simply clamp δ to $(-1, 1)$, resulting in a stable algorithm across hyperparameters. Notice how this is exactly the derivative of the SmoothL1Loss.

We reset the trace whenever the agent takes exploratory actions and when the episode ends, similar to STREAMQand others in the literature. Based on empirical experimentation, we opted not to reset $\boldsymbol{v}$, as doing so yielded slightly worse results. Alg. 1 compares and summarises the updates of Q(λ), Adaptive Q(λ) and DQN using ADAM.

Algorithm 1 Simplified comparison of update rules

For all updates we require: $\delta \leftarrow r + \gamma \max_a q(s', a, \boldsymbol{w}) - q(s, a, \boldsymbol{w})$

▷ *$Q(\lambda)$ update*

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{w}} q(s, a, \boldsymbol{w})$
$\boldsymbol{z} \leftarrow \gamma\lambda \boldsymbol{z} + \boldsymbol{g}$
$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta\delta \boldsymbol{z}$
if ep. done **or** non-greedy **then**
  $\boldsymbol{z} \leftarrow \boldsymbol{0}$
end if

▷ *$AQ(\lambda)$ update*

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{w}} q(s, a, \boldsymbol{w})$
$\boldsymbol{z} \leftarrow \gamma\lambda \boldsymbol{z} + \boldsymbol{g}$
$\boldsymbol{v} \leftarrow \gamma\lambda \boldsymbol{v} + (1 - \gamma\lambda)\boldsymbol{g}^2$
$\boldsymbol{\rho} \leftarrow \boldsymbol{z}/(\sqrt{\boldsymbol{v}} + \varepsilon)$
$\hat{\delta} \leftarrow \text{clip}\,(\delta, -1, 1)$
$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta\hat{\delta}\boldsymbol{\rho}$
if ep. done **or** non-greedy **then**
  $\boldsymbol{z} \leftarrow \boldsymbol{0}$
end if

▷ *Q-learning update (Adam)*

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{w}} \delta^2$
$\boldsymbol{m} \leftarrow \beta_0 \boldsymbol{m} + (1 - \beta_0)\boldsymbol{g}$
$\boldsymbol{v} \leftarrow \beta_1 \boldsymbol{v} + (1 - \beta_1)\boldsymbol{g}^2$
$\boldsymbol{\rho} \leftarrow \boldsymbol{m}/(\sqrt{\boldsymbol{v}} + \varepsilon)$
$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\boldsymbol{\rho}$

AQ(λ) delivers strong results. In MINATAR (Fig. 2) it largely surpasses STREAMQ, both by the average normalised score and the robust median (IQM). The IQM on ATARI (Fig. 1) is largely comparable between the top three algorithms, with AQ(λ) having a small edge on STREAMQ. The average normalised score of AQ(λ) however approaches three times the human baseline, in comparison to just two times the human baseline for C51and STREAMQ. Furthermore, the Mann-Whitney test indicates over 0.65 probability of improvement over STREAMQ.

In conclusion, we believe these results validate our initial hypotheses on the importance of variance adjusted updates and bounded objectives.

6.1 Related work

AQ(λ) draws heavily from the works of Javed et al. (2024); Elsayed et al. (2024); Elelimy et al. (2025), that sparked a renewed interest in the pure online RL setting. It shares many of the components with STREAMQ. Being a λ-return method implemented with eligibility traces, it traces its origin in the works of Sutton (1988) and Mahmood & Sutton (2015).

In reasoning about the role of constraining the size of the updates, we were further inspired by Javed et al. (2024) and, specifically on the role of bounded objectives in batch-RL, by Palenicek et al. (2025) and Farebrother et al. (2024).

Among the recent advances in second-order optimisation for large language model training there's also the observation that adopting the variance-adjusting mechanism of ADAM can still yield improvements (Frans et al., 2025), which also influenced our design decisions.

