DYNAMIC OPTIMISATION OF DISCOUNT AND TRACE DECAY FOR VALUE-BASED RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Hyperparameter optimisation remains a central challenge in reinforcement learning (RL), particularly for the discount factor and trace-decay which directly influence objective estimation and the bias–variance trade-off in learning targets. Given the high computational cost of RL, it is desirable to dynamically adapt hyperparameters within a single training run, thereby avoiding the expense and sample inefficiency of hyperparameter optimisation over multiple independent runs. Recent approaches have explored dynamic hyperparameter adaptation during training; however, they often suffer from brittleness, limited applicability across algorithms, significant computational overhead, or costly policy evaluations. In this work, we propose a unified framework for Dynamical Optimisation Of Discount-factor And Trace-decay (DOODAT). DOODAT is simple, reliable, broadly applicable across algorithms, and incurs minimal computational and sample overhead. We evaluate against state-of-the-art meta-gradient methods and standard non-adaptive baselines on PQN and PPO across multiple environment suites. DOODAT consistently improves performance and exhibits strong robustness. Notably, it recovers high performance even when initialised with suboptimal hyperparameters, far from commonly used values, achieving an IQM of 1.0 on the ALE, whereas static training or baseline methods achieve no learning at all.

1 INTRODUCTION

Reinforcement learning (RL) has been applied successfully to challenging problems across a broad range of domains, including games, scientific discovery, and control (Hafner et al., 2025; Kaufmann et al., 2023; Degrave et al., 2022; Badia et al., 2020; Mnih et al., 2015). Despite successes, the performance of RL algorithms remains brittle, exhibiting high sensitivity to hyperparameters (Obando-Ceron et al., 2024; Eimer et al., 2023; Andrychowicz et al., 2020; Henderson et al., 2017).

Modern RL algorithms are typically grounded in the generalised policy iteration framework, which alternates between policy evaluation and policy improvement (Sutton & Barto, 2018). Within this framework, value functions estimate the expected sum of discounted future rewards and provide the basis for improving the policy. The discount factor γ determines the effective planning horizon by weighting immediate versus future rewards, while the trace-decay parameter λ governs the bias–variance trade-off in return estimation (e.g. via eligibility traces or advantage estimators). Both hyperparameters have been empirically observed to substantially affect learning stability (White & White, 2016; Kearns & Singh, 2000), sample efficiency (Schwarzer et al., 2023; Amit et al., 2020; Schulman et al., 2016), and final performance (Andrychowicz et al., 2020; Tsitsiklis & Van Roy, 1997), and their optimal values are known to vary not only across tasks, but also over the course of training (Mohan et al., 2023; Kearns & Singh, 2000). In particular, Andrychowicz et al. (2020) and Obando-Ceron et al. (2024) have identified the discount factor as one of the most important and environment-sensitive hyperparameters in their studies, recommending per-task tuning.

Given the computational cost and sample complexity of RL, exhaustive per-task tuning is often infeasible; therefore, it is desirable to adapt hyperparameters dynamically within a single training run. In the context of Automated RL (AutoRL; (Parker-Holder et al., 2022)), several approaches have been proposed to achieve this. Among them, meta-gradient methods (Xu et al., 2018; Zahavy et al., 2020; Flennerhag et al., 2021) are the only ones that provide a general mechanism for online hyperparameter optimisation in both off-policy and on-policy settings without requiring separate policy

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

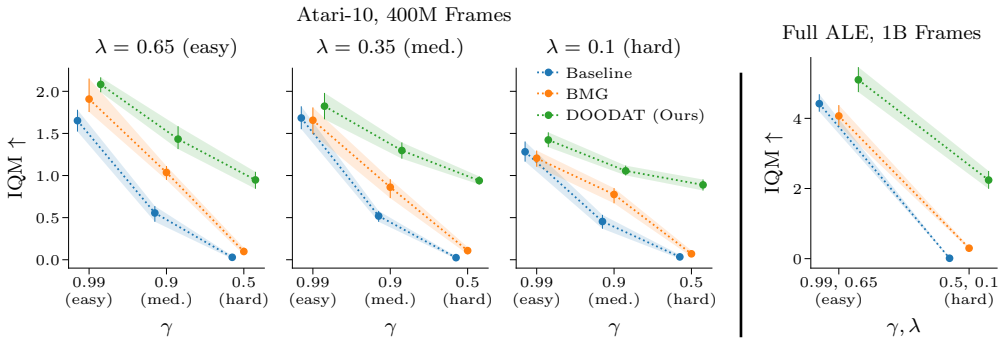

Figure 1: Comparison of our method DOODAT for dynamically tuning the discount factor γ and trace-decay λ for PQN, with static values (baseline) and BMG. Hyperparameter settings are ordered from left to right by increasing difficulty. Left corresponds to (tuned) baseline values. **Left:** On the Atari-10, DOODAT outperforms the baseline and BMG across initialisations. It reliably recovers an IQM of at least 1.0 even when initialising γ and λ far from optimal values. In contrast, neither the baseline nor BMG show any signs of learning for $\gamma = 0.5$. **Right:** Training with 1B environment frames on the full ALE suite, DOODAT still significantly outperforms the baseline and BMG. BMG is not able to learn from hard initialisations and starts to show performance degradation when starting from baseline parameters for longer training. See Section 4.1 for the experimental setup.

evaluations or substantial additional updates. These methods use higher-order gradients by differentiating an outer objective with respect to hyperparameters throughout the optimisation process.

However, meta-gradient approaches exhibit several limitations. Firstly, the outer objective for off-policy learning is based on temporal-difference (TD) errors and thus provides only an indirect proxy for the goal of improved policy performance. Secondly, due to their recurrent dependencies, meta-gradients can become increasingly noisy and unstable when propagated through multiple optimisation steps (Metz et al., 2019; 2022). Consequently, practical implementations often truncate the meta-gradient horizon, resulting in myopic hyperparameter adaptation. Furthermore, this instability makes applying meta-gradients difficult for epoch-based RL algorithms, including widely adopted off- and on-policy methods such as PQN (Gallici et al., 2025) and PPO (Schulman et al., 2017).

To address these limitations, we propose a method for Dynamic Optimisation Of Discount-factor And Trace-decay (DOODAT) within a single training run, applicable to both off-policy and on-policy algorithms. Fundamentally, our approach takes inspiration from the policy improvement theorem; however, rather than necessitating strictly greater action-values across all states, we relax the criterion to prefer policies that yield a higher average expected return under a shared value function. To implement this, we augment the value and policy function with auxiliary heads trained using slightly increased and decreased values of γ and λ. We then compare the performance of the different policies associated with the perturbed hyperparameter values under the value function of the current γ and λ values, moving in the direction of the best-performing ones. This avoids back-propagation through optimisation trajectories and relies on observable differences in policy quality by hyperparameter perturbations, thereby circumventing the myopia of meta-gradients. Moreover, DOODAT requires only minimal modifications to standard RL pipelines, adds negligible computational overhead and little additional sample cost. As a result, DOODAT occupies a distinct point in the AutoRL design space by combining continuous within-run adaptation, low computational overhead, and applicability to value-based algorithms, whereas existing approaches typically lack one or more of these properties. We provide a detailed comparison with related AutoRL methods in Appendix A. Figure 1 depicts the results of DOODAT applied to PQN on the Arcade Learning Environment suite (ALE; (Bellemare et al., 2013)); further details can be found in Section 4.2. Our new method outperforms both static tuning and bootstrapped meta-gradients (BMG; (Flennerhag et al., 2021)) across initialisation regimes of varying difficulty.

Our implementations and instructions for reproducing our results can be found in the repository `https://anonymous.4open.science/r/doodat_submission-29D3`

2 BACKGROUND

We consider the standard RL formulation, wherein an agent interacts with an environment modelled as a Markov Decision Process (MDP) defined by $\mathcal{M} := (\mathcal{S}, \mathcal{A}, p, r, \rho_0)$ (Sutton & Barto, 2018). At each time step t, the agent observes a state $s_t \in \mathcal{S}$, executes an action $a_t \in \mathcal{A}$ sampled from a policy $\pi(a \mid s)$, and transitions to a state $s_{t+1} \sim p(\cdot \mid s_t, a_t)$, receiving a scalar reward r_{t+1}. The objective is to find a policy maximising the expected cumulative return from a starting state distribution.

Value-based RL algorithms approach this objective by explicitly learning parameterised value functions to estimate expected discounted returns, subsequently used for policy improvement. The discount factor $\gamma \in [0, 1)$ is not treated as an intrinsic property of the underlying MDP, but rather as an algorithmic hyperparameter that is commonly tuned to establish the effective planning horizon and stabilise training. The state-value function under a specific policy π is defined as $v_\pi(s) := \mathbb{E}_\pi\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\big]$. Analogously, the action-value function is defined as $q_\pi(s, a) := \mathbb{E}_\pi\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\big]$. Within deep RL, these functions are approximated using neural networks and optimised via TD learning.

By controlling the relative importance of immediate versus distant rewards, the tuned value of γ influences the convergence properties and the scale of the value targets. To construct these targets, algorithms frequently rely on the n-step return, which mitigates the bias of one-step bootstrapping by accumulating observed rewards over a finite trajectory horizon before querying the value approximator: $g_{t:t+n} := \sum_{k=0}^{n-1} \gamma^k r_{t+k+1} + \gamma^n v_\pi(s_{t+n})$. A generalisation of this concept is achieved through the eligibility trace-decay parameter, $\lambda \in [0, 1)$, resulting in a λ-return. The λ-return computes an exponentially weighted average of all n-step returns, defined as $g_t^\lambda := (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} g_{t:t+n}$, and gives a bias–variance trade-off within the target estimation.

3 METHOD

We describe DOODAT for a given hyperparameter pair $\chi \in \{\gamma, \lambda\}$. Figure 2 gives an overview of DOODAT applied to PQN. In Appendix F, we present pseudo-code for PQN and PPO.

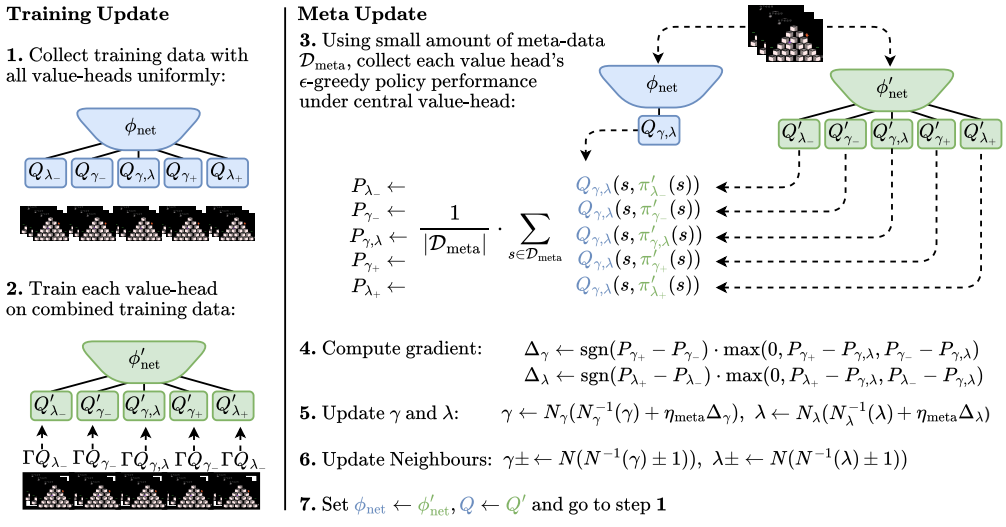

Figure 2: Overview of DOODAT for jointly tuning the discount factor γ and trace-decay λ for PQN (Gallici et al., 2025). **Left:** multi-head training with shared data and separate targets for each hyperparameter setting. **Right**: meta-update based on comparing the policies induced by each head on a held-out meta-dataset. Hyperparameters are adjusted towards better-performing neighbours using the central value function as the common evaluation objective.

3.1 MULTI-HEAD VALUE ARCHITECTURE

To enable dynamic adaptation without parallel network updates, we maintain multiple value and policy predictions in parallel. Let $\phi(s)$ denote the shared representation produced by a neural network, i.e. the output of the encoder layers. For the central hyperparameter χ, we define two perturbed neighbours, $\chi_+ \leftarrow N_\chi(N_\chi^{-1}(\chi) + 1)$ and $\chi_- \leftarrow N_\chi(N_\chi^{-1}(\chi) - 1)$, via a neighbourhood mapping function N_χ (detailed in Section 3.4).

DOODAT maintains three distinct value heads, each comprised of a single linear layer, instantiated as either action-value functions $Q(\phi(s), a)$ or state-value functions $V(\phi(s))$, depending on the underlying algorithm. For actor-critic methods, we introduce an analogous set of three distinct policy heads $\pi(\phi(s))$ that branch from a shared policy representation. During data collection, all candidate policies are deployed across an equal share of the parallel environments. The heads are trained simultaneously on the aggregated data, differing only in the hyperparameter configuration, χ, χ_+, or χ_-, used to construct their respective training objectives. In the on-policy setting, each policy head is optimised during training, using advantage estimates from its corresponding state-value head, and off-policy discrepancies are handled via importance sampling. The feature representations ϕ are trained with the summed losses of all heads. This design integrates into both off- and on-policy algorithms with negligible computational overhead (a running time comparison is in Appendix H).

