Selective Growing Q-Networks (SGQN): Value-Guided Adaptive Discretisation for Continuous Control

Anonymous authors
Paper under double-blind review

Abstract

One way to tackle reinforcement learning for continuous control is to discretise the action space and apply value-based Q-learning, where the resolution of that discretisation fundamentally shapes the exploration–exploitation trade-off. Coarse discretisations facilitate early exploration but limit final precision; fine discretisations enable precise control but slow convergence. Existing adaptive approaches such as Growing Q-Networks (GQN) address this by growing uniformly from coarse to fine resolution, agnostic to which regions of the action space are actually promising. We introduce **Selective Growing Q-Networks (SGQN)**, an algorithm that replaces uniform growth with value-guided selective expansion: bins are refined only in action regions where Q-values are high, while persistently low-value bins are lazily pruned. During action selection, a temperature-annealed softmax over active bins concentrates probability mass on high-value regions as learning matures. SGQN maintains the scalability of decoupled Q-learning while focusing representational capacity where it matters most. On six tasks from the DeepMind Control Suite spanning 2D to 12D action spaces, SGQN achieves competitive or superior performance to strong baselines. These results suggest that value-guided selective discretisation is a competitive approach to continuous control, though per-component attribution of the performance gains requires further ablation.

Keywords: reinforcement learning, continuous control, action space discretisation, Q-learning, adaptive resolution, value decomposition

1 Introduction

Reinforcement learning for continuous control is commonly approached through actor-critic methods, which directly parameterise a policy over a continuous action space (Lillicrap et al., 2016; Haarnoja et al., 2018; Fujimoto et al., 2018; Schulman et al., 2017). While effective, these methods introduce substantial complexity: heuristics for training stabilisation, increased compute requirements, and wide hyperparameter search spaces (Seyde et al., 2022). An alternative is to instead focus on action space discretisation: a natural question is whether Q-learning methods, which avoid explicit policy parameterisation, can achieve comparable performance when equipped with a suitable discretisation of the action space.

Recent work has shown that this is indeed possible for action space discretisation. Seyde et al. (2021) showed that Bernoulli (bang-bang) policies are competitive on many locomotion tasks, drawing on Pontryagin's maximum principle to explain why boundary actions are frequently near-optimal. Building on this, Seyde et al. (2022) demonstrated that Decoupled Q-Networks (DecQN), which decompose the Q-function across action dimensions and apply bang-bang two-bin discretisation, can match state-of-the-art actor-critic performance on standard benchmarks. Seyde et al. (2024) extended this line of work with Growing Q-Networks (GQN), which progressively refine a uniform discretisation from coarse to fine resolution over the course of training, achieving smooth control without sacrificing early exploration.

Despite this progress, a fundamental inefficiency remains: all existing adaptive discretisation methods treat every action dimension and every bin equally at each resolution level. When GQN increases the action resolution, it activates new bins *uniformly* across the entire action range, regardless of which regions have

been visited, which yield high returns, or which the learned Q-function considers promising. This wastes representational capacity on regions that the agent has already determined to be suboptimal.

We propose **Selective Growing Q-Networks (SGQN)**, which addresses this inefficiency through three complementary mechanisms:

1. **Value-guided selective growth.** Rather than activating new bins uniformly at growth steps, SGQN identifies bins with high Q-values and activates their neighbours, concentrating new resolution in promising action regions.
2. **Lazy pruning.** Bins that have been explored but consistently yield low Q-values and receive few visits are deactivated. This keeps the effective action space compact and focused throughout training.
3. **Temperature-annealed softmax selection.** During action selection, probabilities over active bins are weighted by Q-values through a softmax with a decaying temperature, smoothly transitioning from uniform exploration to value-concentrated exploitation.

Together, these mechanisms implement a form of *resource allocation* in action space: the algorithm invests representational and sampling resources where the learned value function suggests they will be most useful. SGQN retains full compatibility with the decoupled Q-learning framework of DecQN and GQN (Seyde et al., 2022; 2024), benefiting from their linear-time complexity in the number of action dimensions and bins, and uses standard off-policy training with prioritised experience replay and double Q-learning.

We evaluate SGQN on six tasks from the DeepMind Control Suite (Tassa et al., 2018) spanning 2D to 12D action spaces. SGQN achieves comparable or superior performance on the majority of tasks, demonstrating that value-guided selective discretisation is a viable and competitive approach to continuous control.

Contributions. This paper makes the following contributions:

1. We identify a structural inefficiency in existing adaptive discretisation methods for continuous control — uniform growth is agnostic to learned value information — and propose value-guided selective growth and lazy pruning as a conceptually motivated remedy.
2. We introduce SGQN, a practical algorithm that integrates selective growth, lazy pruning, and temperature-annealed softmax selection within the decoupled Q-learning framework, and evaluate it empirically against five baselines on six DMControl tasks.

2 Related Work

2.1 Value-Based Methods for Continuous Control

Q-learning (Watkins & Dayan, 1992) and its deep extension DQN (Mnih et al., 2015) are natively suited to discrete action spaces. Extending them to continuous actions requires either approximating the argmax over a continuous domain or discretising the space. Gu et al. (2016) proposed Normalised Advantage Functions (NAF), which represent the Q-function in a form that admits closed-form continuous maximisation. Tavakoli et al. (2018) introduced action branching, treating each action dimension as an independent head of a shared network, enabling Q-learning in factored multi-dimensional spaces with linear rather than exponential complexity.

