# LOCAL REINFORCEMENT LEARNING WITH ACTION-CONDITIONED ROOT MEAN SQUARED Q-FUNCTIONS

### **Anonymous authors**

Paper under double-blind review

### **ABSTRACT**

The Forward-Forward (FF) Algorithm is a recently proposed learning procedure for neural networks that employs two forward passes instead of the traditional forward and backward passes used in backpropagation. However, FF remains largely confined to supervised settings, leaving a gap at domains where learning signals can be yielded more naturally such as RL. In this work, inspired by FF's goodness function using layer activity statistics, we introduce Action-conditioned Root mean squared Q-Functions (ARQ), a novel value estimation method that applies a goodness function and action conditioning for local RL using temporal difference learning. Despite its simplicity and biological grounding, our approach achieves superior performance compared to state-of-the-art local backprop-free RL methods in the MinAtar and the DeepMind Control Suite benchmarks, while also outperforming algorithms trained with backpropagation on most tasks.

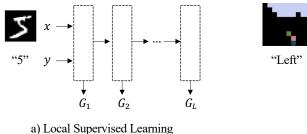
# 1 Introduction

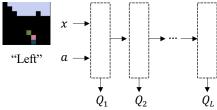
The success of deep learning has relied on *backpropagation* (Rumelhart et al., 1986), a procedure that has significant limitations in terms of biological plausibility as it requires synchronous computations and weight symmetry. Many works have provided backprop-free alternatives for training deep neural networks (Lillicrap et al., 2016; Nøkland, 2016; Nøkland & Eidnes, 2019). Notably, Hinton (2022) proposed the Forward-Forward algorithm (FF), a new approach that performs layerwise contrastive learning between positive and negative samples. This algorithm is lightweight and entirely eliminates the need for backpropagation, thereby addressing some of the biological plausibility concerns.

However, most studies on backprop-free methods are focused on the search for a biologically plausible mechanism for performing gradient updates on supervised tasks. Could a biologically plausible source of learning signals be equally meaningful? Reward-centric environments and temporal-difference (TD) methods (Sutton, 1988) serve as natural candidates for filling this gap. Biological brains have evolved through a series of reward-guided evolution, while ample evidence has shown that our brains could be implementing TD (Schultz et al., 1997a; O'Doherty et al., 2003; Watabe-Uchida et al., 2017; Amo et al., 2022). Since the goodness score in FF models the "compatibility" between the inputs and labels, this local learning paradigm can be readily adapted to a reinforcement learning (RL) setting where we model the value of an input state and an action from each layer's activities. See Figure 1 for a comparison between the supervised learning and RL setups of the forward-forward learning paradigm.

Towards integrating local methods and RL, Guan et al. (2024) recently proposed Artificial Dopamine (AD) that incorporates top-down and temporal connections in an Q-learning framework. Since the local Q-function estimation needs to be explicitly predicted, Guan et al. (2024) uses a dot-product between two sets of mappings from the inputs that produces the value estimate for each action. This design, while backprop-free, makes the architecture more flexible modeling complex inputs. However, AD still relies on the output of the dot-product to be the same dimension as the action space, limiting the performance of the method.

Inspired by FF's local goodness function from using layer statistics, we propose Action-conditioned Root mean squared Q-Function (ARQ), a simple vector-based alternative to traditional scalar-based Q-value predictors designed for local RL. ARQ is composed of two key ingredients: a goodness function that extracts value predictions from a vector of arbitrary size, and action conditioning by





b) Action-Conditioned Local Reinforcement Learning

Figure 1: Local learning paradigms inspired by the Forward-Forward (FF) algorithm (Hinton, 2022). a) The original FF is designed for supervised learning, where each layer models the "goodness" between image x and label y. Information is carried forward only through bottom-up and optionally top-down connections without backpropagation. b) We extend FF local learning for reinforcement learning—each layer takes a state observation x and an action candidate a as input, and estimates the Q value by taking the root mean squared function of the hidden vector.

inserting an action candidate at the model input. ARQ significantly improves the expressivity of a local cell by allowing more neurons at the output layer without sacrificing the backprop-free property. By applying action conditioning, we further unleash the capacity of the network to produce representation specific to each state-action pair. Moreover, ARQ can be readily implemented on AD and take full advantage of their non-linearity and attention-like mechanisms.

We evaluate our method on the MinAtar benchmark (Young & Tian, 2019) and the DeepMind Control Suite, challenging suites designed to test RL algorithms in low-dimensional settings where local methods remain viable. Our results show that our method consistently outperforms current local RL methods and surpasses conventional backprop-based value-learning methods in most games, demonstrating strong decision-making capabilities without relying on backpropagation. Through this contribution, we seek to encourage further exploration of the intersection between RL and biologically plausible learning methods.

