Parameter-Efficient Distributional RL via Normalizing Flows and a Geometry-Aware Cramér Surrogate

Simo Alami C.[1], Rim Kaddah[2], Marie-Paule Cani[1], Jesse Read[1]
[1]LIX, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris,
[2]IRT SystemX
{mohamed.alami-chehboune, marie-paule.cani, jesse.read}@polytechnique.edu,
rim.kaddah@irt-systemx.fr

Distributional Reinforcement Learning (DistRL) improves upon expectation-based methods by modeling full return distributions, but standard approaches often remain far from parsimonious. Categorical methods (e.g., C51) rely on fixed supports where parameter counts scale linearly with resolution, while quantile methods approximate distributions as discrete mixtures whose piecewise-constant densities can be wasteful when modeling complex multi-modal or heavy-tailed returns. We introduce NFDRL, a parsimonious architecture that models return distributions using continuous normalizing flows. Unlike categorical baselines, our flow-based model maintains a compact parameter footprint that does not grow with the effective resolution of the distribution, while providing a dynamic, adaptive support for returns. To train this continuous representation, we propose a Cramér-inspired, geometry-aware distance defined over probability masses obtained from the flow. We show that this distance is a true probability metric, that the associated distributional Bellman operator is a $\sqrt{\gamma}$-contraction, and that the resulting objective admits unbiased sample gradients—properties that are typically not simultaneously guaranteed in prior PDF-based DistRL methods. Empirically, NFDRL recovers rich, multi-modal return landscapes on toy MDPs and achieves performance competitive with categorical baselines on the Atari-5 benchmark, while offering substantially better parameter efficiency.

1. Introduction

Traditional reinforcement learning (RL) algorithms aim to estimate the expected return from a given state or state-action pair [1]. However, this expectation provides only a partial view of the underlying return distribution, omitting critical information about uncertainty, risk, and variability. Distributional reinforcement learning (DistRL) addresses this limitation by modelling the full return distribution, providing a richer and more informative signal for decision-making.

Why distributional RL? Modeling the full return distribution can improve learning and control beyond what is possible with expectation-only value functions. Prior work reports that distributional critics provide richer training signals and can yield stronger empirical performance, while also enabling risk-aware decision making through functionals of the return distribution (e.g., tail probabilities or CVaR) rather than a single mean value [2**? ?**]. In addition, explicit access to distributional information supports robustness considerations and uncertainty-aware policies in stochastic environments, where variability and rare outcomes matter.

A critical challenge in DistRL is achieving an expressive representation of the return distribution without incurring excessive computational or parametric costs. Early approaches such as Categorical DQN (C51) [2] approximate the return distribution using a histogram over a fixed discrete support. This formulation suffers from a fundamental lack of parsimony: the model's parameter count scales linearly with the support resolution. To achieve high precision, one must drastically increase the number of atoms, inflating the model size. Furthermore, the fixed support imposes a

Third Conference on Parsimony and Learning (CPAL 2026).

hard bound on returns, limiting applicability in domains with dynamic or highly variable rewards, such as robotic control [3] or finance [4].

Existing alternatives face theoretical or representational trade-offs. Bellemare et al. [2] showed that the Distributional Bellman operator is a contraction in the Wasserstein metric. However, C51 is trained using KL divergence because empirical Wasserstein minimization suffers from biased sample gradients. This creates a disconnect between the contraction metric and the optimization objective. Moreover, using the KL divergence is also a major drawback in the unbounded setting, where predicted and target distributions may have negligible overlap, resulting in poor gradient signals. Subsequent quantile-based methods like QR-DQN [5] and IQN [6] or FQF [7] address the support limitation by learning quantile locations and train with quantile regression losses (pinball, often with a Huber smoothing). By construction, quantile models handle unbounded returns and optimize a surrogate consistent with the 1-Wasserstein distance in one dimension. While highly effective, these methods approximate distributions as mixtures of Dirac deltas (step functions). This representation can be representationally inefficient for smooth, multi-modal, or heavy-tailed distributions, often requiring a large number of quantiles to approximate simple densities [8].

A complementary approach is to model the return distribution through a continuous probability density. Continuity matters for two reasons. First, it yields an adaptive support and avoids committing to a fixed return range, which is a limitation of categorical methods in domains with dynamic or heavy-tailed rewards. Second, a smooth density facilitates the computation of risk-sensitive criteria (e.g., CVaR or tail probabilities) by direct integration, without introducing an additional discretization level or increasing the number of parameters to refine resolution. This motivates continuous density models for DistRL. While recent work has explored normalizing flows for DistRL [9], achieving a training objective that is both stable in RL and theoretically consistent (contraction and unbiased sample gradients) remains challenging.

Contribution. We propose NFDRL, a parameter-efficient DistRL method that models return distributions using continuous normalizing flows [10]. To train this continuous representation, we introduce a geometry-aware surrogate of the Cramér distance defined over probability masses. Our approach offers three distinct advantages. First, it ensures **representational efficiency:** unlike categorical baselines where model size scales linearly with resolution, NFDRL maintains a compact, constant parameter footprint while offering infinite effective resolution and dynamic support. Second, it guarantees **theoretical soundness**; we prove that our mass-based distance is a true metric that induces a $\sqrt{\gamma}$-contraction in the distributional Bellman operator and admits unbiased sample gradients, resolving the inconsistencies of prior PDF-based approaches. Finally, the method demonstrates superior **expressivity**, capturing complex, multi-modal return landscapes that quantile baselines often fail to resolve.

We validate our approach on toy MDPs and the Atari-5 benchmark [11]. We show that NFDRL achieves performance competitive with categorical baselines while utilizing significantly fewer parameters, offering a theoretically rigorous and parsimonious alternative for Distributional RL.

Scope and positioning. Our objective is not to propose a replacement for strong quantile-based agents such as IQN on Atari, but to study a parsimonious *continuous* return representation with well-controlled metric properties. Quantile methods represent $\eta^\pi(x, a)$ as a finite mixture of Dirac masses; they are highly effective for control, but their discrete nature can make fine-grained characterization of distributional *shape* (e.g., multimodality, tail behavior) dependent on the number of quantiles. In contrast, NFDRL models a smooth density with adaptive support and a constant parameter footprint, and is paired with a Cramér-inspired objective for which we prove (i) metric axioms, (ii) a $\sqrt{\gamma}$-contraction of the distributional Bellman operator, and (iii) unbiased sample gradients. We view NFDRL as a complementary tool with different trade-offs (parsimony, continuous PDFs, and theoretical consistency), particularly relevant when downstream risk-sensitive quantities depend on the detailed shape of the return distribution.

2. Background

We consider the standard RL setting, where the interaction between an agent and an environment is modelled as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, P, \gamma)$ [1]. Here, $\mathcal{X}$ and $\mathcal{A}$ denote the state and action spaces, $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $P(\cdot|x, a)$ is the transition kernel, and $\gamma \in (0, 1)$ is a discount factor. A policy $\pi(\cdot|x)$ maps a state $x \in \mathcal{X}$ to a distribution over actions.

Under a fixed policy π, the return $G^\pi(x)$ is defined as the random variable representing the discounted sum of rewards collected along a trajectory starting from state x:

$$G^\pi(x) := \sum_{t=0}^{\infty} \gamma^t R(x_t, a_t), \quad x_0 = x, a_t \sim \pi(\cdot|x_t), x_{t+1} \sim P(\cdot|x_t, a_t). \tag{1}$$

The goal is to estimate the expected return i.e. the action-value function Q:

$$Q^\pi(x, a) := \mathbb{E}[G^\pi(x, a)] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t), \Big|, x_0 = x, a_0 = a\right].$$

These quantities satisfy the Bellman equation:

$$Q^\pi(x, a) = \mathbb{E}[R(x, a)] + \gamma \mathbb{E}_{X' \sim P(\cdot|x,a), A' \sim \pi(\cdot|X')}[Q^\pi(X', A')]. \tag{2}$$

Distributional Reinforcement Learning

Rather than approximating only the expectation of returns, DistRL aims to model the entire distribution of $G^\pi(x)$, capturing richer information such as variance, multimodality, or higher moments. This has been shown to improve both learning dynamics and empirical performance [2].

We adopt the formulation proposed by Bellemare et al. [12], which distinguishes between the random return $G^\pi(x)$ and its law $\eta^\pi(x)$, defined as:

$$\eta^\pi(x)(S) := \mathbb{P}(G^\pi(x) \in S), \quad \forall S \subseteq \mathbb{R}. \tag{3}$$

To express the Bellman recursion at the level of distributions, they introduce the concept of a push-forward function. Given a function $f : \mathbb{R} \to \mathbb{R}$ and a probability distribution η, intuitively we have, if $Z \sim \eta$, then $f(Z) \sim f_\# \eta$.

Let the bootstrap function defined as $b_{r,\gamma} : \mathbb{R} \to \mathbb{R}$ by $b_{r,\gamma}(z) := r + \gamma z$, for fixed $r \in \mathbb{R}$ and $\gamma \in (0, 1)$. Then the distributional Bellman equation over distributions can be written as:

$$\eta^\pi(x) = \mathbb{E}_{a \sim \pi(\cdot|x), X' \sim P(\cdot|x,a)}[(b_{R(x,a),\gamma})_\# \eta^\pi(X')]. \tag{4}$$

This defines the distributional Bellman operator $\mathcal{T}^\pi : \mathcal{P}(\mathbb{R})^{\mathcal{X}} \to \mathcal{P}(\mathbb{R})^{\mathcal{X}}$, where:

$$(T^\pi \eta)(x) := \mathbb{E}_{a \sim \pi(\cdot|x), X' \sim P(\cdot|x,a)}[(b_{R(x,a),\gamma})_\# \eta(X')]. \tag{5}$$

This approach provides a compact, principled notation for distributional TD methods we use.

Normalizing Flows

NF are a class of generative models that transform a simple base distribution into a more complex one via a sequence of smooth, invertible mappings [10]. Starting from a base sample $z_0 \sim p(z_0)$, a flow applies a sequence of transformations $f_k \circ \cdots \circ f_1$ to obtain z_K, whose density is computed using the change of variables formula:

$$\log p_\theta(z_K) = \log p(z_0) - \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial f_k}{\partial z_{k-1}} \right) \right|. \tag{6}$$

This allows flows to model flexible densities while retaining exact likelihoods and differentiability.

3. Normalizing Flows for Distributional RL

We present our NF-based model for return distribution estimation. While normalizing flows can map in either direction—base to return distribution or vice versa—RL constraints guide this choice. We first discuss both options and justify the appropriate direction, then describe the model architecture and how the distributional Bellman operator is implemented using pushforward distributions.

3.1. Modeling the Forward Flow: Sampling Returns and Evaluating Densities

Let $\mathcal{U}$ be a base distribution with full support over $\mathbb{R}$, and let $z \sim \mathcal{U}$. Let f be a diffeomorphism from $\mathbb{R}$ to itself (i.e., a bijective, differentiable mapping with a differentiable inverse).

The flow f can be interpreted in two equivalent ways, (i) either as mapping return outcomes $y \sim \eta^\pi(x, a)$ into the base space: $z = f(y)$, or (ii) as mapping base samples $z \sim \mathcal{U}$ to return outcomes via: $y = f(z)$, where $y \sim \eta^\pi(x, a)$.

Mapping from return outcomes to base samples (i.e., modeling the inverse flow) is attractive because it allows direct likelihood evaluation of observed returns, which is beneficial for maximum likelihood training. However, in RL, we do not observe true return samples $y \sim \eta^\pi(x, a)$ directly. We must construct them through bootstrapping. This makes the inverse mapping impractical in our context. Instead, we opt to model the forward flow, transforming base noise samples $z \sim \mathcal{U}$ into return outcomes $y = f_\theta(z)$, which allows us to generate samples from $\eta^\pi(x, a)$ and define pushforward distributions suitable for implementing the distributional Bellman operator.