7 Conclusion

In this work, we revisit the streaming reinforcement learning protocol to investigate whether established batch-RL techniques could be competitive. Through extensive benchmarking across MINATAR and 55 ATARI games, we answered this in the affirmative: when appropriately adapted, classic algorithms like DQN and C51 prove to be surprisingly effective.

We then isolate some of the properties that enable this performance. We found that the severe non-stationarity of the streaming setup requires objectives with bounded derivatives to prevent large TD errors from destabilizing learning. Furthermore, we highlighted the necessity of variance-adjusted updates featuring a long gradient history. Finally, we took steps toward understanding the role of ADAM's ε parameter, demonstrating that rather than acting as a simple step-size scalar, it functions as a Signal-to-Noise Ratio (SNR) filter that facilitates smooth convergence when processing sparse or noisy features.

Building upon these insights, we introduced Adaptive $Q(\lambda)$, a small intervention on Q(λ) that synergizes variance adaptation with bounded updates to set a new high-performance baseline in the streaming regime.

A Appendix

A.1 Hyperparameters

Table 2 outlines the complete set of hyperparameter configurations used for our empirical evaluation across the ALE suite. We detail both the globally shared settings, such as the exploration schedule and discount factor, as well as the specific optimizer and algorithmic parameters tuned for each individual method.

Table 2: Hyperparameter configurations for the evaluated algorithms on the ALE suite.

Shared Hyperparameters				
Discount factor (γ)	0.99			
Exploration (ϵ) start $\rightarrow$ end	$1.0 \rightarrow 0.01$			
Exploration decay steps	2.5×10^6			
Algorithm-Specific Hyperparameters				
Parameter	**DQN**	**C51**	**StreamQ**	**AQ(λ)**
Optimizer	Adam	Adam	ObGD	Alg. 1
Learning Rate (η)	2.2×10^{-6}	4.6×10^{-5}	1.0	4.6×10^{-4}
Optimizer ϵ	0.01	0.01	–	0.1
Adam (β_1, β_2)	$(0.999, 0.999)$	$(0.999, 0.999)$	–	–
Eligibility Trace (λ)	–	–	0.8	0.8
ObGD κ	–	–	2.0	–
Number of Atoms (K)	–	200	–	–

A.2 Additional MINATAR results

Quantile Regression DQN. We briefly experimented in MINATAR with an alternate distributional algorithm. Quantile Regression DQN (QR-DQN) (Dabney et al., 2018), models the distribution using N atoms with fixed, uniform probabilities $1/N$, while learning their locations $\theta_i(s, a, \boldsymbol{w})$ instead. The practical objective in QR-DQN is then the minimisation of the quantile Huber loss. Although our initial MINATAR with Quantile Regression were very promising, C51 demonstrated superior performance upon transitioning to the ATARI-5* subset. We do however present some of these results here.

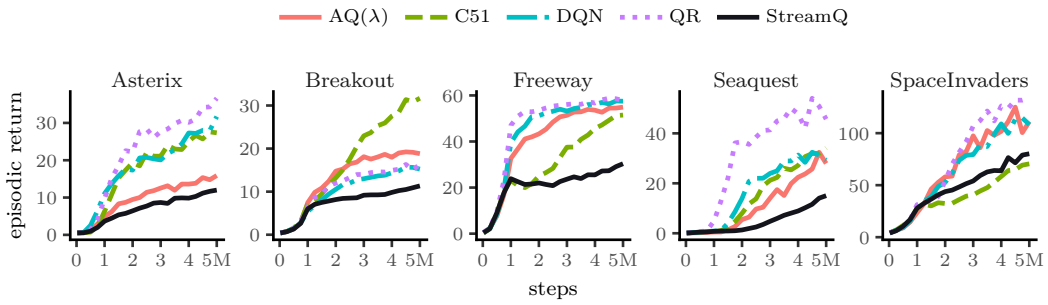

Figure 6: MINATAR results separated by game. The solid lines represent the mean evaluation return averaged over 9 independent runs

