OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Path Integral Optimiser: Global Optimisation via Neural Schrödinger-Föllmer Diffusion

Max McGuinness MGM52@CAM.AC.UK
Eirik Fladmark EF454@CAM.AC.UK
Francisco Vargas FAV25@CAM.AC.UK
University of Cambridge

Abstract

We present an early investigation into the use of neural diffusion processes for global optimisation, focusing on Zhang et al.'s Path Integral Sampler. One can use the Boltzmann distribution to formulate optimization as solving a Schrödinger bridge sampling problem, then apply Girsanov's theorem with a simple (single-point) prior to frame it in stochastic control terms, and compute the solution's integral terms via a neural approximation (a Fourier MLP). We provide theoretical bounds for this optimiser, results on toy optimisation tasks, and a summary of the stochastic theory motivating the model. Ultimately, we found the optimiser to display promising per-step performance at optimisation tasks between 2 and 1,247 dimensions, but struggle to explore higher-dimensional spaces when faced with a 15.9k parameter model, indicating a need for future work on adaptation in such environments.

1. Introduction

Motivation: Most tasks in machine learning can be characterised as *high-dimensional non-convex stochastic optimisation problems*. Stochastic gradient descent (SGD) [21] is the archetypical approach in differentiable settings but neglects second-order gradient information. Many extensions have been proposed to incorporate such information—such as Adam [11], Adagrad [7], and momentum [16, 22]. However, while effective in the early stages of training, these approaches struggle to generalise as broadly as SGD [10], and their theoretical convergence properties remain poorly understood for non-convex global minimisation [5, 13]. With diffusion models being shown to be highly effective at sampling from high-dimensional structured distributions [18], and offering promising theoretical guarantees [24, 27], this paper wishes to apply those benefits to optimisation.

Contributions We propose one of the first practical diffusion-based optimisers, the *Path Integral Optimiser (PIO)*: a neural Schrödinger-Föllmer diffusion process trained as a HyperNetwork. We outline PIO in **Section 3**; derive theoretical guarantees for PIO as an optimiser in **Section 4**; perform an empirical study comparing PIO and PIO-like models to popular optimisers at classic machine learning tasks, in some cases matching or exceeding their performance, in **Section 5**; and conclude with recommendations for future work in **Section 6**. We also publish our codebase.[1]

1. https://github.com/mgm52/Path-Integral-Optimiser

2. Preliminary

2.1. The Schrödinger-Föllmer Process

The Schrödinger bridge problem is that of finding the most likely stochastic evolution from an initial distribution $\mathbb{Q}$ into a target $\mathbb{P}$ while minimising KL-divergence to a Wiener process W.

Definition T2.1 *In **stochastic control terms**, the **explicit dynamic Schrödinger bridge problem** for probability measures $\delta_0 : \mathbb{R}^n \to \mathbb{R}$ and $\mathbb{P} : \mathbb{R}^n \to \mathbb{R}$ is that of finding the minimal drift μ for an identity-variance Itô process $X = \{X_t \mid t \in [0,1]\}$, $X_t : (\mathbb{R}^n, \mathcal{F}) \to (\mathbb{R}^n, \mathcal{F})$ to transform δ_0 into $\mathbb{P}$. For $X_T = (X_0 \sim \delta_0) + \int_0^T \mu_t dt + \int_0^T dW_t$, we seek the minimal drift to reach $\mathbb{P}$:*

$$\mu = \arg\min_{\mu} \mathbb{E}\left[\frac{1}{2}\int_0^1 \|\mu_t\|^2 dt + \log \frac{P_{W_1}(X_1)}{\mathbb{P}(X_1)}\right].$$

When $\mathbb{Q}$ is a dirac distribution, one can solve this problem via Schrödinger-Föllmer diffusion [8, 9].

Definition T2.2 *The **Schrödinger-Föllmer process** is an identity-variance diffusion process that solves the dynamic Schrödinger-Föllmer problem from Dirac $X_0 \sim \delta_0 = \mathbb{Q}$ to arbitrary target $X_1 \sim \mathbb{P}$. Taking $f : \mathbb{R}^n \to \mathbb{R}^n$ to be the ratio $f(x) = \frac{d\mathbb{P}}{dN(0,\mathbf{I}_n)}(x)$:*

$$X_T = \delta_0 + \int_0^T \frac{\mathbb{E}_{Z\sim N(0,I_n)}[\nabla f(X_t + \sqrt{1-t}Z)]}{\mathbb{E}_{Z\sim N(0,I_n)}[f(X_t + \sqrt{1-t}Z)]} dt + \int_0^T dW_t.$$

2.2. Motivating a Neural Approximation

To overcome the drift's expectation terms not having an analytic closed-form (except in specific cases, like the target being a Gaussian mixture), Huang et al. [9] proposed a Monte-Carlo approach. However, Zhang et al. [27] highlight that this approach is analogous to importance sampling[2] of target distribution $X_1 \sim \mathbb{P}$ with proposal distribution $Z_i \sim N(0, I_n)$, which entails two major drawbacks encompassing excessively high variance: the **curse of dimensionality** and the **curse of distribution alignment** (required samples increases exponentially with KL divergence).

Tzen et al. [23] found that, as an alternative to Monte-Carlo estimation, one can use an MLP to efficiently approximate the drift term to arbitrary accuracy, so long as the network can efficiently approximate target density $\mathbb{P}$. Intuitively, a well-formed neural network can increase in depth *linearly* to increase expressivity exponentially [17]. We note an additional advantage of the neural/meta approach: no fixed time commitment. One can pause or extend the approximation network's training at any point and then take a viable trajectory; by contrast, the Monte-Carlo method spends all compute running one trajectory to exactly $t = T$, no shorter or longer.

2.3. Path Integral Sampler

Zhang et al. [27] describe a complete implementation of neural-approximated discrete SFP in the **Path Integral Sampler (PIS)**. PIS learns parameters θ by simply re-using the drift goal in Definition **T2.1** as the loss in a standard gradient-based optimisation algorithm (e.g. Adam [11]).

2. To be specific, it may be more apt to compare the SF process to *annealed* importance sampling, which transforms an initial distribution into a target distribution by interpolating the geometric average of the two distributions.