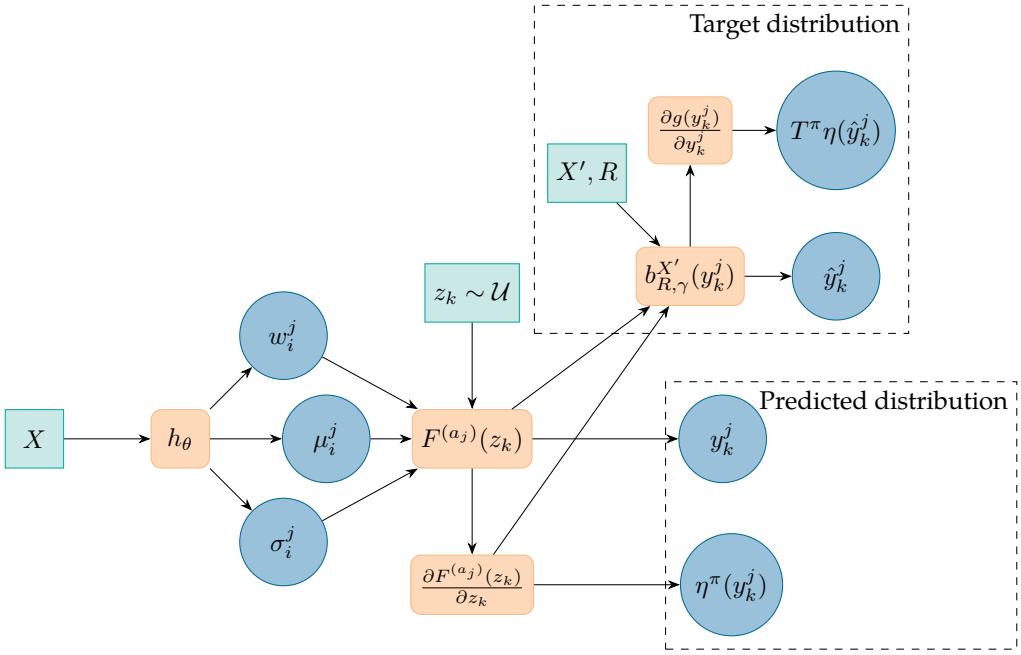


Figure 1: Architecture of the conditional flow model. A neural network h_θ maps each state x to parameters $\{(w_i^j, \mu_i^j, \sigma_i^j)\}_{i=1}^n$ for each action $a_j \in \mathcal{A}$, defining a mixture CDF $F^{(a_j)}$. This CDF acts as a flow that transforms base noise samples $z_k \sim \mathcal{U}$ into return samples $y_k^j = F^{(a_j)}(z_k)$. The change of variable formula then makes use of the flows derivative to approximate the return distribution $\eta^\pi(y_k^j)$. To estimate the target distribution, the bootstrap function is implemented as a flow and takes the reward and next state as input. It finally outputs target distribution $T^\pi\eta$.

While choosing to represent return samples as $y = f_\theta(z)$, with $z \sim \mathcal{U}$, makes sampling straightforward, it is also possible to compute the probability density of a given return value using the change of variable formula. Let $f_\theta : \mathbb{R} \to \mathbb{R}$ be a flow parameterized by θ, and $p_\mathcal{U}$ the density of the base

distribution $\mathcal{U}$. Then for $y = f_\theta(z)$, we have:

$$\log \eta^\pi(x,a)(y) = \log\left(p_{\mathcal{U}}(z)\left|\frac{\partial y}{\partial z}\right|^{-1}\right) = \log\left(p_{\mathcal{U}}(z)\left|\frac{\partial f_\theta(z)}{\partial z}\right|^{-1}\right) = \log p_{\mathcal{U}}(z) - \log\left|\frac{\partial f_\theta(z)}{\partial z}\right| \quad (7)$$

This formulation yields a closed-form expression for the learned density. Computing the log-density of sampled returns requires only the base log-density and the derivative of the flow function. Choosing a flow function with easily computable derivatives is crucial.

3.2. A CDF-Based Flow Architecture for Conditional Return Modeling

We propose an architecture in which a conditional flow function, constructed from a CDF, maps base samples to return values. Since return distributions $\eta^\pi(x,a)$ are one-dimensional, we design a 1D flow $F_\theta(x,a) : \mathcal{Z} \to \mathbb{R}$, where z is drawn from a fixed base distribution, and y = $F_{\theta(x,a)}(z)$ is a return sample conditioned on the state-action pair (x,a).

To model this conditional transformation, we use a neural network h_θ that takes as input the state x and outputs, for each discrete action $a_j \in \mathcal{A}$, the parameters of a Gaussian mixture distribution: $\{(w_i^{(j)}, \mu_i^{(j)}, \sigma_i^{(j)})\}_{i=1}^n$. These parameters define a mixture CDF, and with $z \sim \mathcal{U}$, we get:

$$y^{a_j} = F^{(a_j)}(z) = \sum_{i=1}^{n} w_i^{(j)} \Phi\left(\frac{z - \mu_i^{(j)}}{\sigma_i^{(j)}}\right)$$

Why CDF-based flows? Since $\eta^\pi(x,a)$ is one-dimensional, we adopt a CDF-based parameterization to balance expressivity and numerical stability. Concretely, we model the conditional transformation through a Gaussian-mixture CDF, which is strictly monotone by construction and whose derivative is a Gaussian-mixture PDF. This yields closed-form density evaluation via the change-of-variables formula while retaining a flexible, non-linear family in 1D.

Although this CDF does not admit a closed-form inverse in general, strict monotonicity enables robust inversion by 1D binary search when evaluating likelihoods. In our setting, the forward map $z \mapsto y$ is cheap and is the only operation required for sampling-based Bellman updates; the inverse is used only for density evaluation and can be computed reliably with logarithmic-time search. Finally, we emphasize that this design choice is motivated by tractability in the 1D return setting rather than by universal superiority over other flow families (e.g., affine/squared flows or spline flows). Different parameterizations may be preferable under different computational or modeling constraints; we therefore present Gaussian-mixture CDF flows as a simple and expressive choice well suited to our use case.

In contrast, our CDF-based approach naturally supports smooth, continuous outputs and avoids discretization altogether. It is conceptually closest to the method in [13], which uses a logistic mixture CDF composed with an inverse sigmoid. Like ours, it supports expressive, invertible transformations without sacrificing differentiability.

3.3. Rescaling the Flow Output: Addressing the Bounded Support of CDFs

Because the chosen flow function is based on a CDF, its output is restricted to $[0,1]$. However, one of our goals is to allow the predicted return distribution to be continuous and adaptable, reflecting the nature of real-world return values in RL. To overcome this limitation, we introduce an additional transformation step that rescales the output of the CDF-based flow.

h_θ is extended to output an additional parameter $G^{\max}_{(x,a)}$, which defines the upper bound of the return range for a given state-action pair. The model can hence flexibly adapt the support of the predicted distribution without manual specification. In this case, we apply an affine transformation to map the CDF output $y \in [0,1]$ to the desired return range: $f(y) = 2 \cdot y \cdot G^{max} - G^{max}$. The function f is considered as a flow function that comes after the first CDF flow F.

We are now composing two flows (F and f) and as per (6), this transformation introduces an additional Jacobian term in the log-likelihood of (7):

$$\log \eta^{\pi}(x,a)(y) = \log p_{\mathcal{U}}(z) - \log \left| \frac{dF_{\theta}(z)}{dz} \right| - \log |2 \cdot G^{max}|. \tag{8}$$

3.4. Constructing the RL Target Distribution

Before proceeding, we clarify terminology: in normalizing flows, the target distribution is the flow's output. In our RL setting, this is the predicted return distribution for a state-action pair. During training, it is compared to a separate RL target distribution, which serves as the learning signal. Unless otherwise noted, target distribution refers to this RL target.

We begin by reviewing how target distributions are commonly constructed in DistRL. We then show why directly applying existing methods to our flow-based model does not yield a valid distribution. Finally, we propose a principled solution by introducing a *target flow* that enables a coherent learning objective within our framework.

As shown in (5), the Bellman operator T^{π} applies the bootstrap function $b_{r,\gamma}(y) = r + \gamma y$ to samples $y = F_{\theta}(z)$. However, this scaling distorts probability mass, breaking the change-of-variables formula. **Our key contribution is to treat $b_{r,\gamma}$ as a flow layer**. Being affine and invertible, it integrates naturally into the model, preserving normalization. The full composed flow then yields a valid target distribution under the change-of-variable formula.

Let $F_{x',a'}$ denote the flow that predicts the return distribution for the next state-action pair, and let $\tilde{y} = g(y) = r + \gamma y$ be the bootstrap flow. We now have a composition of 3 flows (F, f and g) and applying (6), The target log-density becomes:

$$\log T^{\pi}\eta(x,a)(\tilde{y}) = \log p_z(z) - \log \left| \frac{\partial F_{x',a'}(z)}{\partial z} \right| - \log |2 \cdot G^{max}| - \log \left| \frac{\partial g(F_{x',a'}(z))}{\partial F_{x',a'}(z)} \right| \tag{9}$$

$$= \log p_z(z) - \log \left| \frac{\partial F_{x',a'}(z)}{\partial z} \right| - \log |2 \cdot G^{max}| - \log(\gamma). \tag{10}$$

This approach guarantees that the target distribution is properly normalized, thanks to the compositionality of flows and the use of the change-of-variable formula. Additionally, the extra terms $\log(\gamma)$ and $\log |2 \cdot G^{max}|$ are constant with respect to z and do not incur computational overhead.

Applying the target flow alone does not yield a complete RL target distribution suitable for training, as the predicted distribution $\eta^{\pi}(x,a)$ and the target distribution $T^{\pi}\eta(x,a)$ are defined over different supports. As in the C51 algorithm, which requires projecting the Bellman update onto a fixed categorical support, we must ensure alignment between the two distributions to enable valid comparisons. Hence, we introduce an alignment procedure based on kernel density estimation (KDE). More specifically, we use a KDE to evaluate η^{π} on the support of $T^{\pi}\eta(\tilde{y})$, this is denoted as $\hat{\eta}^{\pi}$. The same is done for $T^{\pi}\eta$ evaluated on the support of η^{π} (denoted $\hat{T}^{\pi}\eta$)[1]. This alignment allows to estimate the target probability masses v_i required for the loss defined in Section 3.5.

We then have:

$$\mathcal{L}(\eta^{\pi}(x,a), T^{\pi}\eta(x,a)) = \sum_{i=1}^{N} D\Big(\eta^{\pi}(x,a)(y_i) \;\Big\|\; \hat{T}^{\pi}\eta(x,a)(y_i)\Big) + \sum_{j=1}^{M} D(\hat{\eta}^{\pi}(x,a)(\tilde{y}_j) \parallel T^{\pi}\eta(x,a)(\tilde{y}_j)). \tag{11}$$

Most importantly, as detailed in the appendix section F, the KDE is an essential tool allowing our proposed loss (eq. (11)) and metric (eq.(12)) to behave like a transport distance.

[1]This is an abuse of notation as η^{π} and $\hat{\eta}^{\pi}$ are the same distribution but evaluated on a different set of points. However as both are obtained using different processes; one is the direct output of the model and the other is obtained using a KDE, we use this notation to mark the difference.

Final state Gaussian reward approximation In standard TD methods, the value target at the final time step is simply the immediate reward r. In the distributional RL setting, however, we require a full target distribution. Since r is a scalar, it can be viewed as a degenerate distribution — a Dirac delta centered at r. To enable learning with continuous distributions, we approximate this Dirac using a Gaussian $\mathcal{N}(r, \sigma)$. The alignment process is illustrated in the appendix figure 2 and detailed in appendix algorithm 1.