#### 2 RELATED WORKS

Backprop-free learning methods & FF: In recent years, several backprop-free training algorithms have been proposed to address the limitations of traditional backpropagation in neural networks (Lillicrap et al., 2016; Nøkland, 2016; Nøkland & Eidnes, 2019; Belilovsky et al., 2019; Baydin et al., 2022; Ren et al., 2023; Fournier et al., 2023; Singhal et al., 2023; Innocenti et al., 2025). One notable method is the Forward-Forward Algorithm (FF) (Hinton, 2022), which offers a biologically plausible, energy-efficient alternative to backpropagation. To extend the capabilities of FF, Ororbia & Mali (2023) proposed the Predictive Forward-Forward Algorithm, showing that a top-down generative circuit can be trained jointly with FF. Tosato et al. (2023) found that models trained with FF objectives generate highly sparse representations. This pattern closely resembles the observations of neuronal ensembles in cortical sensory areas, suggesting FF may be a suitable candidate for modeling biological learning. Recently, Sun et al. (2025) proposed DeeperForward, integrating residual connections (He et al., 2016), the mean goodness function, and a channel-wise cross-entropy based objective function (Papachristodoulou et al., 2024) into FF. DeeperForward yields 87% on CIFAR-10 with a 17-layer deep architecture.

Value Estimation in Deep Neural Networks: TD methods for value estimation have been particularly useful in the recent decade as the rise of deep neural networks offers a powerful function approximator. Mnih et al. (2013) introduced DQN, where a deep neural network is applied to approximate the Q-Function. They showed that this method significantly outperformed earlier methods on the Atari 2600 games, initiating a family of methods built upon this architecture (Van Hasselt et al., 2016; Wang et al., 2016; Dabney et al., 2018b; Hessel et al., 2018; Fortunato et al., 2017; Dabney et al., 2018a; Hausknecht & Stone, 2015). In actor-critic architectures, it is also common to use a deep neural network for value and advantage estimation (Schulman et al., 2017; 2015a;b; Lillicrap et al., 2015; Mnih et al., 2016; Haarnoja et al., 2018b;a; Fujimoto et al., 2018; Gruslys et al., 2017; Abdolmaleki et al., 2018; Kostrikov et al., 2020; Yarats et al., 2021). For planning-based methods

 using either Monte Carlo tree search (MCTS) or a learned model, value estimation is also significant in driving the planning process (Schrittwieser et al., 2020; Silver et al., 2016; 2017; Hansen et al., 2023; Sacks et al., 2024; Ye et al., 2021; Hafner et al., 2023; 2020; 2019a;b) Yet, few works have investigated the capability of local learning on value estimation.

Action Conditioning of Value Estimators: An important design choice in value estimation is whether the network is conditioned on the action. Early neural value estimation methods Riedmiller (2005) incorporated action conditioning by incorporating both state and action as model inputs. With the advent of deep neural network approaches such as DQN, practices began to diverge. Purely value-based methods like DQN are typically only state-conditioned, with action-specific predictions produced at the output layer by indexing over action values. This design is computationally efficient and well-suited for discrete tasks with low-dimensional action spaces. In contrast, actor—critic methods developed for high-dimensional continuous control tasks Lillicrap et al. (2015); Haarnoja et al. (2018a) condition on both state and action at the input of their critic networks. Although this distinction is largely arbitrary in backpropagation-based architectures and can be adapted to the task, we show that action conditioning at model inputs is strictly preferable for local RL.

Local and Decentralized Reinforcement Learning: The concept of decentralized RL can be dated back to the dawn of RL. Klopf (1982) introduced the idea of the hedonistic neuron, which hypothesized that each of our neurons may be guided by their independent rewards. Instead of being a miniscule part of a large operating neural system, each neuron may be an RL agent itself. In modern RL literature, the localized formulation of RL methods can be related to the multi-agent RL (MARL) setup, where multiple independent agents can be designed to cooperate well toward maximizing their joint rewards (Tan, 1993; Foerster et al., 2017; Palmer et al., 2017; Su et al., 2022; Lauer & Riedmiller, 2000; Jiang & Lu, 2023; De Witt et al., 2020; Su & Lu, 2022; Su & Lu; Arslan & Yüksel, 2016; Jin et al., 2021). Conveniently, we can frame the problem of training RL using local objectives as a MARL problem where each agent represents different modules within a network. Recently, Seyde et al. (2022) has explored a similar approach for the continuous control problem, showing that using a separate critics network for each fixed action after action discretization works surprisingly well. Guan et al. (2024) builds upon the FF architecture, showing that a network with nonlinear local operations, decentralized objectives, and top-down connections across the temporal dimension can exceed state-of-the-art methods trained end-to-end. We extend upon this literature of decentralized methods for value estimation.

## 3 BACKGROUND

**Forward-Forward (FF):** The FF Algorithm (Hinton, 2022), as its name denotes, uses two forward passes instead of one forward pass and one backward pass used in backpropagation. The first forward pass carries the positive data, or real data, while the second pass carries the negative data, or fake data either manually defined or synthetically generated by the network. The network is then trained by maximizing the goodness of each layer in the positive pass, while minimizing the goodness of each layer in the negative pass. The definition of goodness based on a hidden vector z is as follows:

$$G_z = \sum_{z_i \in z} z_i^2. \tag{1}$$

In layman's terms, this equation represents the sum of squares of all activations over L, a measure of the magnitude and orientation of the activation vector. By training its layers greedily, FF is biologically plausible and could serve as a model for our future discovery of the inner mechanisms of the human brain.

