EFFICIENT CONSISTENCY MODEL TRAINING FOR POLICY DISTILLATION IN REINFORCEMENT LEARNING

Bowen Fang
Columbia University
New York, NY 10025
bf2504@columbia.edu

Xuan Di
Columbia University
New York, NY 10025
sharon@columbia.edu

ABSTRACT

This paper proposes an efficient consistency model (CM) training scheme tailored for the policy distillation step common in reinforcement learning (RL). Specifically, we leverage the Probability Flow ODE (PF-ODE) and introduce two novel training objectives designed to improve CM training efficiency when target policy log-probabilities are available for a limited set of reference actions. We propose Importance Weighting (IW) and Gumbel-Based Sampling (GBS) as strategies to refine the learning signal under these limited sampling budgets. Our approach enables more efficient training by directly incorporating target probability estimates, which aims to reduce variance and improve sample efficiency compared to standard CM training that relies solely on samples. Numerical experiments in a controlled setting demonstrate that our proposed methods, particularly IW, outperform conventional CM training, achieving more accurate policy representations with limited reference data. These findings highlight the potential of using CMs, trained with our proposed objectives, as an efficient alternative method for the policy distillation component within RL algorithms.

1 INTRODUCTION

Deep Generative Models (DGMs), especially diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020), excel at modeling complex distributions but their iterative sampling hinders use in Reinforcement Learning (RL) where frequent environment interactions are needed. Consistency Models (CMs) (Song et al., 2023) offer a faster alternative, capable of few-step sampling, making them promising for RL policy representation (Ding & Jin, 2023). Prior work has explored score-based models and CMs in RL, often within value-based or imitation learning frameworks (Chi et al., 2023; Wang et al., 2022; Hansen-Estruch et al., 2023; Yang et al., 2023; Ding et al., 2024; Psenka et al., 2023; Chen et al., 2023).

While this strategy works well in certain contexts, it is not ideal for RL algorithms that require explicit policy probabilities. Policy gradient RL methods (Sutton et al., 1999; Schulman et al., 2017; Abdolmaleki et al., 2018) and model-based approaches (Schrittwieser et al., 2020; Hubert et al., 2021) utilize explicit action log-probabilities during acting or training. These methods often involve an **improvement** step yielding a target policy p^*, and a **projection** (distillation) step updating the parameterized policy π_θ towards p^*. In this distillation phase, we have access to target log-probabilities $\log p^*(x_i)$ for a limited set of reference actions $\{x_i\}$. Standard generative model training often only uses samples, but we argue that leveraging available $\log p^*(x_i)$ values can make CM training more efficient for this distillation task.

We utilize the Probability Flow ODE (PF-ODE) (Song et al., 2020) formulation, which allows estimating sample density, and focus on CMs (Song et al., 2023) for their sampling speed. We propose two novel CM training losses for policy distillation that incorporate $\{\log p^*(x_i)\}$ to reduce variance compared to standard single-sample CM training (Appendix A.1), especially with limited reference actions. We demonstrate their effectiveness in controlled experiments and briefly explore integration

with PPO (Appendix A.3). Our primary focus is on improving the efficiency of the distillation step ($\mathcal{P}$) in a controlled setting, laying the groundwork for future application in broader RL scenarios.

2 PRELIMINARY

2.1 PROBLEM STATEMENT

Policy gradient methods can be cast as repeated applications of two main operators (Ghosh et al., 2020): an *improvement* operator $\mathcal{I}$ and a *projection* operator $\mathcal{P}$. Specifically:

Improvement $\mathcal{I}$. Given a current policy π_θ (parametrized by a neural network with parameters θ), the operator $\mathcal{I}$ produces an improved policy $\mathcal{I}\pi$ that achieves strictly higher returns. Formally, for all states $s \in S$, $v^{\mathcal{I}\pi}(s) \geq v^{\pi}(s)$, where v^π is the value function under policy π. Examples of such operators include MCTS-based methods (Hubert et al., 2021; Grill et al., 2020; Bertsekas, 2022), which refine a given policy into an improved one by searching over possible actions.

