Learning the Supports for Categorical Critic in Reinforcement Learning

Jen-Yen Chang , Takayuki Osa , Tatsuya Harada

Keywords: Classification-Based Value Learning, HL-Gauss, Upper bound of Bellman operator

Summary

Value functions are an essential component in actor-critic based deep reinforcement learning (RL). Conventionally, these functions are trained as a regression task by minimising the mean squared error (MSE) relative to bootstrapped target values. Meanwhile, in distributional RL, a distribution of returns is modelled based on the distributional Bellman operator. This work investigates the Gaussian Histogram Loss (HL-Gauss), a recent approach that reframes value estimation as classification by encoding each scalar Bellman target as a Gaussian-smoothed categorical target. Despite its potential, applying histogram-based losses to RL presents inherent challenges, most notably the requirement to pre-define a fixed support interval, which is often complicated by the non-stationary and stochastic nature of target values typically found in RL tasks. In this work, we propose an approach that dynamically learns the lower and upper bounds of the support instead of assigning them beforehand. We derive an objective that jointly learns these bounds whilst learning the categorical representation of the scalar values, and we show that this objective forms an upper bound on the mean-squared Bellman error. Our theoretical analysis further shows that this bound is tighter than that of fixed supports of HL-Gauss. Empirically, the proposed objective enables stable adaptation of the support interval and matches HL-Gauss-based actor-critic algorithms on most continuous-control tasks whilst improving on a subset, without requiring a pre-specified support interval.

Contribution(s)

1. We demonstrate that the mean-squared Bellman error, commonly used in RL, is upper-bounded by HL-Gauss, a classification-based value-learning loss.
   Context: The tightness of this upper bound is directly correlated with the absolute values of the support interval. This theoretical connection motivates our approach to actively seek the narrowest feasible support interval to tighten the bound.
2. We provide a novel insight that eliminates the reliance on pre-defined support intervals by formulating dynamic support learning as a constrained optimisation problem.
   Context: We discovered that dynamically learning supports requires balancing two opposing forces: minimising the tightness of the upper-bound of the mean squared error to HL-Gauss, whilst simultaneously maximising the truncated Gaussian coverage constraint.
3. We formulate the constrained optimisation problem as an adversarial min-max game to derive a tractable learning objective.
   Context: By introducing a Lagrangian multiplier to enforce a probability mass coverage constraint, we allow the neural network to automatically expand the supports to capture the target distribution's leaked mass, whilst penalising excessive width.
4. We evaluate the proposed dynamic support learning objective on a range of continuous control tasks.
   Context: Our results show that dynamic support learning is competitive with the fixed-support HL-Gauss baseline on most tasks and improves over it on several tasks.
5. We provide evidence that a dynamically adapted support can outperform any fixed support.
   Context: We show that no single fixed support is optimal across all policies encountered during training, motivating adaptation over a hand-picked interval.

Learning the Supports for Categorical Critic in Reinforcement Learning

Jen-Yen Chang [1,2], **Takayuki Osa** [2], **Tatsuya Harada** [1,2]
chou@mi.t.u-tokyo.ac.jp, takayuki.osa@riken.jp,
harada@mi.t.u-tokyo.ac.jp

[1]**Graduate School of Information Science and Technology, The University of Tokyo**
[2]**RIKEN AIP**

Abstract

Value functions are an essential component in actor-critic based deep reinforcement learning (RL). Conventionally, these functions are trained as a regression task by minimising the mean squared error (MSE) relative to bootstrapped target values. Meanwhile, in distributional RL, a distribution of returns is modelled based on the distributional Bellman operator. This work investigates the Gaussian Histogram Loss (HL-Gauss), a recent approach that reframes value estimation as classification by encoding each scalar Bellman target as a Gaussian-smoothed categorical target. Despite its potential, applying histogram-based losses to RL presents inherent challenges, most notably the requirement to pre-define a fixed support interval, which is often complicated by the non-stationary and stochastic nature of target values typically found in RL tasks. In this work, we propose an approach that dynamically learns the lower and upper bounds of the support instead of assigning them beforehand. We derive an objective that jointly learns these bounds whilst learning the categorical representation of the scalar values, and we show that this objective forms an upper bound on the mean-squared Bellman error. Our theoretical analysis further shows that this bound is tighter than that of non-learned supports of HL-Gauss. Empirically, the proposed objective enables stable adaptation of the support interval and matches HL-Gauss-based actor-critic algorithms on most continuous-control tasks whilst improving on a subset, without requiring a pre-specified support interval.

1 Introduction

Accurate value-function estimation remains a cornerstone of deep reinforcement learning (RL). Traditionally, value functions are approximated through scalar regression using mean-squared error (MSE) with the Bellman update (Fujimoto et al., 2018; Haarnoja et al., 2018). By minimising the scalar error, the network parameters are updated to drive the point estimate towards the empirical mean of the target. Rather than projecting the Bellman update onto a scalar mean, distributional reinforcement learning (distributional RL) (Morimura et al., 2010a;b; Bellemare et al., 2017; Dabney et al., 2018a;b; Yang et al., 2019; Rowland et al., 2019; Kuznetsov et al., 2020; Sun et al., 2024) aims to learn the full probability distribution of the random return variable. This paradigm shift fundamentally alters the learning objective from minimising scalar error to minimising the discrepancy between probability distributions, employing metrics such as the Wasserstein distance for quantile-based regression methods or cross-entropy loss for categorical approximations.

Recent literature indicates that framing regression learning as a classification problem, specifically utilising the Gaussian Histogram Loss (HL-Gauss), yields smoother gradients in common regression tasks (Imani & White, 2018; Imani et al., 2024) and offers superior scalability in RL tasks

(Farebrother et al., 2024). Despite these advantages, histogram-based value-learning methods share a fundamental limitation: the necessity of a pre-defined support interval $[\nu_{\min}, \nu_{\max}]$. In practice, however, the true range of expected returns is rarely known *a priori*. This uncertainty is especially pronounced in environments characterised by sparse rewards, unbounded state spaces, or highly non-stationary returns as the policy evolves. Crucially, the return magnitudes induced by early exploratory policies and by the final converged policy can differ substantially, so no single fixed interval is well-suited to every policy encountered during learning.

Consequently, specifying the support interval beforehand introduces an inherent trade-off. Should the pre-defined support interval be excessively narrow, the target distribution will inevitably suffer from boundary truncation. Such truncation discards critical information regarding extreme, yet potentially highly informative, returns, potentially inducing suboptimal policy convergence. Conversely, if the support interval is conservatively set excessively broad to encompass all conceivable returns, the fixed number of bins k must span a vastly broad domain. This dilution reduces the representational resolution of each bin, weakening the learning signal.

To overcome these limitations, this work introduces a mechanism to dynamically learn the support interval for categorical critics to form the Dynamic Support Endpoint Learning (DySEL) algorithm. The crux of DySEL lies in framing this goal as a constrained optimisation problem. Specifically, the objective is to minimise the width of the learnt support interval in order to maintain high resolution, whilst ensuring it remains sufficiently broad to encapsulate the target distribution, mitigating severe truncation. Empirically, we find that learning the support is competitive with the fixed-support HL-Gauss baseline across most tasks and yields clear gains on a subset of tasks. In summary, the primary contributions of this work are as follows:

- We demonstrate that the mean-squared Bellman error, commonly used in RL, is upper-bounded by HL-Gauss, used for classification-based value learning in RL. The tightness of this upper bound is directly correlated with the absolute values of the support interval. This theoretical connection motivates our approach to actively seek the narrowest feasible support interval to tighten the bound.
- We provide a novel insight that eliminates the reliance on pre-defined support intervals in categorical critics by formulating dynamic support learning as a constrained optimisation problem. We discovered that dynamically learning supports requires balancing two opposing forces: minimising the tightness of the upper-bound of the mean squared error to HL-Gauss, whilst simultaneously maximising the truncated Gaussian distribution coverage.
- We formulate the constrained optimisation problem as an adversarial min-max game to derive a tractable learning objective. By introducing a Lagrangian multiplier to enforce a probability mass coverage constraint, we allow the neural network to automatically expand the supports to capture the target distribution's leaked mass, whilst penalising excessive width.
- We evaluate the proposed dynamic support learning objective on a range of continuous control tasks. Our results show that dynamic support learning is competitive with the fixed-support HL-Gauss baseline on most tasks and improves over it on several, most notably the humanoid tasks.
- We provide evidence that a dynamically adapted support can outperform any fixed support, where no single fixed support is optimal across all policies encountered during training, motivating adaptation over a hand-picked interval.