3.5. Using the Cramér Distance as a Loss Function

Unlike C51, which uses KL divergence, our method operates on adaptable supports where KL is ill-suited. KL fails when predicted and target distributions do not overlap—common early in training—and lacks translation sensitivity, making it ineffective for disjoint supports. While the Bellman operator is a contraction under the maximal form of the Wasserstein metric [2], this metric is hard to optimize with stochastic gradients. Quantile regression [5] addresses this for the 1-Wasserstein case but requires learning quantile values, incompatible with our full-PDF approach. Instead, we use the Cramér distance, which is translation-sensitive, invariant to mass-preserving transforms while providing unbiased sample gradients—making it ideal for our setting [14]. Definitions and key properties are recalled in the appendix.

Recall that for two distributions with CDFs $F(x)$ and $G(x)$, the squared Cramér distance is:

$$l_2^2(P, Q) = \int_{-\infty}^{\infty} (F(x) - G(x))^2 \, dx$$

Computing the Cramér distance necessitates the CDFs of the predicted and target distributions which are not available. While using the available PDFs of the predicted and target distributions, it is possible to compute the CDFs then the exact Cramér distance; however the involved sorting operation on returns generates overhead and becomes tricky or ill-defined in higher dimensions, because multivariate CDFs are not straightforward, and ordering does not generalize well.

We propose a surrogate for the Cramér distance that is computed over the support of the distributions. Crucially, to preserve the contraction property of the Bellman operator, our loss is defined over probability masses rather than raw densities. We propose the following geometry-aware loss:

$$D(\eta^\pi(x, a), \hat{T}^\pi \eta(x, a)) = \left(\frac{1}{N^2} \sum_{i,j} \big(w_i - v_i\big)^2 \cdot |y_i - y_j| \right)^{1/2} \tag{12}$$

where w_i and v_i represent the discrete probability masses of the predicted and target distributions at support point y_i, respectively. This distinction is critical for theoretical convergence: when the return scale changes (e.g., by the discount factor γ), the support density scales inversely ($\propto 1/\gamma$) while the interval width scales proportionally ($\propto \gamma$). Consequently, the probability mass w_i remains invariant.

These masses can be obtained via a Riemann approximation linking the continuous PDF to the discrete support: $w_i \approx \eta^\pi(x, a)(y_i) \cdot \Delta y$ and $v_i \approx \hat{T}^\pi \eta^\pi(x, a)(y_i) \cdot \Delta y$. In the specific case where the support is generated via Monte Carlo sampling (as in our base distribution sampling), each point is implicitly assigned a constant mass of $1/N$, which is inherently scale-invariant.

This formulation, using the linear transport cost $|y_i - y_j|$, retains the scale sensitivity of the original Cramér distance ($l_2(cX, cY) = \sqrt{c}\, l_2(X, Y)$) and allows for stochastic optimization with unbiased gradients. In Appendix sections D and E, we prove that this surrogate acts as a valid distance function and preserves contraction properties under the distributional Bellman operator.

4. Results

In this section, we empirically validate the proposed DistRL method (NFDRL). We illustrate its ability to model return distributions with tunable variance, showcasing the flexibility offered by

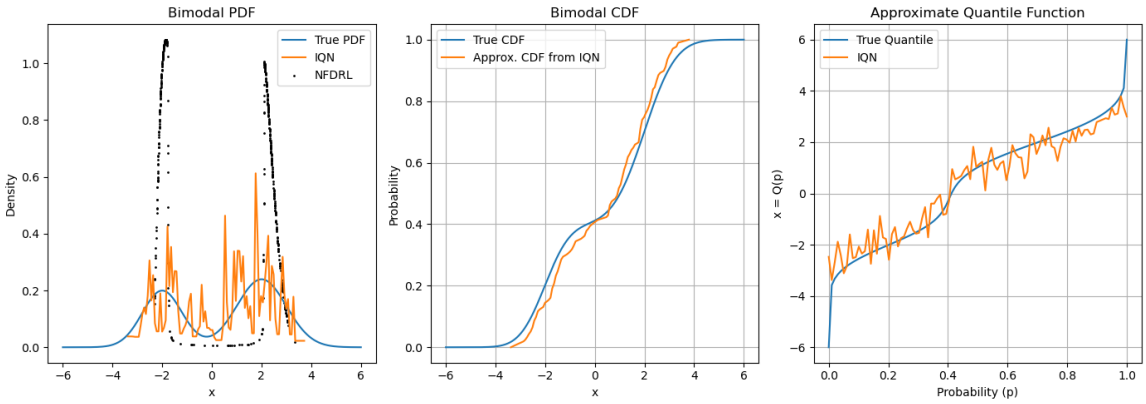


Figure 2: Return distributions learnt for the final state of MDP_3 using IQN and NFDRL. **Right**: true quantile function (blue) and quantile function learnt using IQN, reproducing the results of [15]. **Middle**: True CDF and CDF approximated for the quantile function obtained from IQN. **Left**: True PDF, and approximations obtained using IQN and NFDRL. While IQN produces a noisy distribution blurring the models, our method outputs a smoother distribution that makes the modes apparent.

our parameterization and the effect of surrogate optimization. We then investigate more complex settings, demonstrating the ability to learn multi-modal distributions. A particular focus is placed on a simple MDP taken from [15] with a bimodal final state distribution (see Appendix figure 4), where quantile-based methods struggle to learn smooth distributions. In contrast, our method successfully handles this challenge.

Building on these controlled experiments, we evaluate our method on a discrete stochastic environment, FrozenLake, to better visualize the multimodal distributions output by our model. We then scale to larger benchmarks by reporting results on a selection of Atari 2600 games, establishing the practicality of our approach in deep reinforcement learning settings.

Finally, we compare C51 and NFDRL model size and provide an empirical analysis of the sample size from base distribution to ensure effective training.

4.1. Modeling Expressiveness using simple MDPS

Toy MDPs We evaluate expressiveness on two controlled MDPs (setup detailed in Appendix C). In the deterministic MDP1, NFDRL demonstrates tunable granularity (Figure A5). While the surrogate loss yields slightly wider distributions than the exact Cramér loss—a consequence of minimizing a conservative upper bound over discrete masses—it crucially retains robust gradient signals via the geometry-aware term $|y_i - y_j|$. This avoids the vanishing gradients of KL-based methods on disjoint supports while mitigating the artificial broadening of C51 or the Dirac-mixture limitations of quantile baselines (see Appendix sections F and G). Additionally, the stochastic MDP2 confirms that NFDRL accurately captures multi-modal dynamics, recovering clear bimodal distributions (Figure A7) where unimodal or quantile approximations often blur the density.

MDP3 We reproduce the MDP described in [15] consisting of 4 successive states with only one possible action (Appendix Figure 4). The reward is nil for all states except the last where $R \sim \left(\frac{1}{2}\mathcal{N}(-2,1) + \frac{1}{2}\mathcal{N}(+2,1)\right)$. Figure 2 displays the distributions learnt for that final state return using IQN and NFDRL. As noted in [15], quantile regression approximates the inverse CDF with high variance, especially at extremes, leading to noisy and blurred PDF estimates that obscure bimodality. In contrast, our method produces smooth, clearly bi-modal PDFs, albeit with sharper peaks.

Frozen Lake To further assess the expressiveness of our model, we evaluate it on the Frozen Lake environment [16], a grid-world domain with inherent randomness. In figure A11, we display the

learned return distributions for each state-action pair. The environment's stochasticity naturally induces multimodal and skewed return distributions, which our model captures accurately.

4.2. Benchmarking

Environments We opted to conduct our experiments with the Atari Learning Environment (ALE) [17] and specifically the Atari-5 sub-benchmark [11]. We also report human-normalized scores. Implementation details can be found in the appendix (section H).

Results We evaluate our proposed method, NFDRL, in both its exact (NFDRL-E) and surrogate (NFDRL-S) variants across five Atari games, comparing against established baselines including DQN, C51, and IQN. Performance is measured using human-normalized scores (table 1), computed based on the raw game scores of human players and random agents (Appendix table 1). Both NFDRL variants significantly outperform traditional baselines (DQN and C51) on all games, achieving mean scores of 394 (NFDRL-E) and 407 (NFDRL-S) versus 189 (DQN) and 320 (C51). Notably, NFDRL-S, which uses a surrogate Cramér loss, slightly outperforms the exact version on average, highlighting its favorable tradeoff between computational efficiency and performance.

Although NFDRL-S uses a surrogate Cramér loss, its slightly better performance compared to the exact version is consistent with trends in deep learning. The exact Cramér loss introduces sampling noise, discretization errors, and sensitivity to distribution tails, which can destabilize training. In contrast, the surrogate loss may better handle these challenges, implicitly reweighting or smoothing the distribution, making it more amenable to optimization.

While IQN remains the top-performing method overall with a mean score of 525, our method performs competitively, particularly in games such as Double Dunk and QBert*, where NFDRL-S achieves or exceeds IQN performance. In sum, these results validate that the Cramér-based approach is a powerful alternative to quantile-based distributional RL, capable of modeling complex return distributions while maintaining strong empirical performance.

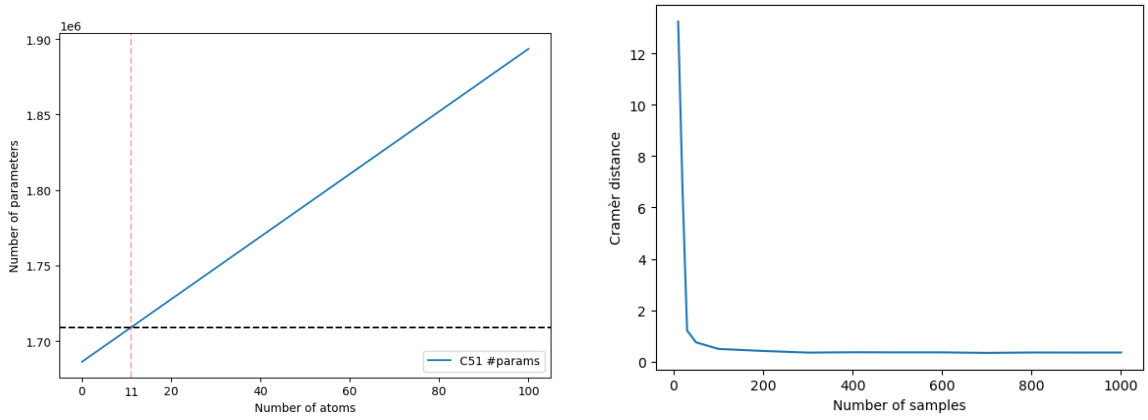
We emphasize that NFDRL is not intended to outperform IQN on Atari, but to provide a parsimonious continuous alternative with different trade-offs: constant parameter footprint and adaptive support, together with a metric-matched objective enjoying contraction and unbiased-gradient guarantees. This perspective is especially relevant in settings where decisions depend on the *shape* of $\eta^{\pi}(x, a)$ (e.g., multimodality or tail risk), rather than only on expected return.

Games	**DQN**	**C51**	**IQN**	**NFDRL-E**	**NFDRL-S**
Battle Zone	79	76	**115**	87	**95**
Double Dunk	545	959	1100	1200	**1243**
Name this Game	103	178	**354**	232	**259**
Phoenix	119	258	**862**	280	**301**
Q*Bert	97	178	193	169	**193**
Mean	189	330	**525**	394	**418**

Table 1: Human-normalized performance on ATARI-5. The best performing agent is highlighted in blue. We also compare our approach directly with C51 as they are both approximating the PDF (best one written in bold font) while IQN is quantile based.

4.3. Parameter efficiency

We evaluate parameter efficiency by comparing our model to C51. Unlike C51, whose parameter count grows with the number of atoms, our model maintains a constant size and already matches C51 with just 11 atoms, highlighting its superior efficiency (Figure 3 (left)). We further examine the impact of the number of base distribution samples in MDP2, observing that performance improves as the sample count increases and stabilizes around 100 samples, which offers a good balance between accuracy and computational cost (Figure 3 (right)).