**Value Estimation in Deep RL:** Estimation of the value function is core to RL. In layman's terms, the value function measures the expected sum of future rewards after discounting given a current state. A similar formulation can be constructed when we are interested in the goodness of a stateaction pair, which is usually termed the q-function. Formally,

$$Q_{\pi}(s,a) = \mathbb{E}_{\pi} \left[ \left. \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right| S_t = s, A_t = a \right]. \tag{2}$$

A widely used class of methods for value estimation is temporal difference (TD) learning (Sutton, 1988), which bootstraps value estimates by blending immediate rewards with future predictions,

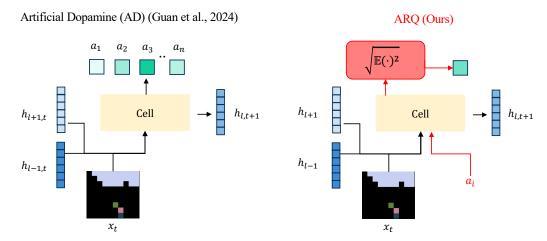


Figure 2: High-level computation diagram between Guan et al. (2024) and ARQ. Key implementations of ARQ are highlighted in red. AD cells take activations (highlighted in blue, darker color means earlier layer) and the state observation as input and produces a vector of size  $n_a$ , each indicating the value prediction of an action candidate. Our ARQ takes activations, the state observation, and the action candidate as input, and produces a hidden vector of arbitrary size, before passing it through a root mean squared function to yield a scalar prediction.

allowing for online, incremental updates. This method paved the way for the development of many subsequent approaches, particularly Q-learning. Take a q-function Q(s,a). To update the function given an experience  $(S_t,a,r,S_{t+1})$ , Q-learning makes the following iterative update

$$Q^{(i+1)}(S_t, A_t) = Q^{(i)}(S_t, A_t) + \alpha (R_t + \gamma \max_{a'} Q^{(i)}(S_{t+1}, a') - Q^{(i)}(S_t, A_t)),$$
(3)

where  $\gamma$  is a discounting factor,  $\alpha$  is a pre-determined learning rate, and a' represents any possible actions in the next step.

Recently, the rise of neural networks pushed q-learning to new heights. Mnih et al. (2013) proposed DQN, approximating q-function values using a deep neural network. Based on the Bellman equation, DQN constructs a mean squared error function as the objective, namely

$$L_{\theta} = \left(R_t + \gamma \max_{a'} Q_{\theta}(S_{t+1}, a') - Q_{\theta}(S_t, A_t)\right)^2. \tag{4}$$

Mnih et al. (2013) tested their agents on the Atari 2600 environment, and show that a convolutional neural network trained in this fashion is able to achieve near-human performance level from raw pixel inputs, a feat previously considered far-fetched.

Artificial Dopamine (AD): AD (Guan et al., 2024) trains a local RL agent using Q-learning. An AD network is consisted of multiple AD cells, each of which makes an independent estimation of  $Q(S_t, A_t)$ . To yield a scalar estimation, each AD cell adopts an attention-like mechanism to compute a weighted sum of its hidden activations using weights from a separate linear projection, effectively incorporating nonlinearity while maintaining backprop-free. Additionally, each AD cell takes inputs from the layer below, the layer above, and also the raw state observation, enabling skip connections, top-down connections, and information flow throughout the temporal dimension in an RL environment. Mathematically, an AD cell at depth l conducts the following operations,

$$X = \operatorname{concat}(s_t, h_t^{l-1}, h_{t-1}^{l+1}), \tag{5}$$

$$h_t^l = \text{ReLU}(W_h X),$$
 (6)

$$Q(s_t, a_t) = \tanh(W_{att}X)^T h_t^l, \tag{7}$$

where  $h_t^l$  represents the activation of the AD cell at time t and depth l. While this attention-like mechanism brings exciting nonlinearity to a single AD cell without the need for backpropagation, the scalar nature of  $Q(s_t, a_t)$  implies that the dimensionality of  $W_{att}$  must be limited by the size of the action space. We aim to remove this constraint.

# ARQ: ACTION-CONDITIONED ROOT MEAN SQUARED Q-FUNCTION

In the context of FF, the goodness function measures the likelihood of the observation to come from the postive distribution. In the context of RL, the concept of value measures the expected sum of future rewards for the trajectories starting from a given state. We observe a connection—both denote a measure of the current input's desirability to an agent. Could the association between goodness and value be exploited to unleash the capacity of local RL networks? In this section, we introduce a novel vector-based training mechanism for local value estimation that can be used out-of-the-box. We term it the Action-conditioned Root mean squared Q-function (ARQ).

#### 4.1 ARQ

216

217 218

219

220

222

224 225

226

227

228

229

230

231

232

233

234

235

236

237 238 239

240 241

242 243

244 245

246

247 248

249

250

251

253

254

255

256

257

258

259

260

261 262

263

264

265 266

267

268

269

Take a state s and an action a. Based on the Bellman equation, we are interested in finding

$$Q_*(s,a) = \mathbb{E}_{\pi} \left[ R_t + \gamma \max_{a'} Q_*(S_{t+1}, a') \middle| S_t = s, A_t = a \right].$$
 (8)

Inspired by the association between the concept of goodness from FF and the concept of value in RL, we directly approximate Q(s, a) using the goodness function. Given a hidden vector z, which can be either an intermediate action or an output embedding from a neural network. Instead of taking the sum of each vector unit squared, we make a small modification and take the root mean squared (RMS) function of the vector after mean subtraction to prevent its goodness values from exploding as we scale up the number of units. This is equivalent to the standard deviation of the hidden vector. In mathematical terms, we compute the estimated value of applying action a on state s using

$$\mu_y = \mathbb{E}_{y_i \in y} y_i, \ Q_{\theta}(s, a) = \sqrt{\mathbb{E}_{y_i \in y} (y_i - \mu_y)^2},$$
 (9)

where  $\theta$  denotes the parameters of the network and z denotes a hidden vector produced by the

To train this network, we update our weights using the same mean squared objective function as

previous Q-learning methods (Mnih et al., 2013). Namely,
$$L_{\theta} = \left(R_t + \gamma \max_{a'} Q_{\theta}(S_{t+1}, a') - Q_{\theta}(S_t, A_t)\right)^2. \tag{10}$$

Note that it is possible to sample positive and negative data in order to train in the same contrastive fashion as the original FF algorithm, particularly when our method is used with a training mechanism that maintains a replay buffer. We leave this for future investigations to keep our method versatile.