2 Preliminaries

A standard RL problem is defined as an infinite-horizon Markov Decision Process (MDP) = $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$, where the RL agent at time t observes a state s_t from a set of states $\mathcal{S}$, chooses an action a from a set of actions $\mathcal{A}$, and receives a reward r according to a mapping of the reward function $\mathcal{R}$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The environment then transitions into a state s_{t+1} with a transition probability

function $P(s_{t+1}|s_t, a_t)$ and the interaction continues. We also define the replay buffer $\mathcal{D}$ containing the state, action, reward, and next state at timestep t as $\mathcal{D} = (s_t, a_t, r_t, s_{t+1})$. The objective of an RL agent is to find a policy π that maximises the discounted expected return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma_t \mathcal{R}(s_t, a_t)]$, which is the expected cumulative sum of rewards when following the policy in the MDP, with the importance of the horizon is determined by a discount factor $\gamma \in [0, 1)$.

Distributional Reinforcement Learning. Whilst standard RL is principally concerned with the expected value of the return $Q^\pi(s, a) = \mathbb{E}[\mathcal{Z}^\pi(s, a)]$, distributional reinforcement learning (distributional RL) (Morimura et al., 2010a;b) seeks to characterise the full probability distribution of the random return variable, denoted as $\mathcal{Z}^\pi(s, a) = \sum_{t=0}^{\infty} \gamma_t \mathcal{R}(s_t, a_t)$. The distributional RL perspective allows for learning the intrinsic randomness of the environment and the policy, also called aleatoric uncertainty.

Categorical Value Learning via The Gaussian Histogram Loss (HL-Gauss). To represent a categorical distribution within a neural network, it is necessary to discretise the continuous support of the scalar into a fixed set of k atoms or bins. HL-Gauss re-casts the scalar regression target $Q(s, a)$ into a smoothed, continuous probability mass function over the predefined bins, and the neural network is trained to classify the probabilities of each bin by minimising the cross-entropy loss between the output of the neural network $h_i(x)$ and the re-casted target distribution q_y. Unlike distributional RL, HL-Gauss does not model the random-return distribution $\mathcal{Z}$.

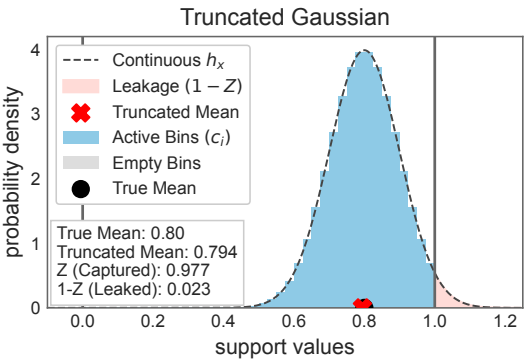

Figure 1: An example of a truncated Gaussian distribution $q(y)$ and its truncation bias. The X-axis is the support interval, limited in (0,1). The Y-axis is the probability density of the truncated Gaussian. Blue denotes bins with probability assigned, whilst grey bins have zero probability. In this work we aim to utilise $(1 - Z)$ as one of the substitutes to learn dynamic support interval.

Consider a support interval on the real line, denoted by $[\nu_{\text{min}}, \nu_{\text{max}}]$ between the endpoints ν_{min} and ν_{max}, which is uniformly partitioned into k bins, each possessing a width $w_i = (\nu_{\text{max}} - \nu_{\text{min}})/k$. Let σ represent the standard deviation, μ be the actual value of samples, the corresponding truncated Gaussian distribution is defined as:

$$q(y) = \frac{1}{Z\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} \tag{1}$$

The normalising term Z, also referred to in this work as covering mass, is defined as the integral of the probability density function (PDF) of the truncated Gaussian distribution over the support interval, and is unrelated to the random-return variable $\mathcal{Z}^\pi$. Therefore, supposedly a fixed and sufficiently broad support interval is given, $Z \approx 1$. With erf denoting the Gaussian error function (Olver et al., 2010). Z is calculated as

$$Z = \frac{\left(\text{erf}\left(\frac{\nu_{\text{max}}-\mu}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{\nu_{\text{min}}-\mu}{\sqrt{2}\sigma}\right)\right)}{2} \tag{2}$$

The weighting c_i of i-th bin is defined as

$$c_i = \frac{1}{2Z}\left(\text{erf}\left(\frac{l_i + w_i - \mu}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{l_i - \mu}{\sqrt{2}\sigma}\right)\right) \tag{3}$$

$$\text{with } l_i = \nu_{\text{min}} + (i-1)\cdot w_i\,, \quad i \in \{1, 2, \ldots, k\} \tag{4}$$

HL-Gauss (Imani & White, 2018; Imani et al., 2024; Farebrother et al., 2024) is defined as

$$\mathcal{L} = -\sum_{i=1}^{k} c_i \log h_i(x) \tag{5}$$

Here $h_i(x)$ is the predicted probability of bin i and $z_i = l_i + w_i/2$ is its centre, so a categorical output decodes to the scalar $\sum_{i=1}^{k} h_i(x)\, z_i$; we write $\mathbb{E}_{q_y}[z]$ for the corresponding decoded mean of the target q_y.

Biases in HL-Gauss. When decoding the categorical representation into a scalar value, several biases, namely the truncation bias and the quantisation bias, are inadvertently introduced. In HL-Gauss, these biases are mostly due to the hyperparameter choices. The truncation bias denotes the difference between the mean of a distribution μ and the mean calculated from the bins as $\mathbb{E}_{q_y}[z]$. That is, truncation bias $= \mathbb{E}_{q_y}[z] - \mu$. The quantisation bias denotes the difference between the continuous Gaussian h_x and its discretised counterpart onto bins, which is proportional to width w_i of each bin quantisation bias $\propto w_i = (\nu_{\text{max}} - \nu_{\text{min}})/k$ (Rowland et al., 2018; Lyle et al., 2019). The quantisation bias also affects the granularity of c_i.

Both biases may therefore be viewed as a result introduced by the support interval $[\nu_{\text{min}}, \nu_{\text{max}}]$. The truncation bias is prevalent when the support interval is too narrow, resulting in the Gaussian distribution being truncated; whilst the quantisation bias, assuming the number of bins k is fixed, is a direct result of w_i, and thus is more prevalent when the support interval is too broad. We show the formulation of both biases as the following and give an intuitive example of a truncated Gaussian distribution and its truncation bias in Figure 1, where the sum of the blue part is the covering mass Z calculated from Equation (2), from which we may recover the truncated mean by $\mathbb{E}_{q_y}[z]$. The orange part is leaked mass $(1 - Z)$, which causes the truncated mean to diverge from the true mean μ. In this work we aim to utilise $(1 - Z)$ as one of the substitutes to learn dynamic support interval.

3 Motivation: Interpretation of the HL-Gauss Loss

Building on the biases of pre-defined support intervals, this section motivates the dynamic adaptation of these intervals for categorical critics in RL.