(a) Parameter count comparison between C51 and NFDRL. C51's parameter count increases with the number of atoms due to its linear output layer. NFDRL maintains a fixed number of parameters, matching C51 only when it uses 11 atoms, thus demonstrating superior parameter efficiency.

(b) Impact of the number of base distribution samples on performance. As the number of samples used for computing the distributional loss increases, NFDRL's loss decreases and performance improves. A plateau is reached around 100 samples, indicating a good trade-off between accuracy and efficiency.

Figure 3: NFDRL Parameter Efficiency.

5. Conclusion

We introduced a new DistRL method that models return distributions as mixtures of Gaussians, with parameters learned via normalizing flows. By optimizing the Cramér loss—exactly or through a surrogate—we capture richer uncertainty and learn more precise value distributions. Empirically, our method achieves competitive or better performance on Atari games while being far more parameter-efficient than C51 and more expressive than quantile-based methods.

Limitations. Nonetheless, our approach bears certain limitations. While mentioned here, these are more detailed in the appendix section 1. (i) Our method shows high training variance, especially on certain games, partly due to sensitivity to learning rate schedules and stochasticity from sampling. (ii) Training convergence is relatively slow, stemming from variance, the indirect nature of flow parameterization, and the inherent cost of training normalizing flows. (iii) Design choices such as using CDF flows and KDE for target distributions introduce modeling constraints, inefficiencies, and non-trainable components.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 08 2017. URL `https://proceedings.mlr.press/v70/bellemare17a.html`.

[3] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/5c48ff18e0a47baaf81d8b8ea51eec92-Paper.pdf`.

[4] Arthur Charpentier, Romuald Elie, and Carl Remlinger. Reinforcement learning in economics and finance, 2020. URL `https://arxiv.org/abs/2003.10014`.

[5] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI*, 2017.

[6] Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1096–1105. PMLR, 07 2018. URL https://proceedings.mlr.press/v80/dabney18a.html.

[7] Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. Fully parameterized quantile function for distributional reinforcement learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f471223d1a1614b58a7dc45c9d01df19-Paper.pdf.

[8] Fan Zhou, Jianing Wang, and Xingdong Feng. Non-crossing quantile regression for distributional reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15909–15919. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b6f8dc086b2d60c5856e4ff517060392-Paper.pdf.

[9] Thibaut Théate, Antoine Wehenkel, Adrien Bolland, Gilles Louppe, and Damien Ernst. Distributional reinforcement learning with unconstrained monotonic neural networks. *Neurocomputing*, 534:199–219, May 2023. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.02.049. URL http://dx.doi.org/10.1016/j.neucom.2023.02.049.

[10] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL http://jmlr.org/papers/v22/19-1028.html.

[11] Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games, 2022. URL https://arxiv.org/abs/2210.02019.

[12] Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023. http://www.distributional-rl.org.

[13] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2722–2730. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ho19a.html.

[14] Marc G. Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients, 2017. URL https://arxiv.org/abs/1705.10743.

[15] Sami Jullien, Romain Deffayet, Jean-Michel Renders, Paul Groth, and Maarten de Rijke. Distributional reinforcement learning with dual expectile-quantile regression, 2024. URL https://arxiv.org/abs/2305.16877.

[16] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[17] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[18] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL `http://jmlr.org/papers/v23/21-1342.html`.

A. Limitations

Variance Our method exhibits high training variance, particularly on games such as Pong. This sensitivity can be partially mitigated by carefully tuning the learning rate schedule, a hyperparameter to which our model is notably sensitive. Figure A1 illustrates a training run using a fixed learning rate: while the model eventually achieves high performance, its learning curve remains highly unstable. Furthermore, because our loss function relies on sampling from base distributions (e.g., for computing the Cramér distance), it introduces additional stochasticity that contributes to this variance.

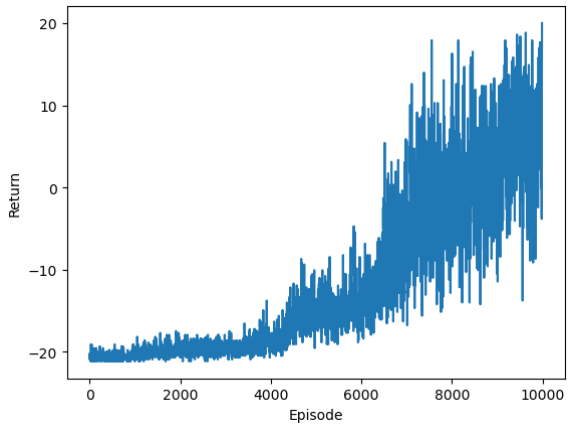


Figure A1: Training curve on PONG without learning rate decay

Learning efficiency As shown in main paper table 1 and Appendix figure A1, our model achieves competitive performance but at the cost of a slow training convergence. We believe this can be due to different factors:

1. The training variance might not help the model converge faster
2. Instead of learning directly the density of given values like C51, or specified values, our model learns flow parametrisations that indirectly lead to return distributions. This indirect relationship might hinder the learning performance by making the task more complex for the model.
3. Normalizing Flows are effective for learning exact likelihoods but they are notoriously slow to train, this fact is confirmed by our empirical results.

CDF Flow While using a CDF as a flow transformation offers advantages in modeling monotonic mappings and enabling efficient computation of the Cramér distance (Main paper section 3.2), it also introduces notable limitations. First, the CDF is inherently bounded, making it ill-suited for modeling unbounded return distributions without an additional projection step to extend its support. Second, since the model parameterizes a mixture of Gaussians, the components can become disjoint—e.g., with widely separated means and variances—which may result in poor overlap with the base distribution (typically standard Gaussian). Consequently, this can lead to inefficient coverage of the learned CDF's support, introducing instability and training inconsistencies. Although chaining multiple flow transformations or adopting alternative flow families may alleviate this issue, doing so increases model complexity and may hinder convergence. Addressing this trade-off remains an open direction for future work.

KDE for Target Distribution To enable the computation of the Cramér distance, we construct a target return distribution via KDE, ensuring that it shares the same support as the predicted distribution. However, this introduces a non-trivial computational burden and a non-trainable step in the pipeline, as gradients do not flow through the KDE. We attempted to mitigate this by reusing

the same set of samples z (used to parameterize the predicted distribution) to build the KDE target, in the hope of improving alignment and efficiency. Unfortunately, this strategy did not yield significant improvements. Ideally, a more integrated approach would avoid the need for KDE altogether, enabling fully end-to-end training and reducing reliance on handcrafted alignment procedures.

B. Alignment Procedure and Algorithm

As mentionned in main paper section 3.4, our method necessitates a support alignment between the predict and target distributions. Indeed for the same samples $z_i \sim \mathcal{U}$, we have $F_{(x,a)}(z_i) = y_i$ and the target distribution is based on subsequent states and actions, therefore $\tilde{y}_i = F_{(x',a')}(z)$. Said otherwise, as the predicted and target distributions are based using different state-action pairs, the output return values are different are also different, without even taking the bootstrap function into account. Therefore, in order to compare the two distributions accurately, we use a KDE to get $\eta^\pi(x,a)(y_i), \eta^\pi(x,a)(\tilde{y}_i), T^\pi\eta(x',a')(y_i), T^\pi\eta(x',a')(\tilde{y}_i)$.

Let $y_i \sim \eta^\pi(x,a)$ be samples from the predicted return distribution and $\tilde{y}_j \sim T^\pi\eta(x,a)$ be samples from the target distribution. First, we evaluate the KDE of $T^\pi\eta(x,a)$ on the predicted support:

$$T^\pi\eta(x,a)(y_i) = \frac{1}{M}\sum_{j=1}^{M} K_h(y_i - \tilde{y}_j), \quad \forall i = 1,\dots,N,$$

and reciprocally:

$$\hat{\eta}^\pi(x,a)(\tilde{y}_j) = \frac{1}{N}\sum_{i=1}^{N} K_h(\tilde{y}_j - y_i), \quad \forall j = 1,\dots,M.$$

The process is illustrated in figure A2 and detailed in algorithm 1.

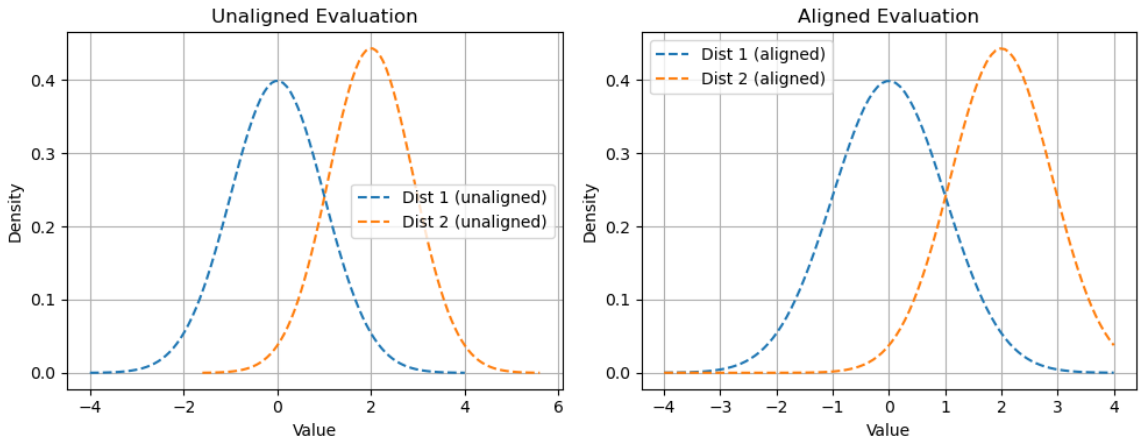


Figure A2: Impact of Support Alignment on Distribution Comparison. Left: Two Gaussian distributions evaluated on different support sets, making direct comparison ill-posed. Right: The same distributions evaluated on a shared support, enabling meaningful density-wise comparison. This illustrates the necessity of support alignment in distributional reinforcement learning, akin to the projection step in C51.

Algorithm 1 Flow-based Target Distribution Construction in Distributional RL

Require: Current return distribution $\eta^\pi(x, a)$, reward r, next state x', terminal indicator d, support z, number of samples N

Compute next-state return distribution:

$$(y', p^\pi(y')) \leftarrow g(F_\theta(x', z), z)$$

if x' is terminal **then**

Replace $(y', p^\pi(y'))$ with Gaussian $\mathcal{N}(r, 0.1)$

end if

Compute expected returns:

$$Q^\pi(x', a') = \sum_{y'} y' \cdot p^\pi(y'))$$

Greedy action: $a^* = \arg\max_{a'} Q^\pi(x', a')$

Select corresponding distribution: $y^* = y^*[a^*]$

Select support $s = [-\max(y, y^*), \max(y, y^*)]$

Estimate KDEs:

$$\hat{p}_\eta(s) \leftarrow \text{KDE}(y, \eta(x, a)(y)), \quad \hat{p}_{T^\pi\eta}(s) \leftarrow \text{KDE}(y^*, T^\pi\eta(x', a^*)(y^*))$$

Interpolate both to support $\{y_i\}_{i=1}^N$:

$$p_\eta(y_i), p_{T^\pi\eta}(y_i) \leftarrow \text{Interpolate}(\hat{p}_\eta, \hat{p}_{T^\pi\eta})$$

Interpolate both to support $\{\tilde{y}_j\}_{j=1}^N$:

$$\tilde{p}_\eta(\tilde{y}_j), \tilde{p}_{T^\pi\eta}(\tilde{y}_j) \leftarrow \text{Interpolate}(\hat{p}_\eta, \hat{p}_{T^\pi\eta})$$

return $\{\tilde{y}_j\}_{j=1}^N, p_\eta(y_i), p_{T^\pi\eta}(y_i), \tilde{p}_\eta(\tilde{y}_j), \tilde{p}_{T^\pi\eta}(\tilde{y}_j)$

C. Toy Markov decision processes

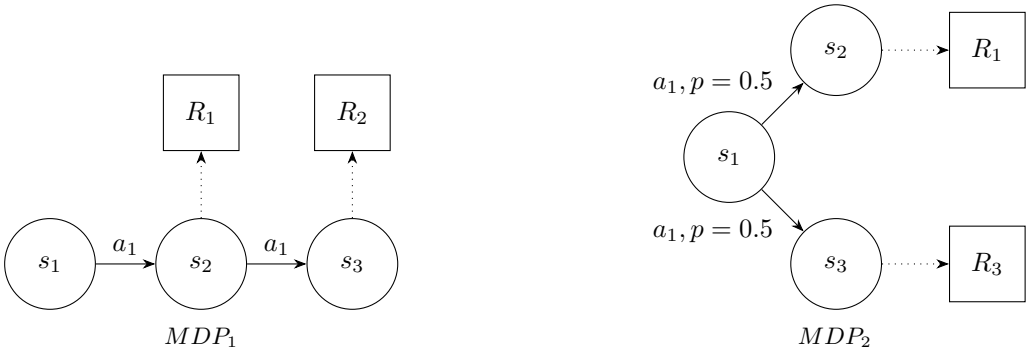


Figure A3: Two example MDPs.