Tang & Agrawal (2020) showed that naive uniform discretisation of continuous action spaces can match policy gradient methods on several tasks when combined with standard Q-learning improvements, challenging the assumption that continuous action representations are necessary. Building on this, Seyde et al. (2022) proposed DecQN, combining bang-bang (two-bin) discretisation with full value decomposition across dimensions. By framing each action dimension as a cooperative agent in a multi-agent RL formulation (Sunehag et al., 2018; Rashid et al., 2018), DecQN achieves linear-time complexity in the number of action dimensions d and bins n per dimension, and competitive performance with SAC and TD3 on standard benchmarks.

2.2 Adaptive Action Space Resolution

The trade-off between exploration efficiency and precision is well documented in continuous control. Coarse discretisations reduce coordination complexity and facilitate exploration (Seyde et al., 2022), while fine discretisations enable precise steady-state behaviour. Several approaches have aimed to get the best of both worlds.

Seyde et al. (2024) proposed GQN, which grows the action space resolution from coarse to fine based on a stagnation criterion: growth is triggered when the moving-average return falls below a threshold $G_{\text{threshold},t} = (1 - 0.05\,\text{sign}(\mu_{\text{MA}}))\,\mu_{\text{MA}} + 0.90\,\sigma_{\text{MA}}$, where μ_{MA} and σ_{MA} are the moving-average mean and standard deviation of recent returns. Growth follows a predefined sequence (e.g., $2 \to 3 \to 5 \to 9$ bins), activating new bins uniformly and warm-starting their Q-values through the shared network trunk.

Seo et al. (2024) proposed Coarse-to-Fine Q-Networks (CQN), which perform hierarchical refinement *within each action selection*: at each of L levels, the agent selects a bin from a B-bin discretisation of the current interval, then zooms into that bin for the next level. CQN achieves resolution B^L with only $L \times B$ Q-value evaluations, but errors compound across levels and early mistakes cannot be corrected.

Our work differs from both GQN and CQN. Relative to GQN, SGQN replaces uniform growth with value-guided selective expansion and adds lazy pruning, concentrating resources on promising regions rather than distributing them uniformly. Relative to CQN, SGQN maintains a persistent global discretisation that is refined over training, rather than performing per-step hierarchical refinement.

2.3 Adaptive Discretisation in Other Contexts

Adaptive discretisation has a long history in function approximation and Monte Carlo methods. In state space, adaptive tile coding (Sutton & Barto, 2018) allocates resolution where the value function varies most rapidly. In tree search, progressive widening (Coulom, 2007) grows the branching factor adaptively. In control, curriculum learning approaches (Czarnecki et al., 2018) start with easier task formulations and progressively increase difficulty. SGQN applies this philosophy specifically to the *action space discretisation* in deep Q-learning.

2.4 Bang-Bang and Boundary-Seeking Policies

Seyde et al. (2021) investigated the empirical observation that trained continuous control agents frequently saturate at action boundaries, drawing theoretical support from optimal control theory: Pontryagin's maximum principle implies that minimum-time controllers are bang-bang. This motivates the use of binary (Bernoulli) action discretisation, which they show achieves competitive performance on locomotion tasks despite its extreme coarseness. SGQN begins from this insight but allows resolution to grow selectively where the Q-function indicates that finer discrimination is valuable.

3 Background

3.1 Markov Decision Processes and Q-Learning

We consider an agent interacting with a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A} \in \mathbb{R}^d$ is the d-dimensional continuous action space, $\mathcal{P}(s' \mid s, a)$ is the transition kernel, $\mathcal{R}(s, a) \in \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The agent seeks a policy $\pi : \mathcal{S} \to \mathcal{A}$ maximising the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$.

The optimal action-value function satisfies the Bellman optimality equation:

$$Q^*(s, a) = \mathbb{E}\big[r + \gamma \max_{a'} Q^*(s', a') \mid s, a\big]. \tag{1}$$

Deep Q-Networks (Mnih et al., 2015) approximate Q^* with a neural network $Q(s, a; \theta)$, minimising the TD loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\big[(y_t - Q(s, a; \theta))^2\big], \tag{2}$$

where the target uses a periodically updated target network θ^-. Double DQN (van Hasselt et al., 2016) uses the online network to select actions but the target network to evaluate them, reducing overestimation bias:

$$a^* = \arg\max_{a'} Q(s_{t+1}, a'; \theta), \qquad y_t = r_t + \gamma\, Q(s_{t+1}, a^*; \theta^-). \tag{3}$$

3.2 Decoupled Q-Networks

For a d-dimensional action space discretised into n bins per dimension, the joint action space has size n^d, which is intractable for large d. Seyde et al. (2022) propose decomposing the Q-function across dimensions:

$$Q_{\text{total}}(s, \mathbf{a}) = \frac{1}{d} \sum_{i=1}^{d} Q_i(s, a^i), \tag{4}$$

where $Q_i(s, a^i)$ evaluates only dimension i of the action. Greedy action selection then decouples:

$$a^{*i} = \arg\max_{a^i \in \mathcal{A}^i} Q_i(s, a^i), \quad i = 1, \ldots, d, \tag{5}$$

reducing complexity from $O(n^d)$ to $O(d \times n)$. The decomposition corresponds to the Individual Global Max (IGM) property from cooperative multi-agent RL (Sunehag et al., 2018), which holds exactly when action dimensions are independent and approximately otherwise.