We begin by examining the relationship between scalar Q-value regression, typically trained using the mean-squared Bellman error, and classification-based value learning, via HL-Gauss in this instance. Through this analysis, we demonstrate that the mean-squared Bellman error is upper-bounded by the HL-Gauss loss. Crucially, the tightness of this bound correlates directly with the absolute values of the support interval. Thus, enforcing a narrower support yields a theoretically tighter bound on the MSE, motivating our search for the narrowest feasible interval.

3.1 Connection between HL-Gauss and the upper bound of the mean-squared Bellman error

Suppose $(\mathcal{T}Q)$ is the Bellman operator, with the neural network outputting samples (x_j, y_j), the target distribution has support interval $[\nu_{\text{min}}, \nu_{\text{max}}]$ and target samples are denoted as (x_j, q_j). Extending (Imani et al., 2024) to RL, the Bellman operator that minimises with MSE is:

$$
\begin{aligned}
&\text{MSE}_{\text{Bellman}} && (6)\\
&:= (Q(s,a) - (\mathcal{T}Q)(s,a))^2 && (7)\\
&= (Q(s,a) - \mathbb{E}_{q_y}[z] + \mathbb{E}_{q_y}[z] - (\mathcal{T}Q)(s,a))^2 && (8)\\
&\le 2(Q(s,a) - \mathbb{E}_{q_y}[z])^2 + 2(\mathbb{E}_{q_y}[z] - (\mathcal{T}Q)(s,a))^2 \quad \because (M+N)^2 \le 2M^2 + 2N^2 && (9)\\
&= 2(\mathbb{E}_{h_x}[z] - \mathbb{E}_{q_y}[z])^2 + 2(\mathbb{E}_{q_y}[z] - (\mathcal{T}Q)(s,a))^2 \quad \because Q(s,a) := \mathbb{E}_{h_x}[z] && (10)\\
&\le \underbrace{8 \max(|\nu_{\text{min}}|, |\nu_{\text{max}}|)^2}_{\text{absolute width}} \cdot \underbrace{\min\left(\frac{1}{2}\mathcal{D}_{KL}(q_y \| h_x), 1 - \exp(-\mathcal{D}_{KL}(q_y \| h_x))\right)}_{\text{distributions matching}}\\
&\quad + \underbrace{2(\mathbb{E}_{q_y}[z] - (\mathcal{T}Q)(s,a))^2}_{\text{truncation bias}} \quad \because \text{Equation (7) in (Imani et al., 2024)} && (11)
\end{aligned}
$$

We may observe that the minimisation of the mean-squared Bellman error on the LHS is upper-bounded by the RHS and may be decomposed into three components: the absolute width of the support interval, defined by $[\nu_{\text{min}}, \nu_{\text{max}}]$; the distribution-matching mechanism, facilitated by the Kullback–Leibler (KL) divergence; and the truncation bias, which represents the error between the projected target value and the true target value.

3.2 Connection to Previous Works

In the literature introducing HL-Gauss for both standard regression (Imani & White, 2018; Imani et al., 2024) and reinforcement learning contexts (Farebrother et al., 2024), a standard simplifying assumption is that the distributions matching term $\min\left(\frac{1}{2}\mathcal{D}_{KL}(q_y\|h_x), 1-\exp(-\mathcal{D}_{KL}(q_y\|h_x))\right)$ may be directly substituted with the cross-entropy loss because, during the simultaneous minimisation on both sides (c.f. Section 2.1 in (Imani et al., 2024)), minimising the cross-entropy effectively minimises the KL divergence $\mathcal{D}_{KL}(q_y\|h_x)$ between the target distribution q_y and the parametrised prediction h_x.

Furthermore, existing literature typically assumes a sufficiently broad and fixed support interval $[\nu_{\text{min}}, \nu_{\text{max}}]$. Under such conditions, the cross-entropy remains the sole term requiring minimisation. This simplification is theoretically justified; when the support interval is sufficiently wide, the truncation bias, which represents the error incurred by target values falling outside the pre-defined support interval (c.f. Figure 1), becomes minimal. That is, the leaked mass $(1 - Z)$ of the target probability distribution effectively becomes negligible. Moreover, because the absolute width of the support interval is fixed, it remains constant w.r.t the trainable parameters of the neural network. Consequently, it does not contribute to the gradient computation, allowing prior works to safely discard the absolute width term and focus exclusively on minimising the cross-entropy objective presented in Equation (5).

Nevertheless, since in this work we are interested in dynamic support interval, we can observe in Equation (11) that an excessively broad support interval results in a loose upper bound on the mean-squared Bellman error, resulting in low granularity and potentially leading to suboptimal convergence. Consequently, we contend that it is imperative to dynamically learn an appropriately scaled support interval that balances the resolution of the distribution against the magnitude of the supports.

4 Practical Algorithm for Learning Dynamic Supports

In this work, we present the Dynamic Support Endpoint Learning (DySEL) algorithm derived from the upper bound of the mean-squared Bellman error in Equation (11). The key to our proposed approach is built on our observation that this upper bound reveals an inherent adversarial tension between the absolute width term $\max(|\nu_{\text{min}}|, |\nu_{\text{max}}|)$ and the truncation bias term $\mathbb{E}_{q_y}[z] - (\mathcal{T}Q)(s,a)$ during minimisation. Specifically, the former term, the absolute width, seeks to minimise the support interval directly to have a tighter upper bound on the mean-squared Bellman error, whereas the latter term, the truncation bias, incentivises the expansion of the supports to encapsulate the full probability mass Z in order to eliminate truncation errors.

To resolve this adversarial tension, we propose to reformulate the objective in Equation (11) as a *constrained optimisation* problem. By making a series of changes, we arrive at our final min-max objective in Equation (16) that balances the requirement for a tighter support interval against the necessity of sufficient distributional coverage.

Assumptions. In the remainder of this work, we operate under the assumption that the number of bins k remains fixed, as it defines the neural network's final output layer, a common assumption in categorical critic literature (Bellemare et al., 2017; Imani & White, 2018). Furthermore, in accordance with previous works in HL-Gauss (Imani & White, 2018; Imani et al., 2024; Farebrother et al., 2024), the Gaussian standard deviation σ is calculated from a pre-determined sigma-to-width ratio, that is, $\sigma \propto w_i$. Consequently, the primary remaining degree of freedom for learning is the support interval $[\nu_{\text{min}}, \nu_{\text{max}}]$, which aligns with our goal and the derived upper bound in Equation (11).

Addressing the product. Firstly, directly minimising the product, in our case $\max(|\nu_{\text{min}}|, |\nu_{\text{max}}|) \cdot \text{CE}$ in Equation (11), is known to cause instability as it constitutes a fundamentally non-convex optimisation problem (Boyd & Vandenberghe, 2004; Sun et al., 2016; Zhu et al., 2017; Jin et al., 2017). Since both the max of absolute values and cross-entropy are always non-negative, we may apply the AM-GM inequality $2\sqrt{X \cdot Y} \leq (X + Y)$ and introduce a positive scaling constant $\alpha > 0$ as follows:

$$4 \max(|\nu_{\text{min}}|, |\nu_{\text{max}}|)^2 \cdot \text{CE} \leq \left(\alpha \max(|\nu_{\text{min}}|, |\nu_{\text{max}}|)^2 + \frac{1}{\alpha} \text{CE} \right)^2 \tag{12}$$

Addressing the truncation bias. Secondly, we argue that minimising the truncation bias may be simplified and transformed into minimising the probability mass residing outside the support interval (leaked mass) $(1 - Z)$, which in turn maximises the covering mass Z. An intuitive understanding of covering mass Z or leaked mass $(1 - Z)$ may be found in Figure 1. Let us denote the expected value of this leaked mass as μ_{out}. By the law of total expectation (Weiss, 2005) and after rearranging,

$$(\mathcal{T}Q)(s, a) = Z \cdot \mathbb{E}_{q_y}[z] + (1 - Z) \cdot \mu_{\text{out}} \tag{13}$$

$$\mathbb{E}_{q_y}[z] - (\mathcal{T}Q)(s, a) = (1 - Z)(\mathbb{E}_{q_y}[z] - \mu_{\text{out}}) \tag{14}$$

Since the target is a thin-tailed truncated Gaussian, the displacement $(\mathbb{E}_{q_y}[z] - \mu_{\text{out}})$ is bounded; driving the leaked mass $(1 - Z) \to 0$ therefore drives the truncation bias to zero, thus it suffices to control $(1 - Z)$ directly.