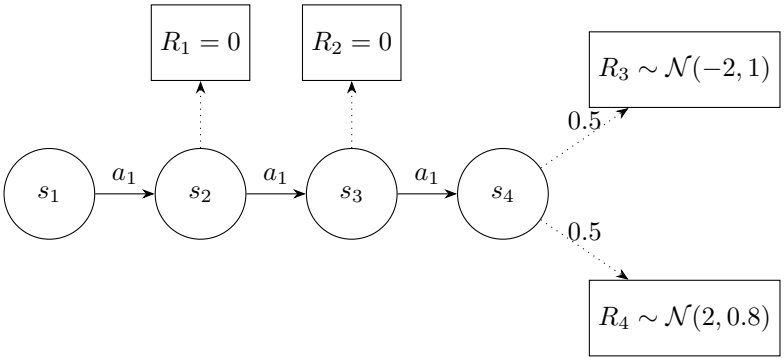
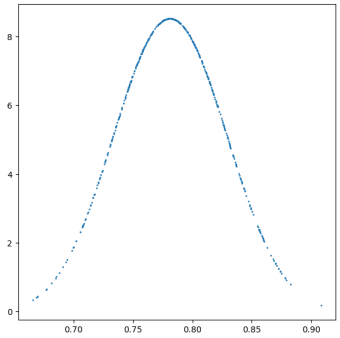
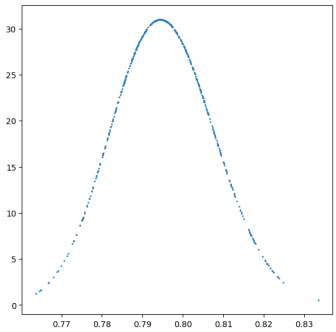


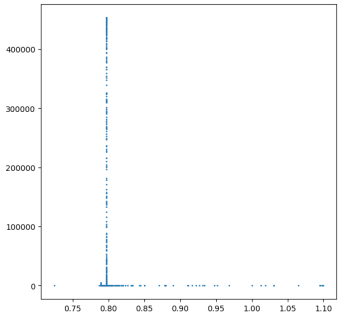
Figure A4: $MDP_3; \gamma = 1$



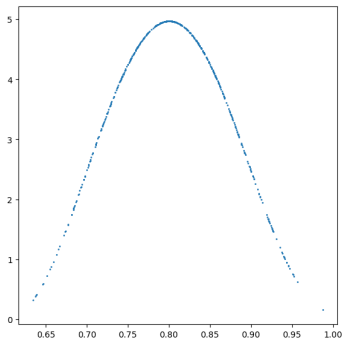
(a) bandwith: 0.05, s.e: 0.1



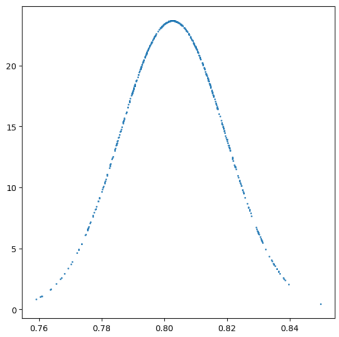
(b) bandwith: 0.01, s.e: 0.01



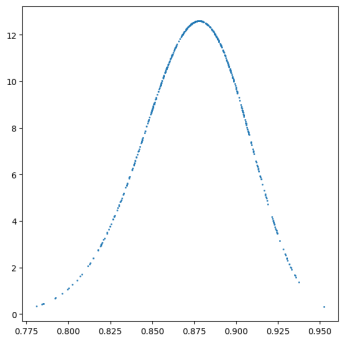
(c) bandwith: 0.001, s.e: 0.01



(d) bandwith: 0.05, s.e: 0.1

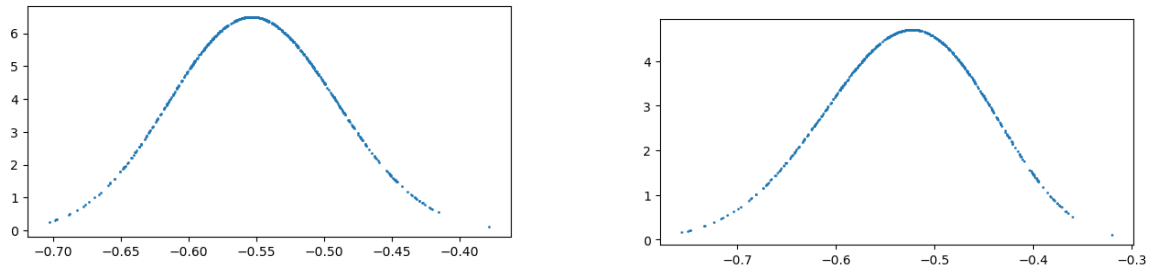


(e) bandwith: 0.01, s.e: 0.01



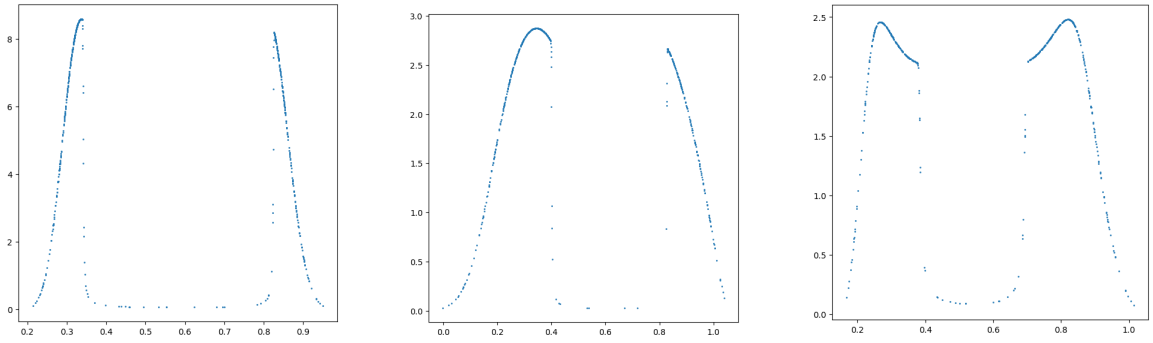
(f) bandwith: 0.001, s.e: 0.01

Figure A5: Learnt return distributions for the state-action pair (s_2, a_1) in MDP_1, under different values of the KDE bandwidth and final state's reward variance. The x-axis is return values and the y axis corresponds to their corresponding densities. The target reward is 0.8. (a,b,c) show distributions learned using the exact Cramér loss, while (d,e,f) show those obtained with our surrogate. Narrower distributions can be achieved with both losses, illustrating the method's flexibility.



(a) Exact Cramér (b) Surrogate

Figure A6: Learnt distributions for (s_1, a_1) in MDP_1; $R_1 = -0.8$, $R_2 = 0.3$; $\gamma = 0.9$. The x-axis is return values and the y axis corresponds to their densities. A KDE bandwidth of 0.05 and a final reward standard error of 0.01 are used.



(a) Exact Cramér (b) Surrogate (c) L^2 loss

Figure A7: Learnt distributions for (s_1, a_1) in MDP_2; $R_1 = 0.8$, $R_2 = 0.3$. The x-axis is return values and the y-axis corresponds to their corresponding densities. We use a KDE bandwidth of 0.05 and a final reward standard error of 0.1.

D. Relevant distance properties

Bellemare et al. [14] show that the Cramér distance holds the following properties:

Scale sensitivity. Consider a divergence $\mathbf{d}$, and for two random variables X, Y with distributions P, Q, write $\mathbf{d}(X, Y) := \mathbf{d}(P, Q)$. We say that $\mathbf{d}$ is scale sensitive (of order β), i.e. it has property $(\mathbf{S})$ if there exists a $\beta > 0$ such that for all X, Y, and a real value $c > 0$,

$$\mathbf{d}(cX, cY) \leq |c|^{\beta} \mathbf{d}(X, Y) \tag{S}$$

Sum invariance: A divergence $\mathbf{d}$ has property $(\mathbf{I})$, i.e. it is sum invariant, if whenever A is independent from X, Y

$$\mathbf{d}(A + X, A + Y) \leq \mathbf{d}(X, Y) \tag{I}$$

A divergence is said ideal if it possesses both $(\mathbf{S})$ and $(\mathbf{I})$.

Unbiased sample gradients: Let $X_1, X_2, \ldots, X_m$ be independent samples from P and define the empirical distribution $\hat{P}_m := \hat{P}_m(X_m) := \frac{1}{m} \sum_{i=1}^{m} \delta_{X_i}$. From this, define the sample loss $\mathbf{d}(\hat{P}_m, Q_\theta)$. We say that $\mathbf{d}$ has unbiased sample gradients when the expected gradient of the sample loss equals the gradient of the true loss for all P and m:

$$\mathbb{E}_{X_m \sim P} \nabla_\theta \mathbf{d}(\hat{P}_m, Q_\theta) = \nabla_\theta \mathbf{d}(P, Q_\theta) \tag{U}$$

If a divergence does not possess $(\mathbf{U})$, then minimising it with stochastic gradient descent may not converge or towards the wrong minimum. Conversely, if $\mathbf{d}$ possesses $(\mathbf{U})$ then we can guarantee that the distribution which minimises the expected sample loss is $Q = P$. From these properties, [14] draws the following propositions:
Proposition 1: *The KL divergence has unbiased sample gradients* (U)*, but is not scale sensitive* (S)*.*
Proposition 2: *The Wasserstein metric is ideal* (I, S)*, but does not have unbiased sample gradients.*

E. Cramér-inspired geometry-aware metric

We introduce a Cramér-inspired, geometry-aware metric on discrete probability masses and show that it enjoys key properties (metric axioms, $\sqrt{gamma}$-contraction, and unbiased sample gradients) mirroring those that make the Cramér distance appealing in DistRL. Recall that for two distributions with CDFs $F(x)$ and $G(x)$, the squared Cramér distance is:

$$l_2^2(P, Q) = \int_{-\infty}^{\infty} (F(x) - G(x))^2 \, dx$$

Let $\delta(t) = p(t) - q(t)$ be the difference in probability densities. We can express the CDF difference as an integral:

$$F(x) - G(x) = \int_{-\infty}^{x} \delta(t) \, dt$$

Substituting this into the distance definition:

$$l_2^2(P, Q) = \int_{-\infty}^{\infty} \left(\int_{-\infty}^{x} \delta(t) \, dt \right)^2 dx$$

We apply the Cauchy-Schwarz inequality to bound the inner term. For any interval $[y, x]$, we have:

$$\left(\int_y^x \delta(t) \cdot 1 \, dt \right)^2 \leq \left(\int_y^x 1^2 \, dt \right) \left(\int_y^x \delta(t)^2 \, dt \right)$$

The first term on the right side evaluates simply to the length of the interval:

$$\int_y^x 1 \, dt = |x - y|$$

Thus, we obtain the upper bound:

$$(F(x) - G(x))^2 \leq |x - y| \int_{-\infty}^{x} \delta(t)^2 \, dt$$

Substituting this back into the outer integral, and swapping the order of integration (Fubini's theorem), we approximate the global distance as a pairwise sum over the support. Discretizing the domain into N support points $\{y_1, \ldots, y_N\}$ with associated probability masses w_i and v_i (where $\delta(y_i) \approx w_i - v_i$), the integral approximates to:

$$\mathcal{L}_{\text{surrogate}} = \sum_{i=1}^{N} \sum_{j=1}^{N} (w_i - v_i)^2 \cdot |y_i - y_j|$$

In the next sections we will investigate the three following questions:

- **Q1:** Is this loss a proper distance? If so, then the loss function would be symmetric and scale sensitive unlike KL divergence.
- **Q2:** Is the distributional Bellman operator a contraction in this case? If so, this would ensure convergence of the distributional Bellman operator η^π towards the random returns T_η^π.
- **Q3:** Does it still possess the unbiased sample gradient estimate property? If so, then SGD can be used to optimise this loss function in a RL context unlike the Wasserstein distance. More specifically, we will be able to learn from sample transitions.