3.3 Growing Q-Networks

GQN (Seyde et al., 2024) maintains a binary active mask $m \in \{0,1\}^{d \times N}$ over a full grid of N bins per dimension, initially activating only $n_0 \ll N$ uniformly-spaced bins. Growth is triggered by a stagnation criterion and advances the active set through a predefined sequence $n_0 \to n_1 \to \cdots \to N$. All newly activated bins are warm-started via the shared network trunk, exploiting the implicit structure learned at coarser resolution. At each step, Q-values are computed and actions selected only over the currently active bins.

4 Selective Growing Q-Networks

SGQN builds on the DecQN and GQN frameworks, replacing uniform growth with three value-guided mechanisms, each corresponding to a distinct training phase (Figures 1–3).

4.1 Action Space Representation

For each action dimension $i \in \{1, \ldots, d\}$, we maintain a fixed full grid $\mathcal{B}^i = \{b_1^i, \ldots, b_N^i\}$ of N uniformly-spaced values covering $[a_{\min}^i, a_{\max}^i]$, and a binary active mask $m^i \in \{0,1\}^N$. Only bins j with $m_j^i = 1$ participate in action selection and Q-value computation. The effective active set for dimension i is $\mathcal{A}_{\text{active}}^i = \{b_j^i : m_j^i = 1\}$.

We initialise by activating n_0 uniformly-spaced bins per dimension:

$$j_k = \text{round}\left(\frac{(k-1)(N-1)}{n_0 - 1}\right), \quad k = 1, \ldots, n_0. \tag{6}$$

4.2 Phase 1: Uniform Exploration

During an initial warm-up of T_{warm} episodes, SGQN uses standard ε-greedy action selection uniformly over the active bins (Figure 1). This phase accumulates sufficient Q-value estimates before value-guided decisions are made.

4.3 Phase 2: Selective Growth and Lazy Pruning

Value-guided growth. Every Δ_g episodes after the warm-up, SGQN evaluates whether to grow each dimension. Let $\bar{Q}_j^i$ denote the running mean Q-value for bin j of dimension i For each dimension i, we identify

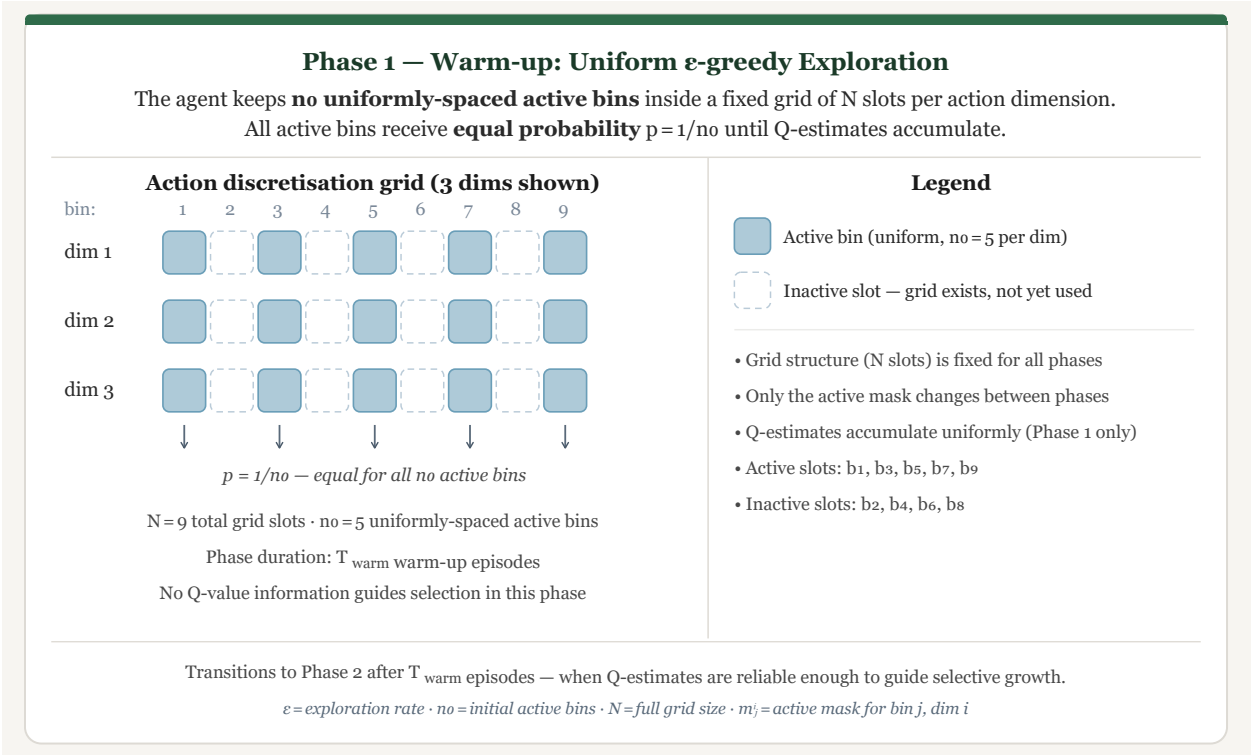


Figure 1: **Phase 1 (Warm-up).** The agent maintains $n_0 = 5$ uniformly-spaced active bins (light blue) within a fixed grid of $N = 9$ slots per action dimension; remaining slots exist in the grid but are inactive (dashed outline). All active bins receive equal selection probability $p = 1/n_0$ under ε-greedy exploration, allowing Q-value estimates to accumulate across the full resolution range before any value-guided decisions are made. *Notation:* n_0 – initial active bins; N – full grid size; ε – exploration rate; T_{warm} – warm-up duration.