Definition T2.3 *The **EM-discretised neural approximation** to the **Schrödinger-Föllmer process** makes use of neural drift term $\widehat{b}_\theta : (\mathbb{R}^n, [0,1]) \to \mathbb{R}^n$, such that $\widehat{b}$ is a feed-forward multilayer neural net with parameters θ, and is defined for K timesteps, with timesteps $t_i = i/K$, as:*

$$X_T = \delta_0 + \sum_{i=0}^{T} \widehat{b}_\theta(X_{t_i}, t_i)/K + \sum_{t=0}^{T} (W_{t_{i+1}} - W_{t_i}).$$

Definition T2.4 *The **Path Integral Sampler** approximates optimal parameters θ^* to solve the explicit dynamic Schrodinger Bridge problem **T2.1** using the **loss function** goal $\arg\min_\theta \mathcal{L}(\theta) \approx \theta^*$, with $\mathcal{L}(\theta) : (\mathbb{R}^n, \mathcal{F}) \to (\mathbb{R}, \Sigma)$ defined as:*

$$\mathcal{L}(\theta) = \frac{1}{2}\int_0^1 \|\widehat{b}_\theta(X_t, t)\|^2 dt + \log \frac{P_{W_1}(X_1)}{\mathbb{P}(X_1)}.$$

The assumption that target distribution $\mathbb{P}$ has an analytic form in the loss term characterises the process as solving the *uniform-density-to-$\mathbb{P}$-density sampling problem*, hence Zhang et al. referring to it as a sampler. To enable unbiased sampling, Zhang et al. [27] perform importance sampling, assigning each trajectory a weight measuring the alignment of its terminal state with $\mathbb{P}$.

2.4. SFP For Optimisation

One can use a Boltzmann transformation to characterise optimisation as a *sampling* problem.

Definition T2.5 *Given loss function $\mathcal{L} : \mathbb{R}^p \to (\Omega \to \mathbb{R})$ and scalar $\sigma \in (0, 1]$, we define the **Boltzmann density** $\mathbb{P}$ as*

$$\mathbb{P}(\theta) = \frac{\exp(-\mathcal{L}(\theta)/\sigma)}{\int_{\mathbb{R}^p} \exp(-\mathcal{L}(x)/\sigma)dx}.$$

For sufficiently small σ, sampling from $\mathbb{P}$ is equivalent to identifying the optimal parameters $\theta_* \in \arg\min_{\theta\in\mathbb{R}^p} \mathcal{L}(\theta)$ such that $\lim_{\sigma\to 0} \mathbb{P}(\theta_*) = 1$.

Because Schrödinger-Föllmer processes (**T2.2**) are scale-invariant with regard to the target distribution—i.e. constant factors in f are cancelled out by the $\nabla f/f$ division in the drift term—one can ignore the Boltzmann distribution's integral denominator $\int_{\mathbb{R}^p} \exp(-\mathcal{L}(x)/\sigma)dx$ and sample with the more readily-computable formula $\mathbb{P}(\theta) = \exp(-\mathcal{L}(\theta)/\sigma)$.

To enable the drift term to appropriately match the scale dictated by σ, Dai et al. [4] rescaled the process's terms using σ and $\sqrt{\sigma}$: $X_1 = \delta_0 + \int_0^1 \sigma\widehat{b}_\theta(X_t, t)dt + \int_0^1 \sqrt{\sigma}dW_t$.[3] Simulating the process targeting a Carrillo function, they found the sampler to be successful at global optimisation, surpassing the Langevin-dynamics sampling method. However, while Dai et al.'s results on the Carrillo function are promising, there has not yet been an attempt to use an SFP for stochastic optimisation, nor machine-learning optimisation, which presents scaling "curses" for the Monte-Carlo approach as mentioned in Section 2.2.

2.5. Alternative Processes

While this paper focuses on PIS as an archetype, it can be seen as part of a greater framework of methods that solve the Schrödinger Bridge problem and can thus be applied to optimisation—albeit with varying theoretical guarantees. Notable alternatives include **Langevin dynamics** of

3. This rescaling is not justified in the paper; it appears that Dai et al. may have been motivated by the rescaling that is necessary for *Langevin* dynamics, though we are not certain that the same is necessary for SFP.

the form $dX_T = -\nabla\frac{1}{2}\nabla \log f(X_t, t)dt + dW_t$ for target density f, which tend to require a far longer mixing time than SFP [4]; and **denoising diffusion** with a time-reversal of the form $dY_t = -\beta_{T-t}(Y_t + 2\sigma^2\nabla \log p_{T-t}(Y_t))dt + \sigma\sqrt{2\beta_{T-t}}dW_t$ from $Y_0 \sim \mathcal{N}_{\text{approx}}(0, \sigma^2)$, which may be more numerically stable than SFP but incurs an additional approximation error due to starting from an approximate gaussian rather than a point [25].