In what follows, the theory treats the masses w_i abstractly; they may arise from a Riemann approximation of a continuous density or from an empirical distribution over Monte-Carlo samples. Our proofs only require that they form valid discrete probability distributions.

E.1. Q1: Is it a Proper Distance?

We verify that the proposed loss function satisfies the four axioms of a metric: Non-negativity, Symmetry, Identity of Indiscernibles, and the Triangle Inequality.

Let the loss be defined over the discrete probability masses $\mathbf{w}, \mathbf{v} \in \mathbb{R}^N$ defined on a fixed support $\{y_1, \ldots, y_N\}$:

$$D(\mathbf{w}, \mathbf{v}) = \sqrt{\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (w_i - v_i)^2 |y_i - y_j|} \tag{13}$$

First, we simplify the expression by factoring out the term that depends only on i. Let Ω_i be the "geometric weight" of the i-th support point, representing the sum of distances from y_i to all other points:

$$\Omega_i = \sum_{j=1}^{N} |y_i - y_j| \tag{14}$$

Note that as long as $N > 1$ and the support points are distinct, $\Omega_i > 0$ for all i.

Substituting this into the loss equation:

$$D(\mathbf{w}, \mathbf{v}) = \frac{1}{N} \sqrt{\sum_{i=1}^{N} \Omega_i (w_i - v_i)^2} \tag{15}$$

This reveals that our loss function is simply a Weighted Euclidean Distance (scaled by $1/N$) between the mass vectors $\mathbf{w}$ and $\mathbf{v}$, with weights Ω_i.

1. **Non-negativity:** Since $(w_i - v_i)^2 \geq 0$ and weights $\Omega_i > 0$, the sum is non-negative. The square root function preserves non-negativity. Thus, $D(\mathbf{w}, \mathbf{v}) \geq 0$.

2. **Symmetry:** The term $(w_i - v_i)^2$ is inherently symmetric: $(w_i - v_i)^2 = (v_i - w_i)^2$. The weights Ω_i depend only on the support geometry, which is fixed for the comparison. Thus, $D(\mathbf{w}, \mathbf{v}) = D(\mathbf{v}, \mathbf{w})$.

3. **Identity of Indiscernibles:** Clearly, if $\mathbf{w} = \mathbf{v}$, then $(w_i - v_i) = 0$ for all i, so $D = 0$. Conversely, assume $D(\mathbf{w}, \mathbf{v}) = 0$. This implies:

$$\sum_{i=1}^{N} \Omega_i (w_i - v_i)^2 = 0$$

   Since $\Omega_i > 0$ strictly (for distinct support points with $N > 1$) and $(w_i - v_i)^2 \geq 0$, the only way the sum can be zero is if every individual term $(w_i - v_i)^2 = 0$. Therefore, $w_i = v_i$ for all i, implying $\mathbf{w} = \mathbf{v}$.

4. **Triangle Inequality:** We must show that $D(\mathbf{w}, \mathbf{u}) \leq D(\mathbf{w}, \mathbf{v}) + D(\mathbf{v}, \mathbf{u})$ for any third distribution $\mathbf{u}$.

   Let us define the transformed vector $\mathbf{x}$ such that $x_i = \sqrt{\Omega_i} w_i$. Similarly, let $x'_i = \sqrt{\Omega_i} v_i$ and $x''_i = \sqrt{\Omega_i} u_i$.

   The loss can be rewritten as the standard Euclidean (l_2) distance between these transformed vectors (ignoring the $1/N$ factor, which scales all terms linearly):

$$N \cdot D(\mathbf{w}, \mathbf{v}) = \sqrt{\sum_{i=1}^{N} (\sqrt{\Omega_i} w_i - \sqrt{\Omega_i} v_i)^2} = ||\mathbf{x} - \mathbf{x}'||_2$$

   The standard Euclidean norm satisfies the triangle inequality:

$$||\mathbf{x} - \mathbf{x}''||_2 \leq ||\mathbf{x} - \mathbf{x}'||_2 + ||\mathbf{x}' - \mathbf{x}''||_2$$

   Substituting back the definitions, we obtain:

$$N \cdot D(\mathbf{w}, \mathbf{u}) \leq N \cdot D(\mathbf{w}, \mathbf{v}) + N \cdot D(\mathbf{v}, \mathbf{u})$$

   Dividing by N, the triangle inequality holds.

Conclusion: The proposed surrogate loss D defines a valid metric on the space of discrete probability distributions over a fixed support.

E.2. Q2: Is the Distributional Bellman Operator a Contraction for D?

We now show that the distributional Bellman operator $\mathcal{T}^\pi$ is a contraction under the metric D defined above.

Recall the definition of the operator. For a transition (x, a, r, x'), the operator applies a shift and scale to the random return $Z(x', a')$. If $Z \sim \eta$, then the target return is $R + \gamma Z$.

In our discrete mass framework, this transformation affects the support locations $\{y_i\}$, while leaving the probability masses invariant (as established in the implementation note).

Let $\mathbf{w}$ and $\mathbf{v}$ be the probability mass vectors for two different return distributions η_1 and η_2 defined on the same support $\{y_i\}$. The distance between them is:

$$D(\eta_1, \eta_2) = \sqrt{\frac{1}{N^2} \sum_{i,j} (w_i - v_i)^2 |y_i - y_j|} \tag{16}$$

Applying the Bellman operator $\mathcal{T}^\pi$ transforms the support points y_i to new locations $y'_i = r + \gamma y_i$. As discussed, the probability masses w_i and v_i associated with these points remain invariant under this transformation.

Thus, the distance between the projected distributions is:

$$D(\mathcal{T}^\pi \eta_1, \mathcal{T}^\pi \eta_2) = \sqrt{\frac{1}{N^2} \sum_{i,j} (w_i - v_i)^2 |y'_i - y'_j|} \tag{17}$$

Substituting $y'_i = r + \gamma y_i$:

$$|y'_i - y'_j| = |(r + \gamma y_i) - (r + \gamma y_j)| = |\gamma(y_i - y_j)| = \gamma |y_i - y_j| \tag{18}$$

(Note: Since $\gamma > 0$, $|\gamma| = \gamma$).

Substituting this back into the distance equation:

$$\begin{aligned} D(\mathcal{T}^\pi \eta_1, \mathcal{T}^\pi \eta_2) &= \sqrt{\frac{1}{N^2} \sum_{i,j} (w_i - v_i)^2 \cdot \gamma |y_i - y_j|} \\ &= \sqrt{\gamma} \cdot \sqrt{\frac{1}{N^2} \sum_{i,j} (w_i - v_i)^2 |y_i - y_j|} \\ &= \sqrt{\gamma} \cdot D(\eta_1, \eta_2) \end{aligned} \tag{19}$$

Conclusion: Since the discount factor satisfies $0 < \gamma < 1$, it follows that $\sqrt{\gamma} < 1$. Therefore, the distributional Bellman operator $\mathcal{T}^\pi$ is a contraction mapping with respect to the metric D, with contraction factor $\sqrt{\gamma}$.

By Banach's Fixed Point Theorem, repeated application of $\mathcal{T}^\pi$ converges to a unique fixed-point distribution.

E.3. Q3: Does d Admit Unbiased Sample Gradient Estimates?

Finally, we verify that our proposed surrogate loss satisfies property (U) (Unbiased Sample Gradients) as defined by [14]. This property ensures that minimizing the loss over stochastic samples (e.g., transitions sampled from the replay buffer) minimizes the true expected loss over the full distribution.

Let $\mathbf{w}(\theta)$ be the predicted probability mass vector parameterized by θ. Let $\mathbf{v}_{\text{true}}$ be the true target probability mass vector. In a reinforcement learning setting, we do not observe $\mathbf{v}_{\text{true}}$ directly; instead, we observe empirical samples (realizations of returns) which form an empirical target distribution $\hat{\mathbf{v}}$.

We define the squared sample loss $\mathcal{L}$ as:

$$\mathcal{L}(\theta, \hat{\mathbf{v}}) = \sum_{i=1}^{N} \sum_{j=1}^{N} (w_i(\theta) - \hat{v}_i)^2 |y_i - y_j| \tag{20}$$

To simplify notation, let $\Omega_i = \sum_{j=1}^{N} |y_i - y_j|$ be the geometric weight associated with support point i. The loss simplifies to a weighted sum of squared errors:

$$\mathcal{L}(\theta, \hat{\mathbf{v}}) = \sum_{i=1}^{N} \Omega_i (w_i(\theta) - \hat{v}_i)^2 \tag{21}$$

We compute the gradient with respect to parameters θ:

$$\nabla_\theta \mathcal{L} = \sum_{i=1}^{N} 2\Omega_i (w_i(\theta) - \hat{v}_i) \nabla_\theta w_i(\theta) \tag{22}$$

We now examine the expected gradient over the distribution of sample targets. Since the expectation is a linear operator:

$$\begin{aligned}\mathbb{E}_{\hat{\mathbf{v}}}[\nabla_\theta \mathcal{L}] &= \mathbb{E}_{\hat{\mathbf{v}}}\left[\sum_{i=1}^{N} 2\Omega_i(w_i(\theta) - \hat{v}_i)\nabla_\theta w_i(\theta)\right] \\ &= \sum_{i=1}^{N} 2\Omega_i(w_i(\theta) - \mathbb{E}[\hat{v}_i])\nabla_\theta w_i(\theta)\end{aligned} \tag{23}$$

Since $\hat{\mathbf{v}}$ is constructed from unbiased samples of the return (e.g., via Monte Carlo sampling or an unbiased empirical distribution), we have $\mathbb{E}[\hat{v}_i] = v_{\text{true},i}$. Therefore:

$$\mathbb{E}_{\hat{\mathbf{v}}}[\nabla_\theta \mathcal{L}] = \nabla_\theta \mathcal{L}(\theta, \mathbf{v}_{\text{true}}) \tag{24}$$

Conclusion: The expected gradient of the sample loss is exactly the gradient of the true loss. Unlike the Wasserstein distance, which generally requires biased sample gradients or dual approximations, our geometry-aware surrogate allows for direct stochastic gradient descent (SGD) on sample transitions.

F. Behaviour of the Geometry-Aware Metric and the Role of KDE

In this section we clarify a limitation of the discrete geometry-aware metric D in a simple "one-hot, disjoint support" setting, and then show how the KDE-based construction used in practice mitigates this issue and yields a loss that behaves more like a transport distance.