Projection $\mathcal{P}$. Once we have an improved policy $\mathcal{I}\pi$, we need to *distill* it back into the parametric policy network π_θ. The operator $\mathcal{P}$ accomplishes this by projecting $\mathcal{I}\pi$ onto the space of policies representable by our chosen function approximator. Typical operator $\mathcal{P}$ could be minimising the cross-entropy between π_θ and the improved policy $\mathcal{I}\pi$: $CE = \mathbb{E}_{x\sim\mathcal{I}\pi}[-\log \pi_\theta]$. This two-step process, $\mathcal{P}\circ\mathcal{I}$, is the core mechanism behind policy gradient algorithms. For example, MuZero (Schrittwieser et al., 2020) applies MCTS ($\mathcal{I}$) at each step to obtain a tree-search policy, and then updates (or *distills*) the policy network to match that improved policy ($\mathcal{P}$).

In this work, we focus on the *projection operator* $\mathcal{P}$. Specifically, we assume an improved policy $\mathcal{I}\pi$ is already given, and our goal is to distill it into a policy represented by *Consistency Model* efficiently. Suppose we have access to the improved policy's log-probabilities, $\log p^*$, over a (typically small) set of actions $\{x_i\}$. A naive application of Consistency Model training to these samples can lead to high variance, although it remains unbiased (see Song et al., 2023, Lemma 1 in Appendix A). One could reduce this variance by drawing more samples from the target policy, but that is often computationally expensive. Thus, we aim to develop a more efficient approach that better leverages the available improved action probabilities.

We adopt the usual Gaussian corruption scheme $x_t = x + \sigma_t z$ from VE-ODE (Song et al., 2020), but specialize it to our setting. We recall the Consistency Model Training in Appendix A.1, given x and x_t, the score function $\nabla \log p_t(x_t)$ can be estimated with $-(x_t - x)/t^2$. In standard Consistency Model training, one uses a single sample $x \sim p_{\text{data}}$ as the training target; here, however, the "data" distribution, which is unknown, is replaced by the improved policy distribution $p^*(x)$. Hence, we aim to approximate the posterior

$$p\big(x \mid x_t\big) \;\propto\; p^*(x)\,\mathcal{N}\big(x_t \mid x,\; \sigma_t^2\mathbf{I}\big),$$

which exploits the limited set of actions (and corresponding log-probabilities) from $\mathcal{I}\pi$ in a more efficient manner.

3 EFFICIENT CONSISTENCY TRAINING FOR POLICY DISTILLATION

In this section, we show how knowledge of the improved policy $\mathcal{I}\pi$'s log-density, $\log p^*(x)$, can help reduce variance and improve efficiency in training Consistency Models.

We highlight two strategies that utilize $\log p^*(x)$ under a limited sampling budget:

- **Importance Weighting** (IW): For a small set of samples $\{x_i\}$ with known $\log p^*(x_i)$, we form an approximate posterior mean by importance weighting.
- **Gumbel-Based Sampling** (GBS): For discrete candidate action sets, we sample from the posterior distribution using the Gumbel trick, which can further reduce variance in the estimated denoising target.

Here we present two training algorithms (Algorithms 1 and 2) showing how to incorporate these refined targets in CM training.

3.1 IMPORTANCE WEIGHTING

Suppose we have access to a set $\{x_i\}$ of reference actions, each with a known (or stored) value of $\log p^*(x_i)$. We would like to approximate the posterior mean

$$\mu(x_t) \;=\; \mathbb{E}[\, x \mid x_t \,]$$

An *importance weighting* (IW) strategy is:

1. For each x_i in the reference set, compute the unnormalized posterior weight
$$w_i \;=\; \exp\Big(\log p^*(x_i)\Big)\, \exp\Big(-\tfrac{1}{2t^2}\|x_t - x_i\|^2\Big). \tag{1}$$
2. Normalize $w_i \leftarrow w_i / \sum_{j=1}^n w_j$.
3. Form the IW estimate of the posterior mean:
$$\widehat{\mu}_{\mathrm{IW}}(x_t) \;=\; \sum_{i=1}^n w_i x_i. \tag{2}$$

This $\widehat{\mu}(x_t)$ is an approximation to $\mathbb{E}[x \mid x_t]$ *restricted* to the finite set $\{x_i\}$. We then use

$$s_{\mathrm{IW}}(x_t) \;=\; -\,\frac{x_t \;-\; \widehat{\mu}(x_t)}{t^2}$$

as an improved target for training the model's denoiser. See Algorithm 1 for pseudocode that integrates this step into a CM-like loss.

3.2 GUMBEL-BASED SAMPLING

An alternative to directly constructing the posterior mean is to draw approximate from $p(x \mid x_t)$, using the Gumbel trick on a finite candidate set. For each reference x_i, draw an i.i.d. Gumbel variable $g_i \sim \mathrm{Gumbel}(0,1)$ and pick

$$i^* \;=\; \arg\max_i \Big[\, \log p^*(x_i) \;-\; \tfrac{1}{2\sigma_t^2}\|x_t - x_i\|^2 \;+\; g_i \Big].$$

And $x_{\text{sample}} = x_{i^*}$. This yields a random draw from the discrete distribution whose unnormalized log-probabilities are $\log p^*(x_i) \;-\; \frac{1}{2\sigma_t^2}\|x_t - x_i\|^2$. By re-sampling g_i multiple times, we can obtain multiple samples $x^{(1)}_{\text{sample}}, x^{(2)}_{\text{sample}}, \ldots$ from the approximate posterior. One may then average them to get an empirical posterior mean, or simply treat the sampled x_{sample} as a ***single-sample*** training target. We adopt the latter one as demonstrated in Algorithm 2.

4 EXPERIMENT

To evaluate our proposed training methods for distilling the improved policy $\mathcal{I}\pi$ into a Consistency Model, we conduct a controlled experiment in a two-dimensional action space. Specifically, we create a mixture of truncated Gaussian components with randomly sampled means, scales, and mixture weights (normalized to sum to one). We constrain the means and scales to lie within the action range $[x_{\min}, x_{\max}]$, and use this mixture distribution as the target policy. For each experiment instance, we sample a set of actions along with their probabilities $\{x_i, \log p^*(x_i)\}_{i=1}^n$ from the target policy, and then train Consistency Models under various loss formulations.

We quantify how well each loss recovers the target distribution by computing the mean-squared error (MSE) between the estimated and true log-likelihoods. We repeat the experiment with 5 distinct instances of the mixture distribution, each with 5 different random seeds. We adopt the same neural network architecture as in Ding & Jin (2023) throughout.

In terms of implementation, we follow the Karras diffusion procedure (Karras et al., 2022), where the probability flow ODE (PF-ODE) is given by

$$\frac{dx}{dt} \;=\; -\frac{x_t - D_\theta(x_t, t)}{t}.$$

Algorithm 1 CM Training with Importance Weighting

Input: Denoiser D_θ, learning rate η, reference set $\{x_i, \log p^*(x_i)\}_{i=1}^n$, noise schedule $\{\sigma_t\}_{t=0}^T$.
repeat
  Sample a batch $\{x^{(b)}\}_{b=1}^B$ from the reference set.
  for each batch element $b = 1, \ldots, B$ **do**
    Sample a time t_b.
    Generate $z^{(b)} \sim \mathcal{N}(0, I)$ and noise the sample $x_t^{(b)} = x^{(b)} + \sigma_{t_b} z^{(b)}$.
    Compute importance weights $\{w_i^{(b)}\}_{i=1}^n$ via Eq. equation 1, using reference set.
    Normalize $w_i^{(b)} \leftarrow w_i^{(b)} / \sum_{j=1}^n w_j^{(b)}$.
    Compute $\widehat{\mu}(x_t^{(b)}) = \sum_{i=1}^n w_i^{(b)} x_i$.
    Denoise $\hat{x}^{(b)} = D_\theta(x_t^{(b)}, \sigma_{t_b})$.
    MSE loss $L_b = \|\hat{x}^{(b)} - \widehat{\mu}(x_t^{(b)})\|^2$.
  end for
  $\theta \leftarrow \theta - \eta \nabla_\theta \frac{1}{B} \sum_{b=1}^B L_b$
until convergence