3. Path Integral Optimiser

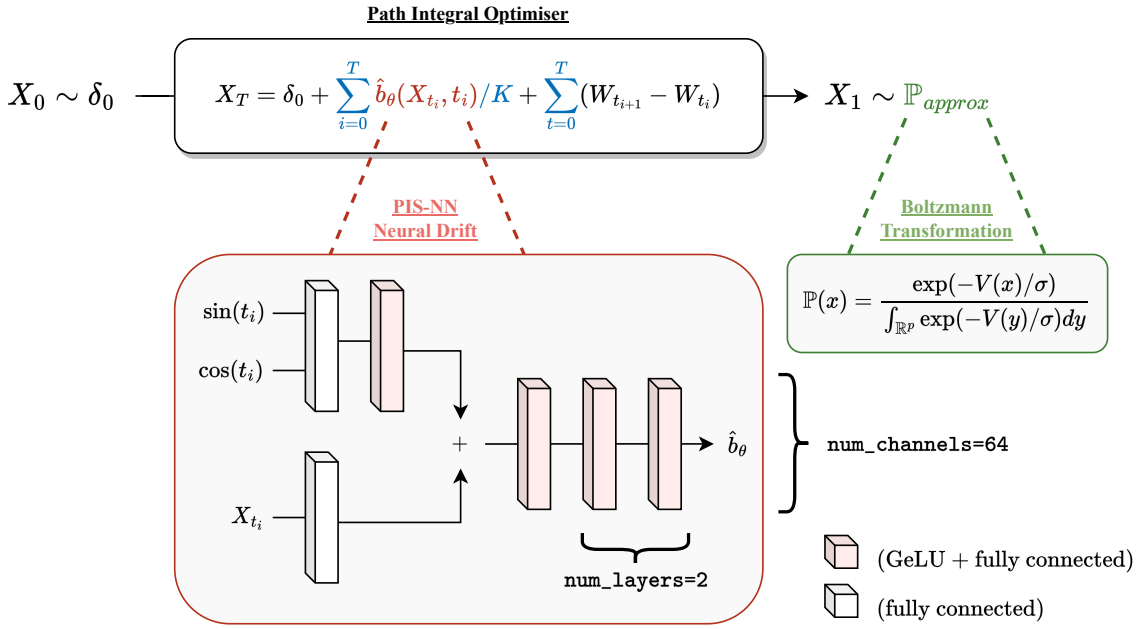


Figure 1: The **Path Integral Optimiser** is a neural-approximated Schrödinger-Föllmer process (**T2.3**) which minimises a loss function V by learning to generate samples from its Boltzmann distribution (green). The drift term is computed by a Fourier MLP (red), and the process as a whole is simulated with Euler-Maruyama discretisation (blue). With correct parameterisation of θ, σ, and K, this architecture is capable of global optimisation as proven in this paper's corollary (**T4.7**).

To perform practical optimisation via a Schrödinger-Föllmer process, alleviating the scaling issues of Dai et al.'s Monte-Carlo optimiser (Section 2.4), we adapt Xiao et al.'s Path Integral Sampler [27] architecture to sample from the Boltzmann transformation of a parameters-to-loss function. We refer to this complete model as a **Path Integral Optimiser** (PIO). Each training step involves computing at least one full trajectory, then Adam-updating PIO's approximation network according to PIS loss (**T2.4**). At validation, we run multiple trajectories and take the argmin validation loss. An unusual advantage of PIO as an optimizer is that, once trained, we effectively get an ensemble of variants for "free" by computing new stochastic trajectories.

Architecturally, PIO uses the *PIS-NN* approach put forward by Zhang et al. [27], in which a single time-conditioned feed-forward network $NN_\theta(X_t, t)$ approximates the entire drift. Alternatively, PIO-Grad uses the *PIS-Grad* formulation, which may be less impractical for general machine learning optimisation due to the cost of re-computing target score $\nabla \log \mathbb{P}$ at each timestep as guidance.

To assist PIO at optimisation, we decrease the Boltzmann parameter σ in lockstep with learning rate, matching higher-variance loss landscapes with finer-grained parameter updates.

4. Theoretical Guarantees

We derive a core theorem for the usability of discretised *neural-approximated* Schrödinger-Föllmer-like processes (**T2.3**) such as PIO for global optimisation, by building on the theoretical results of several SFP papers [4, 23, 24]. These bounds are also relevant to *denoising diffusion* optimizers (2.5), which one may call Föllmer-like due to satisfying the same expressiveness bounds and having a drift term that can be similarly expressed in terms of a semigroup (the semigroup being defined by score) [20].

To improve interpretability, we provide more informal corollaries in the *Remarks* section. Full derivations can be found in the appendix.

Theorem T4.6 *Suppose assumptions **AA.1** - **AA.4** hold. Then, for each $\epsilon \in (0, \tau), \sigma \in (0, 1), \widehat{\epsilon} \in (0, 16L^2/c^2), K \in \mathbb{Z}^+$, there exist constants $C_{\tau,\epsilon,p}$ and $C_{L'}$ (depending on τ, ϵ, p and L' respectively) such that*

$$P_{\widehat{Y}_1}(V(X_1) > \tau) \leq C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) + \sqrt{2\widehat{\epsilon}} + \sqrt{4L'^2(C_{L'}/K + 1/K^2)},$$

for Schrödinger Föllmer process $\widehat{Y}$ EM-discretised with step size $1/K$, whose neural drift $\widehat{b}_\theta(x, t) = \widehat{v}(x, \sqrt{1-t})$ is L'-Lipschitz in both parameters.

4.0.1. REMARKS

Theorem **T4.6** may be more intuitively phrased as *the probability of discrete neural SFP $\widehat{Y}$ being a τ-global minimiser of function V:*

$$P_{\widehat{Y}_1}(V(X_1) < \tau) \geq 1 - \left(C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) + \sqrt{2\widehat{\epsilon}} + \sqrt{4L'^2(C_{L'}/K + 1/K^2)}\right).$$

We can trivially see that Theorem **T4.6** enables global optimisation by considering the fact that, by Dai et al.'s theorem, there exists a neural SFP drift for every $\widehat{\epsilon} > 0$; that by Tzen et al.'s assumptions, we can set $\sigma \in (0, 1]$ arbitrarily; and that by EM-discretisation, we can set $1/K \in (0, 1]$ arbitrarily.

Therefore, the three parameters can tend to 0 to reach a global optimiser:

$$\begin{aligned}
&\lim_{\sigma,\widehat{\epsilon},1/K\to 0} \left[C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) + \sqrt{2\widehat{\epsilon}} + \sqrt{4L'^2(C_{L'}/K + 1/K^2)}\right] \\
&= \lim_{\sigma\to 0} C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) + \lim_{\widehat{\epsilon}\to 0} \sqrt{2\widehat{\epsilon}} + \lim_{1/K\to 0} \sqrt{4L'^2(C_{L'}/K + 1/K^2)} \\
&\to 0 + 0 + 0.
\end{aligned}$$

Additionally, as a more informal result from Theorem **T4.6**, we present the following corollary:

Corollary T4.7 *By Theorem **T4.6**, $\forall 0 < \delta \ll 1$, with probability at least $1 - \sqrt{\delta}$, $\widehat{Y}_1$ is a τ-global minimiser of V, i.e., $V(\widehat{Y}_1) \leq \tau + \inf V(x)$, if for the scaling factor σ, neural error $\hat{\epsilon}$, and number of iterations K, we have:*

$$\mathcal{O}\left(\frac{\tau - \epsilon}{\ln(1/\delta)}\right) \geq \sigma \quad \wedge \quad \mathcal{O}(\delta) \geq \widehat{\epsilon} \quad \wedge \quad \mathcal{O}(L'/\sqrt{\delta}) \leq K.$$