Reformulating as a constrained optimisation problem. We may now begin to formally present DySEL by observing that the minimisation problem in Equation (11) is at heart a *constrained optimisation* problem. Intuitively, the primary objective is to identify the minimum possible width for the support interval, thereby ensuring a tight upper bound on the error. Simultaneously, this interval must satisfy the constraint of sufficiently encapsulating the probability mass, Z, of the truncated Gaussian distribution to prevent detrimental truncation bias. That is, in minimisation terms, we aim to minimise $(1 - Z)$. Additionally, one inherent property of Z, by definition in Equation (2), is that $\max(Z) = 1$, allowing us to utilise $(1 - Z)$ as a constraint.

Let θ denote the parameters of the critic network and ϕ represent the parameters of the support interval network. Applying the series of changes presented, we may reformulate the minimisation of the RHS from Equation (11) as the following *constrained optimisation* problem:

$$\mathcal{J}(\theta, \phi) = \underbrace{\alpha \max(|\nu_{\text{min}}|, |\nu_{\text{max}}|)}_{\text{width penalty}} + \underbrace{\frac{1}{\alpha}(-\sum_{i=1}^{k} c_i \log h_i(x))}_{\text{cross-entropy}}$$

$$\text{subject to the truncation constraint on leaked mass} : (1 - Z) \leq \epsilon \tag{15}$$

Our final objective: min-max optimisation. We are ready to present our final objective. We may optimise the problem in Equation (15) as an adversarial min-max game by introducing a Lagrangian multiplier λ to enforce the probability mass constraint during the gradient descent process. The parameters θ, ϕ, and λ are therefore trained via the following Lagrangian,

$$\mathcal{L}(\theta, \phi, \lambda) = \underbrace{\alpha \max(|\nu_{\text{min}}|, |\nu_{\text{max}}|)}_{\text{width penalty}} + \underbrace{\frac{1}{\alpha}(-\sum_{i=1}^{k} c_i \log h_i(x))}_{\text{cross-entropy}} + \underbrace{\lambda((1 - Z) - \epsilon)}_{\text{mass constraint}} \tag{16}$$

where the optimal parameters θ^*, ϕ^* and λ^* are defined by:

$$\theta^*, \phi^* = \arg\min_{\theta, \phi} \mathcal{L}(\theta, \phi, \lambda) \quad \text{and} \quad \lambda^* = \arg\max_{\lambda \geq 0} \mathcal{L}(\theta, \phi, \lambda) \tag{17}$$

Within Equation (16), as λ is a trainable parameter, the remaining hyperparameter is α, introduced in Equation (12), which governs the trade-off between the absolute width penalty and the cross-entropy loss when minimising our proposed final objective. Please refer to the listing 2 for a reference implementation.

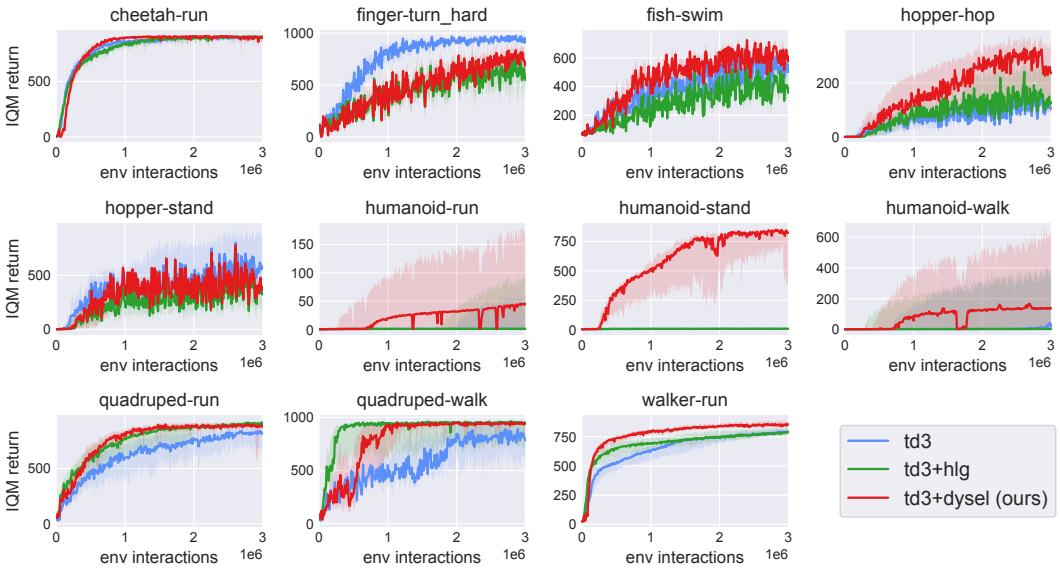

Figure 2: The IQM returns for each task in DM Control of our proposed DySEL versus baselines. Our proposed method TD3+DySEL remains competitive with TD3+HLG, whilst significantly outperforming in some *humanoid* tasks, showing the empirical benefit of learnt supports.

5 Experiments

In this section, we empirically evaluate the performance of our final proposed objective in Equation (16), comparing it to related HL-Gauss works on a variety of challenging continuous control tasks. We show that our proposed DySEL outperforms or remains competitive with previous baselines. We also provide analyses on some design choices and include ablation studies. Detailed hyperparameters and details on the practical implementation used in our experiments are depicted in Section A, with further ablation studies in Section B. A reference implementation of HL-Gauss and DySEL is also included in Section A.

5.1 Experimental Setups

Baselines. For continuous control tasks that we experiment on, we build our algorithm on top of TD3 (Fujimoto et al., 2018). We also adapt HL-Gauss on top of TD3 to facilitate comparison, noted as TD3+HLG. For all experiments, for TD3+HLG, we use support interval = $[-100, 100]$. We apply DySEL to TD3 and denote the resulting algorithm as TD3+DySEL. Furthermore, for both TD3+HLG and TD3+DySEL, we use $k = 128$ bins.

Benchmark and Evaluation Method. We evaluate DySEL on 11 challenging tasks from the commonly used benchmark DeepMind Control Suite (Tunyasuvunakool et al., 2020), where the maximum achievable return for these tasks is 1000. We train 10 seeds, seeds$= \{0, \dots 9\}$, for all tasks and train for three million timesteps whilst evaluating every ten thousand timesteps. We run twenty episodes at each evaluation, and calculate inter-quantile mean (IQM) with shaded area as the upper and lower bound of 95% bootstrapped confidence interval, according to best practices (Agarwal et al., 2021). We report the IQM rather than the mean as it is robust to outlier seeds and is the recommended aggregate of Agarwal et al. (2021). The results are depicted in the next subsection.

5.2 Results and Q&As

Our experiments aim to answer the following questions.

Q1: What is the performance of DySEL for DM-control benchmarking tasks?
A1: DySEL is generally competitive with vanilla HL-Gauss and provides clear gains on several tasks.

The IQM returns for each task are depicted in Figure 2. Compared with baselines, we find that TD3+DySEL generally achieves better performance or yields results comparable to the best-performing baselines.

