informatics

MDPI

Article

Controllable Orthogonalization for Stabilizing Neural Network Training in Deep Reinforcement Learning

Jia Yu [1], Dun Su [2,3,4], Hua Cui [1], Hai Wang [1], Lei Huang [2,3] and Tengwei Song [2,*]

[1] TravelSky Technology Limited, Beijing 101318, China; yujia@travelsky.com.cn (J.Y.); cuihua@travelsky.com.cn (H.C.); wanghai@travelsky.com.cn (H.W.)
[2] State Key Laboratory of Complex & Critical Software Environment, School of Computer Science and Engineering, Beihang University, Beijing 100191, China; suduncs@gmail.com (D.S.); huangleiai@buaa.edu.cn (L.H.)
[3] School of Artificial Intelligence, Beihang University, Beijing 100191, China
[4] BAIYANG AI, Beijing 100080, China
* Correspondence: songtengwei@gmail.com

Abstract

Deep reinforcement learning (DRL) relies on neural networks trained under temporally correlated experience, evolving replay distributions, and bootstrapped targets, conditions that can destabilize neural function approximation. This study investigates controllable orthogonalization as a network-level mechanism for improving training stability. Orthogonalization by Newton's Iteration (ONI) is applied selectively to fully connected layers, with layer placement and Newton iteration count T controlling the location and strength of orthogonalization. We first analyze ONI placement and intensity in Deep Q-Network (DQN), and then evaluate the selected configurations on MinAtar and Atari. The same design principle is applied to the critic of Data-Regularized Q version 2 (DrQ-v2) on DeepMind Control Suite tasks. DQN-ONI improves aggregate normalized scores and learning trajectories across the evaluated discrete-control games. On the twelve-task continuous-control benchmark, Critic-ONI exhibits no learning failures. Learning-rate sweeps and penultimate-layer activation statistics further show a wider usable optimization range and more stable intermediate representations. These results show that selective, intensity-controlled orthogonalization can improve the training stability of DRL neural networks.

Keywords: deep reinforcement learning; neural network architecture; normalization; orthogonalization; training stability

Check for updates

Academic Editor: Olga Kurasova

Received: 31 August 2026
Revised: 28 September 2026
Accepted: 30 September 2026
Published: 2 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Deep reinforcement learning (DRL) combines reinforcement learning with neural function approximation and has become a general approach to sequential decision making [1,2]. The Deep Q-Network (DQN) demonstrated direct action-value learning from Atari images [2,3], while deterministic and entropy-regularized actor–critic methods extended deep learning to high-dimensional continuous control [4–6]. In both settings, training reliability remains a defining practical concern: learning speed and final behavior can change sharply with the task, random initialization, and optimization configuration [7].

This instability is rooted partly in the data and target construction used by DRL. Consecutive observations are correlated, replay distributions evolve with the behavior policy, and temporal-difference targets depend on learned value estimates. A change in the function approximator can therefore alter later targets and, in actor–critic methods, the policy

update itself. Experience replay, target networks, multi-step returns, entropy regularization, and image augmentation address important parts of this feedback loop [2,5,8–10]. They leave open a complementary architectural question: how should the geometry of the neural transformations be controlled during DRL training?

Normalization and weight constraints provide tools for regulating feature and gradient propagation. Batch normalization, layer normalization, weight normalization, spectral normalization, and gradient penalties have each been used to improve neural optimization [11–15]. Their behavior in DRL depends strongly on the interaction with non-stationary data and bootstrapped learning. In particular, batch-dependent statistics can couple normalization to the changing replay distribution, while a fixed spectral constraint controls only the dominant singular direction. Recent studies increasingly treat normalization and parameterization as direct controls on DRL optimization: normalization has been connected to parameter-norm growth and effective learning rates [16]; CrossQ uses Batch Renormalization in off-policy actor–critic learning [17]; weight normalization has been used to stabilize high update-to-data-ratio training [18]; and recent analysis of Batch Normalization modes relates unstable actor–critic configurations to distribution mismatch and Q-value estimation error [19]. Hyperspherical normalization provides a related geometric approach that constrains weight and feature norms for scalable DRL optimization [20]. Complementary stabilization methods target encoder gradients and dormant representations, as illustrated by A-LIX and ReDo [21,22].

Orthogonal transformations offer a distinct form of control because an orthogonal matrix has unit singular values and preserves the Euclidean norm of a linear transformation. Orthogonalization by Newton's Iteration (ONI) constructs the forward weight by iteratively approximating inverse-square-root normalization [23,24]. Its iteration count T determines how closely the nonzero singular values approach one, making the strength of the constraint explicit.

These considerations motivate controllable rather than indiscriminate orthogonalization. Applying an orthogonal constraint too broadly can restrict representational flexibility in comparatively shallow DRL value networks, particularly when task-specific Q-functions require nonuniform transformations. We therefore control orthogonalization along two dimensions: the selected layer determines where the constraint is applied, and T determines its intensity. The resulting design balances geometric control with task-specific representation learning.

We evaluate this principle in two complementary architectures. For DQN, ONI is inserted into selected fully connected layers of the value head and studied on MinAtar and Atari. For Data-Regularized Q version 2 (DrQ-v2), ONI is inserted into the critic because the critic participates in both the bootstrapped target and the actor gradient.

The main contributions are as follows:

- We introduce selective ONI as a controllable network-level intervention for DRL, using layer placement and Newton iteration count as separate architectural controls.
- We instantiate the method in DQN value heads and DrQ-v2 twin critics and evaluate it across MinAtar, Atari, and the DeepMind Control Suite (DMC).
- We show that controllable ONI improves discrete-control learning trajectories, accelerates difficult continuous-control tasks, and reduces observed training failures across challenging DMC tasks.
- We further characterize training stability through random-seed sensitivity, learning-rate robustness, and the ℓ_1 norms of per-dimension activation mean and variance.

2. Related Work

2.1. Algorithmic Stabilization in Deep Reinforcement Learning

Deep reinforcement learning is sensitive to random seeds, hyperparameters, evaluation protocols, and implementation choices [7]. Value-based algorithms address several sources of instability through the learning rule and data pipeline. DQN combines experience replay with a target network [2]; Double DQN reduces overestimation [25]; dueling networks separate state value and action advantage [26]; prioritized replay changes transition sampling [27]; and Rainbow combines these ideas with multi-step and distributional targets [8]. These mechanisms improve target estimation and data use while retaining a neural value function trained from evolving targets.

In continuous control, deterministic policy gradients and Deep Deterministic Policy Gradient (DDPG) optimize an actor through a learned critic [4,6]. Soft Actor-Critic (SAC) adds entropy regularization and stochastic policies [5]. For visual observations, Contrastive Unsupervised Representations for Reinforcement Learning (CURL), Data-Regularized Q (DrQ), and DrQ-v2 use contrastive representation learning or image augmentation to improve sample efficiency [9,10,28]. The critic remains a key feedback component because it defines both a bootstrap target and the direction used to improve the actor.

These approaches act on target estimation, data reuse, and visual representations. Amplification and contraction within the learned value function remain a complementary concern. ONI controls the forward weights while retaining the learning rules of DQN and DrQ-v2.

2.2. Network-Level Stabilization in Reinforcement Learning

Normalization changes either activation statistics or weight geometry. Batch Normalization and layer normalization operate on intermediate features [11,12]; group normalization removes batch-size dependence; and weight normalization reparameterizes each weight vector by direction and magnitude [13]. CrossNorm was developed for off-policy temporal-difference learning by mixing current and replay-state statistics [29]. D2RL (Deep Dense Architectures in Reinforcement Learning) uses dense input connections to improve information flow in deeper reinforcement-learning networks [30], while regularization studies in continuous control show that network design and parameter constraints can materially affect policy optimization [31].

Recent work has broadened network-level stabilization in actor–critic reinforcement learning. CrossQ makes careful use of Batch Renormalization in actor and critic networks and achieves high sample efficiency with a low update-to-data ratio [17]. Palenicek et al. study the behavior of CrossQ at higher update-to-data ratios and show that weight normalization stabilizes the resulting training dynamics while maintaining the effective learning rate [18]. Wang et al. systematically analyze Batch Normalization training and evaluation modes in off-policy actor–critic algorithms, associating unstable configurations with distribution mismatch and inaccurate Q-value estimation and reporting improved learning-rate robustness with mode-aware normalization [19]. Complementary analyses connect normalization to parameter-norm growth and effective learning-rate schedules [16], while hyperspherical normalization constrains weight and feature norms for scalable DRL optimization [20]. Together, these studies strengthen the view that neural parameterization and normalization can directly regulate DRL optimization dynamics.

A-LIX regularizes encoder gradients in pixel-based temporal-difference learning [21], while ReDo recycles dormant neurons to preserve expressivity [22]. Their focus on feature learning motivates examining representation dynamics alongside return when evaluating ONI.