the top-k active bins by Q-value:

$$\mathcal{H}^i = \{j : m_j^i = 1,\ \bar{Q}_j^i > \tau_Q \cdot \max_{j' : m_{j'}^i = 1} \bar{Q}_{j'}^i\}, \tag{7}$$

where $\tau_Q \in (0, 1)$ is a growth threshold (default 0.8). For each bin $j \in \mathcal{H}^i$, we activate its immediate neighbours $j \pm 1$ if they are not already active and the dimension has not yet reached its maximum resolution N:

$$m_{j'}^i \leftarrow 1 \quad \forall j' \in \{j-1, j+1\} \cap \{1, \ldots, N\},\ m_{j'}^i = 0. \tag{8}$$

Newly activated bins are warm-started through the shared Q-network trunk, inheriting the value structure learned at coarser resolution.

Lazy pruning. Every Δ_p episodes (with $\Delta_p > \Delta_g$), bins that have been persistently low-value and rarely visited are deactivated. Formally, bin j in dimension i is pruned if:

$$m_j^i = 1, \quad \bar{Q}_j^i < \tau_p \cdot \max_{j'} \bar{Q}_{j'}^i, \quad \text{count}_j^i < v_{\min}, \tag{9}$$

where $\tau_p \in (0, 1)$ is a pruning threshold (default 0.2), count_j^i is the number of times bin j was selected in the last W episodes, and $v_{\min}$ is the minimum visit count to trigger pruning (default 5). To preserve action space connectivity, we do not prune bins that are the sole remaining bin in their dimension, nor bins at the boundary of the active set if pruning would create an isolated active bin.

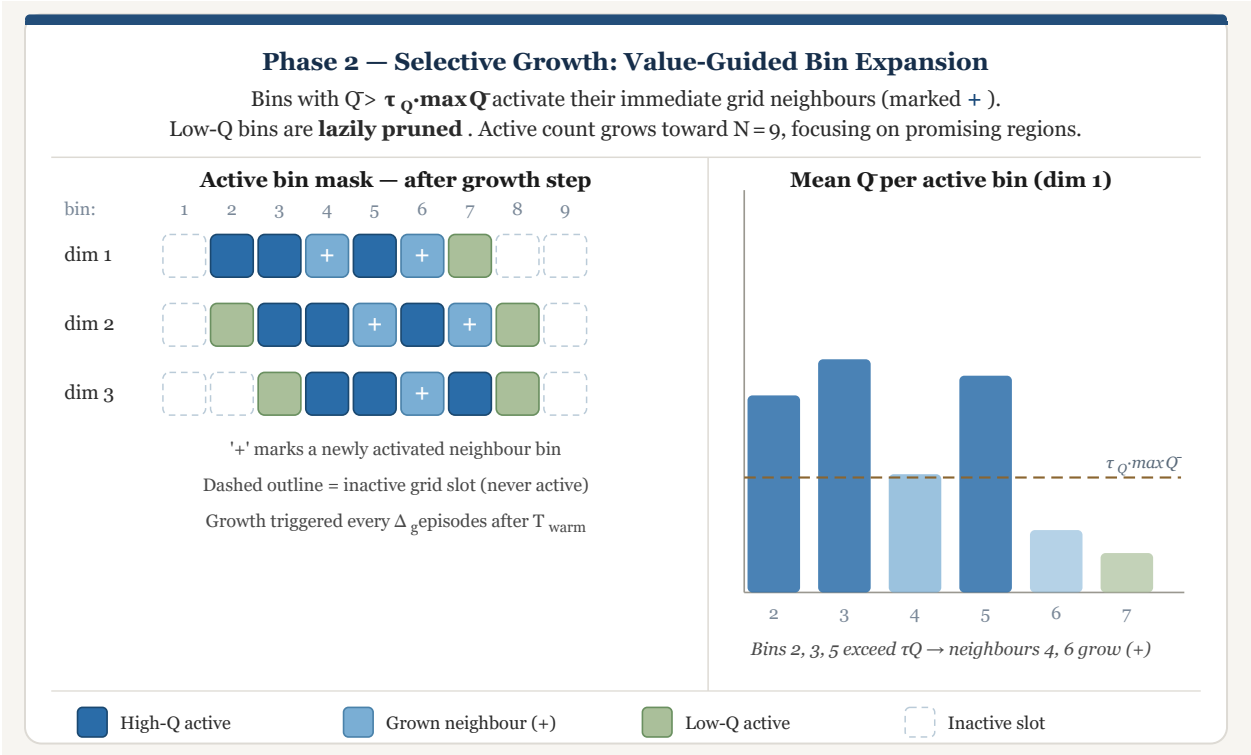


Figure 2: **Phase 2 (Selective growth).** Bins whose mean Q-value exceeds the growth threshold ($\bar{Q}^i_j > \tau_Q \cdot \max_{j'} \bar{Q}^i_{j'}$, dark blue) activate their immediate neighbours in the grid (medium blue, marked +). Bins with persistently low Q-values (green) remain active but are candidates for pruning; slots that have never been activated remain masked (dashed outline). The bar chart shows the mean Q-value $\bar{Q}^i_j$ per active bin for dimension 1; the dashed line marks the growth threshold $\tau_Q \cdot \max \bar{Q}$. *Notation:* $\bar{Q}^i_j$ – mean Q-value for bin j, dimension i; τ_Q – growth threshold; Δ_g – growth interval.