3.2 UNIFIED META-OBJECTIVE FOR POLICY COMPARISON

To adapt χ, we compare the performance of the policies associated with each hyperparameter value, denoted as π_χ, π_{χ_+}, and π_{χ_-}. The core principle of DOODAT relies on policy improvement: a candidate policy is considered superior if it yields a higher expected return when evaluated under a shared baseline. We implement this by evaluating all candidate policies against the central value function. We define a performance estimator $P(\pi_{\chi'})$ over a small meta-dataset $\mathcal{D}_{\text{meta}}$, collected during training with the central policy π_χ, similar to meta-gradient methods.

For value-based, off-policy methods, the estimator is computed directly using the central action-value function Q_χ^{old} before the last update step:

$$P(\pi_{\chi'}) = \frac{1}{|\mathcal{D}_{\text{meta}}|} \cdot \sum_{s_i \in \mathcal{D}_{\text{meta}}} Q_\chi^{\text{old}}\big(\phi^{\text{old}}(s_i), \pi_{\chi'}^{\text{new}}(\phi^{\text{new}}(s_i))\big) \tag{1}$$

For policy-gradient, on-policy methods, where explicit action-value functions are absent, we use a similar comparison using an importance-weighted objective. Utilising advantages A_χ^{old} computed under the central value function V_χ^{old} before the last update, the estimator is defined as:

$$P(\pi_{\chi'}) = \frac{1}{|\mathcal{D}_{\text{meta}}|} \cdot \sum_{(s_i,a_i) \in \mathcal{D}_{\text{meta}}} \frac{\pi_{\chi'}^{\text{new}}(a_i \mid \phi^{\text{new}}(s_i))}{\pi_\chi^{\text{old}}(a_i \mid \phi^{\text{old}}(s_i))} A_\chi^{\text{old}}(\phi^{\text{old}}(s_i), a_i) \tag{2}$$

3.3 HYPERPARAMETER META-UPDATE

Following the evaluation phase, the central hyperparameter χ is updated based on the relative performance of its neighbours. If the central policy π_χ achieves the maximum estimated performance P, χ remains unaltered. Conversely, if a perturbed policy shows better performance, χ is adjusted proportionally to the observed performance difference. For instance, if π_{χ_+} performs best, χ is increased by an amount proportional to $P(\pi_{\chi_+}) - P(\pi_\chi)$, resulting in χ_{new}. This functions as a finite-difference approximation of the performance gradient with respect to the hyperparameter. Furthermore, we perform linear interpolation between the network weights θ_χ of the central head and those of the outperforming auxiliary head $\theta_{\chi*}$. The magnitude of this interpolation is given by $|N_\chi^{-1}(\chi) - N_\chi^{-1}(\chi_{\text{new}})|$. This ensures that the central head assimilates the knowledge of the better-performing hyperparameter value. In practice, γ and λ are updated independently and concurrently.

3.4 NEIGHBOURHOOD FUNCTION

The neighbourhood mapping N_χ is designed to satisfy three criteria: (i) mapping $\mathbb{R}$ to $[0, 1)$, (ii) increasing numerical resolution as the value approaches 1, given that γ and λ operate within exponentially discounted sums, and (iii) providing meaningful perturbations for finite-difference updates.

We define the following clipped power-law function with $\alpha > 0$ controlling the resolution:

$$N_\chi(x) := 1 - \frac{1}{(\alpha \cdot \max(x, 0) + 1)^2}, \quad N_\chi^{-1}(x) := \begin{cases} 0, & x \leq 0, \\ \frac{1}{\alpha}\left(\frac{1}{\sqrt{1-x}} - 1\right), & 0 < x < 1, \\ 1, & x \geq 1. \end{cases} \tag{3}$$

Based on preliminary experimentation, we found $\alpha_\gamma = 2.0$ and $\alpha_\lambda = 0.2$ to be effective in practice. Figure 3 depicts the behaviour of both functions. By using a lower α value for the dynamic adjustment of λ, we achieve a more uniform distribution of values and thus better interpolate between one-step TD and Monte Carlo targets. While our function is not the only function that satisfies our requirements, we found it to be generally robust across experiments.

4 Experiments

Following Castro (2025), we thoroughly examined DOODAT on the ALE, specifically the Atari-10 environments (Aitchison et al., 2022), and then tested generalisation to different training regimes, algorithms and benchmarks. Training curves for all our experiments are presented in Appendix J.2 together with the respective dynamic schedules γ and λ of each training.

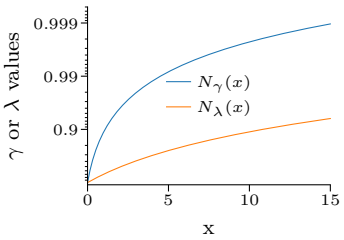

Figure 3: Neighbourhood functions used for γ and λ.

4.1 Experimental Setup

We applied DOODAT to the off-policy RL algorithm PQN (Gallici et al., 2025) and the on-policy algorithm PPO (Schulman et al., 2017). For PQN, we used the implementation in purejaxql (Gallici et al., 2025) and for PPO the purejaxrl implementation (Lu et al., 2022). Unless stated otherwise, all experiments were conducted using five independent pseudo-random number seeds for training. To aggregate results, we report the interquartile mean (IQM) and optimality gap (Agarwal et al., 2022). Individual task performance is reported using the mean performance over all seeds. Confidence intervals for IQM and task performance are always bootstrapped 95% confidence intervals. We benchmarked DOODAT against a static baseline with fixed γ and λ, and against dynamic updates with BMG. Following Flennerhag et al. (2021), we used the Adam optimiser (Kingma & Lei, 2015) for meta-optimisation of DOODAT and BMG. Both DOODAT and BMG use an additional 12.5% of meta-training data to compute meta updates.

To the best of our knowledge, so far, meta-gradients have not been applied to epoch-based RL algorithms such as PQN and PPO in the literature. We adapt BMG to this setting by performing a single meta-gradient update per epoch. Specifically, we backpropagate through the initial minibatch update and utilise the subsequent minibatch updates as bootstrapped targets for the meta-gradient computation. To ensure a fair comparison, the learning rates for both BMG and DOODAT were rigorously tuned (see Figure 5). Further implementation details for BMG are provided in Appendix G.

We evaluated configurations of varying difficulty by initialising γ and λ at different distances from their standard baseline settings. We refer to the initialisation with baseline values as *easy*, an intermediate initialisation as *medium* and the initialisation furthest from the baseline as *hard*. Evaluations are performed across the full ALE (Bellemare et al., 2013), Procgen (Cobbe et al., 2020), and the DeepMind Control Suite (Tassa et al., 2018). For the ALE, we considered a training regime that runs for 400M environment frames and an extended regime training for 1B frames. In the 1B training regime, we ran experiments only for three seeds, due to the substantially higher computational costs. Environment configurations are given in Appendix B and hyperparameter settings are detailed in Appendix C. The computational execution environment is given in Appendix D.

4.2 Feasibility of Optimisation

We investigated the extent to which DOODAT can dynamically learn hyperparameter schedules when starting from easy to hard initialisations. Figure 1 (left) illustrates performance across a grid of initial parameter values on Atari-10 Aitchison et al. (2022), ranging from $\gamma = 0.99$, $\lambda = 0.65$ (easy)

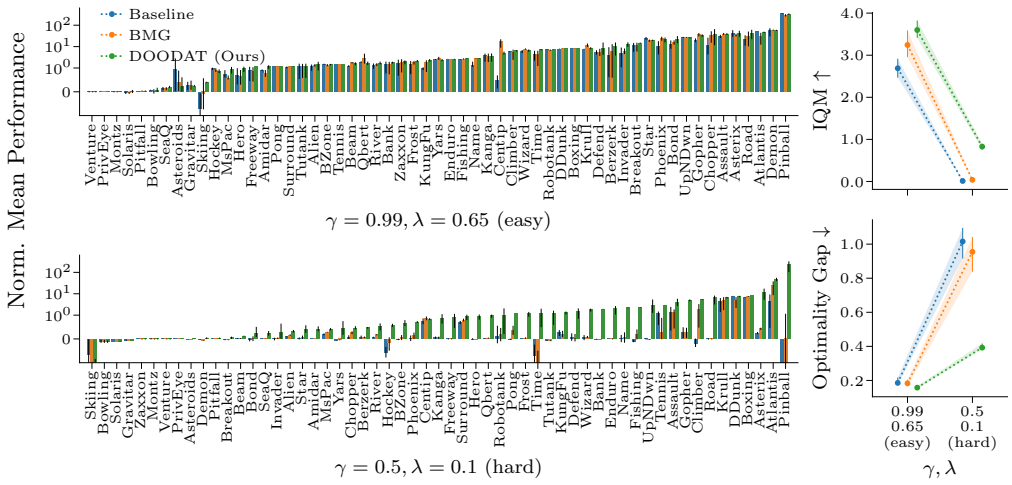

Figure 4: Comparison of DOODAT to static training (baseline) and BMG on the full ALE trained for 400M environment frames on PQN. Barplots showing per environment performance and pointplots aggregated performance. DOODAT achieves substantially improved IQM over both the baseline and BMG and in particular achieves good performance when starting from hard initialisations. In this case, the optimality gap of around 0.4 further shows this is the case across the suite. Additionally, DOODAT is very reliable, given that, except for two environments, it never suffers from substantially degraded (i.e. more than 20% reduced) performance. See Section 4.1 for the experimental setup.

to $\gamma = 0.5$, $\lambda = 0.1$ (hard). DOODAT consistently learns schedules that yield the highest IQM. Notably, with harder initialisations, where static training fails and BMG shows no improvement, DOODAT successfully recovers strong performance, achieving an IQM of at least 1.0. To assess whether DOODAT can further benefit from longer training, we evaluated it on the full ALE suite for 1B environment frames (Figure 1, right). DOODAT continues to achieve improved performance for this extended training, significantly outperforming the baseline in the easy regime. In contrast, we observe that BMG performs worse than the baseline in this regime, showing degraded performance in training regimes for which its meta-learning rate has not been tuned. Similarly, DOODAT recovers a high IQM of over 2.0 in the hard regime, where BMG still shows no sign of learning. Comparing DOODAT against static training using the γ and λ values it converges to (Appendix J.5), the two match in the hard regime, so gains stem from the found configuration, whereas in the easy regime, DOODAT achieves significantly better results, indicating benefit from the dynamic schedule itself.

4.3 Reliability of DOODAT

A practical dynamic tuning method must be reliable, never pushing hyperparameter settings into unstable regions and thus suffer from substantially degraded performance. We assessed this by evaluating DOODAT on the full ALE suite for the default 400M environment frames (Figure 4). For the easy initialisations, DOODAT consistently improves over or matches the performance of both the baseline and BMG. Importantly, our approach proves reliable; with the two sole exceptions of *Asteroids* and *StarGunner*, it never degrades performance more than 20% across the entire suite. Detailed results on the relative improvements of DOODAT and BMG are given in Appendix J.1. Furthermore, the results on the full suite mirror those on the Atari-10 subset when starting from suboptimal values ($\gamma = 0.5$, $\lambda = 0.1$). In particular, DOODAT achieves an optimality gap of around 0.4 when starting from poor initialisations, showing its robustness across the ALE suite.

We have argued in our introduction that directly assessing the performance of the policies resulting from perturbed hyperparameter values allows DOODAT to be more reliable compared to the myopic meta-gradient updates. We already observed this is the case when training for 1B environment frames (Figure 4 right), where DOODAT consistently improves, whereas BMG performance starts to degrade. To better understand this, we conducted a grid search across meta-learning rates in Figure 5. Whereas the performance of BMG quickly degrades when trained for larger meta-learning rates (e.g. 10 times increase), DOODAT shows much more benign behaviour. We further examined

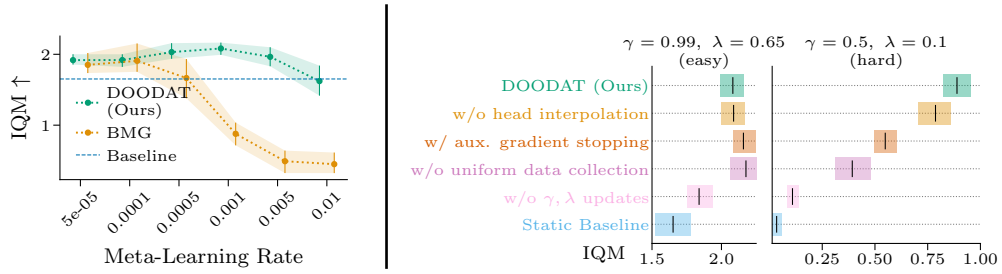

Figure 5: **Left:** Comparison of DOODAT and BMG on PQN under different meta-learning rates on Atari-10 when initialising γ and λ from baseline values. DOODAT exhibits greater robustness to larger meta-learning rates. **Right:** Ablations of DOODAT on PQN when initialised at $\gamma = 0.5$ and $\lambda = 0.1$. Gradient stopping refers to stopping the gradient of the auxiliary value-heads, thus using only the central value-head for learning the feature representations; 'w/o uniform' data collection refers to using only the central policy for collecting training data in contrast to using all value-heads on an equal share of parallel environments. See Section 4.1 for the experimental setup.

the accuracy of the central critic as a proxy for policy performance of full policy rollouts in Appendix E. The results show a consistent pattern that the central critic can reliably select the best-performing value head. This highlights the advantage of our look-ahead approach to better assess the actual performance of adjusted hyperparameter values, resulting in more reliable meta-updates.