Notably, TD3+DySEL demonstrates substantial performance gains over both TD3 and TD3+HLG for *humanoid* tasks. We hypothesise that this improvement is due to the finer granularity afforded by our dynamically learnt support interval, where the finer resolution enables the critic to discern subtle variations in sample returns, nuances that are typically obscured by pre-defined, fixed supports. Indeed, as depicted in Figure 4, employing these learnt intervals as new reference supports for TD3+HLG yields improved empirical benefits for the *humanoid-run* task. We further discuss using the learnt support interval in Q4.

Interestingly, for *finger-turn_hard* and *hopper-stand* tasks, both HL-Gauss-based approaches either underperform or yield no significant improvement over the baseline TD3 algorithm. Upon examining the evolution of the support intervals for these tasks as depicted in Figure 3, the pattern suggests that the underperformance is tied more to the cross-entropy objective than to dynamic support learning itself, since TD3+HLG also underperforms on these tasks.

Q2: What are the important hyperparameters of DySEL?
A2: The most important hyperparameter for DySEL is the choice of α, which balances between the absolute width penalty and cross-entropy in Equation (16).

Given that α determines the magnitude of the absolute width penalty, a larger value of α intuitively yields a narrower support interval during the initial stages of training. We observe that empirical experiments corroborate this hypothesis, and we include how these support intervals change during training in Figure 3. Specifically, within the first few thousand steps, α facilitates the rapid learning of a sufficient support interval; subsequently, as training progresses, the learning support interval may gradually become broader, depending on the task and its best α.

Furthermore, we can achieve dynamic ν_{max} and ν_{min} scheduling, such as gradually increasing the support interval, just by tuning the hyperparameter α. Instead of manually designing a ν_{max} and ν_{min} schedule, which is often infeasible due to the amount of task-specific knowledge required, by using α we can dynamically learn and expand support interval when the distribution requires more coverage on the covering mass Z.

To be precise, three hyperparameter choices remain. Firstly, there is the important term α between the absolute width penalty and cross-entropy. Secondly, the choice of *initialising* support interval, required to project the first target logits to target scalar values. Thirdly, the choice of number of bins k, which is a common hyperparameter for categorical critic methods (Bellemare et al., 2017; Imani & White, 2018). Generally speaking, starting from a smaller (e.g. [-10, 10] or [-5, 5]) initialising support interval is a good enough start, but we show different initialising support intervals do not significantly affect performance as detailed in Section B.

Q3: What are the actual values of the supports learnt?
A3: This is highly task-dependent and closely related to α.

As mentioned previously in Q2, α helps determine the earliest support interval. We include how these support intervals change during training in Figure 3. As we can observe, there are generally speaking two types of behaviours. For some tasks, such as *finger-turn_hard*, *hopper* tasks and etc., the support interval converges to some stable value almost immediately and remains relatively unchanged, whilst for other tasks, such as *cheetah-run*, *fish-swim*, remaining *humanoid* tasks, *quadruped* tasks and etc., the support interval gradually increases as training progresses. The behaviour is highly dependent on the value of α.

Tasks that require a higher α suggest that finer granularity is important during learning, where the finer resolution enables the critic to discern subtle variations in sample returns. Conversely, tasks that require a more stable support interval would need a lower α. Another interesting observation is that our learnt support interval is almost symmetric.

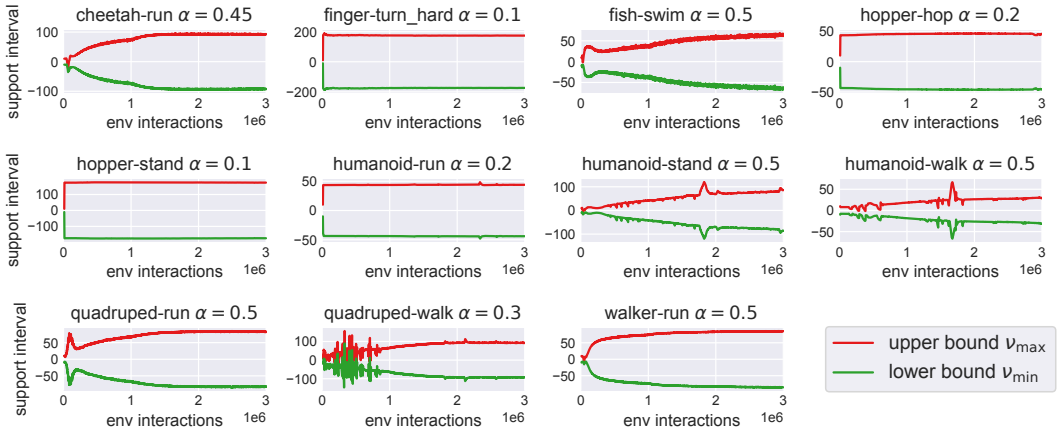

Figure 3: The evolution of support interval of DySEL during training. We can observe that some support intervals remain stable whilst others gradually increase/decrease as training progresses.

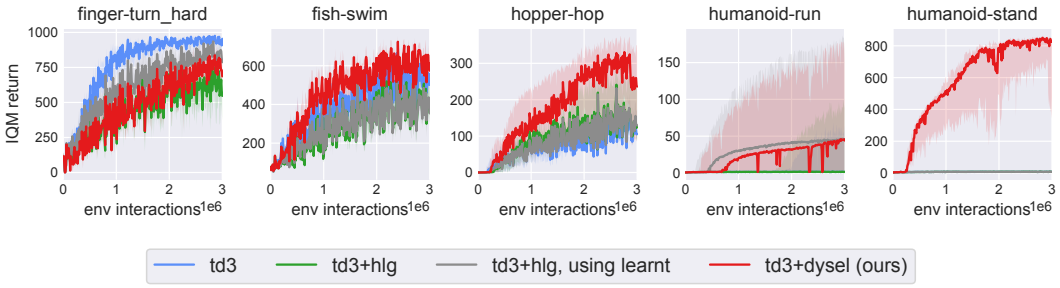

Figure 4: IQM return comparing DySEL with vanilla HL-Gauss and vanilla HL-Gauss using learnt support interval on selected tasks.

Q4: How about using the learnt support interval as reference for vanilla HL-Gauss?
A4: Using the learnt support interval is good enough for vanilla HL-Gauss for some tasks where the support interval remains stable, but not good for tasks where the support interval gradually increases.

Since the support interval changes during training, we use the final support interval that DySEL found (i.e. the support interval at 3M steps), and use it as a referenced support interval for vanilla HL-Gauss for some tasks. We pick these tasks from the two behaviour types we mentioned in Q3. The results are shown in Figure 4.

In tasks such as *finger-turn_hard* and *humanoid-run*, utilising our learnt support intervals enhances the performance of vanilla HL-Gauss. Whilst this indicates that our learnt intervals provide a robust reference for some environments, this falls short in some tasks. Specifically, for tasks where it is more advantageous to allow the support interval to gradually increase over time, namely in *fish-swim* and *humanoid-stand*, DySEL outperforms or remains competitive vanilla HL-Gauss even when using our learnt support interval. We show that DySEL is able to adapt to both behaviours and to find a balanced support interval for training. This highlights a practical advantage of DySEL: it adapts to both regimes and removes the need to tune the support interval.

Q5: How important are the components in the proposed objective?
A5: All components of the proposed objective in Equation (16) are important.

Whilst Section 4 outlines the theoretical justification for these components, this section empirically investigates their individual necessity. To this end, we conduct an ablation study examining two specific scenarios: firstly, the removal of the absolute width penalty, thereby allowing the support interval to expand unrestrictedly; and secondly, the omission of the mass constraint, which reduces the constrained optimisation to an unconstrained problem. Empirical results demonstrate that both elements are critical for stability. For both ablation experiments, we maintain the same balancing coefficient, α, as utilised in Q1. The corresponding results are illustrated in Figure 5.