After a confidence threshold T_{conf} episodes, SGQN replaces ε-greedy with softmax selection over active bins. For dimension i in state s, the selection probability of active bin j is:

$$p^i_j(s) = \frac{\exp(Q_i(s, b^i_j)/\mathcal{T})}{\sum_{j':m^i_{j'}=1} \exp(Q_i(s, b^i_{j'})/\mathcal{T})}, \tag{10}$$

where the temperature $\mathcal{T}$ decays geometrically each episode:

$$\mathcal{T}_{t+1} = \max\big(\mathcal{T}_{\min},\ \mathcal{T}_t \cdot \alpha_{\mathcal{T}}\big), \tag{11}$$

with initial temperature $\mathcal{T}_0$, decay rate $\alpha_{\mathcal{T}}$, and minimum temperature $\mathcal{T}_{\min}$. As $\mathcal{T} \to 0$, the softmax recovers greedy selection; as $\mathcal{T} \to \infty$, it recovers uniform exploration. The annealing schedule thus implements a smooth, continuous transition from exploration to exploitation, replacing the discontinuous ε switch.

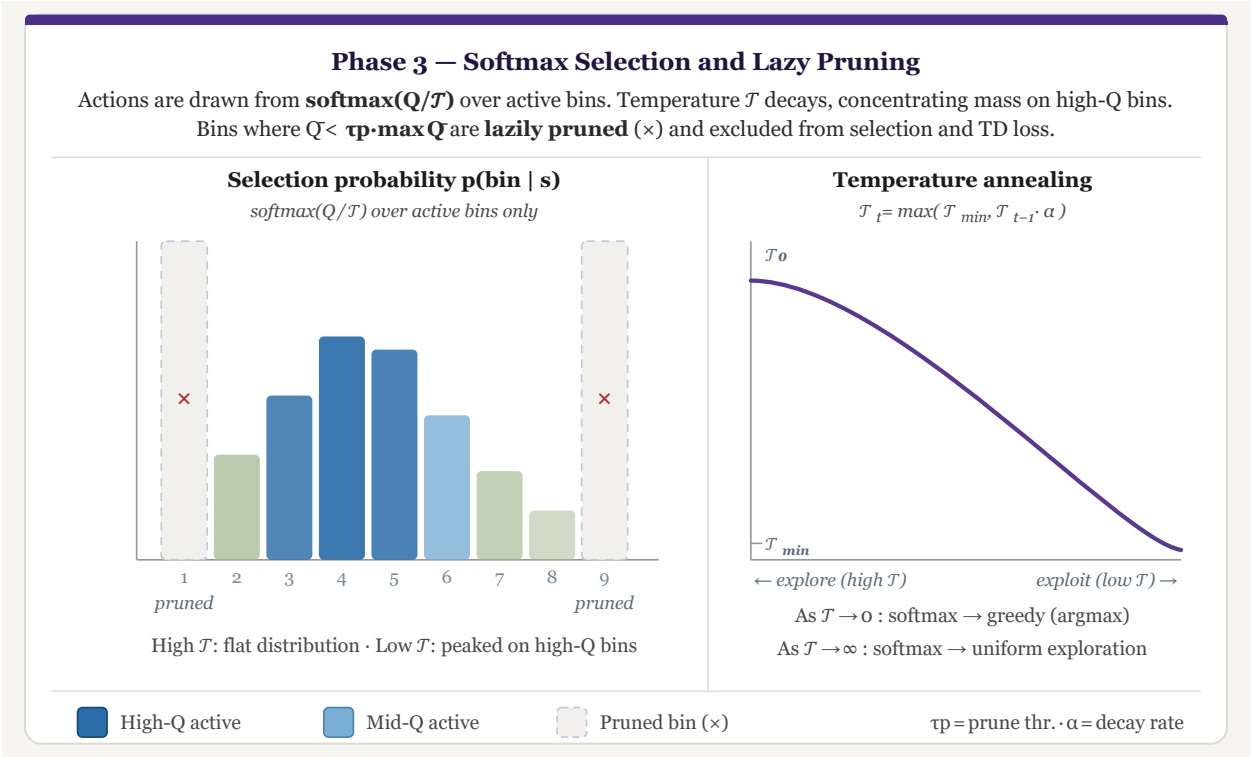


Figure 3: **Phase 3 (Softmax selection + lazy pruning).** The left panel shows the resulting selection probability $p(\text{bin} \mid s) = \text{softmax}(Q/\mathcal{T})$ over active bins: pruned bins (marked $\times$, red) are excluded entirely, while probability mass concentrates on high-Q bins as temperature $\mathcal{T}$ decreases. The right panel shows the geometric temperature decay $\mathcal{T}_t = \max(\mathcal{T}_{\min}, \mathcal{T}_{t-1} \cdot \alpha)$, transitioning smoothly from broad exploration to near-greedy exploitation. *Notation:* $\mathcal{T}$ – softmax temperature; α – decay rate; $\mathcal{T}_0$, $\mathcal{T}_{\min}$ – initial and minimum temperature; $\times$ – pruned bin.

4.5 Training

SGQN uses the same off-policy training infrastructure as DecQN and GQN. A prioritised experience replay buffer (Schaul et al., 2016) samples transitions with probability proportional to their TD error magnitude, with importance-sampling correction annealed over training (full details in Appendix A). The Q-network uses the decoupled architecture of DecQN: a shared trunk with layer normalisation (Ba et al., 2016) feeds per-dimension Q-value heads, each outputting N values (one per bin). Only the $|\mathcal{A}^i_{\text{active}}|$ active bins' outputs are used for selection and loss computation. n-step returns with $n = 3$ provide a temporal smoothing of the TD target. Gradient clipping with norm 40 stabilises training.