4.4 Ablation Study

To evaluate the performance and stability of DOODAT, we ablated its components on the Atari-10 suite. Figure 5 (right) presents the results of these ablations when initialised from the easy (left) and hard (right) configurations. The hard configuration exhibits the clearest differences between variants. Removing head interpolation results in only a modest degradation (orange), whereas two ablations substantially reduce performance: gradient stopping after the auxiliary value heads (i.e. updating the shared feature representation exclusively via the central value head, red) and restricting data collection to the policy induced by the central value head, rather than utilising all policies uniformly with equal shares of the parallel environments (purple). In particular, removing either component may make it harder for the upper value head (γ_+) to learn accurate long-horizon value estimates from a shared representation trained predominantly on shorter-horizon objectives. In the easy domain, i.e. when already starting from well-performing configurations, the same ablations show much smaller effects. Existing literature establishes that collecting data under diverse policies improves off-policy robustness (Dabney et al., 2021) and that auxiliary tasks regularise latent state representations against collapse (Lyle et al., 2021). Moreover, predicting returns at varying discount factors through auxiliary value heads is an established method for improving representation learning (Fedus et al., 2019). Our ablation shows that utilising these heads strictly as auxiliary targets and for data collection, without dynamic tuning of γ and λ (pink), results in complete training failure for the hard initialisation and diminished improvement for the easy initialisation. Thus, the observed empirical gains are directly attributable to the hyperparameter optimisation.

4.5 Generalisation to Other Algorithms and Environments

To verify that DOODAT has not been overfitted to the ALE or the PQN algorithm, we tested its generalisation to the Procgen benchmark using PQN and the DeepMind Control suite using PPO.

PQN with Procgen: For Procgen, we replaced the Nature DQN network with an IMPALA architecture and trained for 200M frames, consistent with the literature. Figure 6 shows that DOODAT significantly outperforms the baseline when starting from standard as well as suboptimal parameter settings. However, we observe multiple Procgen environments (e.g. *Bigfish*) where no improvement is achieved for the hard initialisation regime. This is due to training instability, resulting in a feature-rank collapse, network plasticity loss and high gradient kurtosis, already present in the baseline training and therefore not related to DOODAT. Details can be found in Appendix J.4. Plasticity loss (Castanyer et al., 2025; Lyle et al., 2022) and high gradient kurtosis (Garg et al., 2021) are

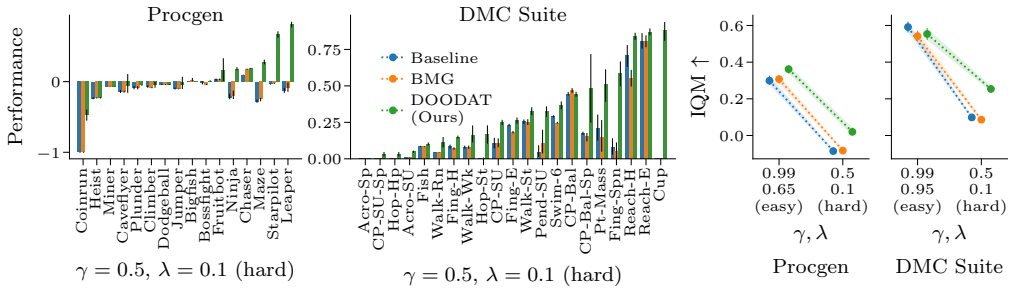

Figure 6: Comparison of DOODAT to static training (baseline) and BMG for PQN with Procgen and PPO with the DeepMind Control suite. The barplots on the left depict per-environment performance for the hard hyperparameter initialisations. The point plots on the right show aggregated performance for both easy and hard initialisation. Performance is normalised by the respective minimal and maximal performance of each task given by (Cobbe et al., 2020) for Procgen and (Tassa et al., 2018) for the DeepMind Control suite. See Section 4.1 for the experimental setup.

known phenomena, and mitigation strategies, such as ReDo (Sokar et al., 2023), periodic network resetting (Nikishin et al., 2022) and GNOM (Garg et al., 2021), are orthogonal to DOODAT. In environments unaffected by this collapse (e.g. *Starpilot*, *Ninja*, *Maze*, *Leaper*, *Coinrun*), DOODAT successfully recovers close-to-baseline performance even from hard initialisations.

PPO with the DeepMind Control: Finally, we adapted our framework for PPO by adding auxiliary policy heads for the neighbouring γ and λ values and evaluating them using the advantage $A_{(\gamma,\lambda)}(s,a)$ from the central value function, as described in Section 3. To ensure stable policy ratios, we apply a clipping objective similar to PPO with the same ϵ-margin. We quantify the off-policy drift of the auxiliary heads and find importance ratios and clipping frequencies comparable to those of the central PPO head (Appendix K.2). Figure 6 presents the results on the DeepMind Control suite. For the hard initialisation regime, DOODAT again shows significant IQM improvements. Although no gains over the standard baseline are achieved, DOODAT does not substantially degrade performance despite collecting data across multiple heterogeneous heads. To further investigate the limited improvement over the tuned baseline, we additionally evaluated 30 static (γ, λ) configurations and found little variation in performance: using the best configuration per task achieves an IQM of 0.61, compared to 0.59 for the default configuration, indicating limited headroom for γ/λ optimisation in this setting (Appendix K.1).

4.6 Robustness of DOODAT to its Hyperparameter Settings

DOODAT has three hyperparameters: the meta-learning rate, α_γ and α_λ. In our experiments, we found that the settings we chose for those hyperparameters generalise well to different training regimes, benchmarks and algorithms. Figure 4 shows that it performs well in different training regimes, consistently benefiting from longer training. In particular, a single meta-learning rate works well across all initialisations of γ and λ. As seen in Figure 5 (left), DOODAT is much less sensitive to its meta-learning rate than BMG. Similarly, the settings for α_γ and α_λ tuned only on the Atari-10, generalise well in our experiments and initialisation regimes. We compared the performance of different α values in Appendix F.2, observing robustness to even substantially larger values. We conclude that this positions DOODAT as an easily applicable method working out of the box.

5 Related Work

Hyperparameters significantly impact training performance in RL (Obando-Ceron et al., 2024; Eimer et al., 2023; Andrychowicz et al., 2020; Henderson et al., 2017; Mohan et al., 2023).

Gradient Free Dynamic Optimisation Methods: Similar to ours, several gradient-free dynamic optimisation methods have already been developed. Population-Based Training uses a population of agents in parallel. After specified intervals, low-performance configurations are resampled and

initialised with high-performance agents in the current population. (Jaderberg et al., 2017; Parker-Holder et al., 2020; 2021; Wan et al., 2022). OHT-ES (Tang & Choromanski, 2020) uses evolutionary strategies to approximate gradients, using multiple perturbations of the current hyperparameter values and full environment rollouts to assess policy performance of perturbations. HOOF (Paul et al., 2019) is a method only applicable to on-policy algorithms, collecting data with a single training policy. Subsequently, a random search over gradient updates with different hyperparameters is conducted and the performance of each resulting policy is estimated using weighted importance sampling over the collected training data. AdaQ (Vincent et al., 2025) applies only to off-policy learning, training a grid of networks with different hyperparameter initialisations. The training data between all networks is shared, and the network with the smallest TD-error after every gradient update is selected as the shared target for the next update. All methods require substantial parallel network updates, with HOOF and AdaQ being limited to specific RL algorithms.

Meta-Gradient Dynamic Optimisation: Meta-gradients provide a gradient-based alternative for online adaptation of continuous hyperparameters (Xu et al., 2018; Zahavy et al., 2020). Similarly to our method, meta-gradients do not require any parallel network updates and only small amounts of additional metadata, making them conceptually the closest to DOODAT. However, the original meta-gradients (Xu et al., 2018) have been developed mainly for single-pass RL algorithms, such as A2C (Mnih et al., 2016) and IMPALA (Espeholt et al., 2018), relying on a continuous stream of online metadata. This makes them inapplicable to epoch-based RL algorithms, such as PQN and PPO. Recently, bootstrapped meta-gradients (BMG) have been introduced as a means of extending the effective meta-learning horizon without backpropagation through all updates (Flennerhag et al., 2021). BMG uses additional network updates as targets for the meta-gradient update, which makes it applicable to epoch-based algorithms. Meta-gradients are known to suffer from instability (Metz et al., 2019; 2022). The meta-loss landscape frequently induces high variance, and backpropagating through multiple temporal steps results in brittle optimisation trajectories (Bonnet et al., 2021).

Temporal Horizon Ensembles: Maintaining multiple temporal horizons via auxiliary value heads improves training (Fedus et al., 2019; Nafi et al., 2024). Recently, Muppidi et al. (2024) introduced TRAC, employing an ensemble of tuners with varying discount factors. Whereas TRAC adapts the underlying optimiser to mitigate plasticity loss, DOODAT optimises RL parameters directly.

6 Discussion

We introduced DOODAT, a reliable framework for dynamically optimising the discount factor γ and trace-decay λ, applicable to value-based RL algorithms. The proposed architecture requires only minimal modifications to training pipelines and negligible computational and sample overhead.

Empirically, DOODAT demonstrates several advantageous properties. It exhibits optimisation stability, preventing hyperparameters from diverging into unstable regions characterised by substantially degraded performance. Furthermore, DOODAT proves effective not only when starting from standard baseline configurations but also from highly suboptimal regions, where it can recover full baseline performance within a single training. This makes it well-suited for new domains, where neither well-tuned hyperparameters nor tuned reward functions are available. Although DOODAT introduces three meta-hyperparameters, experiments across diverse environment suites and algorithms suggest that these exhibit zero-shot generalisation. This positions it as a reliable and efficient alternative to expensive black-box optimisation techniques or meta-gradient formulations.

Limitations. Although DOODAT demonstrates robust empirical performance across diverse tasks, it does not achieve universal success across all environments. Specifically, in environments characterised by learning plateaus with respect to a hyperparameter, DOODAT might get stuck in local optima if not using a larger neighbourhood. However, like all gradient-based methods in non-convex landscapes, meta-gradients suffer from similar failure modes, guaranteeing convergence only to local optima (Flennerhag et al., 2021; Nesterov, 2013). Further, so far, we have not thoroughly studied the neighbourhood size. While our chosen neighbourhood turned out to perform well, the impact of different neighbourhood sizes on training stability has not been thoroughly examined. Finally, while claiming broad applicability in principle due to the simplicity of the adjustments of DOODAT, we have only empirically tested it on PQN and PPO.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experiments and to implement methods. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for mathematical proofs, assist in writing proofs, propose or refine hypotheses, assist with translation, clean or reformat datasets, support qualitative or thematic data analysis, or interpret experimental results. The remaining required-disclosure tasks do not apply to this work.

Additionally, we used generative AI tools for brainstorming, creating or editing software code, and editing/drafting parts of the research paper to improve readability. We reviewed all AI-assisted work and independently verified its correctness and appropriateness. In particular, all AI-assisted code was checked and tested by the authors, including code implementing the methods, and all AI-assisted text and suggestions were reviewed and revised as necessary. The authors remain responsible for the final content of this work, including all text, claims, experimental methodology, results, and software artefacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken several steps to make our results reproducible. An anonymised implementation of DOODAT, together with BMG and static baselines and instructions for reproducing all experiments, is available at `https://anonymous.4open.science/r/doodat_submission-29D3`. The method is specified in Section 3, with full pseudo-code for PQN and PPO in Appendix F and our epoch-based adaptation of BMG in Appendix G. The experimental protocol (base implementations, number of seeds, aggregation metrics, and meta-data budget) is described in Section 4.1. Environment configurations, algorithm and network hyperparameters, DOODAT's meta-hyperparameters (including the neighbourhood-function settings and their sensitivity), and the computing infrastructure and GPU-hour budget are given in Appendices B, C, F, F.2, and D, respectively. Per-environment training curves and the learned γ and λ schedules for every seed are provided in Appendix J.2 and the supplementary material.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Proc. NeurIPS*, 2022.

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In *Proc. ICML*, 2022.

Ron Amit, Ron Meir, and Kamil Ciosek. Discount factor as a regularizer in reinforcement learning. In *Proc. ICML*, 2020.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study. In *Proc. ICLR*, 2020.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *Proc. ICML*, 2020.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013.

Clément Bonnet, Paul Caron, Thomas Barrett, Ian Davies, and Alexandre Laterre. One step at a time: Pros and cons of multi-step meta-gradient reinforcement learning. In *Proc. Workshop on Meta-Learning at NeurIPS*, 2021.