Spectral normalization bounds maximal linear amplification by rescaling the largest singular value [14,32,33], but preserves the ratios between singular values. ONI progressively brings nonzero singular values toward a common scale. This stronger geometric constraint motivates examining both its placement and intensity in bootstrapped value learning.

2.3. Orthogonal Transformations and Controllable ONI

Orthogonal initialization can preserve signals at the start of training [34], while Stiefel-manifold optimization and orthogonal weight normalization maintain orthogonality during training [23,35]. Soft orthogonality penalties encourage, but do not directly construct, orthogonal forward weights [36]. Parseval networks constrain transformations for robustness [37], and gradient penalties regulate local derivatives [15,38].

Orthogonal initialization specifies the starting weights, and soft penalties encourage a constraint through the training loss. ONI instead constructs the forward weight from a trainable proxy at each use [24]. Its finite Newton iteration count controls the strength of orthogonalization, while layer selection determines where the constraint acts. These two controls allow the network to balance propagation control with the flexibility required for value estimation.

3. Materials and Methods

3.1. Problem Setting and Neural Function Approximation

We consider a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is the reward function, and P is the transition kernel. At time t, the agent observes s_t, selects a_t, receives r_t, and transitions to s_{t+1}. The objective is to maximize the expected discounted return

$$G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i, \tag{1}$$

where G_t is the return from time t, r_i is the reward at time i, and $\gamma \in [0, 1]$ is the discount factor. A value-based agent estimates the action value $Q(s, a)$, the expected discounted return for state s and action a, and selects actions from those estimates. An actor–critic agent represents the policy explicitly and uses one or more critics to evaluate actions. In both cases, the value estimate is trained against a bootstrapped target that contains another learned value. The combination of function approximation, bootstrapping, and an evolving data distribution creates a feedback path through which representation changes affect future targets.

Consider a K-layer neural function $F_\theta : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ with

$$\mathbf{h}_k = \mathbf{W}_k \mathbf{x}_{k-1}, \qquad \mathbf{x}_k = \phi(\mathbf{h}_k), \qquad k = 1, \ldots, K, \tag{2}$$

where K is the number of layers, d_{in} and d_{out} are the input and output dimensions, $\mathbf{x}_0 = \mathbf{x}$ is the input, and $\theta = \{\mathbf{W}_k\}_{k=1}^{K}$ collects the layer weights. At layer k, $\mathbf{h}_k$ is the preactivation, $\mathbf{x}_k$ is the activation, and ϕ is the elementwise activation function. Forward activations and backward gradients pass through products of these layer matrices. Singular values substantially above or below one can amplify or attenuate particular directions across repeated transformations. Controlling the nonzero singular values regulates propagation along the represented directions; for a rectangular layer, the relevant orthogonality condition depends on the matrix orientation.

3.2. Orthogonal Reparameterization

For a weight matrix $\mathbf{W}_k \in \mathbb{R}^{d_k \times d_{k-1}}$, where d_k and d_{k-1} are the output and input widths of layer k, the corresponding row-orthogonal set is

$$\mathcal{O}_k = \left\{ \mathbf{W}_k : \mathbf{W}_k \mathbf{W}_k^T = \mathbf{I} \right\}, \tag{3}$$

where $\mathbf{I}$ is the identity matrix and a superscript T denotes transpose. Row orthogonality requires $d_k \leq d_{k-1}$; the column condition $\mathbf{W}_k^T \mathbf{W}_k = \mathbf{I}$ is used for the opposite matrix orientation. Direct manifold optimization is possible [35], but it couples the optimizer to the constraint. ONI instead retains an unconstrained proxy parameter $\mathbf{V}$ and constructs the forward weight as $\mathbf{W} = \Phi_T(\mathbf{V})$, where Φ_T denotes the ONI reparameterization with T Newton steps. Standard optimization updates $\mathbf{V}$ through the gradient propagated across this reparameterization, while the network forward pass uses $\mathbf{W}$.

3.3. Controllable Orthogonalization by Newton's Iteration

Let $\mathbf{V} \in \mathbb{R}^{d_o \times d_i}$ denote the proxy weight matrix of a fully connected layer, with d_o output units and d_i input units. Following ONI, the parameter matrix is centered and spectrum normalized:

$$\mathbf{V}_c = \mathbf{V} - \frac{1}{d_i} \mathbf{V} \mathbf{1} \mathbf{1}^T, \qquad \mathbf{Z} = \frac{\mathbf{V}_c}{\sqrt{\| \mathbf{V}_c \mathbf{V}_c^T \|_F}}. \tag{4}$$

Here, $\mathbf{1} \in \mathbb{R}^{d_i}$ is the all-ones vector, $\mathbf{V}_c$ is the row-centered proxy, $\mathbf{Z}$ is its normalized form, and $\| \cdot \|_F$ is the Frobenius norm. The covariance-like matrix is then

$$\mathbf{S} = \mathbf{Z} \mathbf{Z}^T. \tag{5}$$

For a full-row-rank centered matrix, the normalization places the spectrum in the convergence region $\| \mathbf{I} - \mathbf{S} \|_2 < 1$, where $\| \cdot \|_2$ is the matrix spectral norm. The iteration operates on the nonzero singular directions in the rank-deficient case [24]. ONI then approximates $\mathbf{S}^{-1/2}$ through Newton's Iteration:

$$\mathbf{B}_0 = \mathbf{I}, \qquad \mathbf{B}_t = \frac{1}{2} (3 \mathbf{B}_{t-1} - \mathbf{B}_{t-1}^3 \mathbf{S}), \quad t = 1, 2, \ldots, T. \tag{6}$$

Here, $\mathbf{B}_t$ is the inverse-square-root approximation after Newton step t; the iteration index is local to this construction. The orthogonalized weight is

$$\mathbf{W} = \mathbf{B}_T \mathbf{Z}. \tag{7}$$

Increasing T progressively moves the nonzero singular values toward one [24]. Small T retains a broader singular spectrum, while larger T imposes a stronger constraint. Selecting the layer determines where this transformation acts in the value network.

Algorithm 1 gives the forward construction used by each ONI layer. The proxy $\mathbf{V}$ remains the trainable parameter, and gradients are propagated through the centering, normalization, Newton updates, and final matrix product by the chain rule.

3.4. Backpropagation Through ONI

The ONI transformation is differentiable, so the proxy parameter $\mathbf{V}$ is updated by propagating the loss gradient through the centering, spectrum normalization, Newton iterations, and final matrix product. Given $\partial \mathcal{L} / \partial \mathbf{W}$ and the forward-pass quantities $\mathbf{V}_c$, $\mathbf{Z}$, $\mathbf{S}$, and $\{\mathbf{B}_t\}_{t=1}^T$, Algorithm 2 gives the backward computation used to obtain $\partial \mathcal{L} / \partial \mathbf{V}$. The operator $\mathrm{tr}(\cdot)$ denotes the matrix trace.

Algorithm 1 Forward controllable orthogonalization by Newton's iteration

Input: proxy matrix $\mathbf{V} \in \mathbb{R}^{d_o \times d_i}$; iteration count T

$\mathbf{V}_c \leftarrow \mathbf{V} - \frac{1}{d_i}\mathbf{V}\mathbf{1}\mathbf{1}^T$

$\mathbf{Z} \leftarrow \mathbf{V}_c / \sqrt{\|\mathbf{V}_c\mathbf{V}_c^T\|_F}$

$\mathbf{S} \leftarrow \mathbf{Z}\mathbf{Z}^T$

$\mathbf{B}_0 \leftarrow \mathbf{I}$

for $t = 1$ to T **do**

  $\mathbf{B}_t \leftarrow \frac{3}{2}\mathbf{B}_{t-1} - \frac{1}{2}\mathbf{B}_{t-1}^3\mathbf{S}$

end for

$\mathbf{W} \leftarrow \mathbf{B}_T\mathbf{Z}$

Output: forward weight $\mathbf{W} \in \mathbb{R}^{d_o \times d_i}$

Algorithm 2 Backpropagation through controllable orthogonalization

Input: $\partial\mathcal{L}/\partial\mathbf{W}$ and saved forward quantities $\mathbf{V}_c$, $\mathbf{Z}$, $\mathbf{S}$, $\{\mathbf{B}_t\}_{t=1}^T$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{B}_T} \leftarrow \frac{\partial\mathcal{L}}{\partial\mathbf{W}}\mathbf{Z}^T$$

for $t = T$ down to 1 **do**

$$\frac{\partial\mathcal{L}}{\partial\mathbf{B}_{t-1}} \leftarrow \frac{3}{2}\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t} - \frac{1}{2}\left[\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}(\mathbf{B}_{t-1}^2\mathbf{S})^T + (\mathbf{B}_{t-1}^2)^T\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}\mathbf{S}^T + \mathbf{B}_{t-1}^T\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}(\mathbf{B}_{t-1}\mathbf{S})^T\right]$$