ARO can be implemented out-of-the-box in place of the standard O-learning formulation. Given any intermediate vector produced by an arbitrary neural network architecture, ARQ can extract scalar statistics that serve as a prediction for the estimated value without any parameters. This property allows architectures designed for local RL to enjoy greater flexibility.

**Action Conditioning:** Due to the nature of goodness functions to produce scalar values, it is natural to implement ARO with action conditioning at the model input. Concretely, to estimate  $Q_{\theta}(s,a)$ , the neural network  $\theta$  takes both the state vector s and the action vector a as inputs and outputs a single scalar prediction. This contrasts with implementations such as Mnih et al. (2013) and Guan et al. (2024), where the model receives only the state vector s and produces an output of dimension  $n_a$ , with each entry corresponding to the value of a discrete action. We demonstrate in Section 5 that this minor design decision is critical to the performance of local RL methods. For tasks with discrete action spaces, we use a one-hot vector to represent an action candidate. For tasks with continuous action spaces, we apply bang-bang discretization on the action space following Seyde et al. (2021) and condition the network on the binary action vector.

### 4.2 IMPLEMENTATION

To evaluate our method against state-of-the-art local RL architectures, we implement AR on top of Guan et al. (2024).

Our implementation is consisted of multiple cells stacked together, each of which takes inputs from the layer below, the layer above, the input observation, and an action candidate  $a_t$  to make an estimation of  $Q(s_t, a_t)$ . Each cell adopts a similar attention-like mechanism as Guan et al. (2024). After the attention mechanism, we apply the goodness function on the intermediate vector after the attention computation. Specifically, a cell at depth l conducts the following operations,

273

274

275 276

277 278

279

$$X = \operatorname{concat}(s_t, h_t^{l-1}, h_{t-1}^{l+1}, a_t), \tag{11}$$

$$h_t^l = \text{ReLU}(W_h X), \tag{12}$$

$$y_t^l = \tanh(W_{att}X)^T h_t^l, \tag{13}$$

$$y_t^l = \tanh(W_{att}X)^T h_t^l,$$

$$\mu_y = \underset{y_i \in y}{\mathbb{E}} y_i, \ Q(s_t, a_t) = \sqrt{\underset{y_i \in y_t^l}{\mathbb{E}} (y_i - \mu_y)^2},$$

$$(13)$$

Gradients are passed only within each cell to ensure the architecture is backprop-free. Pseudocode comparising AD and ARQ can be found in Figure 3. Most training choices are inherited from Guan et al. (2024).

# 280 281 282 283

284

285

286

287

288

289 290

291

292 293

295

296 297 298

299

300

301

302

303

304

305 306

307 308

309

310

311

312

313

314

315

316

317

318

319

320 321

322

323

# Algorithm 1 AD (Guan et al., 2024)

1:  $X \leftarrow [s_t, h_t^{l-1}, h_{t-1}^{l+1}]$ 2:  $h_t^l \leftarrow \text{LayerNorm}(\text{ReLU}(W_h X))$ 3:  $Z_1 \leftarrow W_{att_1} X$ 4:  $Z_2 \leftarrow W_{att_2} X \Rightarrow Z_2$  has dimension  $n_a$ 5:  $W \leftarrow Z_2^{\top} Z_1$ 6:  $W \leftarrow \text{LayerNorm}(\text{tanh}(W))$ 

7:  $Q \leftarrow Wh_t^l$ 

 $\triangleright Q$  has dimension  $n_a$ 

# Algorithm 2 ARQ (Ours)

```
1: X \leftarrow [s_t, h_t^{l-1}, h_{t-1}^{l+1}]
```

2:  $h_t^l \leftarrow \text{LayerNorm}(\text{ReLU}(W_h X))$ 

3: Repeat X along batch dim  $n_a$  times

4:  $X \leftarrow [X, a_t]$   $\Rightarrow$  Action 5:  $Z_1 \leftarrow W_{att_1}X$ 6:  $Z_2 \leftarrow W_{att_2}X$   $\Rightarrow$   $Z_2$  has 7:  $W \leftarrow Z_2^{\top}Z_1$ 8:  $W \leftarrow \text{LayerNorm}(\tanh(W))$  $\triangleright Z_2$  has dimension d

9:  $y \leftarrow Wh_t^l$  $\triangleright y$  has dimension d

10:  $Q \leftarrow \text{RMSQ}(y)$ 

Figure 3: Comparison of AD and ARQ implemented on top of AD. For ARQ, action conditioning is applied as part of the input (Line 4,5, Algorithm 2). Note that ARQ allows  $Z_2$  and y to have dimension d, which can be arbitrary, while AD fixes it at  $n_a$ , one for each action output.