Roger Creus Castanyer, Johan Obando-Ceron, Lu Li, Pierre-Luc Bacon, Glen Berseth, Aaron Courville, and Pablo Samuel Castro. Stable gradients for stable learning at scale in deep reinforcement learning. In *Proc. NeurIPS*, 2025.

Pablo Samuel Castro. The formalism-implementation gap in reinforcement learning research. *arXiv preprint arXiv:2510.16175*, 2025.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proc. ICML*, 2020.

Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G. Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. In *Proc. AAAI*, 2021.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 2022.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 2024.

Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *Proc. ICML*, 2023.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proc. ICML*, 2018.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G. Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. In *Proc. RLDM*, 2019.

Sebastian Flennerhag, Yannick Schroecker, Tom Zahavy, Hado van Hasselt, David Silver, and Satinder Singh. Bootstrapped meta-learning (bmg). In *Proc. ICLR*, 2021.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. In *Proc. NeurIPS Dataset and Benchmark Track*, 2021.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *Proc. ICLR*, 2025.

Saurabh Garg, Joshua Zhanson, Emilio Parisotto, Adarsh Prasad, J. Zico Kolter, Zachary C. Lipton, Sivaraman Balakrishnan, Ruslan Salakhutdinov, and Pradeep Ravikumar. On proximal policy optimization's heavy-tailed gradients. In *Proc. ICML*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 2025.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proc. AAAI*, 2017.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. In *Proc. ICML*, 2020.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 2023.

Michael Kearns and Satinder Singh. "bias-variance" error bounds for temporal difference updates. In *Proc. COLT*, 2000.

Diederik P Kingma and Jimmy Lei. Adam: A method for stochastic optimization. In *Proc. ICML*, 2015.

Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. In *Proc. NeurIPS*, 2022.

Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *Proc. AISTATS*, 2021.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *Proc. ICLR*, 2022.

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, C. Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *Proc. ICML*, 2019.

Luke Metz, C. Daniel Freeman, Samuel S. Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proc. ICML*, 2016.

Aditya Mohan, Carolin Benjamins, Konrad Wienecke, Alexander Dockhorn, and Marius Lindauer. Autorl hyperparameter landscapes. In *Proc. AutoML Conference*, 2023.

Aneesh Muppidi, Zhiyu Zhang, and Heng Yang. Fast TRAC: A parameter-free optimizer for lifelong reinforcement learning. In *Proc. NeurIPS*, 2024.

Nasik Muhammad Nafi, Raja Farrukh Ali, William Hsu, Kevin Duong, and Mason Vick. Policy optimization using horizon regularized advantage to improve generalization in reinforcement learning. In *Proc. AAMAS*, 2024.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *Proc. ICML*, 2022.

Johan Obando-Ceron, João G. M. Araújo, Aaron Courville, and Pablo Samuel Castro. On the consistency of hyper-parameter selection in value-based deep reinforcement learning. *Reinforcement Learning Journal*, 2024.

Jack Parker-Holder, Vu Nguyen, and Stephen Roberts. Provably efficient online hyperparameter optimization with population-based bandits. In *Proc. NeurIPS*, 2020.

Jack Parker-Holder, Vu Nguyen, Shaan Desai, and Stephen J Roberts. Tuning mixed input hyperparameters on the fly for efficient population based autorl. In *Proc. NeurIPS*, 2021.

Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, Frank Hutter, and Marius Lindauer. Automated reinforcement learning (AutoRL): A survey and open problems. *Journal of Artificial Intelligence Research*, 2022.

Supratik Paul, Vitaly Kurin, and Shimon Whiteson. Fast efficient hyperparameter tuning for policy gradients. In *Proc. NeurIPS*, 2019.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proc. ICLR*, 2016.

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

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G. Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *Proc. ICML*, 2023.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *Proc. ICML*, 2023.

Richard S. Sutton and Andrew Barto. *Reinforcement Learning: an Introduction*. MIT press, 2018.

Yunhao Tang and Krzysztof Choromanski. Online hyper-parameter tuning in off-policy learning via evolutionary strategies. In *Proc. Workshop on Deep Reinforcement Learning at NeurIPS*, 2020.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 1997.

Théo Vincent, Fabian Wahren, Jan Peters, Boris Belousov, and Carlo D'Eramo. Adaptive q-network: On-the-fly target selection for deep reinforcement learning. In *Proc. ICLR*, 2025.

Xingchen Wan, Cong Lu, Jack Parker-Holder, Philip J. Ball, Vu Nguyen, Binxin Ru, and Michael A. Osborne. Bayesian generational population-based training. In *Proc. AutoML Conference*, 2022.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A highly parallel reinforcement learning environment execution engine. In *Proc. NeurIPS*, 2022.

Martha White and Adam White. A greedy approach to adapting the trace parameter for temporal difference learning. In *Proc. AAMAS*, 2016.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning (mg). In *Proc. NeurIPS*, 2018.

Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. In *Proc. NeurIPS*, 2020.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carlo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer., 2025. URL `https://github.com/google-deepmind/mujoco_playground`.

A Positioning with Respect to Related AutoRL Methods

Existing AutoRL methods typically trade off computational cost, sample efficiency, applicability, and the granularity at which hyperparameters can be updated. Table 1 summarises these differences. DOODAT combines continuous hyperparameter adaptation with low computational overhead and principled applicability across value-based algorithms.

Method	Computational Cost	Sample Efficiency	Update Granularity	Applicability
PBT	Very High	Low	Periodic/Episodic	Broad (Black-Box)
HOOF	Moderate to High	High	Continuous	Requires explicit policy (incompatible with Q-learning)
AdaQ	Moderate to High	High	Continuous	Restricted to Q-learning, not applicable to discount factor and λ
Temporal Horizon Ensembles	Low to Moderate (many aux. heads required)	High	N/A (no tuning)	Broad (Value-Based)
DOODAT (ours)	Low	High	Continuous	Broad (Value-Based)

Table 1: Comparison of DOODAT with related AutoRL approaches.

AdaQ is structurally unsuited to optimising the discount factor and trace-decay parameter. It relies on an ensemble of independently trained Q-networks and dynamically selects the best-performing network to provide target values. For this target selection to remain well-defined, the networks must share the same discount factor and trace-decay parameter. Moreover, DOODAT requires substantially less computation: rather than maintaining multiple full Q-networks in parallel, it introduces only lightweight auxiliary heads within a single training process.

Population Based Training is broadly applicable due to its black-box nature, but this flexibility comes at substantial computational cost. PBT requires a population of independently trained agents, resulting in lower sample efficiency than methods that adapt hyperparameters within a single training run. Its updates are also typically performed periodically, for example at episode or generation boundaries. In contrast, DOODAT continuously adapts the hyperparameters within a single run without maintaining a population of full agents.

HOOF requires an explicit policy representation to compute KL divergences and importance weights, and is therefore not directly applicable to pure value-based methods such as Q-learning. Furthermore, although HOOF avoids backpropagating through optimisation steps, it evaluates a large set of candidate hyperparameter values to perform its updates. For example, the original HOOF evaluation considers 50 randomly sampled values per hyperparameter, corresponding to 100 candidate evaluations when optimising two hyperparameters. DOODAT instead obtains a continuous optimisation signal from only four lightweight auxiliary heads, substantially reducing the evaluation overhead.

Temporal Horizon Ensembles are architecturally related to DOODAT in that they maintain multiple value heads corresponding to different discount factors. However, these methods use the different temporal horizons to form a static mixture for behaviour rather than dynamically optimising the discount factor of the behaviour function or mixture. Consequently, the hyperparameter defining the mixture still has to be selected externally. DOODAT instead uses its auxiliary heads to continuously adapt the hyperparameters during a single training run.

Overall, DOODAT occupies a different point in the AutoRL design space: it retains the sample efficiency and value-based applicability of single-run methods while enabling continuous adaptation at comparatively low computational cost. This combination is particularly relevant for optimising hyperparameters such as γ and λ, which directly affect value estimation and are therefore difficult to optimise using methods that require independently trained populations or explicit policy representations.

B Environment Setups

For the experimental evaluation, we utilised the EnvPool (Weng et al., 2022) integration for both the ALE (Bellemare et al., 2013) and Procgen (Cobbe et al., 2020). Within the ALE, we applied episodic life termination, reward clipping to $[-1, 1]$, a repeat action probability of 0.25, a frame skip of 4, and a maximum of 30 initial no-ops. To facilitate computationally efficient large-scale experimentation, several experiments employed the Atari-10 subset introduced by Aitchison et al. (2022). This subset captures over 90% of the performance variance of the full ALE suite and comprises Amidar, BattleZone, Bowling, DoubleDunk, Frostbite, KungFuMaster, NameThisGame, Phoenix, Qbert, and RiverRaid. For Procgen, we evaluated agents using the hard environmental distribution during both training and testing. Finally, continuous control tasks were evaluated using the DeepMind Control Suite (Tassa et al., 2018) as provided via the MuJoCo Playground (Zakka et al., 2025).

C Hyperparameter Settings

Table 2 details the hyperparameter configurations for both PQN and PPO across the evaluated environments. The PQN hyperparameters were selected to align with those established in (Gallici et al., 2025), whilst the PPO configurations follow the setup outlined in (Zakka et al., 2025). Regarding the network architectures, we utilised the Nature DQN (Mnih et al., 2015) for the ALE and the IMPALA architecture (Espeholt et al., 2018) for Procgen. For the DeepMind Control Suite, we adopted the standard training configuration provided by Google Brax (Freeman et al., 2021). This setup employs separate actor and critic networks. Specifically, the actor comprises a four-layer feed-forward network with a hidden dimension of 32, whilst the critic consists of a five-layer feed-forward architecture with a hidden dimension of 256. The hyperbolic tangent is used as the activation function.

PQN (ALE, Procgen)	
Hyperparameter	Value / Description
Parallel Environments	128
Steps per Environment	32
Batch Size	32
Exploration ϵ Start	1.0
Exploration ϵ Finish	0.001
Exploration ϵ Decay	10% of the total time steps linear decay
Number of Epochs	2
Learning-Rate	$2.5 \cdot 10^{-4}$

PPO (DM Control Suite)	
Hyperparameter	Value / Description
Parallel Environments	2048
Steps per Environment	8
Batch Size	512
Clipping ϵ	0.2
Value Function Coef.	0.5
Entropy Coefficient	0.01
Number of Epochs	4
Learning-Rate	10^{-4}

Table 2: **Left:** Hyperparameters of PQN for ALE and Procgen. **Right:** Hyperparameters of PPO for the DM Control Suite.

D Execution Environment and Resources Used

Due to the computational demands of our work, we pooled the computing resources of two clusters. We conducted a preliminary test that the experimental results are comparable under both setups. Below, we describe the hardware setups used for training.

L40S Cluster:
Compute nodes with CPUs of type Intel Xeon Gold 5418Y, 2.0 GHz, each 2x 24 cores, 1024GB main memory. Each node comes with 4 GPUs of type NVIDIA L40S 48 GB GDDR6, 18,176 CUDA cores, 568 Tensor cores.

H100 Cluster:
Compute nodes with CPUs of type Intel Xeon 8468 Sapphire, 2.1 GHz, each 2x 48 cores, 512GB main memory. Each node comes with 4 GPUs of type NVIDIA H100 with NVLink and 96 GB HBM2e, 16,896 CUDA cores, 528 Tensor cores.

Overall, we used around ~ 6000 GPU hours for the ALE experiments, ~ 2000 hours for the Procgen experiments and ~ 200 GPU hours for the DeepMind Control Suite experiments. The overall number of GPU hours used in this project is higher due to preliminary experiments to tune our baselines and design decisions for our method.

E Central Critic Proxy Validation

We evaluated whether the central critic provides a reliable proxy for rollout-based performance comparisons. It is important to note that DOODAT is a stochastic optimisation procedure due to predicting policy differences on the samples of trajectory data collected during training. Therefore, it is not expected that the performance of rollout returns can correctly identify the best-performing neighbouring policy. This is similar to other approaches of stochastic optimisation, such as policy-gradient methods, where individual gradient estimates can exhibit substantial deviation from the underlying optimisation direction while still supporting effective learning over repeated updates (Ilyas et al., 2020). Instead, DOODAT needs to provide a signal of the best-performing policy that is sufficiently consistent with the rollout-based returns over multiple steps to ensure correct accumulated update directions.

We conducted this analysis on the Atari-10 suite using three independent seeds. At 10%, 50%, and 100% of the training budget, we extracted the network parameters and evaluated all auxiliary head policies using full environment rollouts. We compared the ranking induced by the central critic, computed exactly as during training, with the empirical ranking from rollout returns using Kendall's rank correlation coefficient τ. Additionally, we evaluated whether the proxy identifies the same local hyperparameter update direction as the rollout oracle. For each hyperparameter, γ and λ, the oracle compares the central head with its upper and lower neighbouring heads: it selects an increase, decrease, or no update according to which neighbour achieves the highest rollout return. The proxy applies the same procedure using central critic estimates. We repeat this evaluation over 100 bootstrap batches of 1024 states sampled from the evaluation trajectories. Table 3 reports overall direction agreement, accuracy conditioned on non-zero oracle updates, and the rate of opposite-direction predictions.