Algorithm 2 CM Training with Gumbel-Base Sampling

Input: Denoiser D_θ, learning rate η, reference set $\{x_i, \log p^*(x_i)\}_{i=1}^n$, noise schedule $\{\sigma_t\}_{t=0}^T$.
repeat
  Sample a batch $\{x^{(b)}\}_{b=1}^B$ from the reference set.
  for each batch element $b = 1, \ldots, B$ **do**
    Sample a time t_b.
    Generate $z^{(b)} \sim \mathcal{N}(0, I)$ and noise the sample $x_t^{(b)} = x^{(b)} + \sigma_{t_b} z^{(b)}$.
    Draw Gumbel variables $\{g_i\}_{i=1}^n$, with $g_i \sim \mathrm{Gumbel}(0, 1)$.
    $\ell_i = \log p^*(x_i) - \frac{1}{2\sigma_{t_b}^2} \|x_t^{(b)} - x_i\|^2 + g_i$.
    $i^* = \arg\max_i \ell_i$.
    Evaluate $\hat{x}^{(b)} = D_\theta(x_t^{(b)}, \sigma_{t_b})$.
    MSE loss $L_b = \|\hat{x}^{(b)} - x_{i^*}\|^2$.
  end for
  $\theta \leftarrow \theta - \eta \nabla_\theta \frac{1}{B} \sum_{b=1}^B L_b$
until convergence

We employ Hutchinson's trick to compute the log-likelihood of generated actions, and for each action sample $\{x_i\}$, we obtain an estimated log-likelihood by first adding noise to generate $x_T = x + \sigma_{\max} z$ and then solving the PF-ODE backward in time. We refer to this calculation as "log_probs" in the following experiments. Finally, we compare these estimated log-likelihoods to those derived via the sampling process, thereby assessing how closely the estimated log-likelihoods match each other.

Table 1: Performance comparison of three loss functions. All metrics shown here are lower-is-better.

Loss Function	LogP MSE	LogP MAE Diff(samples-log_probs)	Score MAE
Original	8.3467 ± 1.9236	0.0210	12.9156 ± 5.2907
Gumbel	7.3514 ± 1.6057	0.0201	10.4524 ± 4.5369
Importance	**7.2986 ± 1.4001**	**0.0009**	**9.5578 ± 3.3169**

Table 1 compares three different loss functions with respect to three metrics, all of which are lower-is-better. The *Original* loss function exhibits the highest (i.e., worst) errors across all metrics, whereas the *Gumbel* and *Importance* achieve improvements. The *Importance* loss function yields the lowest error in every category, indicating that it most effectively distills the target distribution. Figure 2a, 2b demonstrates the evolving of LogP MSE and LogP MAE difference between sampling and log_probs calculation across epochs. Overall, the results highlight the advantage of incorporating importance weights, leading to more stable training and improved performance. More results are presented in the Appendix A.2

5 Conclusion

In conclusion, the proposed Consistency Model training losses, Importance Weighting, and Gumbel-Based Sampling, demonstrate clear benefits for policy distillation in reinforcement learning. By reducing variance and improving sample efficiency, the proposed methods outperform conventional Consistency Model training and deliver more accurate policy representations under limited sampling conditions.

REFERENCES

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

Dimitri Bertsekas. *Lessons from AlphaZero for optimal, model predictive, and adaptive control*. Athena Scientific, 2022.