Why ARQ benefits local Q-learning? As demonstrated in Figure 3, ARQ allows the hidden output to have arbitrary dimensions. We conjecture that ARQ's flexibility to account for arbitrary hidden dimensions allows it to take full advantage of non-linearity within each AD cell. Furthermore, ARQ applies action conditioning at the model input, rather than using vector indices at the output layer as conditioning. We conjecture that this allows the entire module to produce representation specific to each state-action pair, rather than action-agnostic information based on only the observation. Combining these two properties, ARQ exploits the full capacity of the attention-like mechanism that modern local RL methods operates on, allowing greater expressivity of each state-action pair.

#### EXPERIMENTS

Benchmarks: We test ARQ on the MinAtar benchmark (Young & Tian, 2019) and the DeepMind Control (DMC) Suite (Tassa et al., 2018) following Guan et al. (2024). MinAtar is a miniaturized version of the Atari 2600 games, using 10x10 grids instead of 210x160 frames as inputs. The DMC Suite is a benchmark for continuous control tasks featuring low-level observations and actions, designed to evaluate the performance of RL methods in physics-based environments. Both benchmarks involve low-dimensional inputs and outputs instead of high-dimensional raw sensory inputs, making them appropriate testbeds for evaluating the decision-making ability of local methods in simple environments.

**Baselines:** For comparisons with cutting-edge local RL methods, we compare our results with AD for both benchmarks. To evaluate our methods against backprop-based algorithms, we also compare our method against DQN for MinAtar. DQN is a widely used baseline that trains deep neural networks to directly compute scalar Q-values through backpropagation. We follow the DQN implementation used by Guan et al. (2024).

Implementation Details: Following Guan et al. (2024), we use a three-layer fully-connected network, with hidden dimensions being 400, 200, and 200 for MinAtar. We use a three-layer network with hidden dimensions 128, 96, and 96 for DMC tasks. We use a replay buffer and a target network for stability. We incorporate skip connections from the input and top-down connections from the

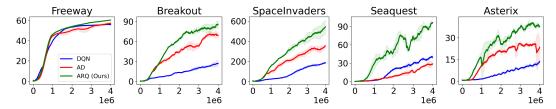


Figure 4: Training performance on the MinAtar games, compared between DQN (blue), AD (orange), and ARQ(green). The x-axis denotes the number of training steps (in millions), and the y-axis indicates average episodic returns. Shaded regions represent standard deviations across 3 seeds. We find that ARQ consistently outperforms AD in all MinAtar games, while outperforming DQN in most games.

layer above. For all experiments, we use an epsilon-greedy policy with linear decay from 1 to 0.01 using an exploration fraction of 0.1. We run our experiments for 4 million steps, where the model starts learning from step 50,000. Learning rate is set fixed at 1e-4. A batch size of 512 is used. For MinAtar, we condition on action candidates by passing them as one-hot vectors into the network. For DMC tasks, we discretize our action space and condition action vectors as model inputs.

Main Results: We present our results in Figure 4 and Figure ??. We run each experiment with three different random seeds and plot their average returns over 100-episode windows along with their standard deviations in shadows. We also calculated the average returns of the last 100 episodes of each training run to obtain a quantitative measure of the final performance of our method, which can be found in Table 2. As demonstrated, ARQ consistently outperforms AD in all MinAtar games. Surprisingly, ARQ also outperforms DQN in all games. In DMC Suite tasks, ARQ achieves superior returns compared to AD, while also exceeding back-prop based methods in most games. We present our numbers in Table 2.

Game Analysis: We note that ARQ outperforms DQN by a wide margin on Breakout and SpaceInvaders. Both of these games operate on similar mechanisms: players aim to remove targets by controlling projectile interactions of objects. To yield higher scores, players need to perform combos of actions to yield higher scores, for instance moving to a sweet spot then waiting for the target to arrive before firing a bullet. We argue that top-down connections in AD provide temporal coherence, which allows our agents to perform sequences of actions smoothly. Additionally, we note that while AD fails to match DQN on Seaquest, ARQ sur-

Table 1: Our method Using Different Nonlinearities Compared in MinAtar Breakout. 'MS' is short for the mean squared function and 'Var' is short for variance. Default ARQ uses the root mean squared (RMS) function.

Nonlinearity	Breakout	SpaceInvaders	
Ours-ARQ	88.83±8.34	555.29±56.97	
Ours-Mean	$79.84 \pm 13.23$	$500.13\pm47.78$	
Ours-MS	$82.10\pm3.28$	$434.88 \pm 14.37$	
Ours-Var	$81.34 \pm 0.39$	$416.46 \pm 66.60$	
AD	$67.40 \pm 4.01$	$369.96 \pm 23.46$	

passes DQN. Seaquest is a game involving firing bullets to remove enemies, with an additional rule that players need to manage an oxygen tank by surfacing above water to refill their tank. This represents that the policy distribution can be bi-modal such that attacking enemies and refilling tanks are both locally optimal policies. We hypothesize that by applying action conditioning, ARQ can capture these policy structures more effectively than AD, which is only state-conditioned.

Effect of Action Conditioning at Input: How does action conditioning affect the performance of local RL methods? To investigate, we conduct ablation experiments on two games from MinAtar, Breakout and SpaceInvaders, using both AD and ARQ. The results can be found in Figure 5. We find a significant improvement when actions are conditioned at the input instead of at the output. Interestingly, this design choice provides only a slight improvement for AD, while yielding a significant increase in performance for ARQ. We conjecture this is due to the increase in the capacity of each cell to capture the granularity within each specific state-action pair, while AD saturates with action-agnostic information.