5. Experimental Results

Sweep Runs/Steps	**Seed** Runs/Steps	**Optimiser**	**Carrillo [2]** ($\lvert\phi\rvert = 2$)	**Moons [1]** ($\lvert\phi\rvert = 41$)	**MNIST [12]** ($\lvert\phi\rvert \approx 15.9k$)
32/32	**10*/100**	**Adagrad [7]**	*0.549* $\pm$ 0.153	0.184 $\pm$ 0.036	**0.425** $\pm$ 0.031
		Adam [11]	0.597 $\pm$ 0.536	**0.057** $\pm$ 0.047	0.476 $\pm$ 0.030
		SGD [21]	**0.491** $\pm$ 0.125	0.356 $\pm$ 0.019	*0.427* $\pm$ 0.034
		PIO	1.458 $\pm$ 0.882	*0.129* $\pm$ 0.121	2.464 $\pm$ 0.026
64/64	**10*/100**	**Adagrad [7]**	0.647 $\pm$ 0.448	0.193 $\pm$ 0.038	*0.347* $\pm$ 0.020
		Adam [11]	*0.498* $\pm$ 0.000	**0.022** $\pm$ 0.010	**0.346** $\pm$ 0.020
		SGD [21]	0.553 $\pm$ 0.149	0.428 $\pm$ 0.026	0.385 $\pm$ 0.011
		PIO	**0.404** $\pm$ 0.417	*0.032* $\pm$ 0.049	2.424 $\pm$ 0.045

Table 1: Min test loss $V(x)$ across optimisers on *mini-batch* training. Bold indicates best-in-column; italics second-best. The 64/64 runs make use of an annealing routine that drops both LR and PIO's σ parameter by 50% on plateaus. *5 runs for MNIST, 10 otherwise.

As a work-in-progress result, in Table 2, we find PIO-Grad to provide more competitive per-step performance at the cost of compute (of around $10\times$ that of PIO for Circles vs Moons). We also compare to DDO, a denoising diffusion sampler [25] (2.5) driven by a PIS-Grad-like network.[4]

Sweep Runs/Steps	**Seed** Runs/Steps	**Optimiser**	**Circles [1]** ($\lvert\phi\rvert = 64$)	**Breast Cancer [6]** ($\lvert\phi\rvert = 171$)	**Covertype [6]** ($\lvert\phi\rvert = 1,247$)
6/300*	**1/300***	**Adam [11]**	*0.998*	0.954	0.772
		AdamW [14]	*0.998*	*0.966*	0.766
		Noisy-SGD [15]	0.997	0.941	0.684
		SGD [21]	0.997	**0.970**	0.769
		SGLD [26]	0.934	0.965	0.656
3/300*	**1/300***	**DDO-Grad**	**0.999**	0.951	*0.801*
		PIO-Grad	*0.998*	0.944	**0.813**

Table 2: Test accuracy % across optimisers on *full-batch* training. Bold indicates best-in-column; italics second-best. *150 steps for Covertype, 300 otherwise.

6. Conclusions

Despite the benefits of being able to stochastically generate alternate parameterisations, the application of diffusion models to ML-optimization has had little attention in literature. We address this by introducing the Path Integral Optimiser (PIO), an application of the Path Integral Sampler to optimisation by use of the Boltzmann transformation. We derive theoretical guarantees for such discretized neural Schrödinger-Föllmer processes at the task of optimization, showing that—for example—the parameter σ should decrease logarithmically with δ to achieve P(global optimisation) over $1 - \sqrt{\delta}$. In empirically evaluating PIO, we match or exceeds classical optimizers in tasks up to 1.5k parameters in size, but the implementation struggles to scale when given an 15.9k-parameter MLP MNIST optimisation problem. Future work should focus on improving scalability by considering larger drift-approximation networks, ensembling PIO's trajectories, pre-training to a known distribution, and better-parallelising the training routine to make PIS-Grad more performant.

4. Results from an independent implementation to Table 1, using a different codebase and experimental setup.

References

[1] Jenny Balfer, Jürgen Bajorath, and Martin Vogt. Compound classification using the scikit-learn library. *Tutorials in Chemoinformatics*, pages 223–239, 2017.

[2] José A Carrillo, Shi Jin, Lei Li, and Yuhua Zhu. A consensus-based global optimization method for high dimensional machine learning problems. *ESAIM: Control, Optimisation and Calculus of Variations*, 27:S5, 2021.

[3] Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

[4] Yin Dai, Yuling Jiao, Lican Kang, Xiliang Lu, and Jerry Zhijian Yang. Global Optimization via Schrödinger-Föllmer Diffusion. *arXiv preprint arXiv:2111.00402*, 2021.

[5] Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration. *arXiv preprint arXiv:1807.06766*, 2018.

[6] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL https://archive.ics.uci.edu/ml.

[7] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[8] Hans Föllmer. An entropy approach to the time reversal of diffusion processes. In *Stochastic Differential Systems Filtering and Control: Proceedings of the IFIP-WG 7/1 Working Conference Marseille-Luminy, France, March 12–17, 1984*, pages 156–163. Springer, 2005.

[9] Jian Huang, Yuling Jiao, Lican Kang, Xu Liao, Jin Liu, and Yanyan Liu. Schrödinger-föllmer sampler: sampling without ergodicity. *arXiv preprint arXiv:2106.10880*, 2021.

[10] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[13] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd international conference on artificial intelligence and statistics*, pages 983–992. PMLR, 2019.

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[15] Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *CoRR*, abs/1511.06807, 2015. URL http://arxiv.org/abs/1511.06807.

[16] Yurii Evgen'evich Nesterov. A method of solving a convex programming problem with convergence rate o\bigl(k^2\bigr). In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences, 1983.

[17] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR, 2017.

[18] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[19] J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[20] Teodora Reu, Francisco Vargas, Anna Kerekes, and Michael M Bronstein. To smooth a cloud or to pin it down: Expressiveness guarantees and insights on score matching in denoising diffusion models. In *The 40th Conference on Uncertainty in Artificial Intelligence*.

[21] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[22] Paul Tseng. An incremental gradient (-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

[23] Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pages 3084–3114. PMLR, 2019.