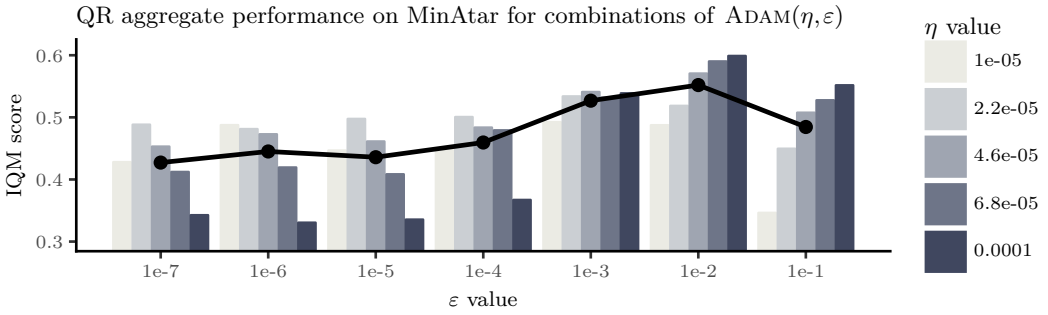

Figure 7: Performance increases with higher values of ε and there is no obvious scaling of the step size η that can compensate for it. • denotes the mean score given a value of ε.