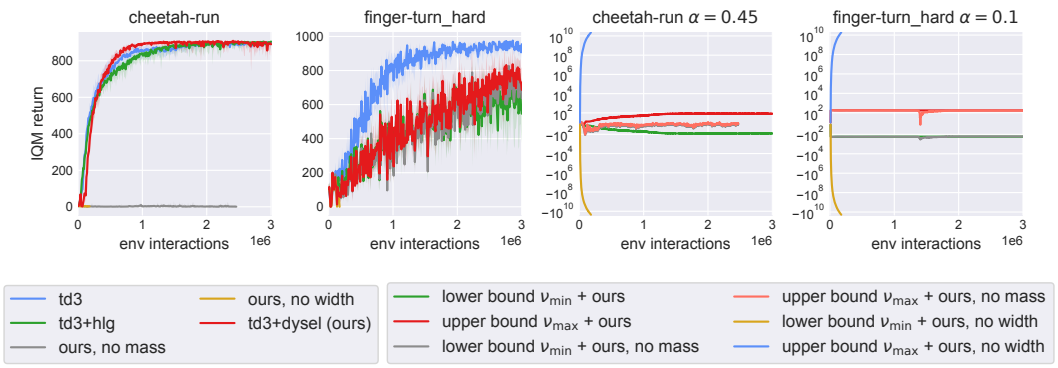

Figure 5: IQM return and its support interval evolution whilst removing components. No-mass means we remove the mass enforcing adversarial penalty. No-width means we remove the absolute width penalty. In both cases, we can show that both elements are critical for stability. For support interval plots on the right, y-axis is in sym-log scale, because the support interval goes to exceedingly large values and diverges without the width penalty.

Observing the results in Figure 5, in the absence of the absolute width penalty, the algorithm diverges. We hypothesise that this is because, rather than learning the target distribution, the cross-entropy loss artificially minimises its objective by exploiting the supports (i.e. the tendency to expand Z instead of truly matching the distribution), expanding the support interval towards infinity as a way to minimise its loss. That is, it artificially minimises the c_i parameter in Equation (5) by having a larger Z coverage, which in turn affects the learnt support interval by making $|\nu_{\text{min}}|, |\nu_{\text{max}}|$ exceedingly large.

Without the Lagrangian penalty enforcing sufficient mass coverage, the natural tendency of the cross-entropy term to arbitrarily widen the support interval is vastly overpowered by the absolute width penalty. This is acceptable for some tasks such as *finger-turn_hard*, where the supports remained relatively unchanged (i.e. smaller α) in Figure 3. However, for tasks where further expansion of the support interval was necessary, such as *cheetah-run*, the absence of the adversarial penalty's expansion mechanism would result in a support interval too narrow to cover the full target distribution, resulting in insufficient probability mass coverage at some point during training.

Intuitively, value targets effectively lie in a bounded interval with probability one; increasing the parametrisation support with a fixed bin size eventually places all targets in a single bin, which trivially yields zero cross-entropy. The width penalty is what prevents this degenerate widening.

6 Related Works

Distributional RL and Classification-Based Value Learning. The conceptual groundwork for distributional RL was established by Morimura et al. (2010a;b), who explored both parametric (Morimura et al., 2010a) and non-parametric (Morimura et al., 2010b) approaches. Indeed, distributional RL has been shown to be a critical component in modern state-of-the-art RL algorithms such as BRO (Nauman et al., 2024) and SimbaV2 (Lee et al., 2025). The seminal work on C51 (Bellemare et al., 2017) introduced the categorical DQN, which discretises the return distribution over a fixed set of atoms or bins. Despite their success, the reliance on a pre-defined support interval introduces a significant dependency that cannot be readily determined *a priori*. On the other hand, HL-Gauss, proposed in (Imani & White, 2018; Imani et al., 2024), reframes the regression task as a classification problem by smoothing target values into a Gaussian distribution over a histogram and was adopted to RL target values in (Farebrother et al., 2024). However, the efficacy of HL-Gauss in RL is still constrained by the requirement of a static support interval. A closely related encoding is the two-hot representation (Farebrother et al., 2024), which places target mass on the two nearest bins; HL-Gauss can be seen as its Gaussian-smoothed generalisation. In this work, we propose to extend the HL-Gauss framework and learn the support interval dynamically.

Dynamic Supports. The most related work to DySEL is the work in Chen et al. (2025), which also has attempted to derive a principled objective based on the upper bound of the mean-squared Bellman error. Whilst they opted for a pure minimisation approach, our approach significantly diverges from theirs where we turn the objective into a constrained optimisation problem, ensuring that the support expands to cover the mass whilst contracting to maintain a tight theoretical bound.

Non-Stationarity and Value Scaling. A primary challenge in RL is the non-stationarity of target values. As the policy improves, the distribution of returns typically shifts and expands, rendering a fixed support interval either too narrow, leading to truncation, or too broad, reducing the resolution per bin. Traditional methods have addressed value scaling through techniques such as PopArt (van Hasselt et al., 2016; Hessel et al., 2019), which uses mean-side statistics to normalise targets. Whilst PopArt is effective for scalar regression, it does not naturally extend to the preservation of distributional shape in categorical critics.

Quantile Regression. Implicit Quantile Networks (IQN) (Dabney et al., 2018a) and Fully Parameterized Quantile Function (FQF) (Yang et al., 2019) avoid using categorical supports completely by learning quantile locations directly, relieving the practitioner from specifying a support range at the price of tail quantiles being hard to estimate. Whilst these types of approaches seem superior as they avoid using fixed pre-defined supports, they suffer from quantile crossing, which may lead to invalid distributions (Zhou et al., 2020), and a lack of inductive bias (Bellemare et al., 2017; Imani & White, 2018; Imani et al., 2024; Sun et al., 2025), which may lead to poor performance. DySEL focuses on the categorical-critic branch by making HL-Gauss supports adaptive.

7 Limitations and Conclusion

The dynamic adaptation of support interval for categorical critics poses a significant challenge for classification-based categorical value learning (Farebrother et al., 2024; Imani & White, 2018; Imani et al., 2024). Despite the advantages of histogram-based losses, their reliance on a pre-defined and fixed support interval remains problematic when confronted with the non-stationary and stochastic target values inherent to RL tasks, partially due to the amount of hyperparameter tuning required.

In this work, we propose the Dynamic Support Endpoint Learning (DySEL) algorithm, where we take the first steps towards learning the support interval dynamically. We link the standard mean-squared Bellman error and HL-Gauss to reformulate the learning of support interval from a pure minimisation task into a constrained optimisation problem. By structuring it as a min-max game, the primary objective compresses the support whilst the Lagrangian adversary safeguards sufficient mass to mitigate truncation. Our approach performs comparably to vanilla HL-Gauss on most tasks and improves over it on several, most notably the *humanoid* tasks, whilst eliminating the need to pre-specify the support interval.

Whilst DySEL successfully learns dynamic supports for classification-based categorical value learning, several limitations remain. Notably, the theoretical guarantee of our proposed optimisation relies on a minimax Lagrangian game to enforce the $(1 - Z)$ truncation constraint. Although this automates the penalty weight λ, the stability of the saddle-point optimisation is sensitive to the dual learning rate. Aggressive dual updates can induce oscillatory behaviour in the bounds network during early training phases. Replacing plain dual ascent with a PID-controlled Lagrangian update (Stooke et al., 2020) could damp these oscillations. Another interesting limitation is that our learnt support interval is almost symmetric. Penalising each bound separately could decouple the two and allow an asymmetric shifted support better matched to the true return range, which we leave to future work. Another potential future direction is exploring adaptive Gaussian σ, allowing finer control over support compression and expansion, and naturally accommodating increasing-γ schedules (François-Lavet et al., 2015). Finally, our evaluation is DM-Control-focused; broader benchmarks are left to future work.