end for

$$\frac{\partial\mathcal{L}}{\partial\mathbf{S}} \leftarrow -\frac{1}{2}\sum_{t=1}^{T}(\mathbf{B}_{t-1}^3)^T\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{Z}} \leftarrow \mathbf{B}_T^T\frac{\partial\mathcal{L}}{\partial\mathbf{W}} + \left(\frac{\partial\mathcal{L}}{\partial\mathbf{S}} + \left(\frac{\partial\mathcal{L}}{\partial\mathbf{S}}\right)^T\right)\mathbf{Z}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{V}_c} \leftarrow \frac{1}{\sqrt{\|\mathbf{V}_c\mathbf{V}_c^T\|_F}}\left(\frac{\partial\mathcal{L}}{\partial\mathbf{Z}} - \mathrm{tr}\left[\left(\frac{\partial\mathcal{L}}{\partial\mathbf{Z}}\right)^T\mathbf{Z}\right]\mathbf{Z}\mathbf{Z}^T\mathbf{Z}\right)$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{V}} \leftarrow \frac{\partial\mathcal{L}}{\partial\mathbf{V}_c} - \frac{1}{d_i}\frac{\partial\mathcal{L}}{\partial\mathbf{V}_c}\mathbf{1}\mathbf{1}^T$$

Output: proxy-parameter gradient $\partial\mathcal{L}/\partial\mathbf{V} \in \mathbb{R}^{d_o \times d_i}$

3.5. ONI in DQN

For DQN, the baseline agent contains a convolutional encoder followed by fully connected layers that predict one Q value per action. Its temporal-difference loss is

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\theta(s,a) - \left(r + \gamma\max_{a'}Q_{\theta'}(s',a')\right)\right)^2\right], \tag{8}$$

where θ denotes the online Q-network parameters, θ' denotes the target Q-network parameters, and $\mathcal{D}$ is the replay buffer.

The target network and replay buffer address target drift and temporal correlation, respectively, while the value head transforms encoded features into Q values. Learnable layers are indexed backward from the output, excluding flattening and activations. DQN-ONI[−2] applies ONI to the hidden fully connected layer; DQN-ONI[−3] applies it to the preceding learnable layer, the final convolution in Figure 1; DQN-ONI[−2&−3] constrains both. The benchmark configuration uses position −2, with T controlling the strength.

Figure 1 shows the DQN-ONI[−2] architecture used in the study. Four stacked input frames are processed by three convolutional layers, followed by a 3136-dimensional flattened representation and a 512-unit hidden layer. ONI is attached to the penultimate

fully connected transformation before the action-value output, where $|\mathcal{A}|$ is the number of discrete actions.

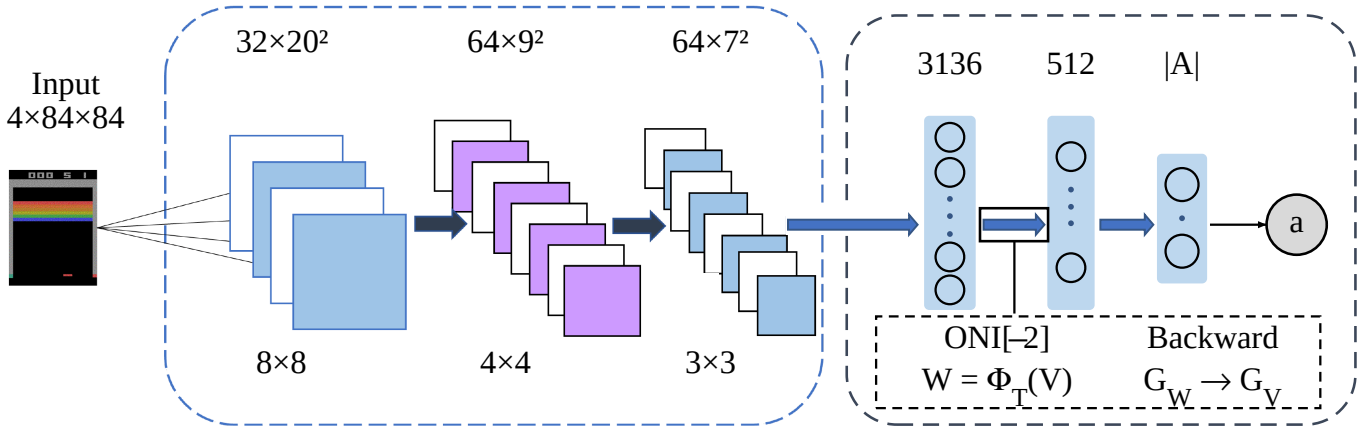


Figure 1. DQN-ONI[−2] architecture. Feature-map labels give channels and spatial dimensions; the squared spatial term denotes equal height and width. The convolution kernels are 8×8, 4×4, and 3×3. The value head maps 3136 features through 512 hidden units to $|\mathcal{A}|$ action values, which determine the selected action. ONI acts on the 3136-to-512 transformation; the inset links its forward and backward computations, with $G_W = \partial\mathcal{L}/\partial\mathbf{W}$ and $G_V = \partial\mathcal{L}/\partial\mathbf{V}$.

3.6. ONI in DrQ-v2

DrQ-v2 is an image-based actor–critic agent that combines data augmentation, n-step returns, and deterministic continuous-control learning [10]. Its encoder supplies a compact feature vector to the actor and twin critics. The two critics are denoted Q_1 and Q_2 and share the same architecture; the equations below use Q to denote a generic critic. The critic update is based on a temporal-difference objective:

$$\mathcal{L}(\theta^Q) = \frac{1}{N}\sum_i \Big[r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'}) - Q(s_i, a_i \mid \theta^Q)\Big]^2. \tag{9}$$

Here, N is the mini-batch size; Q and Q' denote the online and target critics; μ and μ' denote the online and target actors; and θ^Q, $\theta^{Q'}$, θ^μ, and $\theta^{\mu'}$ are their corresponding parameter sets.

The actor is updated through the critic gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a \mid \theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s \mid \theta^\mu)|_{s_i}. \tag{10}$$

where J denotes the actor objective and the gradient is evaluated through the online critic Q.

The critic occupies two feedback paths: it participates in the bootstrapped target and supplies the gradient used to update the actor. Its representation therefore affects value learning and the direction passed to the policy. ONI is applied to the penultimate transformation in each critic, while the encoder, augmentation, target update, and actor structure retain the DrQ-v2 design. We denote this variant Critic-ONI[−2].

Figure 2 shows the shared image encoder and separate actor and critic projections to 50 features. Each critic concatenates its projected representation with the action. Critic-ONI[−2] constrains the penultimate, $1024 \to 1024$ transformation of both critics.

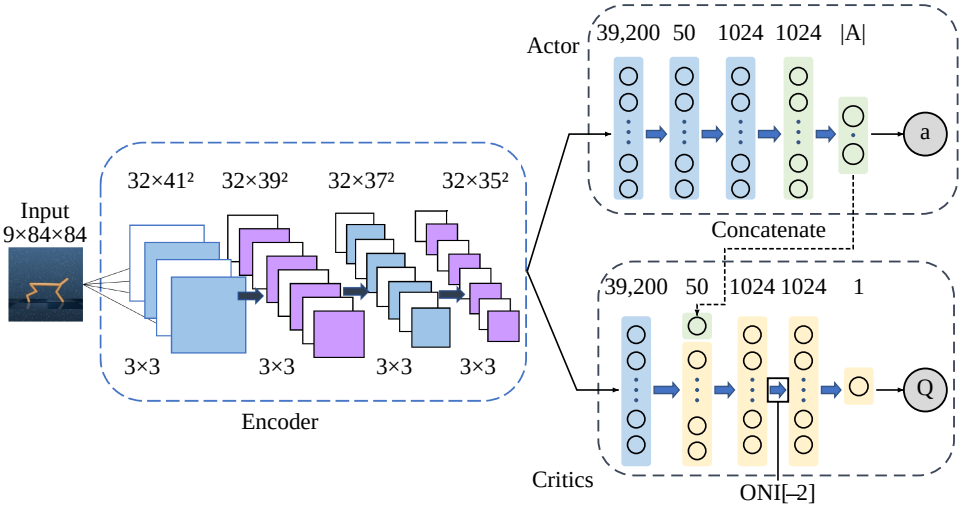


Figure 2. DrQ-v2 architecture with Critic-ONI[−2]. The encoder feeds separate actor and critic projections from 39,200 to 50 features. The action is concatenated with the projected critic features; it is sampled from replay for critic fitting and generated by the actor for policy optimization. One of the two identically structured critics is shown. The outlined connection marks the penultimate, 1024-to-1024 ONI transformation in each critic. Feature-map dimensions use the notation of Figure 1; all convolution kernels are 3×3.