Metric	γ	λ
Kendall correlation (τ)	0.247	0.195
Gradient-direction agreement	58.6%	56.7%
Non-zero update accuracy	35.0%	32.1%
Opposite-direction rate	7.7%	5.6%
Non-harmful decisions	92.3%	94.4%

Table 3: Validation of the central critic proxy against rollout-based evaluations.

The proxy achieves modest positive rank correlations of 0.247 for γ and 0.195 for λ. While modest, this indicates that the central critic preserves some of the relative ordering of neighbouring policies observed under rollout evaluation. This degree of per-step agreement is consistent with observations in RL: Ilyas et al. (2020) find that finite-sample policy-gradient estimates can have very low (≤ 0.1), and at some training stages even negative, cosine correlation with a high-sample estimate of the true gradient, while the corresponding PPO and TRPO agents nevertheless continue to improve.

The central critic selects the same three-way update decision as the rollout oracle in 58.6% of γ cases and 56.7% of λ cases. However, because this metric includes no-update decisions, agreement alone does not imply that the proxy reliably identifies beneficial updates. This is reflected in the lower accuracy conditional on a non-zero oracle update, where the proxy recovers the correct direction in 35.0% and 32.1% of γ and λ cases, respectively. Importantly, genuinely contradictory decisions are relatively uncommon: the proxy selects the opposite direction to the rollout oracle in only 7.7% of γ cases and 5.6% of λ cases. Thus, most disagreements correspond to conservative no-update decisions rather than updates in the direction judged harmful by the rollout oracle.

The same qualitative behaviour is observed across the 10%, 50%, and 100% training checkpoints as well as across the easy and hard subsets. Overall, these results suggest that the central critic does not

Algorithm 1 PQN with DOODAT

Require: Initial γ, λ, meta step size η_{meta}
Require: Neighbourhood mappings N_γ, N_λ
Initialise central config $c \leftarrow (\gamma, \lambda)$
Compute neighbours: $\gamma_\pm \leftarrow N_\gamma(N_\gamma^{-1}(\gamma) \pm 1)$, $\lambda_\pm \leftarrow N_\lambda(N_\lambda^{-1}(\lambda) \pm 1)$
Define neighbourhood: $H \leftarrow \{c, (\gamma_\pm, \lambda), (\gamma, \lambda_\pm)\}$
Initialise shared encoder ϕ and value-heads $\{\theta_h\}_{h \in H}$
for each episode **do**
  Collect $\mathcal{D}_{\text{train}}$ using ϵ-greedy policies $\{\pi_{\theta_h}\}_{h \in H}$ of each value-head with an equal share of environments
  Collect $\mathcal{D}_{\text{meta}}$ using ϵ-greedy policy π_{θ_c} of central value-head
  Save old params: $\phi^{\text{old}}, \{\theta_h^{\text{old}}\}_{h \in H}$
  Network-training:
    Compute returns $\{R_h\}_{h \in H}$
    for epochs $\times$ minibatches $B \subset \mathcal{D}_{\text{train}}$ **do**
      $\mathcal{L} \leftarrow \sum_{h \in H} \sum_{(s,a) \in B} \big(R_h - Q_{\theta_h}(\phi(s), a)\big)^2$
      Update $\phi, \{\theta_h\}$ using $\nabla \mathcal{L}$
    end for
  Meta-evaluation:
    for each $h \in H$ **do**
      $P_h \leftarrow \frac{1}{|\mathcal{D}_{\text{meta}}|} \sum_{s \in \mathcal{D}_{\text{meta}}} Q_{\theta_c^{\text{old}}}\big(\phi^{\text{old}}(s), \pi_{\theta_h}(\phi(s))\big)$
    end for
  Hyperparameter updates:
    Save old hyperparameters: $\gamma^{\text{old}} \leftarrow \gamma, \lambda^{\text{old}} \leftarrow \lambda$
    $\Delta_\gamma \leftarrow \text{sgn}(P_{\gamma_+} - P_{\gamma_-}) \cdot \max(0, P_{\gamma_+} - P_c, P_{\gamma_-} - P_c)$
    $\Delta_\lambda \leftarrow \text{sgn}(P_{\lambda_+} - P_{\lambda_-}) \cdot \max(0, P_{\lambda_+} - P_c, P_{\lambda_-} - P_c)$
    $\gamma \leftarrow N_\gamma(N_\gamma^{-1}(\gamma) + \eta_{\text{meta}} \cdot \Delta_\gamma), \quad \lambda \leftarrow N_\lambda(N_\lambda^{-1}(\lambda) + \eta_{\text{meta}} \cdot \Delta_\lambda)$
  Head interpolation:
    $\delta_\gamma \leftarrow N_\gamma^{-1}(\gamma) - N_\gamma^{-1}(\gamma^{\text{old}}), \quad \delta_\lambda \leftarrow N_\lambda^{-1}(\lambda) - N_\lambda^{-1}(\lambda^{\text{old}})$
    $\theta_c \leftarrow \theta_c + |\delta_\gamma| \cdot \big(\theta_{\gamma_{\text{sgn}(\delta_\gamma)}} - \theta_c\big) + |\delta_\lambda| \cdot \big(\theta_{\lambda_{\text{sgn}(\delta_\lambda)}} - \theta_c\big)$
  Update central config $c \leftarrow (\gamma, \lambda)$
  Recompute neighbourhood H using $\gamma_\pm \leftarrow N_\gamma(N_\gamma^{-1}(\gamma) \pm 1)$, $\lambda_\pm \leftarrow N_\lambda(N_\lambda^{-1}(\lambda) \pm 1)$
end for

Figure 7: Our proposed multi-head hyperparameter tuning DOODAT for PQN. In particular, the training data is collected with all heads on an equal share of parallel environments and the state features are shared between all additional auxiliary heads.

reproduce rollout-based policy comparisons reliably at every individual step, but provides a noisy yet systematically informative local signal that supports repeated hyperparameter adaptation.

F Extended Description of DOODAT

In Algorithm 1 and Algorithm 2, we present pseudo-code for the implementation of our method for PQN and PPO, respectively.

F.1 DOODAT Meta-Optimisation Hyperparameter

Following Flennerhag et al. (2021), we optimise the meta-gradients using Adam (Kingma & Lei, 2015) with a meta-learning rate of 10^{-3} and a gradient clipping threshold of 0.1. In the ALE and Procgen benchmarks, which utilise 128 parallel training environments, we allocate 25 environments to each auxiliary head and 28 to the central configuration of γ and λ. To collect the meta-data, 12 parallel environments are evaluated using the central value head for an identical number of steps as the training data, which facilitates parallel data collection within a single rollout phase. For PPO,

Algorithm 2 PPO with DOODAT

Require: Initial γ, λ, meta step size η_{meta}, clipping parameter ϵ
Require: Neighbourhood mappings N_γ, N_λ
Initialise central config $c \leftarrow (\gamma, \lambda)$
Compute neighbours: $\gamma_\pm \leftarrow N_\gamma(N_\gamma^{-1}(\gamma) \pm 1)$, $\lambda_\pm \leftarrow N_\lambda(N_\lambda^{-1}(\lambda) \pm 1)$
Define neighbourhood: $H \leftarrow \{c, (\gamma_\pm, \lambda), (\gamma, \lambda_\pm)\}$
Initialise shared encoder ϕ and actor-critic heads $\{\theta_h\}_{h \in H}$
for each episode **do**
  Collect trajectories $\mathcal{D}_{\text{train}}$ using stochastic policies $\{\pi_{\theta_h}\}_{h \in H}$ each with equal share of environments
  Collect trajectories $\mathcal{D}_{\text{meta}}$ using stochastic central policy π_{θ_c}
  Save old params: $\phi^{\text{old}}, \{\theta_h^{\text{old}}\}$
  Network-training:
    Compute GAE advantages $\hat{A}_h$ and returns $\hat{R}_h$ for each $h \in H$ using respective γ_h, λ_h
    for epochs $\times$ minibatches $B \subset \mathcal{D}_{\text{train}}$ **do**
      Compute policy ratio $r_h \leftarrow \frac{\pi_{\theta_h}(a|s)}{\pi_{\text{behaviour}}(a|s)}$ for each $h \in H$
      $\mathcal{L}_{\text{actor}} \leftarrow \sum_{h \in H} \min\left(r_h \hat{A}_h, \text{clip}(r_h, 1-\epsilon, 1+\epsilon)\hat{A}_h\right)$
      $\mathcal{L}_{\text{critic}} \leftarrow \sum_{h \in H} \left(V_{\theta_h}(s) - \hat{R}_h\right)^2$
      $\mathcal{L} \leftarrow -\mathcal{L}_{\text{actor}} + c_1 \mathcal{L}_{\text{critic}} - c_2 \sum_{h \in H} \mathcal{H}[\pi_{\theta_h}]$
      Update $\phi, \{\theta_h\}$ using $\nabla \mathcal{L}$
    end for
  Meta-evaluation:
    Compute central GAE advantages $\hat{A}_c$ on $\mathcal{D}_{\text{meta}}$ using γ_c, λ_c
    for each $h \in H$ **do**
      $P_h \leftarrow \frac{1}{|\mathcal{D}_{\text{meta}}|} \sum_{(s,a) \in \mathcal{D}_{\text{meta}}} \text{clip}\left(\frac{\pi_{\theta_h}(a|s)}{\pi_{\theta_c^{\text{old}}}(a|s)}, 1-\epsilon, 1+\epsilon\right) \hat{A}_c$
    end for
  Hyperparameter updates:
    Save old hyperparameters: $\gamma^{\text{old}} \leftarrow \gamma, \lambda^{\text{old}} \leftarrow \lambda$
    $\Delta_\gamma \leftarrow \text{sgn}(P_{\gamma_+} - P_{\gamma_-}) \cdot \max(0, P_{\gamma_+} - P_c, P_{\gamma_-} - P_c)$
    $\Delta_\lambda \leftarrow \text{sgn}(P_{\lambda_+} - P_{\lambda_-}) \cdot \max(0, P_{\lambda_+} - P_c, P_{\lambda_-} - P_c)$
    $\gamma \leftarrow N_\gamma(N_\gamma^{-1}(\gamma) + \eta_{\text{meta}} \cdot \Delta_\gamma), \quad \lambda \leftarrow N_\lambda(N_\lambda^{-1}(\lambda) + \eta_{\text{meta}} \cdot \Delta_\lambda)$
  Head interpolation:
    $\delta_\gamma \leftarrow N_\gamma^{-1}(\gamma) - N_\gamma^{-1}(\gamma^{\text{old}})$, $\delta_\lambda \leftarrow N_\lambda^{-1}(\lambda) - N_\lambda^{-1}(\lambda^{\text{old}})$
    $\theta_c \leftarrow \theta_c + |\delta_\gamma| \cdot \left(\theta_{\gamma_{\text{sgn}(\delta_\gamma)}} - \theta_c\right) + |\delta_\lambda| \cdot \left(\theta_{\lambda_{\text{sgn}(\delta_\lambda)}} - \theta_c\right)$
  Update central config $c \leftarrow (\gamma, \lambda)$
  Recompute neighbourhood H using $\gamma_\pm \leftarrow N_\gamma(N_\gamma^{-1}(\gamma) \pm 1)$, $\lambda_\pm \leftarrow N_\lambda(N_\lambda^{-1}(\lambda) \pm 1)$
end for

Figure 8: Our proposed multi-head hyperparameter tuning DOODAT for PPO. In particular, the training data is collected with all heads on an equal share of parallel environments and the state features are shared between all additional auxiliary heads.

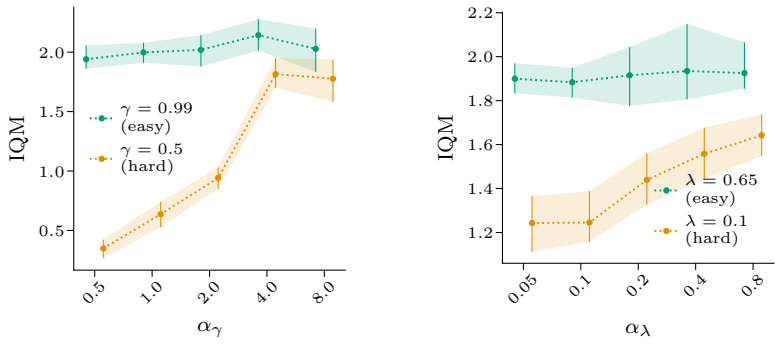

Figure 9: Ablation of different values for α_γ and α_λ for the Atari-10 when starting from the baseline values and the values furthest from the baseline. For both cases, only the respective hyperparameter γ or λ was tuned and the other hyperparameter fixed to the baseline value. The default values in our experiment were $\alpha_\gamma = 2.0$ and $\alpha_\lambda = 0.2$.

which employs 2048 parallel training environments, we assign 409 environments to each auxiliary head and 412 to the central policy. Consistent with the PQN setup, the PPO metadata is collected deterministically using 200 parallel environments for the same number of steps as the training data, thereby maintaining the unified rollout phase. For the neighbourhood function, we use $\alpha_\gamma = 2.0$ and $\alpha_\lambda = 0.2$.