F.1. A limitation of the exact metric for disjoint one-hot distributions

Recall that for a fixed support $\{y_1, \dots, y_N\}$ and two discrete probability mass vectors

$$w = (w_1, \dots, w_N), \qquad v = (v_1, \dots, v_N),$$

our surrogate Cramér distance is

$$D(w, v) \;=\; \sqrt{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(w_i - v_i)^2\,|y_i - y_j|} \;=\; \frac{1}{N}\sqrt{\sum_{i=1}^{N}\Omega_i(w_i - v_i)^2}, \tag{25}$$

with

$$\Omega_i \;:=\; \sum_{j=1}^{N}|y_i - y_j|. \tag{26}$$

Consider now the following simple discrete setting:

- the support $\{y_i\}$ is a fixed grid (e.g., uniform on $[-K, K]$);
- w and v are one-hot distributions:

  $$w_i = \mathbb{1}\{i = a\}, \qquad v_i = \mathbb{1}\{i = b\}$$

  for some indices $a \neq b$.

In this case,

$$(w_i - v_i)^2 = \begin{cases} 1, & i \in \{a, b\}, \\ 0, & \text{otherwise}, \end{cases}$$

and therefore the double sum in (25) collapses to

$$D^2(w, v) = \frac{1}{N^2}\big(\Omega_a + \Omega_b\big). \tag{27}$$

A key observation is that $D^2(w, v)$ in (27) does not depend explicitly on the distance $|y_a - y_b|$ between the two spikes, except through which weights Ω_a and Ω_b happen to be selected. On a symmetric uniform grid, Ω_i is smallest in the centre and largest near the edges, but for two disjoint one-hot distributions the value of $D(w, v)$ is determined by:

1. which bins carry mass (via Ω_a, Ω_b),
2. not by how far apart those bins are.

Concretely, on a grid $\{-10, -9, \ldots, 10\}$, one can compute

$$\Omega_{-10} = \Omega_{+10} = 210, \qquad \Omega_{-9} = 191,$$

so that

$$D^2(\delta_{-10}, \delta_{-9}) \;\propto\; 210 + 191 = 401, \qquad D^2(\delta_{-10}, \delta_{+10}) \;\propto\; 210 + 210 = 420,$$

which are very close despite the spikes being at distance 1 vs 20. In other words:

For disjoint one-hot distributions, the exact metric D behaves as a geometry-weighted Euclidean distance on bin masses, and does not strongly distinguish "near" from "far" spikes in the way an optimal-transport or Cramér distance does.

This is the core limitation that will be addressed by the KDE-based construction used in our practical loss.

F.2. KDE-based construction of the practical loss

In the model, the return distributions are represented as continuous densities via normalizing flows. For a given (x, a), denote

$$p(y) := \eta^\pi(x, a)(y), \qquad q(y) := (T^\pi \eta)(x, a)(y).$$

We do not observe exact masses on a fixed grid; instead we:

1. Sample from the continuous densities:
$$y^{(1)}, \ldots, y^{(N)} \sim p, \qquad \tilde{y}^{(1)}, \ldots, \tilde{y}^{(M)} \sim q.$$
2. Estimate p and q via KDE on each support using a kernel K_h with bandwidth $h > 0$:
on the predicted support $\{y_i\}_{i=1}^N$,
$$\hat{p}(y_i) = \frac{1}{N} \sum_{k=1}^{N} K_h(y_i - y^{(k)}),$$
$$\hat{q}(y_i) = \frac{1}{M} \sum_{j=1}^{M} K_h(y_i - \tilde{y}^{(j)}),$$
on the target support $\{\tilde{y}_j\}_{j=1}^M$,
$$\hat{p}(\tilde{y}_j) = \frac{1}{N} \sum_{k=1}^{N} K_h(\tilde{y}_j - y^{(k)}),$$
$$\hat{q}(\tilde{y}_j) = \frac{1}{M} \sum_{j=1}^{M} K_h(\tilde{y}_j - \tilde{y}^{(j)}).$$
3. Discretize these KDEs into mass vectors on each grid (e.g. by Riemann approximation):
$$w_i^{(y)} \approx \frac{\hat{p}(y_i)\,\Delta y}{\sum_k \hat{p}(y_k)\,\Delta y}, \quad v_i^{(y)} \approx \frac{\hat{q}(y_i)\,\Delta y}{\sum_k \hat{q}(y_k)\,\Delta y},$$
and similarly $w^{(\tilde{y})}, v^{(\tilde{y})}$ on $\{\tilde{y}_j\}$.

The practical loss we use (Eq. (11) in the main text) is then

$$\mathcal{L}(\eta^\pi(x, a), T^\pi \eta(x, a)) \;=\; D\big(w^{(y)}, v^{(y)}\big) + D\big(w^{(\tilde{y})}, v^{(\tilde{y})}\big), \tag{28}$$

where D is exactly the metric defined in (25), applied separately on the predicted and target supports. Thus, in practice, we always apply D to KDE-smoothed mass vectors, not to raw one-hot Diracs on the grid.

F.3. How KDE restores a transport-like behaviour in the one-hot example

We now revisit the "two spikes" example under the KDE-based construction. Consider two sharply peaked continuous distributions with means μ_p and μ_q, and treat them as approximations to Dirac masses:

$$p(y) \approx \delta_{\mu_p}, \qquad q(y) \approx \delta_{\mu_q}.$$

After applying KDE with bandwidth h, we obtain smooth approximations

$$\hat{p}(y) \approx K_h(y - \mu_p), \qquad \hat{q}(y) \approx K_h(y - \mu_q),$$

and hence, on a grid $\{y_i\}$,

$$w_i^{(y)} \;\propto\; K_h(y_i - \mu_p), \qquad v_i^{(y)} \;\propto\; K_h(y_i - \mu_q).$$

Define the difference vector

$$\Delta_i := w_i^{(y)} - v_i^{(y)}.$$

The contribution of the predicted support to the loss is

$$D^2\big(w^{(y)}, v^{(y)}\big) = \frac{1}{N^2} \sum_{i=1}^{N} \Omega_i \, \Delta_i^2. \tag{29}$$

Now compare two regimes:

- **Near modes:** $|\mu_p - \mu_q| \ll h$. The two KDEs $K_h(\cdot - \mu_p)$ and $K_h(\cdot - \mu_q)$ overlap heavily. On most grid points, especially in regions where Ω_i is large, we have

  $$|\Delta_i| = \big|w_i^{(y)} - v_i^{(y)}\big| \approx 0,$$

  so the weighted sum $\sum_i \Omega_i \Delta_i^2$ in (29) is relatively small.

- **Far modes:** $|\mu_p - \mu_q| \gg h$.

  The two KDEs overlap very little. There are two separated regions where one of $w_i^{(y)}, v_i^{(y)}$ is large and the other is close to zero, so

  $$|\Delta_i| \approx \max\big\{w_i^{(y)}, v_i^{(y)}\big\}$$

  on those regions. Since these regions typically lie in parts of the grid where Ω_i is sizeable, many terms $\Omega_i \Delta_i^2$ contribute, making $\sum_i \Omega_i \Delta_i^2$ (and thus D) significantly larger than in the near-mode case.

As a result, for moderate bandwidth h we have qualitatively

$$D_{\mathrm{KDE}}(\delta_{\mu_p}, \delta_{\mu_q}) \quad \text{small if } |\mu_p - \mu_q| \ll h, \qquad D_{\mathrm{KDE}}(\delta_{\mu_p}, \delta_{\mu_q}) \quad \text{large if } |\mu_p - \mu_q| \gg h.$$

The same reasoning applies to the symmetric term $D(w^{(\tilde{y})}, v^{(\tilde{y})})$ on the target support in (28). Taken together, this shows that:

While the exact discrete metric D on one-hot mass vectors does not by itself strongly encode "near vs far" behaviour, the KDE-based practical loss $\mathcal{L}$ applies D to smoothed mass vectors whose differences $(w_i - v_i)^2$ reflect the spatial separation of modes. As a result, $\mathcal{L}$ behaves more like a transport-style distance: it penalizes discrepancies between distributions increasingly as their mass moves further apart.

In the limit $h \to 0$, the KDEs collapse to Dirac masses on the grid and the behaviour reverts to the discrete one-hot case (27). For the moderate bandwidths used in practice, however, KDE smooths each mode and allows the geometry-aware weights Ω_i in (29) to amplify spatially structured discrepancies between the predicted and target return distributions.

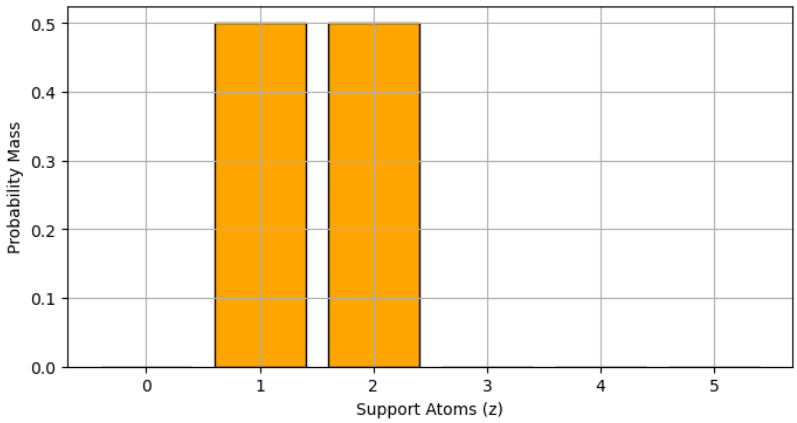


Figure A8: Linear interpolation splits predicted returns among bins in C51

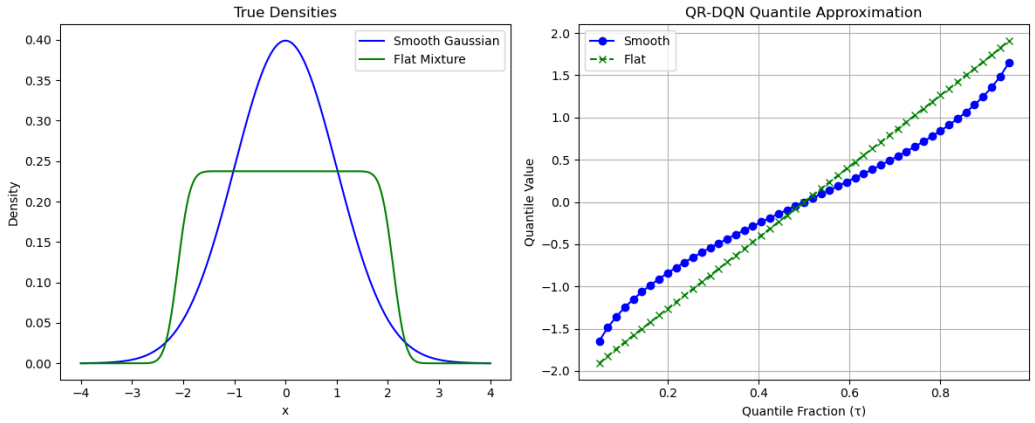
G. C51 tends to learn broader distributions when rewards fall between fixed atoms

Consider a situation where the distribution is represented with bins where each bin spans an interval of length 1. Said otherwise, the categorical representation allocates bin 1 to returns between 0 and 1, bin 2 to returns between 1 and 2. Now consider that the return to predict is exactly 1. This value falls right between bin 0 and bin 1. C51 will allocate the same mass to both bins, effectively displaying a wider distribution than it should. This is illustrated in figure A8 where we used the same mechanism as in C51. We consider $V \in [0, 5]$, which gives 6 atoms with $dz = 1$. When tring to predict a return of 1.5, linear interpolation makes it split into 2 bins. The code used to produce this figure is available in the supplementary material.