4. Experimental Setup

4.1. Evaluation Design and Environments

The experiments cover three groups of environments:

- MinAtar tasks for compact value-based evaluation [39].
- Atari games for image-based DQN evaluation.
- DeepMind Control Suite tasks for continuous-control DrQ-v2 evaluation [40].

The evaluation follows three stages. First, the design stage varies ONI placement and Newton iteration count on MinAtar-asterix and Atari-asterix to identify the configurations used in subsequent experiments. Second, the benchmark stage evaluates the selected configurations across three MinAtar games, eight Atari games, and DMC continuous-control tasks. Third, stability analyses examine sensitivity to random initialization on challenging DMC tasks, learning-rate robustness, and penultimate-layer activation statistics. Episode return is plotted against environment frames; cross-game performance uses the mean MinAtar normalized score (MNS) for MinAtar and the mean human-normalized score for Atari, and the challenging DMC evaluation uses the observed training failure rate.

For MinAtar, the normalization follows Gogianu et al. [32]:

$$\mathrm{MNS}_g = 100 \, \frac{s_g - s_{g,\mathrm{random}}}{s_{g,\mathrm{max}} - s_{g,\mathrm{random}}}, \tag{11}$$

where g indexes the game, s_g is the evaluated agent's episode-return score, $s_{g,\mathrm{random}}$ is the random-policy reference, and $s_{g,\mathrm{max}}$ is the largest baseline-agent score recorded in the reference study. The fixed pairs $(s_{g,\mathrm{max}}, s_{g,\mathrm{random}})$ are $(78.90, 0.49)$ for Asterix, $(360.92, 2.86)$ for Space Invaders, and $(93.91, 0.09)$ for Seaquest. We report their mean normalized score. Atari human normalization instead uses human performance as the upper reference.

Episode return measures policy quality. To assess training stability, we also examine learning failures across DMC initializations, return across learning rates on MinAtar-seaquest, and the l_1 norms of per-dimension activation means and variances. These diag-

nostics characterize learning reliability, optimizer sensitivity, and representation-statistic drift, respectively.

Within each baseline–ONI pair, the optimizer, replay mechanism, target update, and evaluation settings are held fixed. ONI placement and T are the controlled architectural changes. The DQN experiments use a Python 3.10 implementation of the architecture shown in Figure 1; the continuous-control experiments follow the published DrQ-v2 agent and training protocol [10].

4.2. DQN Configuration

The DQN ablation evaluates ONI at learnable-layer positions -2, -3, and both positions, using the indexing defined in Section 3.5. It also varies T from 1 to 5. The benchmark configuration fixes DQN-ONI[-2] with $T = 1$ across the MinAtar games and $T = 2$ across the Atari games, following the representative-task ablations. The plain DQN uses the same encoder, value head, optimizer, replay buffer, and target-network schedule without the ONI reparameterization.

The main DQN hyperparameters are summarized in Table 1.

Table 1. Main DQN hyperparameters used in the experiments.

Hyperparameter	Value
Discount factor γ	0.99
Update frequency	4
Target update frequency	8000
Starting exploration rate	1.0
Final exploration rate	0.01
Exploration decay steps	250,000
Exploration schedule	Linear
Warm-up steps	20,000
Replay size	10^6
Batch size	32
History length	4
Loss function	Mean squared error (MSE)
Optimizer	Adam
Learning rate η	0.00025
Adam damping term ϵ_{Adam}	0.0003125
Adam coefficients (β_1, β_2)	(0.9, 0.999)
Validation steps	125,000
Validation exploration rate	0.001

4.3. DrQ-v2 Configuration

The continuous-control configuration transfers the selected layer position and $T = 2$ from DQN to the penultimate fully connected transformation of both DrQ-v2 critics. This setting is held fixed across the DMC tasks. The actor and encoder retain their original parameterization. The baseline is the corresponding plain DrQ-v2 agent with the same augmentation, replay, update, and exploration settings.

The twelve-task evaluation uses the medium benchmark of DrQ-v2 [10]: acrobot-swingup, cartpole-swingup-sparse, cheetah-run, finger-turn-easy, finger-turn-hard, hopper-hop, quadruped-run, quadruped-walk, reach-duplo, reacher-easy, reacher-hard, and walker-run. Each agent is trained with three random seeds per task, giving 36 runs per agent. A run is classified as a learning failure when its return remains near the initial low level throughout the training horizon, with no sustained improvement. This trajectory-based criterion distinguishes failure to learn from a temporary performance drop. The humanoid experiments provide additional task-level comparisons.

The main DrQ-v2 hyperparameters are summarized in Table 2.

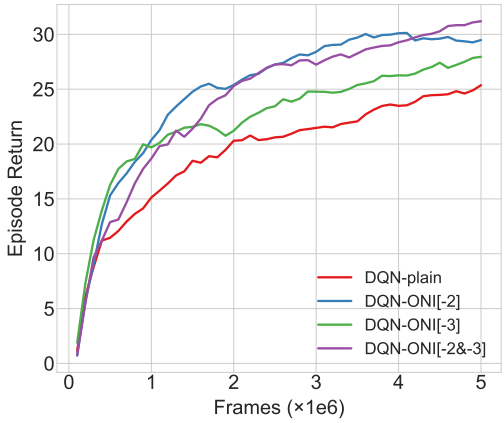
Table 2. Main DrQ-v2 hyperparameters used in the experiments.

Hyperparameter	Value
Replay buffer capacity	10^6
Action repeat	2
Seed frames	4000
Exploration steps	2000
n-step returns	3
Mini-batch size	256
Discount factor γ	0.99
Optimizer	Adam
Learning rate	10^{-4}
Agent update frequency	2
Target critic soft-update rate τ	0.01
Feature dimension	50
Hidden dimension	1024
Exploration standard-deviation clip	0.3

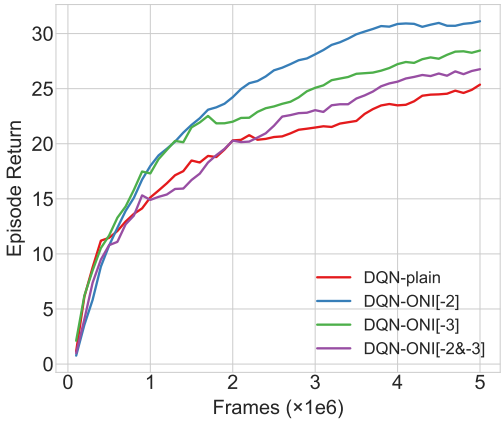
5. Results

5.1. Layer Placement and Orthogonalization Intensity

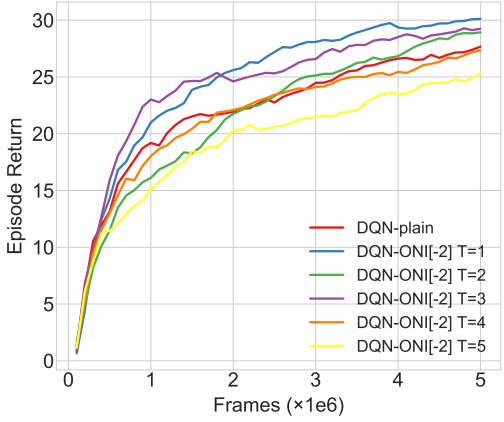
Figure 3 compares ONI placement and strength on MinAtar-asterix. At $T = 1$ and $T = 2$, the penultimate-layer and two-layer variants finish above plain DQN. The intensity sweep favors $T = 1$, selected for the subsequent MinAtar evaluation.



(**a**) Layer placement, $T = 1$



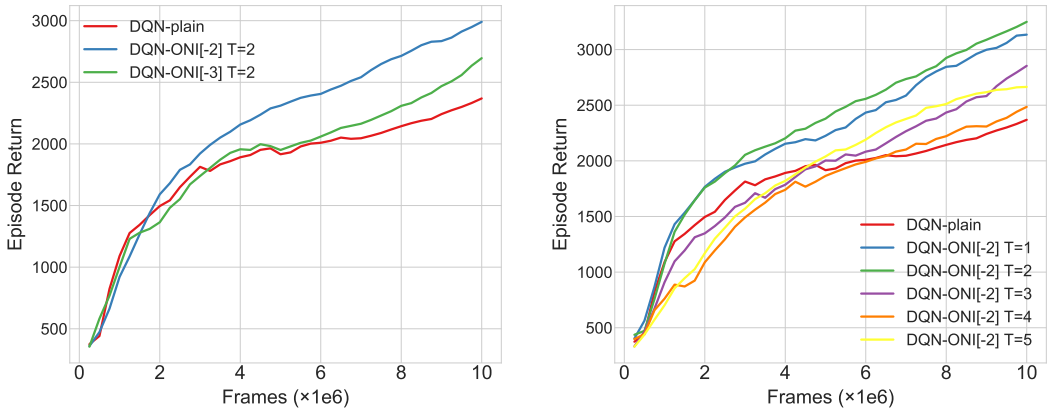
(**b**) Layer placement, $T = 2$



(**c**) Iteration count, ONI[−2]

Figure 3. Layer-position and intensity analysis on MinAtar-asterix. Panels (**a**,**b**) compare ONI placement at $T = 1$ and $T = 2$; panel (**c**) varies T from 1 to 5 for ONI on the penultimate layer.