F.2 Neighbourhood Function Analysis

In Figure 9, we individually examined the impact of different α-values on optimisation performance for both α_γ and α_λ. For the easy initialisation regime, DOODAT is robust to different initialisations, showing no decay in IQM for different values. For the hard initialisation regime, DOODAT appears to consistently improve. This is expected, as particularly increasing the γ and λ in the hard regime allows an algorithm to learn more meaningful policies early on. Specifically, for $\alpha_\gamma = 8.0$, the upper value head is trained with $\gamma \approx 0.98$ right from the start of training compared to $\gamma \approx 0.9$ for $\alpha_\gamma = 2.0$. While higher values appear to remain reliable for the easy initialisations as well, further research needs to examine if they also generalise better to other algorithms and environment suites, compared to our more pessimistic choice of $\alpha_\gamma = 2.0$ and $\alpha_\lambda = 0.2$. Decreasing the α-values further would consistently converge the algorithm to the baseline of static training with γ and λ.

G Meta-Gradient Implementation for Epoch-Based Learning

To adapt Bootstrapped Meta-Gradients (BMG) for epoch-based reinforcement learning algorithms such as PQN and PPO, we distribute the meta-optimisation process across the sequential minibatch updates within each training epoch. Rather than computing meta-gradients across full trajectories, which is standard in online algorithms, we divide the minibatches into blocks of K inner steps and L bootstrap steps.

For BMG, the inner learning phase updates the network parameters θ for K steps using the training data, yielding parameters θ_K. The subsequent L steps continue the optimisation to generate θ_{K+L}, which is further updated on a held-out meta-dataset to produce θ_{K+L+1}. In our implementation, we distinguished two meta-target objectives similar to (Flennerhag et al., 2021):

1. The KL divergence between policies π_{θ_K} and $\pi_{\theta_{K+L+1}}$ for PPO and the ϵ-greedy state-values V_{θ_K} and $V_{\theta_{K+L+1}}$ for PQN.
2. The squared Euclidean distance between the parameters θ_K and parameters θ_{K+L+1}.

Consistent with the findings of Flennerhag et al. (2021), computing the target in prediction space performed better over our tested learning rates compared to the second loss in parameter space, as can been seen in Figure 10. Therefore, we chose the loss in the prediction space in our experiments.

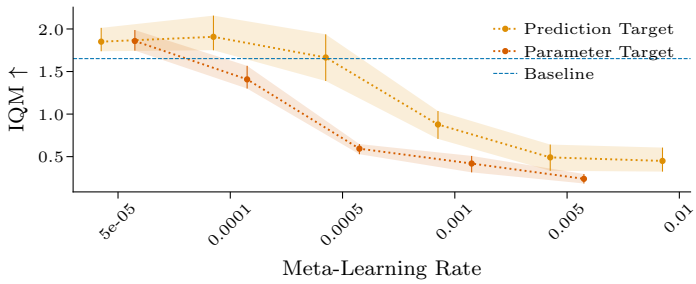

Figure 10: Comparison of the two meta-gradient targets for BMG on the Atari-10 environments for different learning rates.

Furthermore, setting $K = 1$ and defining L as the remaining minibatches within an epoch yielded the best performance. This observation is also consistent with the BMG paper, which showed that increasing the length of the bootstrap target L induces more robust behaviour compared to increasing the number of inner steps K.

Algorithm 3 and Algorithm 4 detail the pseudo-code for the meta-gradient implementations for PQN and PPO, respectively.

H Running Time Comparison

To evaluate additional computational requirements, we conducted a running time comparison between the static baseline, BMG, and DOODAT. Figure 13 illustrates these results for two representative environments of the ALE trained with PQN and the DeepMind Control Suite trained with PPO. Please note that DOODAT and BMG both collect an additional $\sim 10\%$ of training data for the meta optimisation.

Our results indicate that DOODAT introduces negligible computational overhead. Within the ALE, the auxiliary heads contribute a marginal increase in total parameters relative to the Nature DQN network architecture. Consequently, the running time increases by only 8%, even though DOODAT collects 10% of additional training data. BMG increases running time more substantially by 26% This is likely attributable to the complexity of computing second-order meta-gradients within the relatively large Nature DQN architecture.

In the PPO configuration, DOODAT increases running time by 14%. This higher relative overhead is likely due to the much smaller network architecture in our PPO experiments, which makes the auxiliary heads more computationally significant in proportion to the total network size. In this setting, BMG increases running time by only 8%, as meta-gradients are computed more efficiently on smaller architectures.

Overall, these findings demonstrate that DOODAT maintains high computational efficiency and scales favourably with architectural complexity. This benign scaling behaviour distinguishes DOODAT from meta-gradient approaches, which scale poorly when higher-order differentiation must be performed through computationally intensive backbones.

I Applicability of DOODAT across Algorithms

In this paper, we have only demonstrated the applicability of DOODAT empirically to value-based and policy-gradient RL through PQN (off-policy) and PPO (on-policy), respectively. To further substantiate our claim of generalisability, we discuss the principal application of DOODAT to several common RL algorithms. In general, extending DOODAT to other architectures requires only architectural modifications, namely branching the final network layers into multiple auxiliary heads, without introducing algorithm-specific heuristics. The standard loss of the underlying algorithm is then applied independently to each auxiliary head using its corresponding perturbed hyperparameter. Policy quality is evaluated using a shared central value estimate, enabling finite-difference estimates of hyperparameter sensitivity without requiring additional environment interactions.

Algorithm 3 BMG for PQN

Require: Initial γ, λ, meta step size η_{meta}, inner steps K, bootstrap steps L, meta-target type

Initialise network parameters θ and meta-parameters $\chi \leftarrow \{\sigma^{-1}(\gamma), \sigma^{-1}(\lambda)\}$
for each episode **do**
  Collect $\mathcal{D}_{\text{train}}$ using ϵ-greedy policy π_θ
  Collect $\mathcal{D}_{\text{meta}}$ using deterministic policy derived from π_θ
  for epochs **do**
    Shuffle $\mathcal{D}_{\text{train}}$ into minibatches
    for each block of $K + L$ minibatches **do**
      Inner Phase (K steps):
        $\theta_0 \leftarrow \theta$
        for $k = 1$ to K **do**
          Sample minibatch $B \subset \mathcal{D}_{\text{train}}$
          Compute TD loss $\mathcal{L}_{\text{inner}}(\theta_{k-1}, \sigma(\chi))$
          $\theta_k \leftarrow \theta_{k-1} - \eta \cdot \nabla_\theta \mathcal{L}_{\text{inner}}(\theta_{k-1}, \sigma(\chi))$
        end for
      Bootstrap Phase (L steps):
        $\theta'_K \leftarrow \text{stop_gradient}(\theta_K)$
        for $l = 1$ to L **do**
          Sample minibatch $B \subset \mathcal{D}_{\text{train}}$
          Compute TD loss $\mathcal{L}_{\text{inner}}(\theta'_{K+l-1}, \sigma(\chi))$
          $\theta'_{K+l} \leftarrow \theta'_{K+l-1} - \eta \cdot \nabla_\theta \mathcal{L}_{\text{inner}}(\theta'_{K+l-1}, \sigma(\chi))$
        end for
      Meta-Target Computation:
        Compute TD loss $\mathcal{L}_{\text{meta_target}}(\theta'_{K+L}, \sigma(\chi))$ on $\mathcal{D}_{\text{meta}}$
        $\theta'_{K+L+1} \leftarrow \theta'_{K+L} - \eta \cdot \nabla_\theta \mathcal{L}_{\text{meta_target}}(\theta'_{K+L}, \sigma(\chi))$
        $\theta'_{K+L+1} \leftarrow \text{stop_gradient}(\theta'_{K+L+1})$
      Outer Phase:
        if meta-target type is value **then**
          Compute ϵ-greedy state values $V_{\theta_K}(s)$ and $V_{\theta'_{K+L+1}}(s)$ for $s \in \mathcal{D}_{\text{meta}}$
          $\mathcal{L}_{\text{outer}}(\chi) \leftarrow \frac{1}{2}\mathbb{E}_{s \in \mathcal{D}_{\text{meta}}}\left[\left(V_{\theta_K}(s) - V_{\theta'_{K+L+1}}(s)\right)^2\right]$
        else if meta-target type is parameters **then**
          $\mathcal{L}_{\text{outer}}(\chi) \leftarrow \frac{1}{2}\|\theta_K - \theta'_{K+L+1}\|^2$
        end if
      Meta-Update:
        Compute meta-gradients $\nabla_\chi \mathcal{L}_{\text{outer}}(\chi)$ by differentiating through the K inner steps
        $\chi \leftarrow \chi - \eta_{\text{meta}} \cdot \nabla_\chi \mathcal{L}_{\text{outer}}(\chi)$
      Parameter Advancement:
        $\theta \leftarrow \theta_{K+L}$
    end for
  end for
end for

Figure 11: MG and BMG for the PQN algorithm. The inner phase comprises standard parameter updates, wherein the network weights θ are optimised with respect to the TD loss conditioned on the current meta-parameter χ. In contrast, the outer phase performs meta-optimisation by updating χ via gradients propagated through the unrolled inner steps. The MG baseline employs a parameter-space outer loss, minimising the Euclidean distance between θ_K and the target θ'_{K+L+1}. The BMG approach utilises a function-space outer loss, quantifying the squared difference between the ϵ-greedy state values induced by θ_K and θ'_{K+L+1} on a held-out meta-dataset. To ensure that the hyperparameters χ are constrained to $[0, 1]$, the sigmoid function σ is used, similar to (Flennerhag et al., 2021).

For **DQN**, the final linear layer can be replaced with multiple auxiliary Q-heads. Each head is trained using the standard DQN loss under its corresponding perturbed hyperparameter setting. The policies induced by the auxiliary heads can subsequently be evaluated using the central value estimate, analogously to the procedure used for PQN.

Algorithm 4 BMG for PPO

Require: Initial γ, λ, meta step size η_{meta}, inner steps K, bootstrap steps L, meta-target type

Initialise actor-critic parameters θ and meta-parameters $\chi \leftarrow \{\sigma^{-1}(\gamma), \sigma^{-1}(\lambda)\}$
for each episode **do**
  Collect trajectories $\mathcal{D}_{\text{train}}$ using stochastic policy π_θ
  Collect trajectories $\mathcal{D}_{\text{meta}}$
  for epochs **do**
    Shuffle $\mathcal{D}_{\text{train}}$ into minibatches
    for each block of $K + L$ minibatches **do**
      Inner Phase (K steps):
        $\theta_0 \leftarrow \theta$
        for $k = 1$ to K **do**
          Sample minibatch $B \subset \mathcal{D}_{\text{train}}$
          Compute GAE advantages $\hat{A}$ and returns $\hat{R}$ using $\sigma(\chi)$
          Compute PPO loss $\mathcal{L}_{\text{inner}}(\theta_{k-1}, \sigma(\chi))$
          $\theta_k \leftarrow \theta_{k-1} - \eta \cdot \nabla_\theta \mathcal{L}_{\text{inner}}(\theta_{k-1}, \sigma(\chi))$
        end for
      Bootstrap Phase (L steps):
        $\theta'_K \leftarrow \text{stop_gradient}(\theta_K)$
        for $l = 1$ to L **do**
          Sample minibatch $B \subset \mathcal{D}_{\text{train}}$
          Compute GAE advantages $\hat{A}$ and returns $\hat{R}$ using $\sigma(\chi)$
          Compute PPO loss $\mathcal{L}_{\text{inner}}(\theta'_{K+l-1}, \sigma(\chi))$
          $\theta'_{K+l} \leftarrow \theta'_{K+l-1} - \eta \cdot \nabla_\theta \mathcal{L}_{\text{inner}}(\theta'_{K+l-1}, \sigma(\chi))$
        end for
      Meta-Target Computation:
        Compute PPO loss $\mathcal{L}_{\text{meta_target}}(\theta'_{K+L}, \sigma(\chi))$ on $\mathcal{D}_{\text{meta}}$
        $\theta'_{K+L+1} \leftarrow \theta'_{K+L} - \eta \cdot \nabla_\theta \mathcal{L}_{\text{meta_target}}(\theta'_{K+L}, \sigma(\chi))$
        $\theta'_{K+L+1} \leftarrow \text{stop_gradient}(\theta'_{K+L+1})$
      Outer Phase:
        if meta-target type is policy **then**
          Compute policies π_{θ_K} and $\pi_{\theta'_{K+L+1}}$ on $s \in \mathcal{D}_{\text{meta}}$
          $\mathcal{L}_{\text{outer}}(\chi) \leftarrow \mathbb{E}_{s \in \mathcal{D}_{\text{meta}}} \left[\text{KL}(\pi_{\theta'_{K+L+1}}(\cdot|s) \parallel \pi_{\theta_K}(\cdot|s)) \right]$
        else if meta-target type is parameters **then**
          $\mathcal{L}_{\text{outer}}(\chi) \leftarrow \frac{1}{2} \|\theta_K - \theta'_{K+L+1}\|^2$
        end if
      Meta-Update:
        Compute meta-gradients $\nabla_\chi \mathcal{L}_{\text{outer}}(\chi)$ by differentiating through the K inner steps
        $\chi \leftarrow \chi - \eta_{\text{meta}} \cdot \nabla_\chi \mathcal{L}_{\text{outer}}(\chi)$
      Parameter Advancement:
        $\theta \leftarrow \theta_{K+L}$
    end for
  end for
end for

Figure 12: MG and BMG for the PPO algorithm. Analogous to the off-policy setting, the inner phase conducts standard policy and value network updates using the clipped surrogate objective and generalised advantage estimations parametrised by χ. The outer phase executes the meta-parameter update by differentiating the outer objective through the K-step inner optimisation trajectory. Whilst the MG variant relies on the Euclidean distance between network weights (θ_K and θ'_{K+L+1}), the BMG variant constructs a prediction-space outer loss based on the Kullback–Leibler (KL) divergence between the action distributions of the corresponding policies evaluated on the meta-dataset. To ensure that the hyperparameters χ are constrained to $[0, 1]$, the sigmoid function σ is used, similar to (Flennerhag et al., 2021).