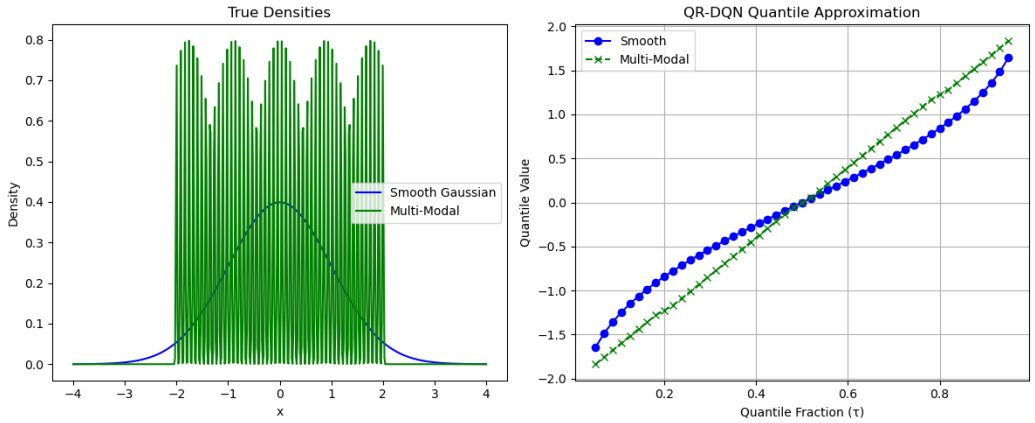
H. Quantile based methods lack control over local spread

QR-DQN predicts a fixed set of return quantiles (e.g., 5th, 15th, …, 95th percentiles), while IQN learns a function that estimates quantiles for arbitrary probabilities sampled from a uniform distribution. Although IQN provides greater flexibility in querying quantiles, both methods ultimately approximate the return distribution as a discrete set of quantile points. These predicted quantiles are best interpreted as Dirac masses of equal weight, forming a piecewise approximation that omits information about the distribution's behavior between points. Consequently, quantile-based methods cannot distinguish between smooth distributions and those with sharp peaks or flat regions, nor can they explicitly model local density or control how probability mass is allocated within intervals.

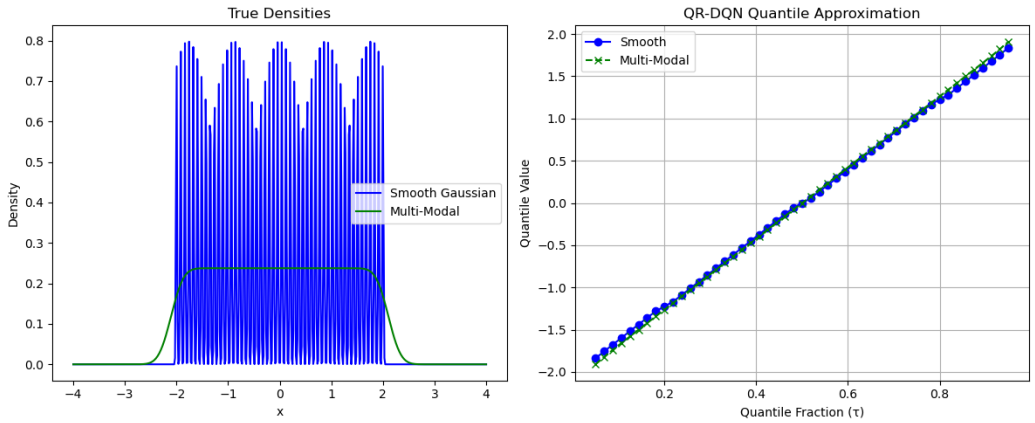
Our flow-based approach addresses these limitations by modeling a continuous PDF, enabling explicit control over probability mass across the return space and allowing the capture of fine-grained structure and local variability that quantile-based methods cannot. Figure A9 illustrates this difference: three comparisons are shown between a standard Gaussian, a flattened Gaussian, and a peaky distribution. While these distributions are clearly distinct, their quantile approximations are almost identical—in particular, the flattened and peaky cases are nearly indistinguishable. Although this is an extreme example, it effectively demonstrates that quantile-based methods lack the ability to represent local spread.



(a) **Left:** A standard Gaussian and a flat mixture density functions. **Right:** Their respective quantile approximations using QR-DQN.



(b) **Left:** A standard Gaussian and an extremely multimodal distribution. **Right:** Their respective quantile approximations using QR-DQN.



(c) **Left:** A flat mixture and an extremely multimodal distribution. **Right:** Their respective quantile approximations using QR-DQN. We observe that they are almost indistinguishable.

Figure A9: QR-DQN quantile approximation for various density functions

I. Additional results

I.1. Frozen Lake

Figure A10 illustrates the Frozen Lake environment, a stochastic grid-world designed to highlight multimodal value distributions. An agent must navigate from a starting point to a goal while avoiding holes in the ice. Due to slipperiness, intended moves may result in perpendicular actions; for example, choosing to move right results in a 1/3 chance of moving right, up, or down. This high level of stochasticity induces significant variability in returns, making Frozen Lake a suitable testbed for visualizing multimodal Q-value distributions. The learned value distributions for each state-action pair, obtained using the surrogate loss, are displayed in Figure A11. States are ordered left to right, top to bottom.

Figure A10: Frozen lake environment

I.2. ATARI-5 Raw scores

Table 2: Raw scores on ATARI-5 Benchmark.

Games	Random	Human	DQN	C51	IQN	NFDRL-E	NFDRL-S
Battle Zone	2,360.0	37,187.5	29,900.0	28742.0	**42244.0**	32800.0	**34540.0**
Double Dunk	-18.6	-16.4	-6.6	2.5	5.6	**7.8**	7.6
Name this Game	2,292.3	8,049.0	8,207.8	12,542.0	**22,682.0**	15,667.2	**17,309.5**
Phoenix	761.4	7,242.6	8,485.2	17,490	**56,599**	18,914	**20,042**
Q*Bert	163.9	13,455.0	13,117.3	23,784	25,750	22,671	**25,852**

I.3. Implemetation Details and hyperparameters

Architecture. We base all baselines and our method on the same underlying neural network. Its architecture and training loop composition follows the structure used in CleanRL [18]. Our method however requires 4 separate heads outputing $\{w_i^j, \mu_i^j, \sigma_i^j\}$ and G_{max}^j. Hyperparameters are displayed in table 3. Our model was trained on an Nvidia RTX A5000 GPU.

As mentioned in main paper section 3.4, we choose $\sigma = 0.05$ to ensure the resulting Gaussian is sharply peaked around r, while still numerically stable. This value is derived by matching the

resolution of the C51 model, which discretizes the support $[0, 10]$ into 51 bins, yielding a bin width of approximately 0.2. To ensure that 95% of the Gaussian's mass lies within the central bin (i.e., $[r - 0.1, r + 0.1]$), we solve $2\sigma = 0.1$, yielding $\sigma = 0.05$. Of course, different values can be chosen to allow for sharper distributions.

Action selection. During training, actions are chosen using an ϵ-greedy strategy: with probability ϵ, a random action is selected; otherwise, the model selects the best action. The value of ϵ decreases over time. In the latter case, action selection follows the approach of C51 or IQN. The model includes one head per possible action, each outputting the parameters of a Gaussian mixture and G^{max}—defining a flow function per action. From these, n samples are drawn from the base distribution and passed through the corresponding flow. This yields n return values and densities per action. The expected return is computed for each, and the action with the highest expected value is chosen.

Hyperparameter	**Value / Description**
`total_timesteps`	10,000,000 (total timesteps of the experiments)
`learning_rate`	5e-5 (learning rate of the optimizer)
`max_norm`	3.0 (maximum allowable gradient norm)
`num_envs`	4 (number of parallel game environments)
`buffer_size`	1,000,000 (size of the replay memory buffer)
`gamma`	0.99 (discount factor)
`target_network_frequency`	1 (steps between target network updates)
`batch_size`	64 (batch size from replay memory)
`start_e`	1.0 (starting epsilon for exploration)
`end_e`	0.01 (final epsilon for exploration)
`exploration_fraction`	0.2 (fraction of timesteps for epsilon decay)
`learning_starts`	30,000 (timestep to start learning)
`train_frequency`	4 (training frequency)
`hidden_size_1`	512 (size of first hidden layer)
`hidden_size_2`	256 (size of second hidden layer)
`n_flows`	1 (number of flows)
`n_components`	4 (number of components in the mixture)
`n_samples`	500 (samples drawn from the base distribution)
`final_reward_variance`	0.1 (final state reward normal distribution variance)
`bandwidth`	0.05 (KDE bandwidth)

Table 3: List of hyperparameters used in our experiments.

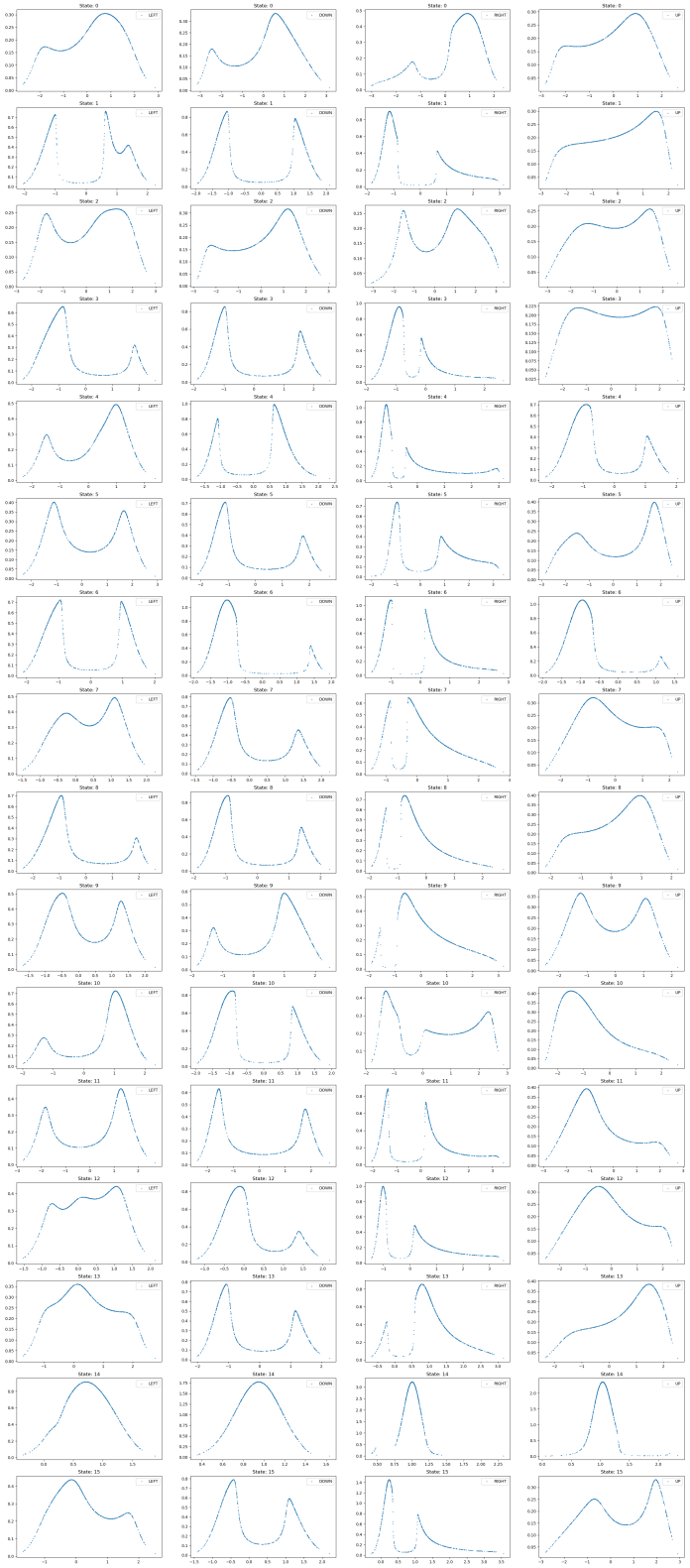


Figure A11: Learnt value distributions using the surrogate loss for each state-action pair of the Fozen Lake environment. States are numbered from left to right and up to down, i.e upper left is 0, bottom right is 16.