Yuhui Chen, Haoran Li, and Dongbin Zhao. Boosting continuous control with consistency policy. *arXiv preprint arXiv:2310.06343*, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *arXiv preprint arXiv:2405.16173*, 2024.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023.

Dibya Ghosh, Marlos C Machado, and Nicolas Le Roux. An operator view of policy gradient methods. *Advances in Neural Information Processing Systems*, 33:3397–3406, 2020.

Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. Monte-carlo tree search as regularized policy optimization. In *International Conference on Machine Learning*, pp. 3769–3778. PMLR, 2020.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pp. 4476–4486. PMLR, 2021.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

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. *Software Impacts*, 6:100022, 2020.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

A Appendix

A.1 Recall of Consistency Model Training

Recall that the Consistency Training avoids the pre-trained score model by leveraging the unbiased estimator:

$$\nabla \log p_t(x_t) \;=\; -\mathbb{E}\Big[\frac{x_t - x}{t^2}\,\Big|\, x_t\Big]. \tag{3}$$

where $x \sim p_{\text{data}}$ and $x_t \sim \mathcal{N}(x_t \mid x, t^2\boldsymbol{I})$. Then given x and x_t, $\nabla \log p_t(x_t)$ can be estimated with $-(x_t - x)/t^2$. It participates in the Consistency Training loss $\mathcal{L}_{CT}^N(\theta, \theta^-)$ as:

$$\begin{aligned}
\mathcal{L}_{CT}^N(\theta,\theta^-) &= \mathbb{E}\Big[\lambda(t_n)\, d\big(f_\theta(x_{t_{n+1}}, t_{n+1}),\, f_{\theta^-}\big(x_{t_{n+1}} + (t_n - t_{n+1})t_{n+1}\,\frac{x_{t_{n+1}} - x}{t_{n+1}^2},\, t_n\big)\big)\Big] \\
&= \mathbb{E}\Big[\lambda(t_n)\, d\big(f_\theta(x_{t_{n+1}}, t_{n+1}),\, f_{\theta^-}\big(x_{t_{n+1}} + (t_n - t_{n+1})z,\, t_n\big)\big)\Big] \\
&= \mathbb{E}\Big[\lambda(t_n)\, d\big(f_\theta(x + t_{n+1}z, t_{n+1}),\, f_{\theta^-}\big(x + t_{n+1}z + (t_n - t_{n+1})z,\, t_n\big)\big)\Big] \\
&= \mathbb{E}\Big[\lambda(t_n)\, d\big(f_\theta(x + t_{n+1}z, t_{n+1}),\, f_{\theta^-}\big(x + t_n z,\, t_n\big)\big)\Big]
\end{aligned}$$

leveraging $z := \frac{x_{t_{n+1}} - x}{t_{n+1}} \sim \mathcal{N}(0, \boldsymbol{I})$.

A.2 Experiment

We evaluate the performance of different sampling methods by comparing their empirical distributions against the true distribution in Figure 1. Each subplot represents a different experiment instances. The x-axis corresponds to the action space, while the y-axis shows the density on a logarithmic scale.

Overall, the results demonstrate that incorporating Gumbel and importance weighting strategies improves the quality of sampled actions, yielding distributions that better approximate the true target.

A.3 Consistency Model as PPO Actor

We evaluate a consistency-model-based Proximal Policy Optimization (PPO), referred to as CM-PPO, on the dm_control (Tunyasuvunakool et al., 2020) `cheetah:run` task using five different random seeds. The key modification lies in the PPO loss function: in addition to the standard clipped surrogate objective, we incorporate a conventional consistency loss scaled by the advantage and modulated by a time-dependent factor that decreases as training progresses. Formally, the combined objective is expressed as:

$$L_{\text{policy}} = L_{\text{clipped\_PPO}} + \lambda(t)\, \mathbb{E}_{(s,x)\sim\pi_\theta}\big[A(s,x)\, L_{\text{consistency}}(s,x)\big],$$