A.3 Full ATARI results

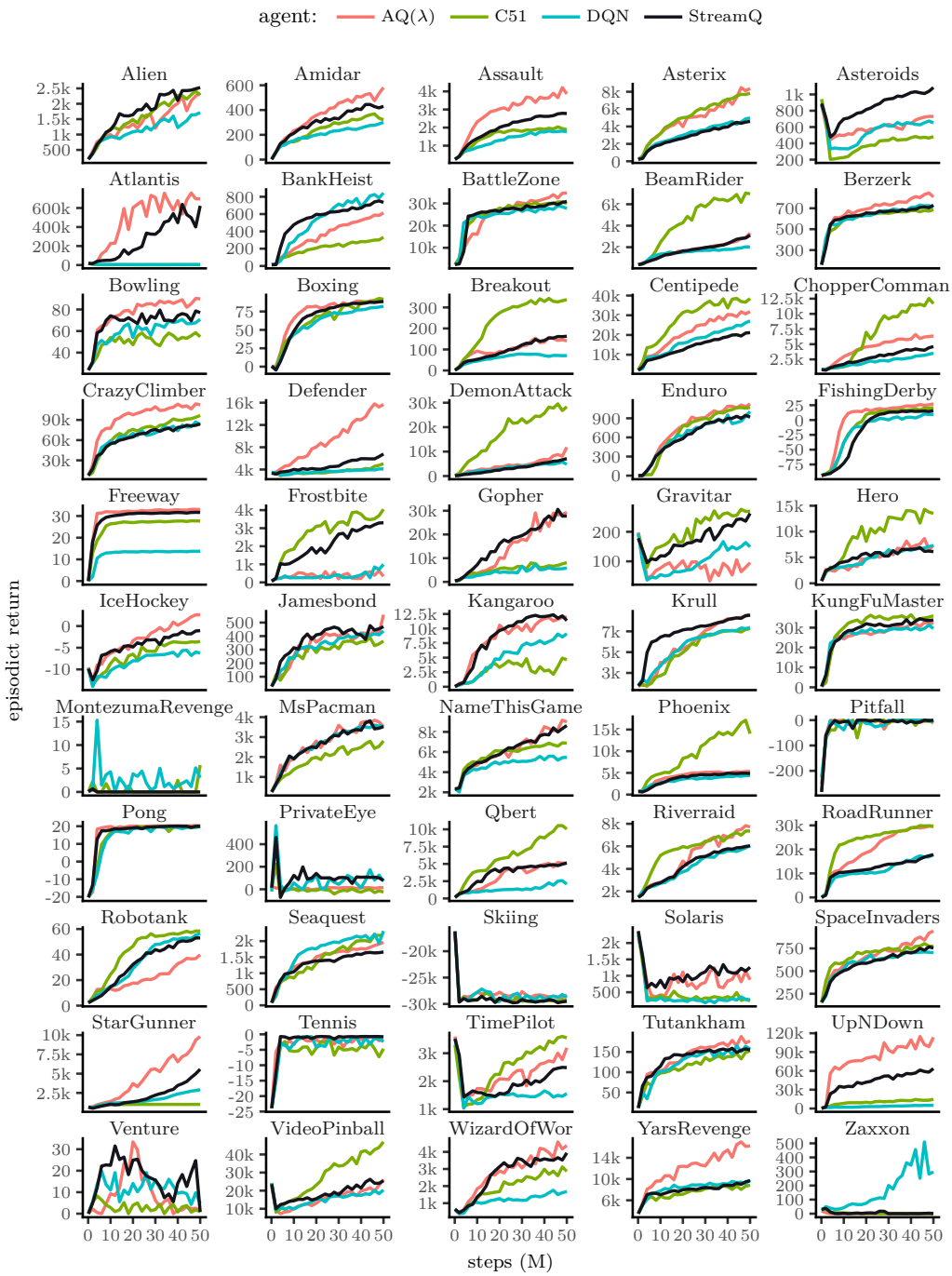

Figure 8: ATARI results separated by game. The solid lines represent the mean evaluation return averaged over 7 independent runs

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 421–438. PMLR, 2023. URL https://proceedings.mlr.press/v202/aitchison23a.html.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013. DOI: 10.1613/JAIR.3912. URL https://doi.org/10.1613/jair.3912.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *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*, pp. 449–458. PMLR, 2017. URL http://proceedings.mlr.press/v70/bellemare17a.html.

Michael Bowling and Esraa Elelimy. Rethinking the foundations for continual reinforcement learning. *CoRR*, abs/2504.08161, 2025a. DOI: 10.48550/ARXIV.2504.08161. URL https://doi.org/10.48550/arXiv.2504.08161.

Michael Bowling and Esraa Elelimy. Rethinking the foundations for continual reinforcement learning. In *Reinforcement Learning and Decision Making (RLDM) 2025*, 2025b. URL https://youtu.be/Vjx7r9mo0H0?si=BreJyfp-JuQwJEmj&t=556. Video presentation. Start time: 9:16.

Johan Samir Obando Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1373–1383. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ceron21a.html.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *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*, pp. 2892–2901. AAAI Press, 2018. DOI: 10.1609/AAAI.V32I1.11791. URL https://doi.org/10.1609/aaai.v32i1.11791.

Brett Daley and Christopher Amato. Reconciling λ-returns with experience replay. In *Conference on Neural Information Processing Systems*, pp. 1131–1140, 2019. URL http://papers.nips.cc/paper/8397-reconciling-returns-with-experience-replay.

Esraa Elelimy, Brett Daley, Andrew Patterson, Marlos C. Machado, Adam White, and Martha White. Deep reinforcement learning with gradient eligibility traces. *CoRR*, abs/2507.09087, 2025. DOI: 10.48550/ARXIV.2507.09087. URL https://doi.org/10.48550/arXiv.2507.09087.

Mohamed Elsayed, Gautham Vasan, and A. Rupam Mahmood. Streaming deep reinforcement learning finally works. *CoRR*, abs/2410.14606, 2024. DOI: 10.48550/ARXIV.2410.14606. URL https://doi.org/10.48550/arXiv.2410.14606.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. In *International Conference on Machine Learning*, pp. 13049–13071, 2024. URL https://openreview.net/forum?id=dVpFKfqF3R.

Kevin Frans, Pieter Abbeel, and Sergey Levine. What really matters in matrix-whitening optimizers? *CoRR*, abs/2510.25000, 2025. DOI: 10.48550/ARXIV.2510.25000. URL https://doi.org/10.48550/arXiv.2510.25000.