The full SGQN procedure is summarised in Algorithm 1.

4.6 Computational Complexity

Like GQN and DecQN, SGQN maintains $O(d \times N)$ Q-values in the network output layer. The growth and pruning operations scan the active mask in $O(d \times N)$ time per check, which is negligible relative to the neural network forward pass. The effective number of active bins per dimension at any time is $n_{\text{active}} \leq N$, and is typically far smaller during early and mid training, reducing the softmax computation in Eq. 10 and the TD loss computation accordingly.

Algorithm 1 Selective Growing Q-Networks (SGQN)

Require: Full grid $\mathcal{B}^i$ for each dimension i; initial bins n_0; growth/pruning intervals Δ_g, Δ_p; thresholds τ_Q, τ_p; temperature schedule $\mathcal{T}_0, \alpha_{\mathcal{T}}, \mathcal{T}_{\min}$

```
Initialise active masks m^i with n_0 uniform bins per dimension
Initialise Q-network θ, target network θ^- ← θ, replay buffer D
for episode = 1, 2, ... do
    Collect episode using ε-greedy (Phase 1) or softmax (Phase 3) over active bins
    Store transitions in D; update running Q-means Q̄^i_j and visit counts
    for each training step do
        Sample batch from D with prioritised replay
        Compute decoupled Q-loss (Eq. 4) over active bins only
        Update θ via gradient descent; periodically sync θ^- ← θ
    end for
    if episode > T_warm and episode mod Δ_g = 0 then
        Grow: activate neighbours of high-Q bins (Section 4.3)
    end if
    if episode > T_warm + Δ_p and episode mod Δ_p = 0 then
        Prune: deactivate low-Q, low-visit bins (Section 4.3)
    end if
    if episode > T_conf then
        Switch to softmax selection (Eq. 10); decay T
    end if
end for
```

5 Experiments

We evaluate SGQN on six tasks from the DeepMind Control Suite (Tassa et al., 2018), comparing against four baselines spanning the spectrum of discretisation strategies. All experiments use 7 random seeds; we report mean episode return and shaded ± 1 standard deviation.

5.1 Experimental Setup

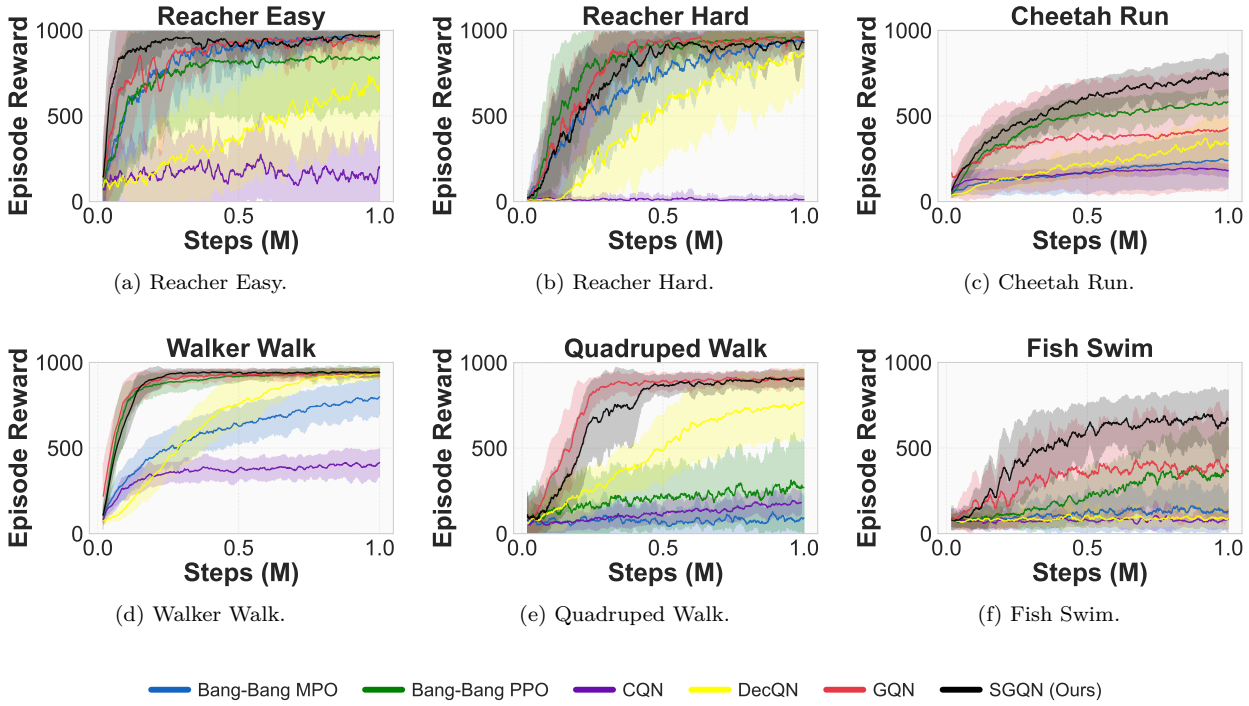
Tasks. We select tasks covering a range of action dimensionalities and control difficulties: **Reacher Easy** and **Reacher Hard** (2D action space), **Cheetah Run** and **Walker Walk** (6D), **Quadruped Walk** (12D), and **Fish Swim** (5D). This suite spans simple reaching tasks, locomotion tasks of varying difficulty, and a high-dimensional control task that stresses action space scaling.