The Atari-asterix ablation in Figure 4 shows the same design pattern. DQN-ONI[−2] with $T = 2$ achieves the strongest later learning trajectory among the displayed placement variants, while the intensity sweep again favors a tuned value of T. Across the two ablations, the penultimate value-head layer provides the effective placement. Layer location and iteration count therefore function as complementary controls over ONI.

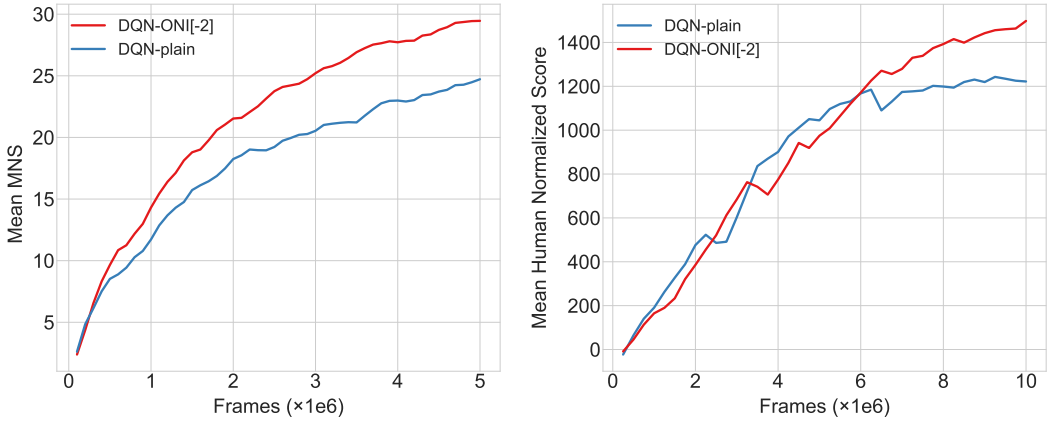


(**a**) Layer placement, $T = 2$ (**b**) Iteration count, ONI[−2]

Figure 4. ONI placement and iteration-count ablation on Atari-asterix. Panel (**a**) compares layer positions at $T = 2$; panel (**b**) compares $T = 1, \ldots, 5$ for DQN-ONI[−2].

5.2. DQN Evaluation on MinAtar and Atari

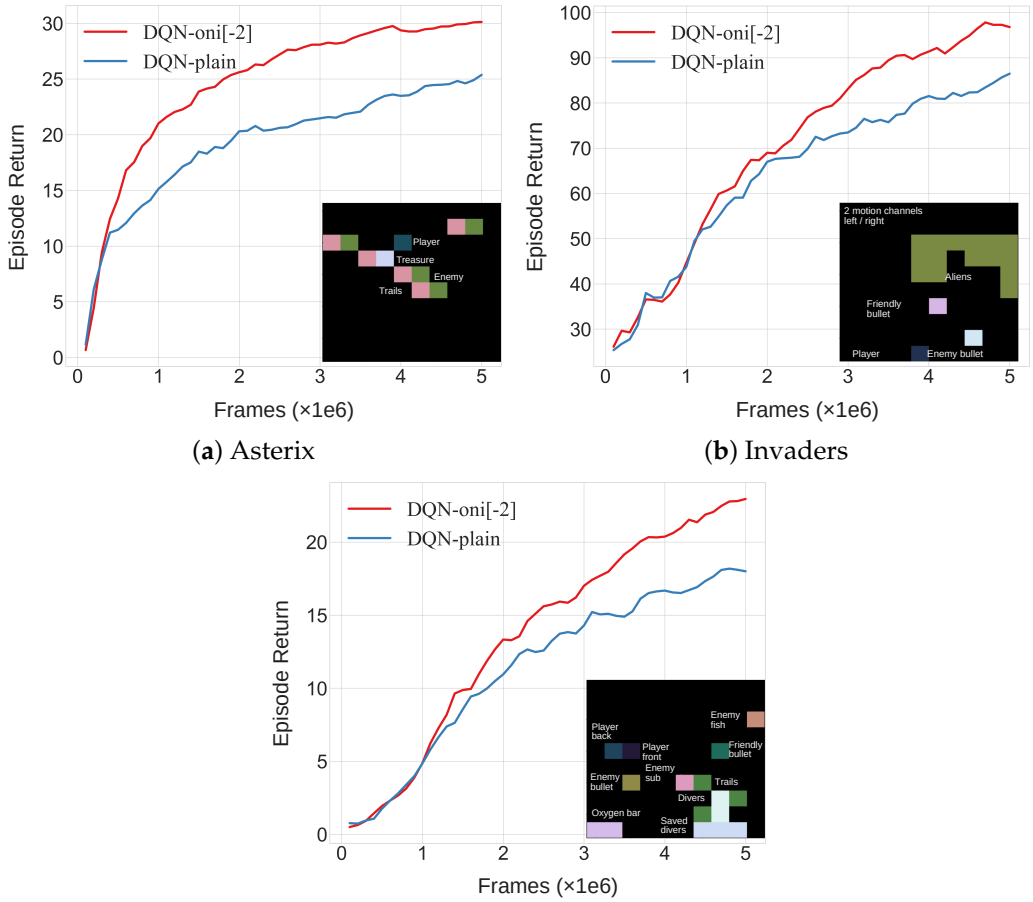
Figure 5 provides the aggregate view. DQN-ONI[−2] achieves a higher mean MNS for the three-game MinAtar group and a higher mean human-normalized score for the eight-game Atari group throughout most of training. The per-task curves in Figures 6–8 show how this aggregate gain develops across different learning profiles.



(**a**) MinAtar, $T = 1$ (**b**) Atari, $T = 2$

Figure 5. Aggregate DQN performance. Panel (**a**) reports the mean MinAtar normalized score over three games using DQN-ONI[−2], $T = 1$; panel (**b**) reports the mean human-normalized score over eight Atari games using DQN-ONI[−2], $T = 2$.

Across the three MinAtar tasks in Figure 6, DQN-ONI[−2] with $T = 1$ finishes above plain DQN. The difference is largest in asterix and remains visible in invaders and seaquest, showing that the selected configuration transfers from the ablation game to the other MinAtar tasks.



(**a**) Asterix (**b**) Invaders

(**c**) Seaquest

Figure 6. DQN and DQN-ONI[−2], $T = 1$, on MinAtar: (**a**) asterix, (**b**) invaders, and (**c**) seaquest.

The first four Atari curves are shown in Figure 7. DQN-ONI[−2] with $T = 2$ ends above the baseline in asterix, battle zone, boxing, and breakout. Battle zone exhibits a temporal crossover in which the ONI curve overtakes the baseline during training, illustrating an improvement in both the attained return and the timing of learning.