**Effect of Goodness Nonlinearities:** One question that naturally arises is the choice of the goodness function. Does the RMS function perform superiorly compared to other functions? We ablate on this

Table 2: Previous methods and ARQ compared in MinAtar and DeepMind Control (DMC) tasks.

MinAtar	Freeway	Breakout	SpaceInvaders	Seaquest	Asterix	
w/ back-prop DQN	$55.86 \pm 0.32$	$27.09 \pm 5.74$	$188.03 \pm 15.81$	$37.96 \pm 9.28$	$13.60 \pm 1.08$	
w/o back-pro	р					
AD	$57.64 \pm 1.49$	$67.40 \pm 4.01$	$369.96 \pm 23.46$	$30.32 \pm 4.79$	$24.05 \pm 5.44$	
ARQ (Ours)	$\textbf{60.77} \pm \textbf{0.32}$	$\textbf{88.93} \pm \textbf{5.90}$	$555.29 \pm 56.97$	$100.81 \pm 6.23$	$37.89 \pm 1.68$	
DMC	Walker Walk	Walker Run	Hopper Hop	Cheetah Run	Reacher Hard	
w/ back-prop						
TD-MPC2	$958.80 \pm 1.29$	$834.07 \pm 10.13$	$348.55 \pm 26.65$	$808.46 \pm 92.10$	$934.84 \pm 6.86$	
SAC	$980.43 \pm 1.63$	$895.02 \pm 46.35$	$319.46 \pm 31.21$	$917.40 \pm 2.45$	$980.01 \pm 1.19$	
w/o back-prop						
AD	$975.25 \pm 0.87$	$761.11 \pm 0.17$	$485.75 \pm 57.89$	$831.57 \pm 15.90$	$954.34 \pm 7.30$	
ARQ (Ours)	$\textbf{976.26} \pm \textbf{2.23}$	$\textbf{771.63} \pm \textbf{1.15}$	$515.45 \pm 21.18$	$881.30 \pm 17.33$	$972.45 \pm 6.04$	

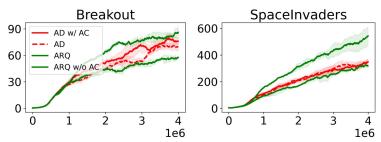


Figure 5: Ablation on action conditioning for AD and ARQ. Action conditioning substantially improves performance. Note that this improvement is particularly significant for ARQ, with average returns of  $\sim$ 85 vs.  $\sim$ 55, a 50% improvement. This indicates that the combination of the RMS function and action conditioning makes ARQ effective.

design choice and conduct experiments on two games from MinAtar, Breakout and SpaceInvaders. As shown in Table 1, we find that using the RMS goodness functions yields superior performance, followed by the mean and the mean squared function. We conjecture that a smaller magnitude in the goodness can enhance stability of training. However, we note that all functions perform superiorly compared with AD, which demonstrates the versatility of our method. We leave it for future work to study the intricate effect each function has on training.

Is it because ARQ has more hidden units? Compared with AD, ARQ employs a larger number of parameters since ARQ allows an arbitrary dimension for its hidden vectors. Could ARQ, however, simply achieve the same improvement with mere scaling? We conduct experiments on AD and ARQ with the same number of total parameters to answer this question. Across different ratios of total parameters (compared with the original AD as a baseline),

Table 3: AD vs. ARQ Across Multiple Scales for MinAtar Breakout.

Scale Ratio	AD	ARQ	
0.5×	$66.34 \pm 5.15$	$\textbf{68.12} \pm \textbf{5.65}$	
$1 \times$	$64.20 \pm 1.90$	$\textbf{86.26} \pm \textbf{0.66}$	
1.5×	$56.63 \pm 5.39$	$\textbf{70.40} \pm \textbf{3.98}$	
2×	$59.79 \pm 4.77$	$\textbf{83.26} \pm \textbf{2.32}$	

we run both AD and ARQ on the MinAtar Breakout game with two different random seeds. As shown in Table 3, ARQ consistently outperforms AD across all scales. This verifies the effectiveness of our method beyond scale.

**Neurons Are Sensitive to Different Scenarios:** How does our method learn through a goodness function? We investigate its inner mechanism by visualizing the activations at each layer under different states. As illustrated in Figure 6, we find that the hidden activations tend to show larger magnitudes under "correct" state-action pairs. For instance, in scenarios where the agent should move right to accurately catch the incoming ball, neurons in the hidden activations show the largest magnitude when the action input matches correspondingly. Interestingly, we observe that different neurons are, in general, activated to different degrees for various action candidates. This implies

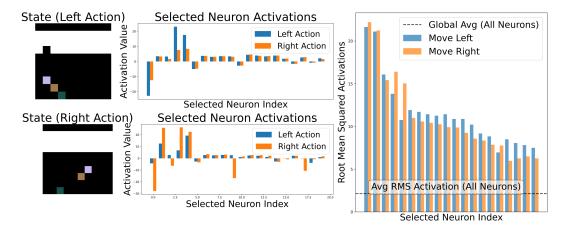


Figure 6: Visualization of Neurons in Layer 0 under Different Scenarios in Breakout Game. 20 neurons w/ highest average activities are visualized. Top Left: When the ball is approaching towards the left side of the brick, neurons show larger magnitude when the action candidate is to "move left", prompting the agent to move towards the ball. Bottom Left: When the ball approaches the right side of the brick, neurons show larger magnitude when the action candidate is to "move right". Right: The average root mean squared (RMS) activations of 20 top neurons across 100 states is collected. Note that top neurons exhibit significantly larger RMS activations than the average RMS activation, implying that these neurons are "dominant" neurons. While most neurons demonstrate similar magnitude between both actions, some neurons appear to be more specialized. our objective function could be encouraging specialized neurons, each of which is responsible for recognizing certain categories of positive signals.