[24] Francisco Vargas, Andrius Ovsianas, David Fernandes, Mark Girolami, Neil D Lawrence, and Nikolas Nüsken. Bayesian learning via neural schrödinger-föllmer flows. *arXiv preprint arXiv:2111.10510*, 2021.

[25] Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

[26] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. Omnipress, 2011. URL https://icml.cc/2011/papers/398_icmlpaper.pdf.

[27] Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

Appendix A. Proof of Main Theorem T4.6

A.1. General Approach

We can build up our result by considering several Schrödinger-Föllmer processes:

- X: A continuous analytic SFP (**T2.2**).
- $\widehat{X}$: A continuous neural SFP (**T2.3** but with integrals instead of EM-discretisation).
- $\widehat{Y}$: A discrete neural SFP (**T2.3**).

To evaluate $\widehat{Y}$'s proficiency at global optimisation for a given loss function V, the aim is to find a tight upper bound on the probability of $V(\widehat{Y}_1) > \tau$. In other words, to find the probability of our optimiser finding a solution with loss worse than τ.

Using the pushforward measure $P_{\widehat{Y}_1}$, one can write this as[5]

$$P_{\widehat{Y}_1}(V(X_1) > \tau).$$

To consider $\widehat{Y}$ a global optimiser, it should be possible to reduce this probabiliy to near-0 for any τ with correct choice of neural drift parameters θ as used in Definition **T2.3**.

A.2. Assumptions

To make use of the results of prior authors, we will first conglomerate their assumptions. To begin, to ensure that the SDE for X admits a unique and strong solution, Dai et al. [4] assume:

Assumption AA.1 *$V(x)$ is twice continuous differentiable on $\mathbb{R}$ and $V(x) = \|x\|_2^2/2$ outside a ball B_R, where B_R denotes the ball centred at origin with radius $R > 0$.*

Additionally, Tzen et al. [23] make several assumptions to ensure that the ratio f used in the analytic SFP definition **T2.2** can be efficiently approximated by a neural net:

Assumption AA.2 *For the analytic SFP definition (**T2.2**), f is differentiable, f and ∇f are L-Lipschitz, and there exists a minima $c \in (0, 1]$ such that $f \geq c$ everywhere.*

Assumption AA.3 *For any $R > 0$ and $\hat{\epsilon} > 0$, there exists a feedforward neural net $\widehat{f}$ with size polynomial in $(1/\hat{\epsilon}, p, L, R)$, such that*

$$\sup_{x \in \mathrm{B}^p(R)} |f(x) - \widehat{f}(x)| \leq \hat{\epsilon} \quad \textit{and} \quad \sup_{x \in \mathrm{B}^p(R)} \|\nabla f(x) - \nabla \widehat{f}(x)\| \leq \hat{\epsilon}.$$

Assumption AA.4 *For neural SFP $\widehat{X}$, the drift $\widehat{b}_\theta(x, t) = \widehat{v}(x, \sqrt{1-t})$ is determined by a neural net $\widehat{v} : \mathbb{R}^p \times [0, 1] \to \mathbb{R}^p$ with size polynomial in $(1/\hat{\epsilon}, p, L, c, 1/c)$, such that the activation function of each neuron is an element of the set $\{\mathrm{S}, \mathrm{S}', \mathrm{ReLU}\}$, where S is the sigmoid function.*

It is worth noting that none of our assumptions require V to be convex within the ball B_R; thus the global optimisation problem remains challenging.

5. Note that this takes advantage of the fact that $V(X_1)$ and $V(\widehat{Y}_1)$ have the same event space, thus $V(X_1) > \tau$ is the same event as $V(\widehat{Y}_1) > \tau$.

A.3. Proof of Main Theorem T4.6

We can phrase our optimisation probability in terms of its distance from the continuous neural SFP terminal distribution $\widehat{X}_1$:

$$\begin{aligned} P_{\widehat{Y}_1}(V(X_1) > \tau) &= P_{\widehat{X}_1}(V(X_1) > \tau) + P_{\widehat{Y}_1}(V(X_1) > \tau) - P_{\widehat{X}_1}(V(X_1) > \tau) \\ &\leq P_{\widehat{X}_1}(V(X_1) > \tau) + |P_{\widehat{Y}_1}(V(X_1) > \tau) - P_{\widehat{X}_1}(V(X_1) > \tau)|. \end{aligned} \tag{1}$$

Next, consider the total variation norm as given in Definition TA.8.

Definition TA.8 *For probability measures P and Q in measurable space $(\Omega, \mathcal{F})$, the total variation norm is defined as $2\times$ the upper bound on the distance between P and Q:*

$$\|P - Q\|_{TV} = 2 \sup_{A \in \mathcal{F}} |P(A) - Q(A)|.$$

Naturally, as the total variation norm is an upper bound, we have $|P(A) - Q(A)| = |Q(A) - P(A)| \leq \|Q - P\|_{TV}$ for all events $A \in \mathcal{F}$. Considering $(V(X_1) > \tau)$ as an event in the event space of the probability measures for $\widehat{X}_1$ and $\widehat{Y}_1$, we can apply the bound to Equation 1, giving us:

$$\begin{aligned} &P_{\widehat{X}_1}(V(X_1) > \tau) + |P_{\widehat{Y}_1}(V(X_1) > \tau) - P_{\widehat{X}_1}(V(X_1) > \tau)| \\ &\leq P_{\widehat{X}_1}(V(X_1) > \tau) + \|P_{\widehat{X}_1} - P_{\widehat{Y}_1}\|_{TV}. \end{aligned} \tag{2}$$

We can then move from the neural approximation $\widehat{X}$ to the analytic process X using the same trick:

$$\begin{aligned} &P_{\widehat{X}_1}(V(X_1) > \tau) + \|P_{\widehat{X}_1} - P_{\widehat{Y}_1}\|_{TV} \\ &\leq P_{X_1}(V(X_1) > \tau) + \|P_{X_1} - P_{\widehat{X}_1}\|_{TV} + \|P_{\widehat{X}_1} - P_{\widehat{Y}_1}\|_{TV}. \end{aligned} \tag{3}$$

Next, we can obtain a KL divergence representation by applying Pinsker's inequality, which I define in TA.9.