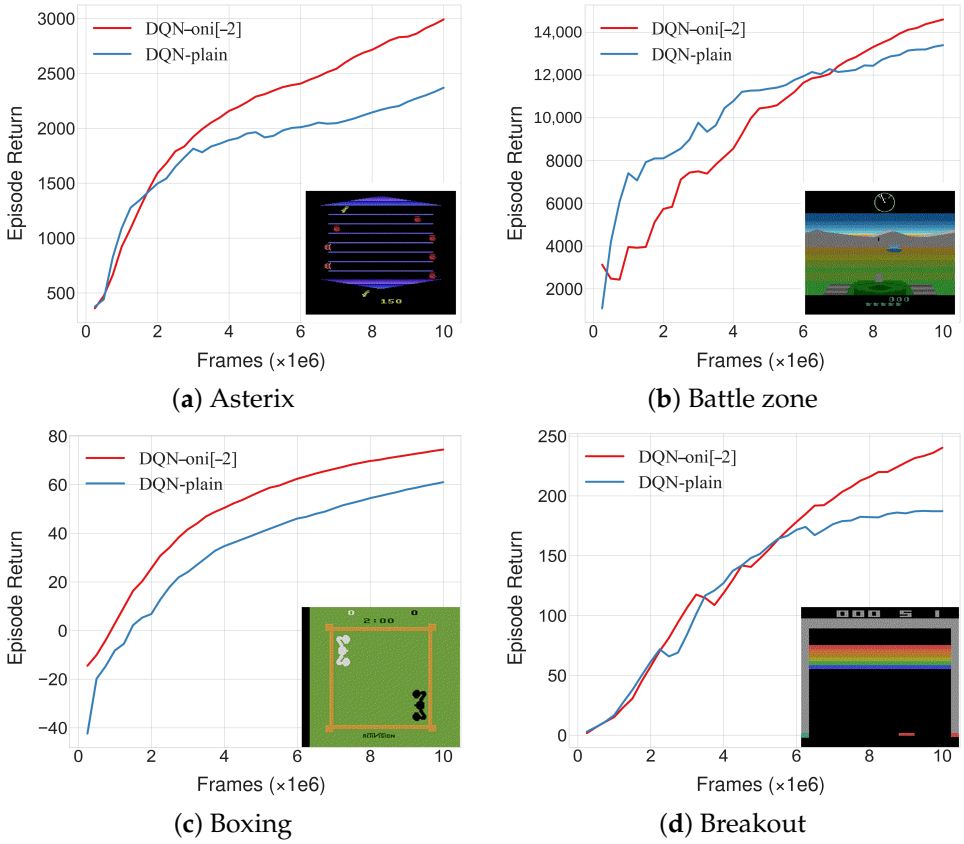
Figure 8 contains the remaining games. The ONI curve ends higher in gopher, name this game, robotank, and seaquest. In gopher, the ONI trajectory overtakes the baseline later in training and finishes higher, illustrating a complementary late-stage gain.

Overall, the selected DQN-ONI configurations improve aggregate normalized scores and final learning trajectories across the evaluated MinAtar and Atari games. The per-task curves show complementary gains in learning speed and final return across the evaluated games.

5.3. DrQ-v2 Evaluation and Training Stability on DeepMind Control Suite

The continuous-control experiment transfers the intervention from a DQN value head to the twin critics of DrQ-v2. Figure 9 shows three difficult humanoid tasks. In humanoid-stand and humanoid-run, Critic-ONI[−2] begins sustained learning substantially earlier. In humanoid-walk, the ONI critic rises earlier and finishes higher. Across these tasks, ONI primarily changes the onset and continuity of successful learning.

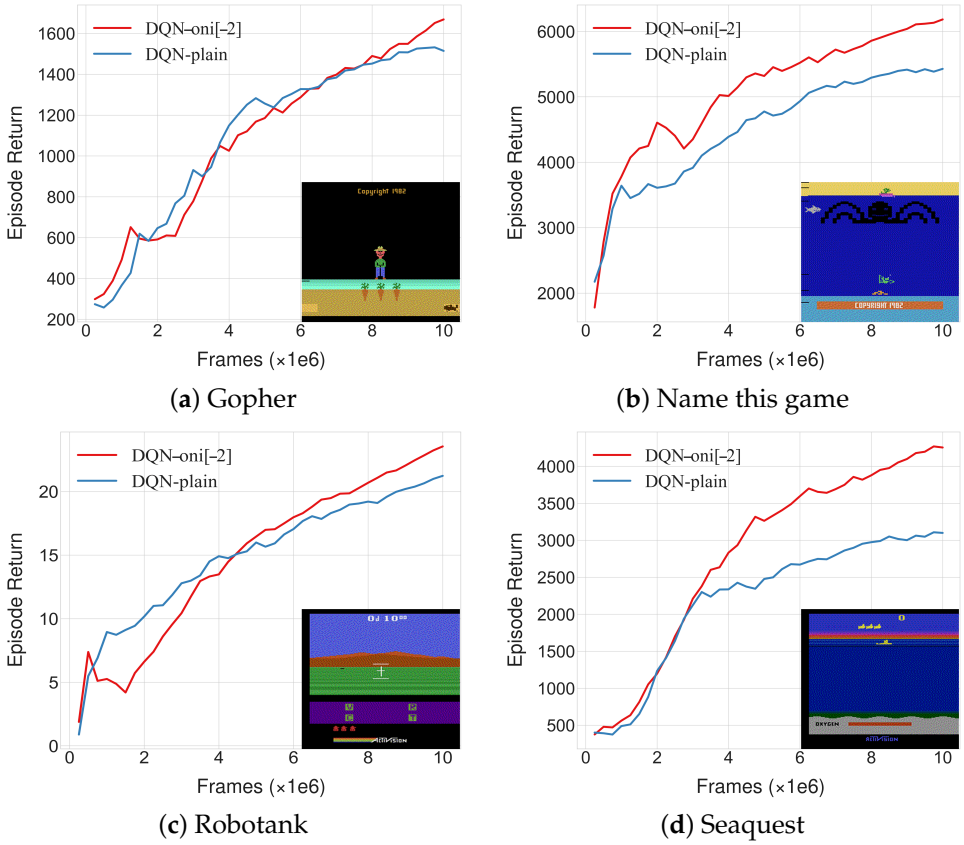
The additional tasks in Figure 10 separate failure recovery from ordinary acceleration. In cartpole-swingup-sparse and hopper-hop, plain DrQ-v2 remains near its initial return while the ONI variant reaches a sustained positive trajectory. In quadruped-run and reach-duplo, both agents learn and Critic-ONI[−2] improves learning speed or the attained return.



(**a**) Asterix (**b**) Battle zone

(**c**) Boxing (**d**) Breakout

Figure 7. DQN and DQN-ONI[−2], $T = 2$, on Atari: (**a**) asterix, (**b**) battle zone, (**c**) boxing, and (**d**) breakout. Red curves denote DQN-ONI[−2]; blue curves denote plain DQN.

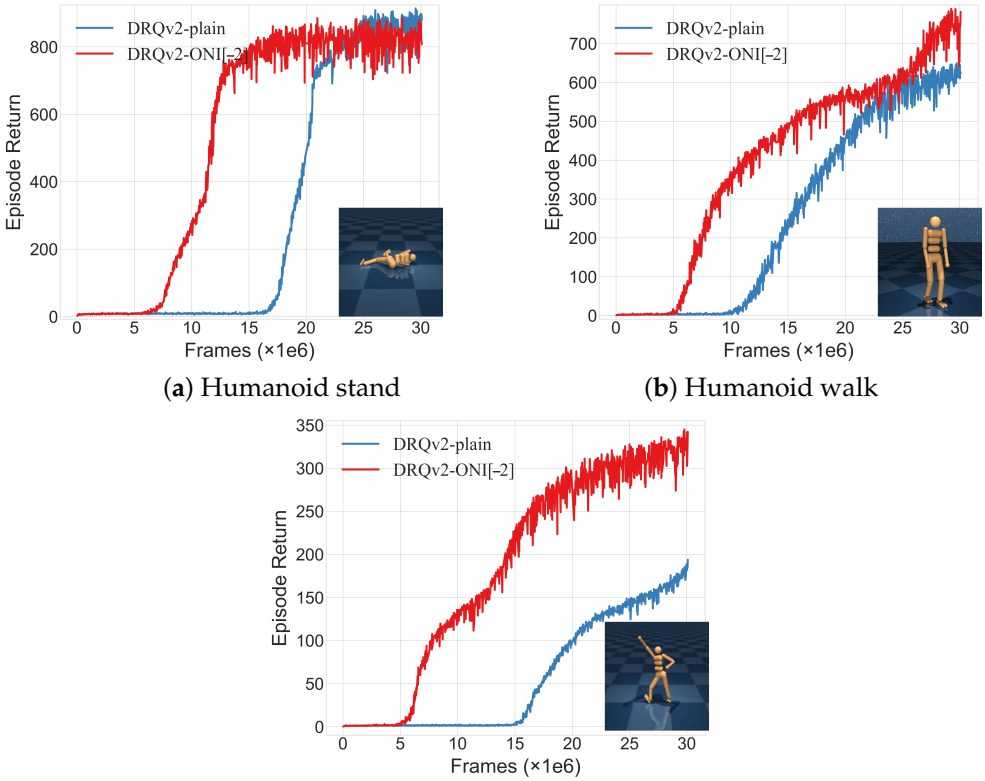


(**a**) Gopher (**b**) Name this game

(**c**) Robotank (**d**) Seaquest

Figure 8. DQN and DQN-ONI[−2], $T = 2$, on Atari: (**a**) gopher, (**b**) name this game, (**c**) robotank, and (**d**) seaquest. Red curves denote DQN-ONI[−2]; blue curves denote plain DQN.