#### 6 DISCUSSION

Previous studies on biologically plausible learning have largely focused on the search for a biologically plausible mechanism for performing gradient updates. As we approach the era of experience, we argue that a biologically plausible paradigm for learning can be equally meaningful to guide us towards the mystery behind how biological brains learn. Reward-centric environments provide a biologically grounded paradigm, aligning with the evolutionary role of survival signals and behavioral shaping through positive or negative reinforcement. The structure of such environments mirrors the ecological settings in which animals adaptively refine behavior through trial-and-error interactions, suggesting that learning systems shaped by rewards may naturally emerge in both artificial and biological agents. Additionally, temporal difference methods are an ideal candidate. It has been shown that biological neurons learn through temporal difference, with hormones conveying the prediction error as a source of learning signal to independent neurons (Schultz et al., 1997b; Bayer & Glimcher, 2005). On the other hand, reinforcement learning has largely focused on learning through interactions with an agent's surrounding environment, and maximizing its rewards through centralized value estimation. Yet, increasing neuroscientific evidence has shown that neurons make decentralized, independent value estimations (Tsutsui et al., 2016; Knutson et al., 2005). Few work in the RL community has investigated whether this biological phenomenon has practical implications. ARO is an effort towards this direction as each cell in our network can be seen as a decentralized value estimator.

# 7 Conclusion

This work proposes Action-conditioned Root mean squared Q-function (ARQ), a vector-based alternative to scalar Q-learning for backprop-free local learning. ARQ enables arbitrary hidden dimensions and improved expressivity by extracting value predictions from hidden activations and applying action conditioning at the model input. We show that, when applied on RL environments, ARQ performs superiorly compared to current local methods, while also outperforming backprop-based methods on most games. Whereas current biologically plausible algorithms are mostly based on the supervised setting, our study suggests that exploring local learning within reinforcement learning may provide a promising avenue for future research in both domains.

# LLM USAGE STATEMENT

LLMs were used in this work to refine certain textual phrasing and to generate minor code elements related to visualization and checkpointing.

# REPRODUCIBILITY STATEMENT

We provide training code for all experiments as supplementary material. All hyperparameters, random seeds, and implementation details are described in Section 5. Our experiments were run on single NVIDIA A4000 GPUs or NVIDIA L40S GPUs with training times ranging from 8-72 hours depending on the task. We used only publicly available benchmarks (MinAtar and DeepMind Control Suite). Together, these resources should enable full reproduction of our results.

#### REFERENCES

- Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.
- Ryunosuke Amo, Sara Matias, Akihiro Yamanaka, Kenji F Tanaka, Naoshige Uchida, and Mitsuko Watabe-Uchida. A gradual temporal shift of dopamine responses mirrors the progression of temporal difference error in machine learning. *Nature neuroscience*, 25(8):1082–1092, 2022.
- Gürdal Arslan and Serdar Yüksel. Decentralized q-learning for stochastic teams and games. *IEEE Transactions on Automatic Control*, 62(4):1545–1558, 2016.
- Atilim Günes Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip H. S. Torr. Gradients without backpropagation. *CoRR*, abs/2202.08587, 2022.
- Hannah M Bayer and Paul W Glimcher. Midbrain dopamine neurons encode a quantitative reward prediction error signal. *Neuron*, 47(1):129–141, 2005.
- Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, 2019.
- Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pp. 1096–1105. PMLR, 2018a.
- Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018b.
- Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.
- Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, pp. 1146–1155. PMLR, 2017.
- Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. arXiv preprint arXiv:1706.10295, 2017.
- Louis Fournier, Stéphane Rivaud, Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Can forward gradient match backpropagation? In *International Conference on Machine Learning*, pp. 10249–10264. PMLR, 2023.
  - Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actorcritic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

- Audrunas Gruslys, Will Dabney, Mohammad Gheshlaghi Azar, Bilal Piot, Marc Bellemare, and Remi Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. arXiv preprint arXiv:1704.04651, 2017.
  - Jonas Guan, Shon Verch, Claas Voelcker, Ethan Jackson, Nicolas Papernot, and William Cunningham. Temporal-difference learning using distributed error signals. *Advances in Neural Information Processing Systems*, 37:108710–108734, 2024.
  - Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018a.
  - Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.
  - Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.
  - Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.
  - Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.
  - Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2023. *URL https://arxiv. org/abs/2301.04104*, 2023.
  - Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.
  - Matthew J Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposia*, volume 45, pp. 141, 2015.
  - Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.
  - Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
  - Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv* preprint *arXiv*:2212.13345, 2(3):5, 2022.
  - Francesco Innocenti, El Mehdi Achour, and Christopher L Buckley.  $\mu$ pc: Scaling predictive coding to 100+ layer networks. *arXiv preprint arXiv:2505.13124*, 2025.
  - Jiechuan Jiang and Zongqing Lu. Best possible q-learning. arXiv preprint arXiv:2302.01188, 2023.
  - Chi Jin, Qinghua Liu, Yuanhao Wang, and Tiancheng Yu. V-learning–a simple, efficient, decentralized algorithm for multiagent rl. *arXiv preprint arXiv:2110.14555*, 2021.
  - A. Harry Klopf. *The Hedonistic Neuron: A Theory of Memory, Learning and Intelligence*. Washington: Hemisphere Pub. Corp., 1982.
  - Brian Knutson, Jonathan Taylor, Matthew Kaufman, Richard Peterson, and Gary Glover. Distributed neural representation of expected value. *Journal of Neuroscience*, 25(19):4806–4812, 2005.
  - Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