Definition TA.9 *For probability measures P and Q, Pinsker's inequality states that*

$$\|P - Q\|_{TV} \leq \sqrt{2 D_{KL}(P \parallel Q)}.$$

Thus, by Pinsker's inequality, we have:

$$\begin{aligned} &P_{X_1}(V(X_1) > \tau) + \|P_{X_1} - P_{\widehat{X}_1}\|_{TV} + \|P_{\widehat{X}_1} - P_{\widehat{Y}_1}\|_{TV} \\ &\leq P_{X_1}(V(X_1) > \tau) + \sqrt{2 D_{KL}(P_{X_1} \parallel P_{\widehat{X}_1})} + \sqrt{2 D_{KL}(P_{\widehat{X}_1} \parallel P_{\widehat{Y}_1})}. \end{aligned} \tag{4}$$

To continue, we will take assumptions AA.1 - AA.4 and apply three results from prior authors:

- *Dai et al.* [4] (Analytic optimisation bound 3.5):
  $P_{X_1}(V(X_1) > \tau) \leq C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right)$ under assumption AA.1.
- *Tzen et al.* [23] (Neural approximation bound 3.1):
  $D_{KL}(P_{X_1} \parallel P_{\widehat{X}_1}) \leq \widehat{\epsilon}$ under assumptions AA.2 - AA.4.

- *Vargas et al.* [24] (EM-Discretisation bound A6):
  $D_{KL}(P_{\widehat{X}_1} \parallel P_{\widehat{Y}_1}) \leq 2L'^2(C_{L'}/K + 1/K^2)$ under assumptions **AA.2** - **AA.4**.

Inserting these into Equation 4, we obtain:

$$P_{X_1}(V(X_1) > \tau) + \sqrt{2D_{KL}(P_{X_1} \parallel P_{\widehat{X}_1})} + \sqrt{2D_{KL}(P_{\widehat{X}_1} \parallel P_{\widehat{Y}_1})}$$
$$\leq C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) + \sqrt{2\widehat{\epsilon}} + \sqrt{4L'^2(C_{L'}/K + 1/K^2)}. \tag{5}$$

In summary, we have obtained a bound for $\widehat{Y}$ as an optimiser, proving Theorem **T4.6**:

$$P_{\widehat{Y}_1}(V(X_1) > \tau) \leq C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) + \sqrt{2\widehat{\epsilon}} + \sqrt{4L'^2(C_{L'}/K + 1/K^2)}. \tag{6}$$

Appendix B. Proof of Main Corollary T4.7

We can examine the terms of our core result **T4.6** in isolation to derive bounds on parameter σ, neural error $\widehat{\epsilon}$ and step count K for optimisation accuracy. In particular, suppose we wish to achieve τ-global minimisation with probability at least $1 - \sqrt{\delta}$ for arbitrary δ, meaning

$$P_{\widehat{X}_1}(V(X_1) > \tau) \leq C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) + \sqrt{2\widehat{\epsilon}} + \sqrt{4L'^2(C_{L'}/K + 1/K^2)} \leq \sqrt{\delta}. \tag{7}$$

We can achieve this by dividing $\sqrt{\delta}$ into three parts using constants $C_1 + C_2 + C_3 = 1$, such that our bound in Equation 7 is equivalent to solving the system:

$$C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) \leq C_1\sqrt{\delta} \quad \wedge \quad \sqrt{2\widehat{\epsilon}} \leq C_2\sqrt{\delta} \quad \wedge \quad \sqrt{4L'^2(C_{L'}/K + 1/K^2)} \leq C_3\sqrt{\delta}.$$

I will now address each part individually; careful inequality handling can lead each term to a big-$\mathcal{O}$ bound, proving Corollary **T4.7**:

1. Firstly, manipulating the σ bound (corresponding to the gap between sampling and optimisation via Definition **T2.5**):

$$\begin{aligned}
C_{\tau,\epsilon,p} \exp\left(-\frac{\tau - \epsilon}{\sigma}\right) &\leq C_1\sqrt{\delta} \\
\ln\left(\frac{C_1}{C_{\tau,\epsilon,p}}\sqrt{\delta}\right) &\geq -\frac{\tau - \epsilon}{\sigma} \\
\frac{1}{2}\ln\left(\frac{C_1^2}{C_{\tau,\epsilon,p}^2}\delta\right)\sigma &\geq -(\tau - \epsilon) \\
-\frac{2(\tau - \epsilon)}{\ln\left(\frac{C_1^2}{C_{\tau,\epsilon,p}^2}\delta\right)} &\geq \sigma \\
\mathcal{O}\left(\frac{\tau - \epsilon}{\ln(1/\delta)}\right) &\geq \sigma.
\end{aligned}$$

2. Secondly, manipulating the $\widehat{\epsilon}$ bound (corresponding to the gap between the analytical SFP and its neural approximation):

$$\begin{aligned}
\sqrt{2\widehat{\epsilon}} &\leq C_2\sqrt{\delta} \\
\widehat{\epsilon} &\leq C_2^2\delta/2 \\
\mathcal{O}(\delta) &\geq \widehat{\epsilon}.
\end{aligned}$$

3. Finally, manipulating the K bound (corresponding to the gap between continuous and EM-discretised neural SFP):

$$\begin{aligned}
\sqrt{4L'^2(C_{L'}/K + 1/K^2)} &\leq C_3\sqrt{\delta} \\
(4L'^2)(C_{L'}/K + 1/K^2) &\leq C_3^2\delta \\
(4L'^2)(KC_{L'} + 1) &\leq K^2C_3^2\delta \\
(4L'^2)(C_{L'} + 1) &\leq K^2C_3^2\delta \\
\frac{(C_{L'} + 1)(4L'^2)}{C_3^2\delta} &\leq K^2 \\
\mathcal{O}(L'/\sqrt{\delta}) &\leq K.
\end{aligned}$$

Appendix C. Evaluation Details

In evaluating PIO, we wish to account for the fact that a practical application of the optimiser would involve some degree of hyperparameter tuning; we also account for inherent stochasticity in both the diffusion model and dataset by re-running the optimisation over multiple seeds; and evaluate the optimiser at multiple tasks to assess generalisation.