Acknowledgments

This work was supported by JST Moonshot R&D Grant Number JPMJPS2011 and JSPS KAKENHI Grant Number JP25K03176.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. URL https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://github.com/openai/gym.

Chen Chen, Daming Shi, Yixiu Mao, and Xiangyang Ji. Adaptive HL-gaussian: A value function learning method with dynamic support adjustment, 2025. URL https://openreview.net/forum?id=cGu5LtGcRD.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018a.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018b.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *International Conference on Learning Representations (ICLR)*, 2023.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep rl. In *International conference on machine learning (ICML)*, 2024.

Vincent François-Lavet, Raphaël Fonteneau, and Damien Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *NIPS 2015 Deep Reinforcement Learning Workshop*, 2015.

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Ehsan Imani and Martha White. Improving regression performance with distributional losses. In *International conference on machine learning (ICML)*, 2018.

Ehsan Imani, Kai Luedemann, Sam Scholnick-Hughes, Esraa Elelimy, and Martha White. Investigating the histogram loss in regression, 2024. URL https://arxiv.org/abs/2402.13425.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. In *International Conference on Machine Learning (ICML)*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Kuznetsov. Controlling overestimation bias with truncated quantile critics. In *International Conference on Machine Learning (ICML)*, 2020.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2025.

Clare Lyle, Marc G. Bellemare, and Pablo Samuel Castro. A comparative analysis of expected and distributional reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013.

Tetsuro Morimura, Hirotaka Hachiya, Masashi Sugiyama, Toshiyuki Tanaka, and Hisashi Kashima. Parametric return density estimation for reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010a.

Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Nonparametric return distribution approximation for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2010b.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Frank WJ Olver, Daniel W Lozier, Ronald F Boisvert, and Charles W Clark. *NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY, 2010. ISBN 978-0-521-19225-5.

Mark Rowland, Marc Bellemare, Will Dabney, Remi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G. Bellemare, and Will Dabney. Statistics and samples in distributional reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning (ICML)*, 2020.

Ju Sun, Qing Qu, and John Wright. When are nonconvex problems not scary?, 2016. URL https://arxiv.org/abs/1510.06096.

Ke Sun, Yingnan Zhao, Wulong Liu, Bei Jiang, and Linglong Kong. Distributional reinforcement learning with regularized wasserstein loss. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Ke Sun, Yingnan Zhao, Enze Shi, Yafei Wang, Xiaodong Yan, Bei Jiang, and Linglong Kong. Intrinsic benefits of categorical distributional loss: Uncertainty-aware regularized exploration in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. In *Software Impacts*, 2020.

Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Neil A. Weiss. A course in probability. In *A Course in Probability*, pp. 380–383. Pearson / Addison-Wesley, 2005.

Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. Fully parameterized quantile function for distributional reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Fan Zhou, Jianing Wang, and Xingdong Feng. Non-crossing quantile regression for distributional reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Zhihui Zhu, Qiuwei Li, Gongguo Tang, and Michael B. Wakin. The global optimization geometry of low-rank matrix optimization. *arXiv:1703.01256*, 2017. URL https://arxiv.org/abs/1703.01256.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Experiments Implementation Details & Hyperparameters

Our implementation is done in JAX (Bradbury et al., 2018). Specifically, the versions of important libraries we use in our experiments are: JAX 0.6.2 (Bradbury et al., 2018), MuJoCo 3.3.7 (Todorov et al., 2012), DeepMind Control Suite 1.0.34 (Tunyasuvunakool et al., 2020) and gym 0.23.1 (Brockman et al., 2016). Nevertheless, we do expect similar empirical performance even if the library versions do not exactly follow ours. We also plan to release our source code upon publication.

Shared across all algorithms. The replay buffer size is set to 10^6, and the discount factor γ is set to 0.99. The target update rate τ for target network(s) is 0.005. We have initial random collect steps of 10000. To ensure a fair comparison, all methods employ a batch size of 256, and all neural networks used two hidden layers consisting of 256 units each. All methods use ReLU (Nair & Hinton, 2010) as activation function. We use Adam (Kingma & Ba, 2015) as optimiser for all neural networks with the learning rate set to 0.0003.

TD3. We implemented TD3 (Fujimoto et al., 2018) closely following excellent public repositories such as JAXRL `https://github.com/ikostrikov/jaxrl` and high-replay-ratio (D'Oro et al., 2023) `https://github.com/proceduralia/high_replay_ratio_continuous_control`. We use the default hyperparameters provided in these implementations.

HL-Gauss. For HL-Gauss, we follow the implementation listed in the appendix in Farebrother et al. (2024). Whilst the base algorithm was DQN in the original work, we adapted the methodology to TD3 for our experiments involving continuous control tasks. For the critic network, the last layer outputs k values. Furthermore, whereas the baseline employs a fixed support interval of $[\nu_{\text{min}}, \nu_{\text{max}}] = [-10, 10]$ for Atari benchmarks (Mnih et al., 2013), we expanded this interval to $[\nu_{\text{min}}, \nu_{\text{max}}] = [-100, 100]$ to accommodate the broader return scales inherent to the DeepMind Control Suite. This range is a principled rather than arbitrary choice: DM Control rewards lie in $[0, 1]$ per step, so with a discount factor of $\gamma = 0.99$ the discounted return is bounded by $1/(1-\gamma) = 100$. The interval $[-100, 100]$ therefore covers the full range of attainable values and is essentially free of truncation bias for the baseline; the limitation it does incur is resolution, since realised returns for most tasks and policies remain well below this bound, leaving much of the fixed support—and hence the available bin resolution—unused.

Additionally, the original HL-Gauss work utilises $k = 51$ bins, following the convention established by C51 (Bellemare et al., 2017). Given that this configuration would yield relatively low granularity across our expanded support (equating to a bin width of nearly 4), we increased the resolution by expanding the number of bins to $k = 128$. Finally, consistent with the original authors, we apply a sigma-to-bin-width ratio of 0.75 to determine the standard deviation of the truncated Gaussian. A reference implementation of TD3+HL-Gauss can be found in listing 1.

TD3+DySEL. We implement our support interval network as a standard three-layer MLP, with hidden dims (256, 256) and the final layer outputting two values. The final layer is initialised according to the initialising support interval, which for all tasks we set to $[-10, 10]$. The outputted two values were sorted to ensure that the support interval is valid. The first value is always the lower bound ν_{min} and the second value is always ν_{max}. For the specific adversary training, we use Adam (Kingma & Ba, 2015) with learning rate of $1e-03$. A reference implementation of our proposed min-max optimisation in Equation (16) can be found in listing 2.

Since our support intervals are dynamically learnt, in Equation (16) the remaining relevant hyperparameter is α that balances the minimisation of absolute width versus the minimisation of cross-entropy. We use $k = 128$ bins and sigma-to-width ratio = 0.75, same as in our HL-Gauss imple-

mentation. Furthermore, we use $\epsilon = 0.005$ for all tasks for our proposed min-max optimisation Equation (16) and did not tune any further.

Selection of α. For each task, we selected α via a grid search over the range $[0.1, 0.2, 0.3, 0.4, 0.45, 0.5]$; the chosen per-task values are listed in Table 1. We found this range sufficient across all tasks, with smaller α favouring a stable, narrow support and larger α permitting gradual expansion, as characterised by the sensitivity analysis in Figure 7. All other hyperparameters are shared with the HL-Gauss baseline, so α is the only method-specific quantity tuned; in effect, DySEL replaces the task-specific tuning of the support interval $[\nu_{\text{min}}, \nu_{\text{max}}]$ required by HL-Gauss with the tuning of a single scalar α. We note that α was selected on the same seeds subsequently used to report results in Figure 2; the per-task performance of DySEL should therefore be read as an optimistic (upper-bound) estimate rather than an unbiased one. We report the full sensitivity of performance to α in Figure 7 so that the variation across the searched range can be assessed directly.