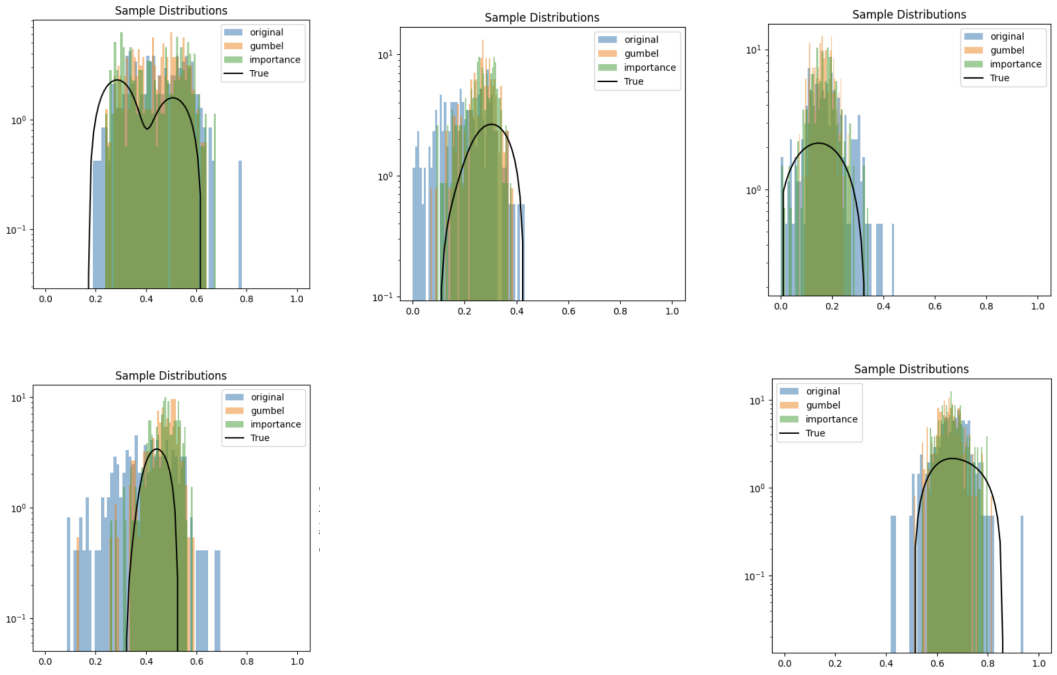
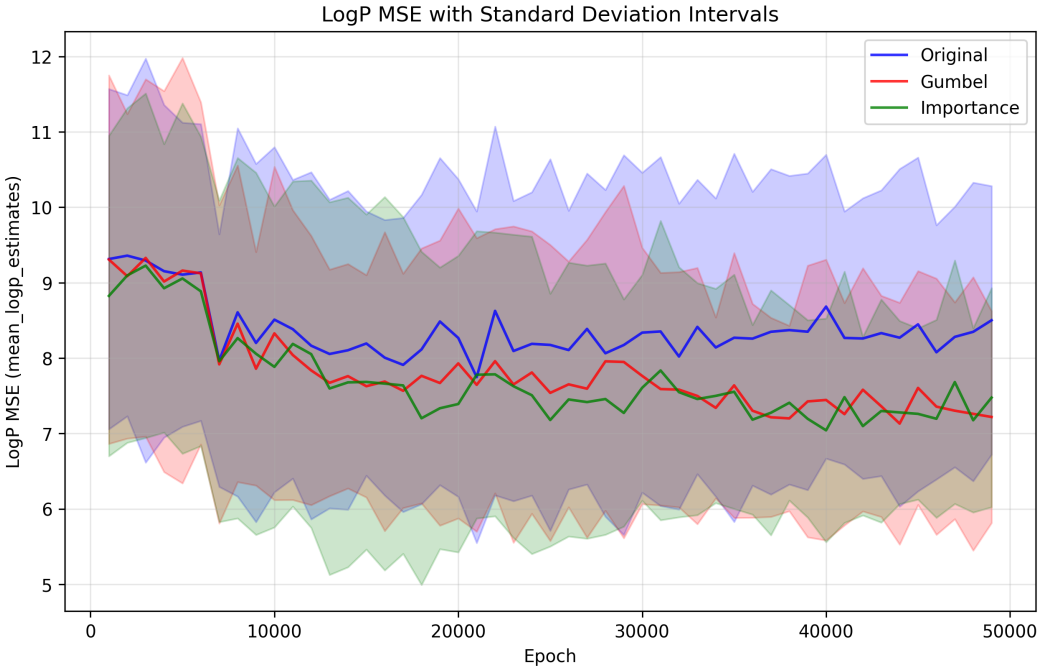