C.1. Hyperparameter Sweeps

Optimisers are considerably challenging to compare fairly—in 2020, Choi et al. found that minor alterations to hyperparameter search space can **reverse classic optimisation results**, emphasising that hyperparameter relationships should not be neglected [3]. For example, they found a tendency for researchers to underestimate Adam's performance by not including its full breadth of parameters in a sweep.

To address this, we consider two distinct sweeps, each performed using the `BayesOpt` Bayesian optimisation algorithm included in Meta's Nevergrad library [19]. Firstly, **Narrow** $\times$**32**, a 32-run sweep with each run lasting 32 training steps. Secondly, **Wide** $\times$**64**: a 64-run sweep, each run lasting 64 training steps and making use of an LR-scheduling algorithm that drops LR by 50% on plateaus, and scales PIO's σ parameter to match.

- **Narrow** $\times$**32**: a 32-run sweep, each run lasting 32 training steps, over the following hyperparameters:
  - PIO: `lr, grad_clip_val, batch_size, sigma`.
  - Others: `lr, grad_clip_val, batch_size`.

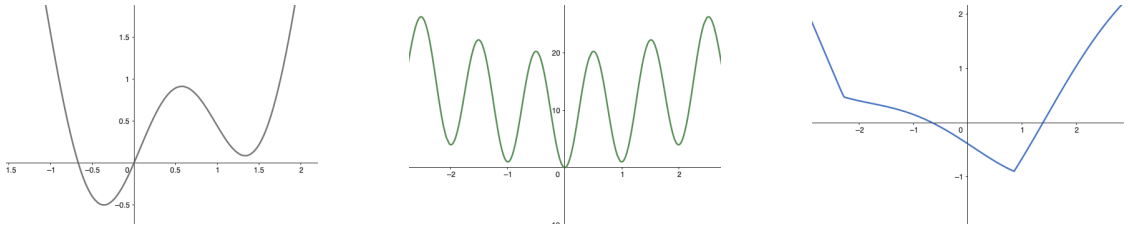
- **Wide** $\times$**64**: a 64-run sweep, each run lasting 64 training steps and making use of an LR-scheduling algorithm that drops LR by 50% on plateaus, and scales PIO's σ parameter to match. This is performed over the following hyperparameters:
  - PIO: `lr, grad_clip_val, batch_size, batch_laps, sigma, num_layers, K`.
  - Adam: `lr, grad_clip_val, batch_size, batch_laps, beta_1, beta_2`.
  - Others: `lr, grad_clip_val, batch_size, batch_laps`.

The idea with **Wide** $\times$**64** is that we aim to fully exploit each optimiser by including a wide variety of parameters. The unusual `batch_laps` parameter entails re-using the same mini-batch over multiple consecutive training steps, which may improve numerical stability. Figure **??** exemplifies parameters for the **Narrow** $\times$**32** sweep.

Appendix D. Relevant concepts

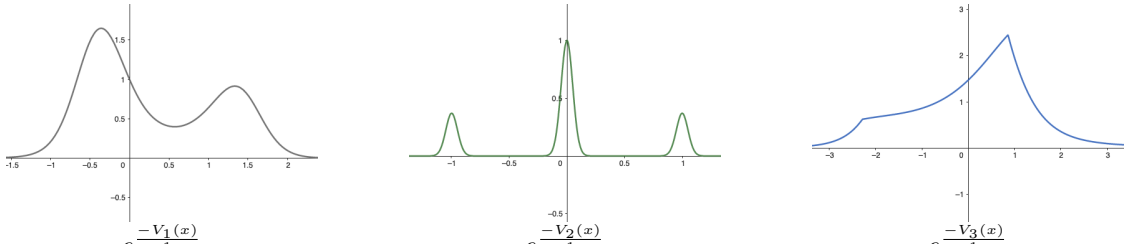
D.1. Boltzmann transformation and σ reduction

The following graphics are provided to give a visual representation of how the Boltzman transformation changes a univariate optimisation target (2), into a Boltzman distribution (3), and how reducing our optimisation parameter σ makes these distributions significantly sharper (4).



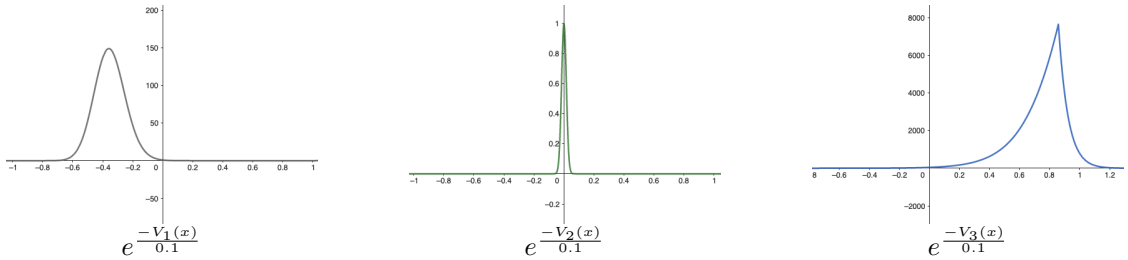
$V_1(x)$ $V_2(x)$ $V_3(x)$

Figure 2: Univariate optimisation target functions.



$e^{\frac{-V_1(x)}{1}}$ $e^{\frac{-V_2(x)}{1}}$ $e^{\frac{-V_3(x)}{1}}$

Figure 3: Univariate optimisation target functions following the Boltzman transformation with $\sigma = 1$.



$e^{\frac{-V_1(x)}{0.1}}$ $e^{\frac{-V_2(x)}{0.1}}$ $e^{\frac{-V_3(x)}{0.1}}$

Figure 4: Univariate optimisation target functions following the Boltzman transformation with $\sigma = 0.1$.

Appendix E. Further Results

E.0.1. NARROW ×32 SWEEP

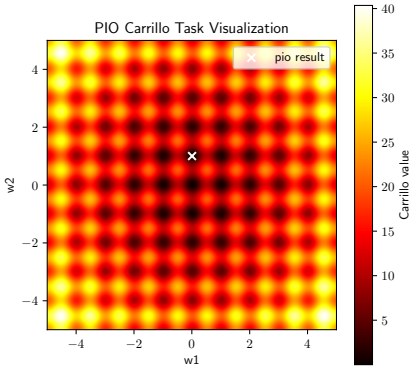


Figure 5: The Carrillo point found by the best PIO run in its ×32 hyperparameter sweep; the optimiser appears stuck in a local minima.