Table 1: Hyperparameter α that balances the weights of absolute width penalty versus the weights of cross-entropy when minimising.

Task	cheetah-run	finger-turn_hard	fish-swim	hopper-hop
α	0.45	0.1	0.5	0.2
Task	hopper-stand	humanoid-run	humanoid-stand	humanoid-walk
α	0.1	0.2	0.5	0.5
Task	quadruped-run	quadruped-walk	walker-run	
α	0.5	0.3	0.5	

One important implementation detail is that we clip the minimum value of σ. If we do not perform this clipping, when minimising $(1 - Z)$, the network may cheat by making σ smaller in the Z formula (restated from Equation (2) below for convenience) especially at the beginning steps of training. Specifically, we clip the minimum to $\sigma = 0.3$.

$$\text{(Equation (2) restated) } Z = \frac{1}{2}\left(\text{erf}\left(\frac{\nu_{\text{max}} - \mu}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{\nu_{\text{min}} - \mu}{\sqrt{2}\sigma}\right)\right)$$

B Additional Experiments

As mentioned in Section 5, three choices of hyperparameters, the initialising support interval, α, and number of bins k remain. We show that α has a non-trivial effect on the training performance for some tasks. Furthermore, we also tried different initialising support intervals for the task, which generally has not much effect apart from some harder tasks.

B.1 Ablation on the initialising support interval

For all tasks, we previously used an initialising support interval of $[-10, 10]$. Empirically, we found that starting from an excessively large initialising support interval can lead to unstable training. We hypothesise that this is because of absolute width penalty of the proposed objective Equation (16), which would be too large to appropriately learn the cross-entropy loss. Nevertheless, we find the ability to be able to start with arbitrary narrower initialising support interval to be more important in practice. To this end, we compared using three different sets of initialising support interval: $[-1, 1]$, $[-5, 5]$ and $[-10, 10]$ and train on some tasks. The results are shown below.

We can observe that the support interval evolution is generally the same regardless of the initialising support interval. Only for *humanoid* tasks where initially there is some instability. This instability may encourage the policy to explore more. We leave this for future work.

```
import jax.numpy as jnp

...
# next q
next_q1_logits, next_q2_logits = target_critic.apply_fn(
    target_critic.params,
    batch.next_observations,
    next_actions,
)
next_q1 = hl_gauss.transform_from_logits_to_value(next_q1_logits)
next_q2 = hl_gauss.transform_from_logits_to_value(next_q2_logits)
next_q = jnp.minimum(next_q1, next_q2)

# target q
target_q = batch.rewards + discount * batch.masks * next_q
target_probs = hl_gauss.transform_from_value_to_probs(target_q)

def critic_loss_fn(trainable_params):
    q1_logits, q2_logits = critic.apply_fn(
        trainable_params,
        batch.observations,
        batch.actions,
    )

    # cross entropy
    ce_loss1 = optax.softmax_cross_entropy(q1_logits, target_probs)
    ce_loss2 = optax.softmax_cross_entropy(q2_logits, target_probs)
    critic_loss = (ce_loss1 + ce_loss2).mean()
...
```

Listing 1: A reference implementation of HL-Gauss (Imani & White, 2018) in JAX.

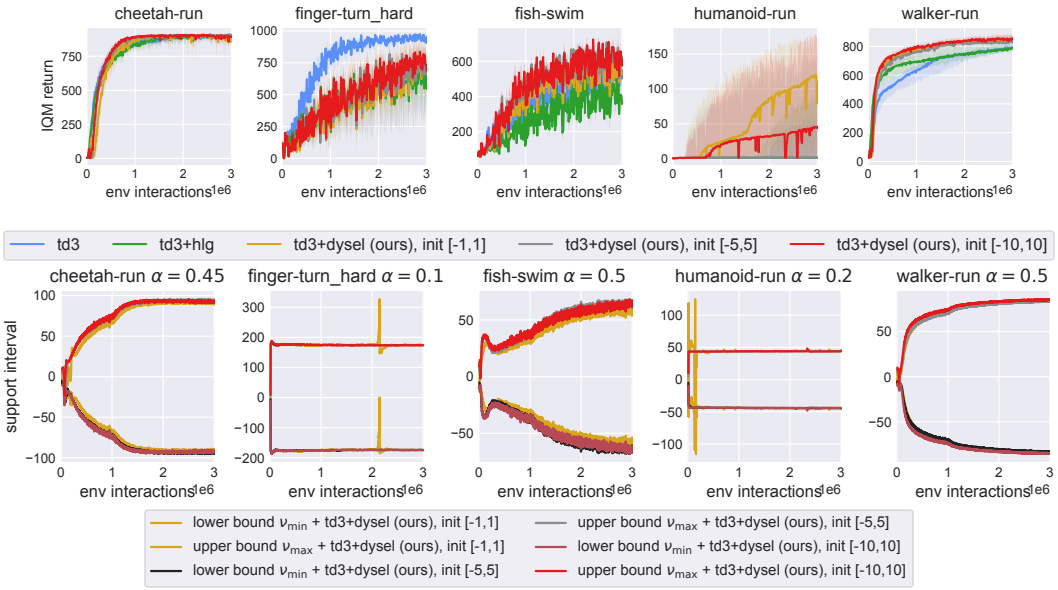

Figure 6: IQM and bounds evolution of different initialisation. For all tasks, the supports evolution are similar, but those initialised with smaller initialising supports have some instability.

B.2 Ablation on the sensitivity to α

For all tasks, we searched for different α, which balances between the absolute width penalty and cross-entropy in Equation (16). For some tasks, smaller α is required to maintain a stable support

```python
import jax.numpy as jnp

...

def critic_loss_fn(trainable_params):
    ...

    # widths
    lower, upper = q_bounds[:, 0], q_bounds[:, 1]
    M = jnp.maximum(jnp.abs(lower), jnp.abs(upper))
    first_term = M * bounds_weight + ce_loss / bounds_weight

    # bias penalty
    projection_bias = 1.0 - target_z
    second_term = adv_lambda * jax.nn.relu(projection_bias - epsilon)

    # total
    critic_loss = first_term + second_term
    critic_loss = critic_loss.mean()
    ...

# we only need the (1-Z-epsilon) for training adversarial loss
projection_bias = 1.0 - target_z
violation = projection_bias - epsilon

def adv_loss_fn(trainable_params):
    adv_lambda = adversary.apply_fn(trainable_params)
    loss = -(adv_lambda * violation).mean()

...
```

Listing 2: A reference implementation of our proposed objective in Equation (16) in JAX.

interval; whilst for some tasks where lower granularity is required, larger α is preferred. We include sensitivity to α on some tasks in Figure 7.

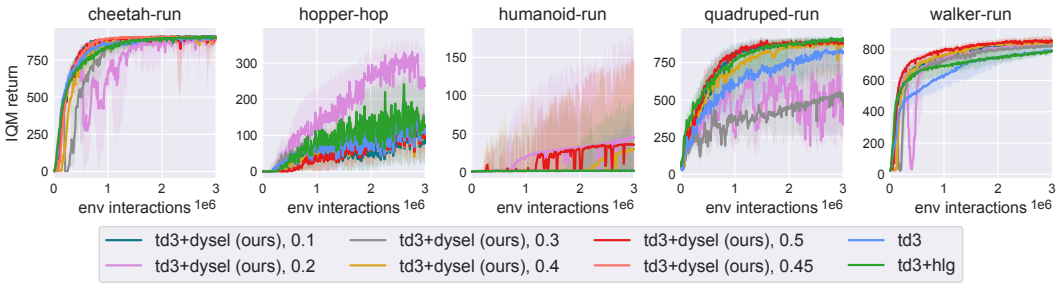

Figure 7: IQM and bounds evolution of different α.