Baselines.

- **Bang-Bang PPO / MPO**: Bernoulli policies (Seyde et al., 2021) trained with PPO (Schulman et al., 2017) and MPO (Abdolmaleki et al., 2018) respectively, using binary $\{-1, +1\}$ discretisation.
- **DecQN**: Fixed 2-bin decoupled Q-learning (Seyde et al., 2022).
- **GQN**: Uniform adaptive growth from 2 to 9 bins (Seyde et al., 2024).
- **CQN**: Hierarchical coarse-to-fine Q-learning, 3 levels $\times$ 5 bins (Seo et al., 2024). We use the hyperparameters from the original paper; however, our CQN results are substantially below those reported by Seo et al. (2024) on several tasks, most notably Reacher Hard. We attribute this to differences in the evaluation protocol (episode length, observation normalisation) rather than an implementation error, but acknowledge that a direct comparison with the original CQN results may not be appropriate. We include CQN as a reference point rather than a definitive benchmark.

Hyperparameters. All methods share a common backbone: layer normalisation, ELU activations, two hidden layers of 512 units, Adam optimiser with learning rate 10^{-4}, discount $\gamma = 0.99$, replay buffer size 10^6, batch size 256, 3-step returns, target network hard update every 100 steps, prioritised replay with $\alpha = 0.6$, β annealed from 0.4 to 1.0. SGQN-specific hyperparameters are listed in Table 2 (Appendix A).

5.2 Main Results



(a) Reacher Easy. (b) Reacher Hard. (c) Cheetah Run.

(d) Walker Walk. (e) Quadruped Walk. (f) Fish Swim.

Figure 4: Learning curves on all six DeepMind Control Suite tasks (mean ± std over 7 seeds, 1M environment steps). SGQN consistently matches or outperforms all baselines across 2D to 12D action spaces.

Table 1 reports final episode return at 1M steps.

Table 1: Episode return at 1M environment steps, read from learning curves (mean ± std, 7 seeds). **Bold** = highest mean per task; underline = second highest. CQN results are lower than those reported in Seo et al. (2024); see Section 5 for discussion.

Method	Reacher Easy	Reacher Hard	Cheetah	Walker	Quadruped	Fish
BB-PPO	830 ± 200	$\mathbf{975} \pm 60$	580 ± 100	940 ± 30	300 ± 100	340 ± 150
BB-MPO	200 ± 150	875 ± 100	240 ± 120	800 ± 80	130 ± 50	130 ± 50
DecQN	650 ± 250	820 ± 200	300 ± 80	940 ± 50	640 ± 150	100 ± 40
GQN	985 ± 15	960 ± 60	430 ± 100	950 ± 30	$\mathbf{900} \pm 50$	430 ± 200
CQN	200 ± 100	30 ± 20	170 ± 80	390 ± 80	150 ± 80	80 ± 60
SGQN	$\mathbf{995} \pm 15$	960 ± 60	$\mathbf{780} \pm 80$	$\mathbf{975} \pm 30$	870 ± 50	$\mathbf{680} \pm 200$

Reacher tasks (2D). SGQN (995) leads on Reacher Easy ahead of GQN (985). On Reacher Hard, BB-PPO achieves the highest return (975), with SGQN and GQN both at 960. The low action dimensionality means uniform discretisation is already tractable; all methods except CQN perform competitively (Figure 4a–4b).

Locomotion tasks (6D). On Cheetah Run, SGQN achieves 780, ahead of BB-PPO (580) and GQN (430). On Walker Walk, SGQN (975) is ahead of GQN (950) and both BB methods (940). BB-PPO is competitive on Walker Walk, where boundary actions align with minimum-time locomotion, but underperforms on Cheetah (Figure 4c–4d).

High-dimensional control (12D). Quadruped Walk is the most demanding task. GQN achieves the highest return (900), with SGQN second (870) and DecQN third (640). CQN (150), BB-PPO (300), and BB-MPO (130) perform poorly, consistent with difficulty scaling to 12D (Figure 4e).

Smooth control (5D). Fish Swim requires coordinated fin control. SGQN (680) achieves the highest return, ahead of GQN (430) and BB-PPO (340). BB-MPO (130) and DecQN (100) perform poorly, consistent with the difficulty of bang-bang and fixed-resolution approaches on smooth 5D control (Figure 4f).

6 Discussion

Why selective growth may help. In a decoupled Q-function, each action dimension's Q-values are hypothesised to form a unimodal or low-modal landscape around the optimal action. If so, activating new bins uniformly wastes resolution in regions already assigned low value, while selective growth concentrates new bins near Q-value peaks where finer discrimination is most likely to improve the greedy action. This intuition motivates the design, though direct ablation evidence is left to future work.

The role of lazy pruning. Pruning is designed to remove bins that have been explored but found uninformative, preventing them from polluting the softmax distribution or consuming replay buffer samples. Whether this is empirically beneficial relative to a system without pruning is a question for ablation studies; the motivation here is conceptual.

Pruning criterion and visit counts. The pruning condition requires a bin to be both low-value *and* infrequently visited (count $< v_{\min}$). Under softmax selection, low-Q bins naturally receive fewer visits, so one might ask whether the visit-count condition is redundant. The purpose is epistemic, not behavioural: a bin that has genuinely low Q-value but has been visited very few times may have an unreliable Q-estimate. Requiring a minimum visit count ensures that pruning is applied only to bins whose low value has been confirmed by sufficient experience, not to bins that happen to have low initial estimates simply because they have rarely been tried. This prevents premature pruning during early training when Q-estimates are noisy.