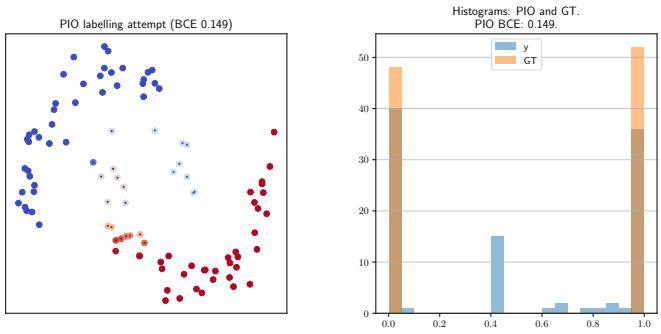


Figure 6: On the left: PIO's final attempt at the Moons labelling task after ×32 sweeping, with ground truth marked with a small dot in the center of each circle. On the right: a histogram of PIO's labels compared to the ground truth, demonstrating room for improvement.

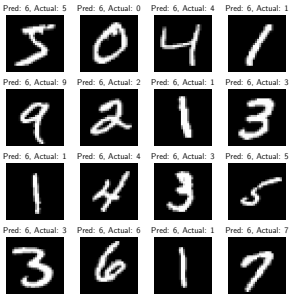


Figure 7: The MNIST labels predicted by PIO's best run in its ×32 hyperparameter sweep; every label is the same, implying that the optimiser is stuck in a very early local minima.

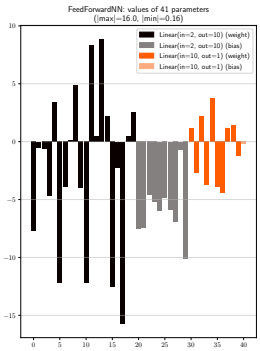


Figure 8: The parameters ϕ found by PIO for the Moons problem. All biases are negative, which may imply underutilisation.

Figure 9 demonstrates the training loss for the best-found configurations, averaged over a number of runs, reduced for MNIST due to its computational cost.

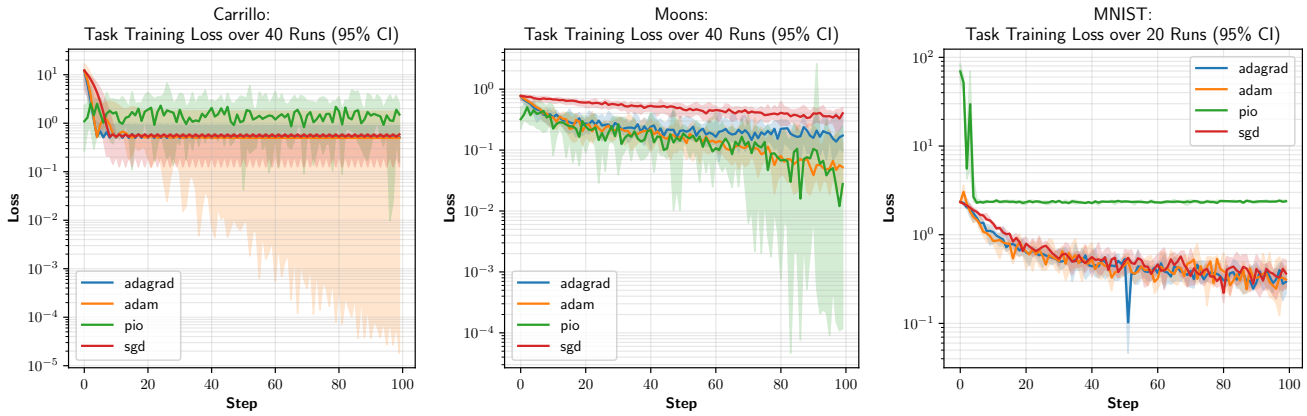


Figure 9: Training loss $V(x)$ for various optimisers, tuned with a **Narrow** $\times$**32** hyperparameter sweep of 32 runs. Confidence intervals are computed by re-running the best-found configuration across multiple seeds.

E.0.2. WIDE $\times$64 SWEEP

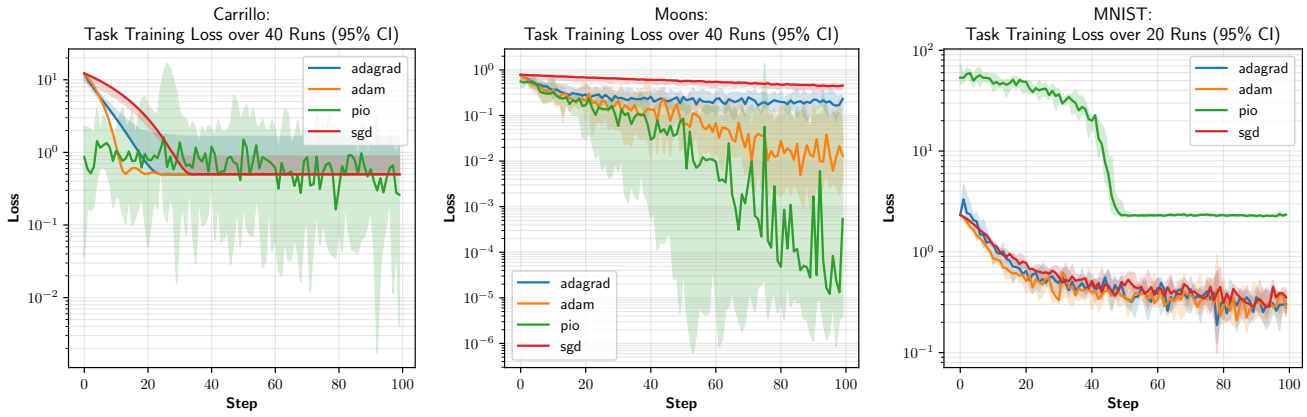


Figure 10: Training loss $V(x)$ for various optimisers, tuned with a **Wide** $\times$**64** hyperparameter sweep of 64 runs, at learning three different task environments. Confidence intervals are computed by re-running the best-found configuration across multiple seeds. Notably, PIO still fails to learn the MNIST task.