- Martin Lauer and Martin A Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the seventeenth international conference on machine learning*, pp. 535–542, 2000.
  - Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv* preprint arXiv:1509.02971, 2015.
  - Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1): 13276, 2016.
  - Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
  - Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.
  - Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.
  - Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *International conference on machine learning*, pp. 4839–4850. PMLR, 2019.
  - John P O'Doherty, Peter Dayan, Karl Friston, Hugo Critchley, and Raymond J Dolan. Temporal difference models and reward-related learning in the human brain. *Neuron*, 38(2):329–337, 2003.
  - Alexander Ororbia and Ankur Mali. The predictive forward-forward algorithm. *arXiv preprint arXiv:2301.01452*, 2023.
  - Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient multi-agent deep reinforcement learning. *arXiv preprint arXiv:1707.04402*, 2017.
  - Andreas Papachristodoulou, Christos Kyrkou, Stelios Timotheou, and Theocharis Theocharides. Convolutional channel-wise competitive learning for the forward-forward algorithm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14536–14544, 2024.
  - Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. In *ICLR*, 2023.
  - Martin Riedmiller. Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pp. 317–328. Springer, 2005.
  - David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by backpropagating errors. *nature*, 323(6088):533–536, 1986.
  - Jacob Sacks, Rwik Rana, Kevin Huang, Alex Spitzer, Guanya Shi, and Byron Boots. Deep model predictive optimization. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 16945–16953. IEEE, 2024.
  - Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
  - John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.
  - John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv* preprint *arXiv*:1506.02438, 2015b.

- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
  - Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997a.
  - Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997b.
  - Tim Seyde, Igor Gilitschenski, Wilko Schwarting, Bartolomeo Stellato, Martin Riedmiller, Markus Wulfmeier, and Daniela Rus. Is bang-bang control all you need? solving continuous control with bernoulli policies. *Advances in Neural Information Processing Systems*, 34:27209–27221, 2021.
  - Tim Seyde, Peter Werner, Wilko Schwarting, Igor Gilitschenski, Martin Riedmiller, Daniela Rus, and Markus Wulfmeier. Solving continuous control via q-learning. *arXiv preprint arXiv:2210.12566*, 2022.
  - David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
  - David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.
  - Utkarsh Singhal, Brian Cheung, Kartik Chandra, Jonathan Ragan-Kelley, Joshua B Tenenbaum, Tomaso A Poggio, and Stella X Yu. How to guess a gradient. *arXiv preprint arXiv:2312.04709*, 2023.
  - Kefan Su and Zongqing Lu. f-divergence policy optimization in fully decentralized cooperative marl.
  - Kefan Su and Zongqing Lu. Decentralized policy optimization. *arXiv preprint arXiv:2211.03032*, 2022.
  - Kefan Su, Siyuan Zhou, Jiechuan Jiang, Chuang Gan, Xiangjun Wang, and Zongqing Lu. Ma2ql: A minimalist approach to fully decentralized multi-agent reinforcement learning. *arXiv* preprint *arXiv*:2209.08244, 2022.
  - Liang Sun, Yang Zhang, Weizhao He, Jiajun Wen, Linlin Shen, and Weicheng Xie. Deeperforward: Enhanced forward-forward training for deeper and better performance. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kOYnXVQCtA.
  - Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.
  - Ming Tan. Multi-agent reinforcement learning: independent versus cooperative agents. In *ICML'93:* Proceedings of the Tenth International Conference on International Conference on Machine Learning, pp. 330–337, 1993.
  - Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv* preprint arXiv:1801.00690, 2018.
  - Niccolò Tosato, Lorenzo Basile, Emanuele Ballarin, Giuseppe de Alteriis, Alberto Cazzaniga, and Alessio Ansuini. Emergent representations in networks trained with the forward-forward algorithm, 2023.
  - Ken-Ichiro Tsutsui, Fabian Grabenhorst, Shunsuke Kobayashi, and Wolfram Schultz. A dynamic code for economic object valuation in prefrontal cortex neurons. *Nature communications*, 7(1): 12554, 2016.

Under review as a conference paper at ICLR 2026 Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016. Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In International conference on machine learning, pp. 1995–2003. PMLR, 2016. Mitsuko Watabe-Uchida, Neir Eshel, and Naoshige Uchida. Neural circuitry of reward prediction error. Annu. Rev. Neurosci., 40:373-394, 2017. Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous con-trol: Improved data-augmented reinforcement learning. arXiv preprint arXiv:2107.09645, 2021. Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. Advances in neural information processing systems, 34:25476–25488, 2021. Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. arXiv preprint arXiv:1903.03176, 2019.