Why not start with softmax from the beginning? One could use a high-temperature softmax throughout, dispensing with Phase 1's ε-greedy entirely. The reason for the phased structure is stability: early in training, Q-values are near-random and a softmax over them would assign probability mass almost uniformly regardless of temperature, providing little benefit over ε-greedy but adding the cost of computing softmax probabilities over all bins. More importantly, Phase 1 deliberately collects diverse experience before the value-guided mechanisms (growth, pruning, annealing) activate. Starting softmax too early risks locking in misleading probability mass before reliable Q-estimates exist. That said, this design choice is worth revisiting experimentally for settings where warm-up is too costly.

Limitations. SGQN introduces additional hyperparameters compared to GQN (τ_Q, τ_p, and the temperature schedule) and an extended phase structure. While our defaults worked well across all tested tasks, hyperparameter sensitivity in novel environments remains a concern. The independence assumption of decoupled Q-learning (Eq. 4) can also be limiting for tasks with strongly coupled action dimensions, such as underactuated systems where joint torques must be precisely coordinated.

Component design rationale. SGQN integrates three mechanisms — selective growth, lazy pruning, and temperature-annealed softmax — whose individual contributions are not separately quantified in this work. The design choices are motivated by the arguments in the preceding paragraphs: selective growth targets new resolution where Q-values are highest; pruning removes confirmed low-value bins; annealing smoothly transitions from exploration to exploitation. A systematic per-component ablation is left to future work.

Future directions. Several extensions are promising: (1) learning per-dimension growth rates, rather than using a shared schedule; (2) extending selective growth to the state space, growing the resolution of state representations in high-value regions; (3) combining selective discretisation with model-based planning; (4) applying SGQN to real robotic systems where sample efficiency and discrete actuation align naturally.

7 Conclusion

We introduced Selective Growing Q-Networks (SGQN), a value-based continuous control algorithm that addresses the inefficiency of uniform action space growth by directing new representational capacity toward high-Q action regions and removing it from persistently low-Q regions. SGQN achieves competitive or superior performance to strong baselines across six DeepMind Control Suite tasks, demonstrating that value-guided selective discretisation is a viable approach to continuous control.

More broadly, these results are consistent with the hypothesis that value-based methods with selective discretisation can be competitive with actor-critic methods for continuous control. The guiding principle — that discretisation resolution should be concentrated where the value function indicates it matters most — is simple and practically motivated, and warrants further investigation including per-component ablation.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Rémi Coulom. Computing elo ratings of move patterns in the game of go. In *Computer Games Workshop*, 2007.

Wojciech M Czarnecki, Siddhant M Jayakumar, Max Jaderberg, Leonard Hasenclever, Yee Whye Teh, Nicolas Heess, Simon Osindero, and Razvan Pascanu. Mix and match – agent curricula for reinforcement learning. In *International Conference on Machine Learning*, 2018.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, 2016.

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*, 2018.

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. In *International Conference on Learning Representations*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for decentralised multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Younggyo Seo, Jafar Uruç, and Stephen James. Continuous control with coarse-to-fine reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024.

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. In *Advances in Neural Information Processing Systems*, 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.

Tim Seyde, Peter Werner, Wilko Schwarting, Markus Wulfmeier, and Daniela Rus. Growing Q-networks: Solving continuous control tasks with adaptive control resolution. In *International Conference on Learning Representations*, 2024.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. In *AAAI Conference on Artificial Intelligence*, 2020.

Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*, 2016.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

A Full Hyperparameter Tables

Table 2: SGQN-specific hyperparameters.

Hyperparameter	Value
Full grid bins N	9
Initial bins n_0	5 (uniform)
Growth interval Δ_g	50 episodes
Pruning interval Δ_p	100 episodes
Growth threshold τ_Q	0.8
Pruning threshold τ_p	0.2
Minimum visit count $v_{\min}$	5 (over last 50 episodes)
Warm-up T_{warm}	50 episodes
Confidence threshold T_{conf}	50 episodes
Initial temperature $\mathcal{T}_0$	10.0
Temperature decay $\alpha_{\mathcal{T}}$	0.995
Minimum temperature $\mathcal{T}_{\min}$	0.5
Epsilon (Phase 1)	$0.2 \rightarrow 0.05$

Table 3: Shared hyperparameters for all value-based methods.

Hyperparameter	Value
Optimiser	Adam
Learning rate	1×10^{-4}
Discount γ	0.99
Replay buffer size	10^6
Batch size	256
n-step returns	3
Target update period	100 steps
Prioritised replay α	0.6
Prioritised replay β	$0.4 \rightarrow 1.0$
Gradient clip norm	40.0
Hidden layers	2×512
Activation	ELU
Normalisation	Layer norm

B Environment Details

All experiments use the standard DMControl state-based observation protocol with no frame stacking. Episodes run for 1000 steps; we report mean episode return averaged over the full episode. The 7 random seeds control both the environment initialisation and the neural network initialisation. No domain randomisation is applied.

Table 4: DeepMind Control Suite task details.

Task	Action Dim.	State Dim.
Reacher Easy	2	7
Reacher Hard	2	7
Cheetah Run	6	17
Walker Walk	6	24
Fish Swim	5	26
Quadruped Walk	12	78