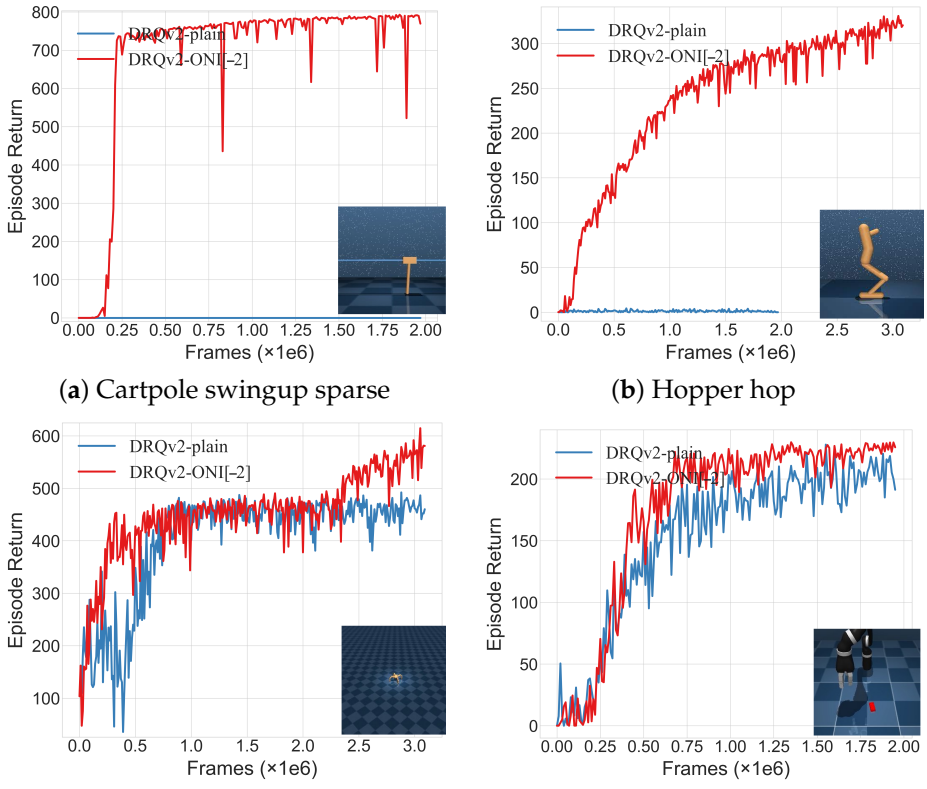
https://doi.org/10.3390/informatics13100162



(**a**) Humanoid stand (**b**) Humanoid walk

(**c**) Humanoid run

Figure 9. DrQ-v2 with and without Critic-ONI[−2], $T = 2$, on (**a**) humanoid-stand, (**b**) humanoid-walk, and (**c**) humanoid-run. The ONI agent is denoted DrQ-v2-ONI[−2] in the legends.



(**a**) Cartpole swingup sparse (**b**) Hopper hop

(**c**) Quadruped run (**d**) Reach duplo

Figure 10. Selected challenging DMC tasks using $T = 2$: (**a**) cartpole-swingup-sparse, (**b**) hopper-hop, (**c**) quadruped-run, and (**d**) reach-duplo. Red curves denote DrQ-v2 with Critic-ONI[−2]; blue curves denote plain DrQ-v2.

Critic-ONI[−2] exhibits no learning failures in the 36 runs of the twelve-task benchmark. Plain DrQ-v2 exhibits failed-learning trajectories on cartpole-swingup-sparse and hopper-hop (Figure 10). These outcomes support improved training reliability under the evaluated initializations.

5.4. Learning-Rate Robustness and Activation Statistics

The two diagnostics in Figure 11 address optimizer robustness and internal representation dynamics. The learning rates are logarithmically spaced from 1.0×10^{-5} to 2.1544×10^{-3}. DQN-ONI[−2] achieves a higher average episode return over most tested rates and retains effective learning at the largest rate, widening the usable optimization range.

For the activation analysis, let $\boldsymbol{\mu}_t$ and $\boldsymbol{\nu}_t$ denote the vectors of per-dimension mean and variance, respectively, for the penultimate-layer output at training step t. The right panel plots $\|\boldsymbol{\mu}_t\|_1$ and $\|\boldsymbol{\nu}_t\|_1$. Here, $\|\cdot\|_1$ denotes the sum of absolute vector components. Both quantities grow rapidly for plain DQN, while DQN-ONI[−2] with $T = 2$ keeps them lower and closely aligned. Together, the learning-rate and activation-statistic analyses show a wider usable optimization range and more stable intermediate feature statistics with controllable ONI.

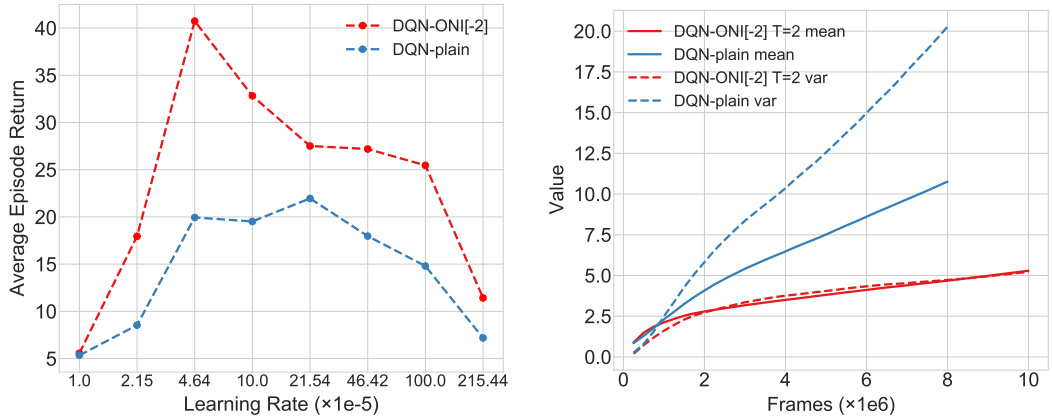


Figure 11. Diagnostic stability analysis. (**Left**): DQN learning-rate sensitivity on MinAtar-seaquest using logarithmically spaced rates and DQN-ONI[−2], $T = 1$. (**Right**): l_1 norms of the per-dimension activation mean (solid) and variance (dashed) at the penultimate layer, using DQN-ONI[−2], $T = 2$.

6. Discussion

6.1. From Geometric Control to Stable Learning

The results reveal a consistent pattern across the two architectures. In DQN, placement and intensity ablations show that selective orthogonalization of a late value-head transformation provides an effective balance between geometric control and representation flexibility. The selected configurations improve aggregate and task-level learning trajectories. Applying the same design principle to the DrQ-v2 critics produces earlier learning and a lower observed failure rate on difficult continuous-control tasks. The learning-rate sweep and activation statistics further connect these task-level effects to optimization robustness and intermediate feature stability.

Orthogonalization constrains directional amplification because the singular values of the selected transformation move toward one. For a feed-forward network, this geometric property is compatible with smoother propagation through the value head. In DQN, it can regulate how changes in encoded states affect Q values and subsequent temporal-difference targets. In DrQ-v2, the corresponding control in the critic also influences the gradient delivered to the actor.

The activation-statistic analysis provides the clearest empirical connection to the internal representation. DRL continually changes both the replay distribution and its bootstrapped targets, so stable hidden statistics can reduce repeated adaptation to a moving feature distribution. The ℓ_1-norm trajectories in Figure 11 show lower growth of the per-dimension activation mean and variance with ONI. Together with the learning-rate sweep, this behavior supports the interpretation that controllable orthogonalization stabilizes intermediate feature distributions during DRL training.

Activation variance in this analysis concerns features within one network. Learning outcomes across initializations, reported in Section 5.3, provide the separate evidence on across-run reliability.

6.2. Comparison with Recent Normalization Methods

Compared with individual-vector weight normalization or hyperspherical norm constraints, ONI also regulates relationships between weight directions through the Gram matrix. This matrix-level control provides a way to influence the representation dynamics observed here. Unlike the batch-statistic and target-network changes in CrossQ [17,18], or the distributional-value and reward-scale design in SimbaV2 [20], selective ONI retains the baseline replay and target-update mechanisms. It is computed from weights independently of batch activation statistics.

6.3. Practical Implications and Design Guidance

The ablations favor finite, small iteration counts rather than the strongest constraint. They suggest starting with $T \in \{1, 2\}$ at the penultimate value or critic layer and comparing learning on a representative task at a matched training budget. When performance is comparable, smaller T reduces computation.

Matrix conditioning governs the change produced by each Newton step [24]; width affects cost, while depth and the number of constrained layers affect how constraints accumulate. Task type and complexity enter through the value function's required flexibility.

The selected fully connected layers directly affect value prediction and the critic gradient while limiting the number of constrained matrices. Leaving the visual encoder unconstrained preserves its flexibility but also leaves its weight conditioning unchanged. The position ablations include the preceding convolutional layer, so the head-only configuration reflects the placement comparison.