For **TD3**, the final layers of both the actor and critic networks are branched into auxiliary heads. Each auxiliary critic is trained using the standard TD3 critic loss under its perturbed hyperparameter setting and provides the corresponding evaluation signal for its auxiliary actor. The central critic can

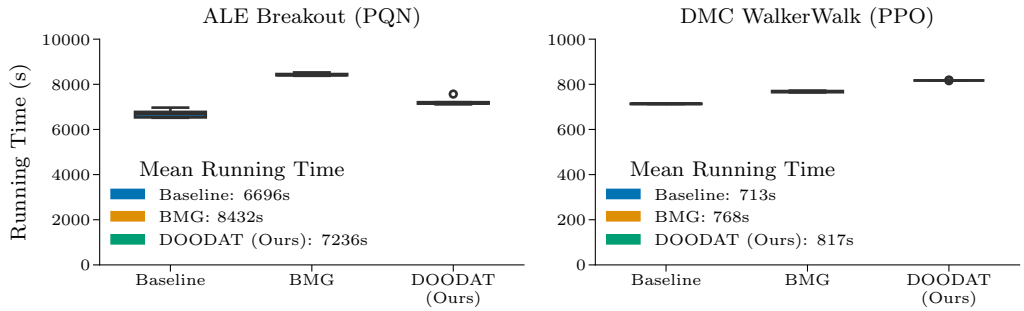

Figure 13: Running time comparison of the static baseline against BMG and DOODAT across PQN and PPO for representative environments. Both BMG and DOODAT collect 12.5% additional environment frames relative to the static baseline for the meta-optimisation of γ and λ. Results are aggregated over five independent runs performed on the L40S cluster (see Appendix D). Refer to Section 4.1 for the experimental setup.

then be used to evaluate the policies produced by the different actor heads, allowing DOODAT to estimate the effect of the hyperparameter perturbations while sharing the underlying representation and data.

For **SAC**, the architectural modification is analogous to TD3, with auxiliary actor and critic heads trained under perturbed hyperparameter settings. A particular consideration is the entropy temperature α. Unlike parameters that only affect the optimisation procedure, α directly changes the soft objective and therefore must be shared across auxiliary heads; using separate temperatures would cause different heads to optimise different soft objectives, making their finite-difference comparison inconsistent. The shared α can nevertheless be updated concurrently without compromising the computational efficiency of DOODAT. Specifically, its standard temperature loss can be computed using the entropy of the central policy head and SAC's target entropy. The resulting shared temperature is then used to compute the soft Bellman targets for all auxiliary critics.

DOODAT also naturally extends to **distributed actor–critic architectures such as IMPALA with V-trace**. The policy and value heads can be branched into multiple auxiliary heads, with each head computing its own V-trace targets under its corresponding perturbed hyperparameter setting. This modification does not require any changes to the underlying distributed data-collection mechanism.

Finally, the same principle can be applied to **offline RL**, including value-based methods such as CQL. The required modification is again to branch the final network layers into auxiliary heads and train each using the standard algorithmic objective under a perturbed hyperparameter setting. Since DOODAT evaluates policy quality through a shared baseline value function rather than requiring additional environment rollouts, this formulation is particularly well suited to offline settings.

Overall, these examples illustrate that DOODAT is not tied to a particular RL objective or data-collection paradigm. Its core requirement is the ability to share a common representation while maintaining multiple auxiliary heads corresponding to perturbed hyperparameter settings. The standard learning objective of the underlying algorithm can then be retained, while the shared central value estimate provides a consistent basis for comparing the resulting policies.

J Additional Results

We provide additional training curves on the ALE, Procgen and the DeepMind Control suite.

J.1 Relative Improvement Plots

To better understand the reliability of DOODAT, we show the relative improvements over the static baseline tuning from our experiments. Figure 14 depicts the relative improvement for the full ALE suite over the easy and hard initialisation of γ and λ for both training with 400M and 1B environment frames. Figure 15 and 16 show the same results for PQN on Procgen and PPO on the DeepMind control suite for both the easy and hard initialisation regimes.

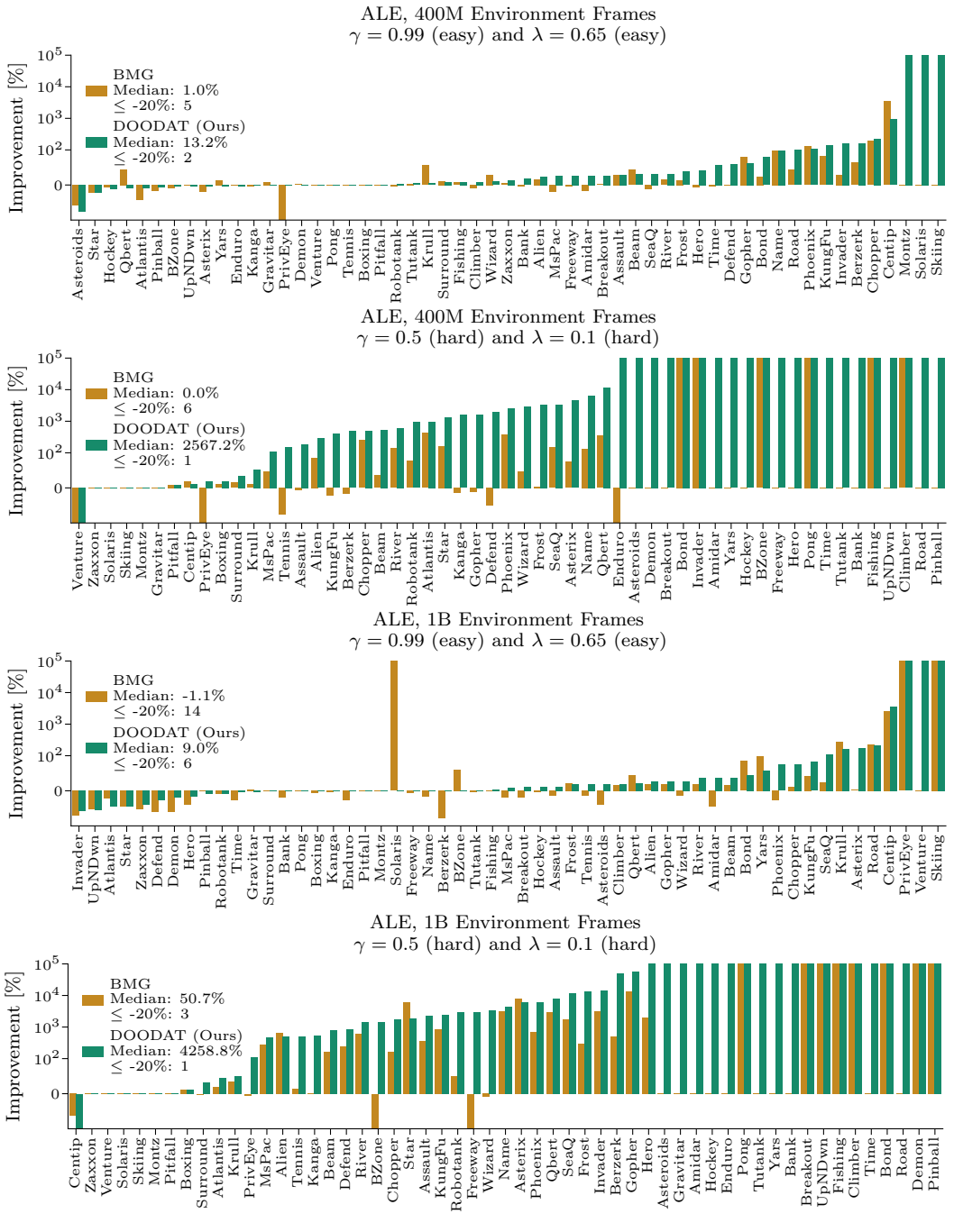

Figure 14: The relative improvements of both DOODAT and BMG for PQN on the ALE over the static baseline for our different initialisation and training regimes. Each plot states the number of environments where performance degrades more than 20% and the median improvement. See Section 4.1 for the experimental setup.

J.2 Environment Training Curves

We present learning curves for the full suites for both the easy and hard initialisation regimes of γ and λ for all environment suites. For the ALE, we present results for both training with 400M environment frames and 1B environment frames. Due to the large size of these plots and to ensure readability, we present the plots in our GitHub repository `https://anonymous.4open.`

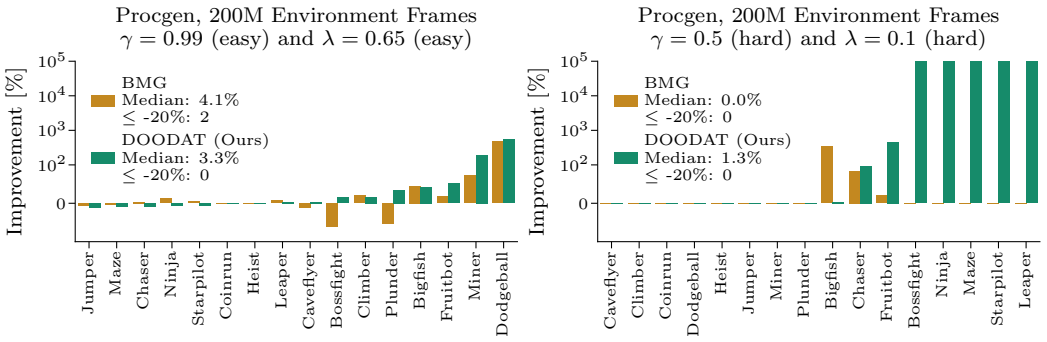

Figure 15: The relative improvements of both DOODAT and BMG for PQN on Procgen over the static baseline for our different initialisation regimes. Each plot states the number of environments where performance degrades more than 20% and the median improvement. See Section 4.1 for the experimental setup.

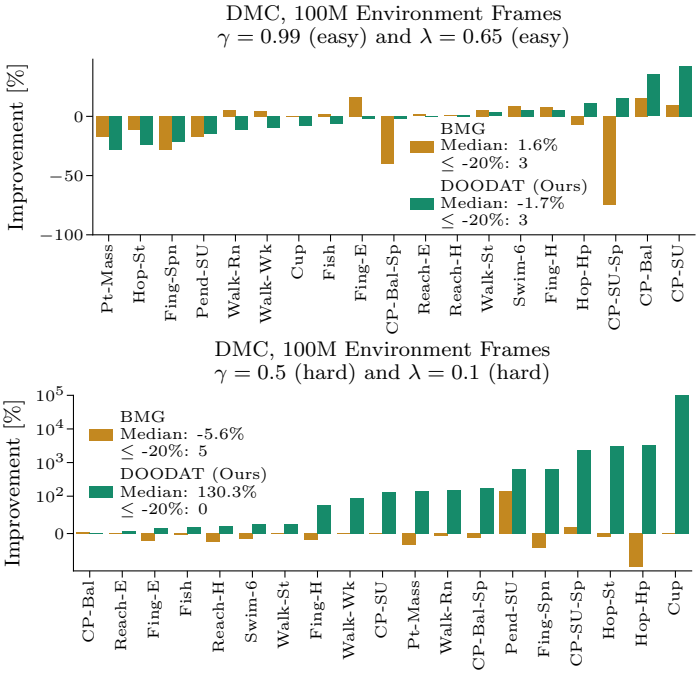

Figure 16: The relative improvements of both DOODAT and BMG for PPO on the DeepMind Control Suite over the static baseline for our different initialisation regimes. Each plot states the number of environments where performance degrades more than 20% and the median improvement. See Section 4.1 for the experimental setup.

`science/r/doodat_submission-29D3` and the supplementary material of this paper. For each environment, we present the learned schedules of γ and λ per seed. To allow linking the hyperparameter schedule of a training seed to its respective performance, we report the per-seed performance in addition to the mean-aggregated performance.