Figure 1: Sample distributions for Consisntecy Models trained with different losses, compared to the target distribution

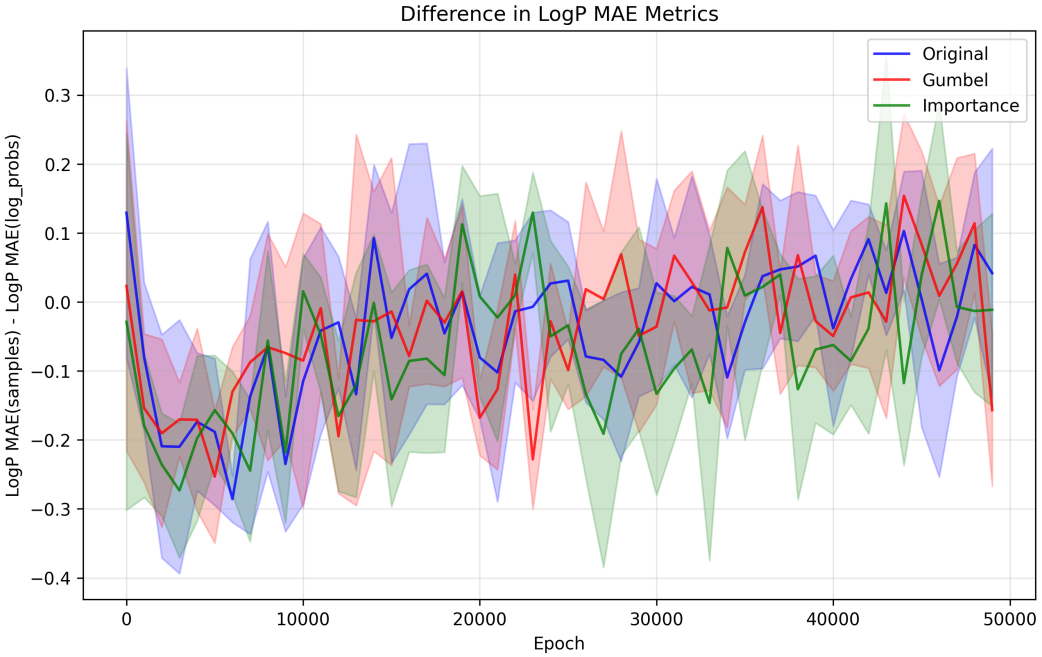
Table 2: Summary of experiment parameters.

Parameter	**Value**
Number of mixture components	2,3
Action dimension	2
Number of epochs	10^5
Learning rate	1×10^{-4}
Size of reference set	16
Batch size	64
Number of diffusion steps	10
$\sigma_{\min}$	0.002
$\sigma_{\max}$	80
Action range	[0, 1]

where $A(s, x)$ denotes the advantage. While the resulting policy demonstrates some capacity for control, as Figure 3 shows, its performance exhibits high variance and remains notably below that of a standard continuous policy. These observations suggest that additional techniques may be necessary to fully leverage the potential of Consistency Models and probability flow in policy gradient methods.



(a) LogP MSE across training epochs



(b) Difference in LogP MAE from sampling and log_probs calculation across training epochs, closer to zero is better

Figure 3: Episode Returns with Confidence Interval across Actor Episodes