For convolutional extension, a kernel of size $c_o \times c_i \times k_h \times k_w$ can be reshaped into a $c_o \times (c_i k_h k_w)$ matrix [24], where c_o, c_i are output and input channels and k_h, k_w are kernel height and width. Orthogonality of flattened filters differs from orthogonality of the full convolution operator because of spatial overlap, stride, and padding [41]; kernel and channel dimensions also determine the added matrix cost.

6.4. Computational Cost

For an $m \times n$ weight matrix, where m and n are its row and column dimensions, Gram-matrix formation and weight reconstruction each require $O(m^2 n)$ operations. Each Newton step uses a constant number of $m \times m$ matrix products, giving an additional forward cost of $O(m^2 n + T m^3)$ per reconstruction; $O(\cdot)$ denotes asymptotic operation count. Back-propagation has the same asymptotic dependence, with $O(T m^2)$ intermediate matrix storage in a direct implementation. Thus, iteration-dependent work and storage increase linearly with T at fixed dimensions. Total training time also includes network evaluation, replay sampling, and environment interaction. In the displayed architectures, $(m, n) = (512, 3136)$ for DQN and $(1024, 1024)$ for each of the two DrQ-v2 critic transformations.

7. Conclusions

This study presents controllable orthogonalization as a network-level mechanism for stabilizing neural network training in DRL. ONI is inserted selectively into DQN value heads and DrQ-v2 critics, with layer placement controlling where the intervention acts and Newton iteration count controlling its strength. The selected configurations improve aggregate and task-level DQN learning, produce earlier learning on difficult continuous-control tasks, and reduce observed training failures across the challenging DMC evaluation. The learning-rate sweep and activation-statistic analysis further show a wider usable optimization range and more stable intermediate feature statistics. Together, these results provide a concrete way to regulate neural weight geometry while retaining the algorithmic structure of value-based and actor–critic learning.

Author Contributions: Conceptualization, J.Y., D.S., L.H. and T.S.; methodology, D.S., L.H. and T.S.; software, D.S.; validation, D.S.; formal analysis, D.S. and T.S.; investigation, J.Y., D.S., H.C. and H.W.; resources, J.Y., H.C., H.W. and L.H.; data curation, D.S.; writing—original draft preparation, J.Y., D.S. and T.S.; writing—review and editing, H.C., H.W., L.H. and T.S.; visualization, D.S.; supervision, L.H. and T.S.; project administration, J.Y., L.H. and T.S.; funding acquisition, L.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant numbers 62441617 and 62476016; and the Fundamental Research Funds for the Central Universities.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The benchmark environments used in this study are publicly available through the Atari Learning Environment, MinAtar, and DeepMind Control Suite. The experimental results supporting the findings are presented in the figures and tables of this article.

Conflicts of Interest: Authors Jia Yu, Hua Cui, and Hai Wang were employed by TravelSky Technology Limited. Author Dun Su was employed by BAIYANG AI. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ALE	Arcade Learning Environment
DMC	DeepMind Control Suite
DQN	Deep Q-Network
DRL	Deep reinforcement learning
DrQ-v2	Data-Regularized Q version 2 agent
MDP	Markov decision process
ONI	Orthogonalization by Newton's Iteration
TD	Temporal difference

References

1. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
2. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
3. Bellemare, M.G.; Naddaf, Y.; Veness, J.; Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.* **2013**, *47*, 253–279. [CrossRef]

4. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
5. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
6. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 387–395.
7. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep reinforcement learning that matters. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
8. Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
9. Kostrikov, I.; Yarats, D.; Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv* **2020**, arXiv:2004.13649.
10. Yarats, D.; Fergus, R.; Lazaric, A.; Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv* **2021**, arXiv:2107.09645.
11. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
12. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
13. Salimans, T.; Kingma, D.P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 901–909.
14. Miyato, T.; Kataoka, T.; Koyama, M.; Yoshida, Y. Spectral normalization for generative adversarial networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
15. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved training of Wasserstein GANs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5767–5777.
16. Lyle, C.; Zheng, Z.; Khetarpal, K.; Martens, J.; van Hasselt, H.; Pascanu, R.; Dabney, W. Normalization and effective learning rates in reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 106440–106473. [CrossRef]
17. Bhatt, A.; Palenicek, D.; Belousov, B.; Argus, M.; Amiranashvili, A.; Brox, T.; Peters, J. CrossQ: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
18. Palenicek, D.; Vogt, F.; Watson, J.; Peters, J. Scaling off-policy reinforcement learning with batch and weight normalization. *Adv. Neural Inf. Process. Syst.* **2025**, *38*, 61948–61970. [CrossRef]
19. Wang, L.; Sudun; Zhang, X.; Wu, W.; Huang, L. An investigation of batch normalization in off-policy actor-critic algorithms. *arXiv* **2025**, arXiv:2509.23750. [CrossRef]
20. Lee, H.; Lee, Y.; Seno, T.; Kim, D.; Stone, P.; Choo, J. Hyperspherical normalization for scalable deep reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning, Vancouver, BC, Canada, 13–19 July 2025*; PMLR: London, UK, 2025; Volume 267, pp. 33352–33403.
21. Cetin, E.; Ball, P.J.; Roberts, S.; Celiktutan, O. Stabilizing off-policy deep reinforcement learning from pixels. In *Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022*; PMLR: London, UK, 2022; Volume 162, pp. 2784–2810.
22. Sokar, G.; Agarwal, R.; Castro, P.S.; Evci, U. The dormant neuron phenomenon in deep reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023*; PMLR: London, UK, 2023; Volume 202, pp. 32145–32168.
23. Huang, L.; Liu, X.; Lang, B.; Yu, A.W.; Wang, Y.; Li, B. Orthogonal weight normalization: Solution to optimization over multiple dependent Stiefel manifolds in deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
24. Huang, L.; Liu, L.; Zhu, F.; Wan, D.; Yuan, Z.; Li, B.; Shao, L. Controllable orthogonalization in training DNNs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 6429–6438.
25. van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016*; AAAI Press: Palo Alto, CA, USA, 2016; Volume 30.
26. Wang, Z.; Schaul, T.; Hessel, M.; van Hasselt, H.; Lanctot, M.; de Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
27. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.

28. Laskin, M.; Srinivas, A.; Abbeel, P. CURL: Contrastive unsupervised representations for reinforcement learning. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 5639–5650.
29. Bhatt, A.; Argus, M.; Amiranashvili, A.; Brox, T. CrossNorm: Normalization for off-policy TD reinforcement learning. *arXiv* **2019**, arXiv:1902.05605.
30. Sinha, S.; Bharadhwaj, H.; Srinivas, A.; Garg, A. D2RL: Deep dense architectures in reinforcement learning. *arXiv* **2020**, arXiv:2010.09163.
31. Liu, Z.; Li, X.; Kang, B.; Darrell, T. Regularization matters in policy optimization: An empirical study on continuous control. *arXiv* **2019**, arXiv:1910.09191.
32. Gogianu, F.; Berariu, T.; Rosca, M.C.; Clopath, C.; Busoniu, L.; Pascanu, R. Spectral normalisation for deep reinforcement learning: An optimisation perspective. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021.
33. Bjorck, N.; Gomes, C.P.; Weinberger, K.Q. Towards deeper deep reinforcement learning with spectral normalization. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 8242–8255.
34. Saxe, A.M.; McClelland, J.L.; Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv* **2013**, arXiv:1312.6120.
35. Absil, P.-A.; Mahony, R.; Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*; Princeton University Press: Princeton, NJ, USA, 2008.
36. Bansal, N.; Chen, X.; Wang, Z. Can we gain more from orthogonality regularizations in training deep networks? In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 4266–4276.
37. Cisse, M.; Bojanowski, P.; Grave, E.; Dauphin, Y.; Usunier, N. Parseval networks: Improving robustness to adversarial examples. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 854–863.
38. Gouk, H.; Frank, E.; Pfahringer, B.; Cree, M.J. Regularisation of neural networks by enforcing Lipschitz continuity. *Mach. Learn.* **2021**, *110*, 393–416. [CrossRef]
39. Young, K.; Tian, T. MinAtar: An Atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv* **2019**, arXiv:1903.03176.
40. Tassa, Y.; Doron, Y.; Muldal, A.; Erez, T.; Li, Y.; de Las Casas, D.; Budden, D.; Abdolmaleki, A.; Merel, J.; Lefrancq, A.; et al. DeepMind Control Suite. *arXiv* **2018**, arXiv:1801.00690.
41. Wang, J.; Chen, Y.; Chakraborty, R.; Yu, S.X. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*; IEEE: Piscataway, NJ, USA, 2020; pp. 11505–11515.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.