J.3 Full Procgen and DeepMind Control Suite Results

In addition to Figure 6, we present the full plots for both easy and hard initialisation regimes for Procgen and the DeepMind Control Suite, in Figure 17 and Figure 18

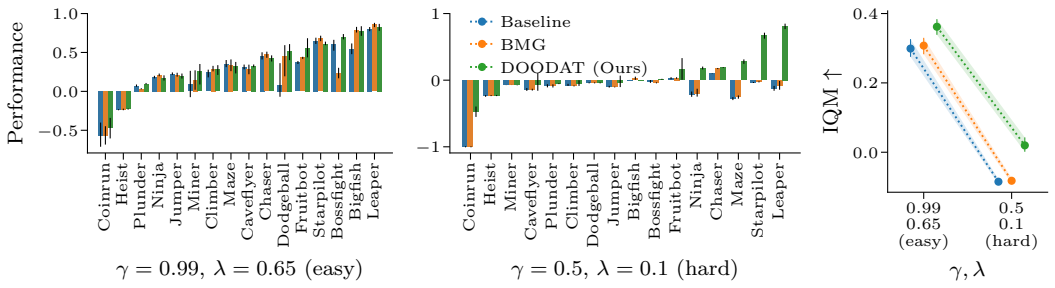

Figure 17: The bar-plots on the left depict the per-environment performance of the baseline trained with static γ and λ values and dynamic tuning with BMG and DOODAT for both Procgen. The point plots on the right depict the aggregated performance as IQM. Performance is normalised using the performance values given in (Cobbe et al., 2020), with 1 being the best performance. See Section 4.1 for the experimental setup.

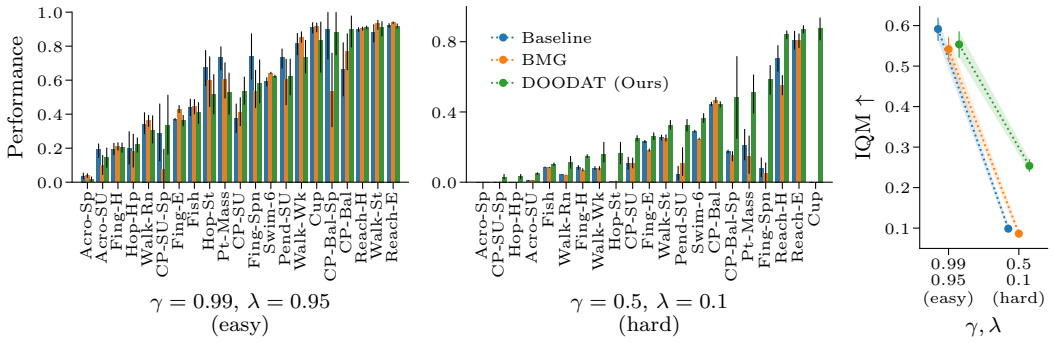

Figure 18: The bar-plots on the left depict the per-environment performance of the baseline trained with static γ and λ values and dynamic tuning with BMG and DOODAT for the DeepMind Control Suite. The point plots on the right depict the aggregated performance as IQM. Performance is normalised to 0 and 1 using the minimal and maximal possible performance. See Section 4.1 for the experimental setup.

J.4 Procgen Plasticity and Feature Rank Collapse

For Procgen, we observe several environments where no performance gains could be achieved when starting from the hard initialisation regime. To better understand this behaviour, we measured the feature rank of observations, the percentage of dormant neurons similar to (Sokar et al., 2023) (i.e. the plasticity loss of the network) and the gradient kurtosis (Garg et al., 2021) during training. The results are depicted in Figure 19 for selected environments. The static baseline without dynamic adjustment of γ and λ already suffers from a feature-rank collapse, high network dormancy and high gradient kurtosis. While DOODAT still behaves more benignly compared to BMG, especially concerning gradient kurtosis, it is not able to recover from the training collapse. Particularly for BMG, training failed in some cases due to high gradient kurtosis. Since the baseline already suffers from these problems, it is not caused by our method. Methods for dealing with plasticity loss (Dohare et al., 2024; Sokar et al., 2023) and gradient kurtosis (Garg et al., 2021) have been developed independently and are orthogonal to our method.

J.5 Benefit of Dynamic Hyperparameter Optimisation

To disentangle the benefit of dynamic optimisation from that of the final hyperparameter configuration, we compared DOODAT against static training using the final hyperparameters discovered by DOODAT on the Atari-10 benchmark. As shown in Figure 20, in the hard regime, DOODAT achieves an IQM of 0.881, compared to 0.880 for static training, with no statistically significant difference due to non-overlapping 95% confidence intervals. This indicates that, in this regime, the performance gain is primarily attributable to discovering an effective final configuration.

In contrast, in the easy regime, DOODAT achieves an IQM of 2.11, compared to 1.95 for static training using the same final hyperparameters, with non-overlapping confidence intervals. Thus, when starting from already well-tuned configurations, the dynamic schedule provides a statistically significant improvement beyond what can be obtained from the final hyperparameters alone. These results, illustrated in Figure 20, demonstrate that DOODAT can both discover effective static configurations and, where beneficial, exploit dynamic hyperparameter schedules during training.

We emphasise that discovering a robust static configuration and learning a dynamic schedule within a single training run are both valuable outcomes. Even under modest computational constraints, black-box optimisation methods in AutoRL typically necessitate tens of independent evaluation runs. Consequently, these methods incur substantially higher computational costs than DOODAT, rendering them infeasible for per-environment tuning across extensive benchmark suites such as the ALE or DMC.

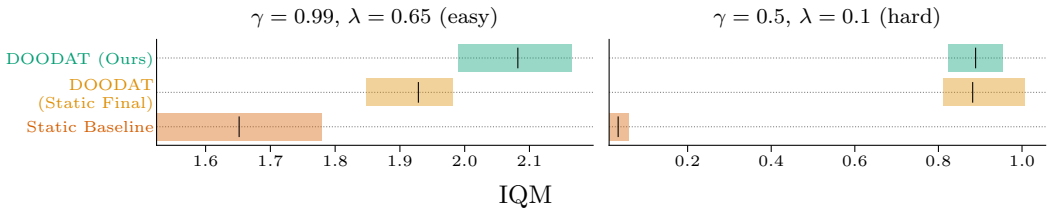

Figure 20: Comparison between DOODAT and static training using the final hyperparameters discovered by DOODAT on the Atari-10 benchmark. In the hard regime, the two approaches achieve comparable performance, indicating that the final hyperparameter configuration accounts for most of the improvement. In the easy regime, DOODAT achieves higher performance, demonstrating an additional benefit from dynamically adapting the hyperparameters during training.

K DeepMind Control Suite Analysis

To better contextualise the results of DOODAT on the DMC, we examine the training sensitivity to the discount and trace-decay and quantify the off-policyness of the auxiliary heads.

K.1 Discount and Trace-decay Sensitivity

To investigate the limited improvement of DOODAT over tuned baselines on the DMC suite, we conducted additional static training runs across a grid of combinations of the discount factor γ and trace-decay parameter λ. Specifically, we evaluated six values of γ (0.95, 0.97, 0.98, 0.99, 0.992, and 0.999) and five values of λ (0.8, 0.9, 0.95, 0.975, and 0.99), resulting in 30 static configurations in total. The default configuration is $(\gamma, \lambda) = (0.99, 0.95)$. The resulting configurations exhibit very similar performance. Selecting the best-performing static configuration for each environment yields an IQM of 0.61, compared to 0.59 for the default static tuning baseline and 0.56 for DOODAT. The small performance variation across static configurations indicates that the DMC easy regime provides limited scope for improvement through hyperparameter scheduling, explaining the weaker gains compared to ALE and Procgen, where the choice of hyperparameters has a substantially larger impact on performance.

K.2 Off-Policy Quantification for DOODAT with PPO

The off-policy drift of the auxiliary heads in PPO requires further discussion. We implemented logging for the importance sampling ratios and the clipping frequency within our PPO implementation of DOODAT, conducting experiments across all environments in the DMC and both easy and hard initialisation regimes to quantify the off-policy drift.

The empirical data demonstrate that off-policy divergence is minimal and clipping occurs infrequently. The central head exhibits a clipping fraction of 4.23% and a mean importance ratio of 99.98%. For the auxiliary heads, the clipping fractions remain highly comparable at 5.13% for the upper head, 4.69% for the lower head, 4.20% for the upper head, and 4.27% for the lower head.

Notably, both lower heads are further from the central head, but exhibit reduced clipping fractions. Furthermore, the mean importance ratios for all auxiliary heads remain tightly bounded between 99.97% and 99.98%.

Because the maximum absolute difference in clipping activation is at most 0.9 percentage points compared to the central head, and the central head itself already has a low clipping fraction of 4.23%, the clipping mechanism efficiently regulates these minor distributional shifts without saturating.

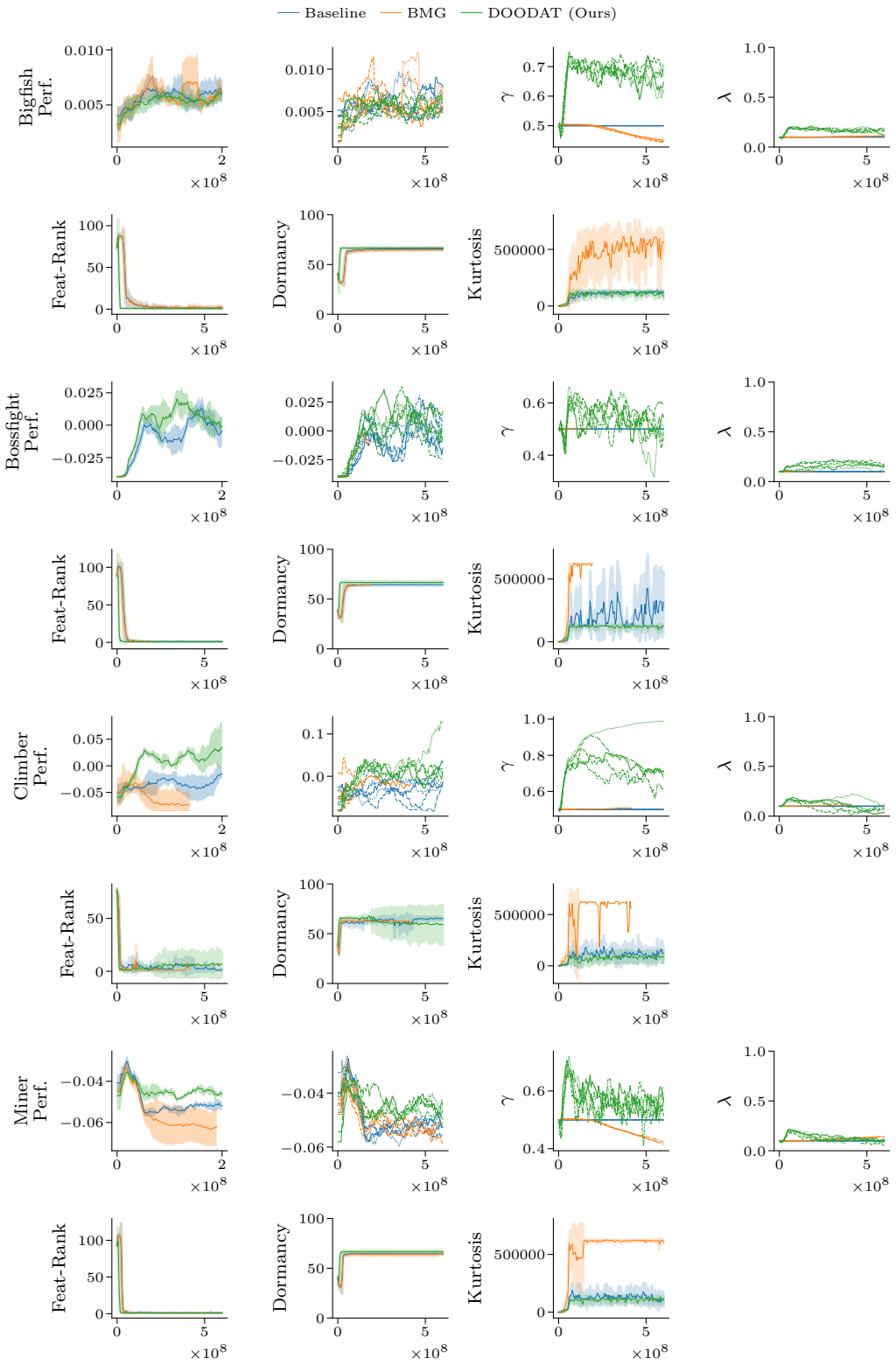

Figure 19: Selected environments of PQN on Procgen showcasing failure modes of both our method DOODAT and BMG when starting from the hard initialisation of $\gamma = 0.5$ and $\lambda = 0.1$. For the shown environments, the baseline with static hyperparameters already suffers from feature-rank degradation, a loss of plasticity and high gradient kurtosis. DOODAT is not able to recover from this training collapse in the beginning. See Section 4.1 for the experimental setup.