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 (eds.), *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*, pp. 3215–3222. AAAI Press, 2018. DOI: 10.1609/AAAI.V32I1.11796. URL https://doi.org/10.1609/aaai.v32i1.11796.

Khurram Javed and Richard S. Sutton. The big world hypothesis and its ramifications for artificial intelligence. URL https://api.semanticscholar.org/CorpusID:274524455.

Khurram Javed, Arsalan Sharifnassab, and Richard S. Sutton. SwiftTD: A fast and robust algorithm for temporal difference learning. *RLJ*, 2:840–863, 2024. URL https://rlj.cs.umass.edu/2024/papers/Paper111.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Alex Lewandowski, Adtiya A. Ramesh, Edan Meyer, Dale Schuurmans, and Marlos C. Machado. The world is bigger! A computationally-embedded perspective on the big world hypothesis. *CoRR*, abs/2512.23419, 2025. DOI: 10.48550/ARXIV.2512.23419. URL https://doi.org/10.48550/arXiv.2512.23419.

Ashique Rupam Mahmood and Richard S. Sutton. Off-policy learning based on weighted importance sampling with linear computational complexity. In Marina Meila and Tom Heskes (eds.), *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands*, pp. 552–561. AUAI Press, 2015. URL http://auai.org/uai2015/proceedings/papers/165.pdf.

Henry B. Mann and Douglas R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947. URL https://api.semanticscholar.org/CorpusID:14328772.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver 0001, 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. URL https://www.wikidata.org/entity/Q27907579.

Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. Xqc: Well-conditioned optimization accelerates deep reinforcement learning. *arXiv preprint arXiv:2509.25174*, 2025.

Boris Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 1964. URL https://api.semanticscholar.org/CorpusID:120243018.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL https://api.semanticscholar.org/CorpusID:205001834.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3:9–44, 1988. DOI: 10.1007/BF00115009. URL https://doi.org/10.1007/BF00115009.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - An introduction, 2nd Edition*. MIT Press, 2018. URL http://www.incompleteideas.net/book/the-book-2nd.html.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6a: Overview of mini-batch gradient descent, 2012. URL https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. Course: Neural Networks for Machine Learning (CSC321), University of Toronto.

Hado van Hasselt, Ashique Rupam Mahmood, and Richard S. Sutton. Off-policy TD(l) with a true online equivalence. In Nevin L. Zhang and Jin Tian (eds.), *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27, 2014*, pp. 330–339. AUAI Press, 2014. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2468&proceeding_id=30.

Hado van Hasselt, Sephora Madjiheurem, Matteo Hessel, David Silver 0001, André Barreto 0001, and Diana Borsa. Expected eligibility traces. In *AAAI Conference on Artificial Intelligence*, pp. 9997–10005, 2021. URL https://doi.org/10.1609/aaai.v35i11.17200.

Harm van Seijen and Richard S. Sutton. True online td(lambda). In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 692–700. JMLR.org, 2014. URL http://proceedings.mlr.press/v32/seijen14.html.

Harm van Seijen, Ashique Rupam Mahmood, Patrick M. Pilarski, Marlos C. Machado, and Richard S. Sutton. True online temporal-difference learning. *Journal of Machine Learning Research*, 17:145:1–145:40, 2016. URL https://jmlr.org/papers/v17/15-599.html.

Martha White and Adam White. A greedy approach to adapting the trace parameter for temporal difference learning. In Catholijn M. Jonker, Stacy Marsella, John Thangarajah, and Karl Tuyls (eds.), *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, pp. 557–565. ACM, 2016. URL http://dl.acm.org/citation.cfm?id=2937006.

Kenny Young and Tian Tian. MinAtar: An atari-inspired testbed for more efficient reinforcement learning experiments. *CoRR*, abs/1903.03176, 2019. URL http://arxiv.org/abs/1903.03176.